Dynamically Consistent Online Adaptation of Fast Motions for Robotic Manipulators

Alexander Pekarovskiy, *Student Member, IEEE,* Thomas Nierhoff
Sandr Hirche, *Senior Member, IEEE,* and Martin Buss, *Fellow, IEEE*

***Abstract*—The planning and execution of real-world robotic tasks largely depend on the ability to generate feasible motions online in response to changing environment conditions or goals. A spline deformation method is able to modify a given trajectory so that it matches the new boundary conditions, e.g. on positions, velocities, accelerations, etc. At the same time, the deformed motion preserves velocity, acceleration, jerk or higher derivatives of motion profile of precalculated trajectory. The deformed motion possessing such properties can be expressed by translation of original trajectory and spline interpolation. This spline decomposition considerably reduces the computational complexity and allows the real-time execution. Formal feasibility guarantees are provided for the deformed trajectory and for the resulting torques. These guarantees are based on the special properties of Bernstein polynomials used for the deformation and on the structure of the chosen computed torque control scheme. The approach is experimentally evaluated in a number of planar volleyball experiments using 3-DoF robots and human participants.**

***Index Terms*—Manipulation planning, motion adaptation, motion control, path planning for manipulators.**

I. Introduction

FOR many years robotic tasks used preplanned repetitive motions for industrial applications. However, nowadays there is a deep need for robots with the capability of doing a variety of tasks with desired encoded behavior but without offline recomputation of the whole motion. It becomes increasingly important to generate motions in accordance with changing goals and adapt to the changing environments. Thus, motion planning for reactive real-world scenarios needs to fulfill several requirements. On the one hand, it needs to comply with the imposed constraints such as position, velocity or acceleration limits, be able to avoid obstacles and at the same time prevent torque saturation. On the other hand, resulting solutions should be generated online, if not real-time, as a response to the updated sensory data. That is particularly important for dynamic manipulation, motion imitation and autonomous driving.

There exist different conceptual approaches to motion generation, however, none of them is capable of satisfying all the requirements. Learning methods for motion generation can be fast, but they generally do not include robot model and guaranteed online constraint satisfaction. On the contrary, planning and replanning based on classical optimal control may address all type of constraints including the nonlinear robot dynamics, i.e. they are dynamically consistent. However, it is computationally expensive and in realistic cases too slow for online application. Instead of complete replanning we propose to resort to efficient approximate methods. The key scientific challenge is to incorporate the different types of constraints for a modified motion under specified time limits.

The subject of this article is online motion deformation which preserves the derivative profile of an initial (planned or learned) motion, incorporates desired boundary conditions and checks violation of trajectory and torque constraints. The resulting approach provides a reactive motion generation suitable for robotic manipulators with non-negligible dynamics.

A. Related Work

Existing motion generation methods only partially possess the desired features for online motion adaptation.

1) Optimal Control and Optimization Methods: Optimal control allows setting task goals, imposing a large variety of constraints and achieving the desired performance. Optimal Control Problems (OCPs) are oftentimes solved with numerical approximate methods such as direct collocation or multiple shooting [1]. Due to the complexity of OCPs, such a recalculation cannot be done online, especially for nonlinear dynamic systems and nonlinear constraints. Therefore, some relaxation of OCP conditions is required to improve the computation time. For instance, the method for creating optimal motions with the satisfaction of continuous inequality constraints offline and replanning them for humanoid robots is implemented in [2]. For these safe motions, solutions are found using semi-infinite programming which is computationally costly, whereas replanning them might be faster but it works only in the vicinity of the safe motions in its joint space. Another option is to relax the original OCP to a trajectory optimization problem, essentially assuming that the control will do the job of perfectly tracking the trajectory. The nonlinear

Manuscript received December 30, 2016; accepted September 23, 2017. This paper was recommended for publication by Associate Editor P. Fraisse and Editor A. Billard upon evaluation of the reviewers' comments. This work was supported in part by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. [267877], partly from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement n. [643433], project "Robotic Assistant for MCI Patients at home (RAMCIP)" and in part by the Technical University of Munich - Institute for Advanced Study (www.tum-ias.de), funded by the German Excellence Initiative.(Corrsponding author: Alexander Pekarovskiy.)

A. Pekarovskiy and M. Buss are with the Chair of Automatic Control Engineering and the TUM Institute for Advanced Study, Technical University of Munich, Lichtenbergstr. 2a, 85748 Garching, Germany
{a.pekarovskiy, mb} at tum.de

T. Nierhoff and S. Hirche are with the Chair of Information-oriented Control, Technical University of Munich, D-80333 Munich, Germany
{tn, hirche} at tum.de

OCP is transformed into a large-scale optimization program in [3] with B-splines used for joint positions. Time intervals and corresponding polynomials are used for approximation of inequality constraints and the cost. However, this method does not consider the influence of external disturbance or system parameter uncertainties on the tracking controller. Thus, motion feasibility can't be fully guaranteed. For sampled trajectories, optimization can be performed with gradient descent methods [4], [5] or with sequential convex optimization [6]. Optimization of polynomial trajectories of the end-effector with inequality contact constraints can be done by varying the boundary conditions [7]. Sampling based RRT replanners are used in [8], [9] to compute multiple feasible motions and to choose the best one based on predefined metrics. These methods are inherently of a randomized nature and thus require excessive computation time to generate multiple motions and check for obstacle avoidance.

2) Time-optimal Path Planners: In time-optimal path tracking [10], trajectories are created through replanning of a given trajectory in the temporal domain. This method satisfies torque constraints but requires recalculation of a boundary curve for maximum velocity for every newly prescribed path. Moreover, it is only applicable to robots with perfect model knowledge. The method has been extended in [11], [12] and [13] to take also admissible acceleration regions and torque constraints into account. Online reactive motion planner [14] produced trajectories with different time constants of visual and force loops using cubic curves. Time optimality along with time-synchronization is achieved in [15] by calculating minimum synchronization times and corresponding polynomials for open-loop joint trajectories every computation cycle, but it doesn't take robot dynamics into account.

3) Learning Methods: Learning methods can be applied for trajectory generation using attractor dynamics. For instance, Dynamic Movement Primitives (DMPs) [16], [17], [18] and Gaussian mixture models (GMM) [19] are commonly used for trajectory generation with motion imitation. A mixture of kinesthetically learned motion primitives is implemented in [20]. Motion generalization is provided only in the vicinity of the learned DMPs. Thus, a large database of demonstrations that scales with the number of new situations and controlled parameters is required. Learning approaches are easy to program, but they become computationally inefficient when constraints' satisfaction for the produced motions must be checked.

4) Deformation Methods: Retargeting or deformation of previously calculated motions can be implemented to preserve desired features of the task. Gleicher [21] introduced retargeting motions for characters in computer animation using B-splines and large space-time constraint optimization. A multilevel B-spline fitting technique [22] is proposed to incorporate a big set of position and orientation constraints. A non-rigid registration [23] method is applied to the tasks with variable scene geometry and initial conditions. This approach minimizes the geometrical distance between original and target sets of points and additionally provides smoothness of the mapping function using thin plate spline regularizer. Deformations based on affine transformations [24] are suggested for human-related applications as they preserve affine-invariant features of the trajectories. Laplacian trajectory editing [25] preserves the local trajectory properties through a least squares approach while keeping a set of positional constraints fixed. This method is based on a Laplacian mesh optimization [26] from computer graphics to approximate 3D surface meshes, but it is applied for path deformation.

5) Boundedness of the Robot Motion: Once the desired robot motion is obtained, trajectory and torque bounds along with a trajectory tracking controller play a critical role to execute it properly. Reactive control and robustification for the running robot are done in [27] by combining deadbeat controller and the sequential composition [28]. This method works for discrete behavioral policies and chooses goal states taken from a discrete set. Modeling errors are transformed in a joint acceleration disturbance [29] to track a trajectory with a prescribed accuracy. Uniform boundedness of inertia [30], [31] and Coriolis/centripetal [32] matrices is important for designing controllers and providing global Lyapunov stability. A computed torque controller yields tracking error dynamics as a linear system, where an error is dependent on the inverse of the mass-inertia matrix, the disturbances and the feedback controller parametrization [33].

The question is how to provide a computationally efficient substitute for the optimal control solver with the specific cost function that preserves dynamic properties of the existing robotic motions. It must also incorporate boundary conditions with their respective time instances and major inequality constraints for kinematic and dynamic robot parameters.

B. Contributions

A novel method based on splines for online recomputation of fast reactive robotic motions and formal guarantees for feasibility of the resulting robot motions is derived. Trajectories obtained with the spline deformation method preserve the desired properties of the original trajectories that depend on one or several derivatives of the motion profile. This motion adaptation provides minimization of velocity deviation for linear splines, minimization of acceleration deviation for cubic splines, jerk deviation minimization for quintic splines, etc. Depending on the application, different motion derivatives are of interest: for example, the velocity for mobile robot navigation, the acceleration in dynamic tasks such as batting or locomotion, and the jerk for natural human motion imitation. After the deformation, a feasibility verification method provides upper bounds for task space positions, velocities and accelerations, as well as joint torques. Tight bounds for torques are derived using interval-based analytical expressions for individual torque components and a mixture of analytical and ∞-norm. By implementing such a mixture, we provide a trade-off between tightness of the bounds for inequality constraints and computational time. The bounds for the feedback error and the control signal are derived for a controller depending on the parametric uncertainties and the external disturbance.

Experimental validation showing online capability, adaptability and robustness of the method is provided for several robot-robot and robot-human planar volleyball scenarios, where discontinuous short contacts with a ball require high accuracy and precise timing of the motion. Choice of the best hitting action and the best-precalculated motion are derived based on the chosen game strategy. An overall torque analysis is carried out for the given robotic system.

In [34] we presented a preliminary approach for retargeting of specially designed optimal trajectories in constrained environments using cubic splines. Still, it wasn't capable of keeping desired derivative profiles and providing boundary conditions higher than velocities. In [35] we derived infinity-norm boundaries for robot joint torques in the case of the cubic spline interpolation. These bounds have a closed-form solution but are quite conservative in presence of external disturbance. Based on these preliminary results, in this article, we introduce a method which is capable of motion adaptation with a broad range of dynamic behaviors and online feasibility guarantees for the resulting robotic motions.

The remainder of the article is as follows: Sec. II introduces a framework for motion control using spline deformation. Sec. III describes the trajectory deformation process. In Sec. IV, the feasibility of trajectory tracking for kinematic and dynamic constraints is analyzed. The experiments for a planar volleyball scenario are described in Sec. V. Finally, Sec. VI discusses the presented approach and suggests ideas for further expansion.

II. Spline Deformation Framework for Motion Control

We consider online motion generation for reactive robotic tasks with prescribed behavior. For such types of motions, it is important to satisfy specific boundary conditions for free motion or contacts and to preserve the chosen derivative of the motion profile. Additionally, generated motions need to comply with position, velocity, acceleration and joint torque constraints. All the following requirements are incorporated in the spline deformation approach, which is proposed here.

The motion generator is based on precomputed motion or set of motions that are calculated offline and the spline deformation method that incorporates new boundary conditions obtained from updated sensory data. Figure 1 illustrates a scenario for a robot that moves from its new initial state to the new end state in a task space.

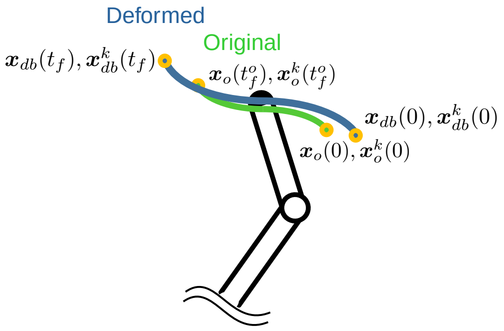

Deformed
Original
$\boldsymbol{x}_{db}(t_f), \boldsymbol{x}^k_{db}(t_f)$
$\boldsymbol{x}_o(t^o_f), \boldsymbol{x}^k_o(t^o_f)$
$\boldsymbol{x}_{db}(0), \boldsymbol{x}^k_{db}(0)$
$\boldsymbol{x}_o(0), \boldsymbol{x}^k_o(0)$

Figure 1: Robotic manipulator operating within its task space. The task space trajectory $\boldsymbol{x}_o$ (green) of the end effector is deformed with the new boundary conditions $\boldsymbol{x}_{db}, \boldsymbol{x}^k_{db}$. The resulting deformed trajectory $\boldsymbol{x}_d$ is depicted with the blue curve.

A. Problem Formulation for Trajectory Deformation

The overall deformation procedure with preservation of the desired derivative can be formulated as an optimization problem. Find a trajectory and its derivatives $\boldsymbol{x}_d(t), \dot{\boldsymbol{x}}_d(t), \ddot{\boldsymbol{x}}_d(t)$ which minimize the difference between the n-th derivative of this trajectory and the n-th derivative of the original trajectory $\boldsymbol{x}_o$ for all task space coordinates, given the original motion, its time duration and the new boundary conditions up to $(n-1)$-th derivative for $n \geq 1$:

$$\underset{\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d, \ddot{\boldsymbol{x}}_d}{\text{minimize}} \quad J = \int_0^{t_f} \left\| \frac{\mathrm{d}^n \boldsymbol{x}_d}{\mathrm{d}t^n} - \frac{\mathrm{d}^n \boldsymbol{x}_o}{\mathrm{d}t^n} \right\|^2 \mathrm{d}t \tag{1a}$$

$$\text{subject to} \quad \boldsymbol{x}_d(\boldsymbol{t}_b) = \boldsymbol{x}_{db}, \tag{1b}$$

$$\boldsymbol{x}^{(k)}_d(\boldsymbol{t}_b) = \boldsymbol{x}^{(k)}_{db}, \quad k \in \{1, \dots, n-1\}, \tag{1c}$$

$$t_f = t^o_f, \tag{1d}$$

$$\boldsymbol{g}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t), \boldsymbol{\tau}(t)) \leq 0, \quad t \in [0; t_f], \tag{1e}$$

where $\boldsymbol{x}_d \in \mathbb{R}^m$ is the overall deformed trajectory, m is the dimensionality of the robot task space coordinates, n is the derivative order of the motion profile that needs to be preserved, t_f is the fixed final time, and t^o_f is the time duration of the original trajectory. Boundary and switching conditions $\boldsymbol{x}_{db}, \dot{\boldsymbol{x}}_{db}, \dots, \boldsymbol{x}^{(n-1)}_{db}$ need to match at $\boldsymbol{t}_b = \begin{bmatrix} 0 & \boldsymbol{t}_s & t_f \end{bmatrix}$, where switching times $\boldsymbol{t}_s = \begin{bmatrix} t_{s1} & \dots & t_{sn} \end{bmatrix}$ are assigned when the deformed motion consists of several phases.

In contrast to OCPs, the overall optimization (1) does not explicitly consider the dynamics of the robot. Therefore, it is important to have feasibility check to cope with inequality constraints $\boldsymbol{g}$ for produced joint positions, velocities, accelerations $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ and torques $\boldsymbol{\tau} : |\boldsymbol{\tau}| \leq \boldsymbol{\tau}_{\text{nom}}$, which implicitly encode the constraints imposed by the underlying robot dynamics including actuator constraints. Thus, inequality constraints (1e) guarantee dynamic consistency of the deformed trajectory, which has to be followed by the real-world manipulator. The cost function (1a) provides motion derivative preservation. Equality constraints (1b)-(1d) impose boundary conditions for initial, final and intermediate time instances of the motion.

B. Conceptual Approach

The block diagram of the spline deformation framework for the robot control is presented in Fig. 2. It consists of an offline trajectory generator in task space and a model based algorithm that sets new fixed boundary conditions. It is beneficial to keep several precomputed trajectories in the database to have several distinctive motions with various motion derivative profiles and execution times. The new trajectory $\boldsymbol{x}_d$, $\dot{\boldsymbol{x}}_d$, $\ddot{\boldsymbol{x}}_d$ is generated by spline deformation and then followed by a feedback controller.

The solution of the spline deformation problem is decomposed into several parts that will be discussed further in the article. First, we precompute trajectories that possess desired properties. Second, the deformation algorithm is applied to preserve derivatives of the motion profile and fit new boundary conditions. Third, feasibility guarantees are provided for trajectories obtained through deformation.

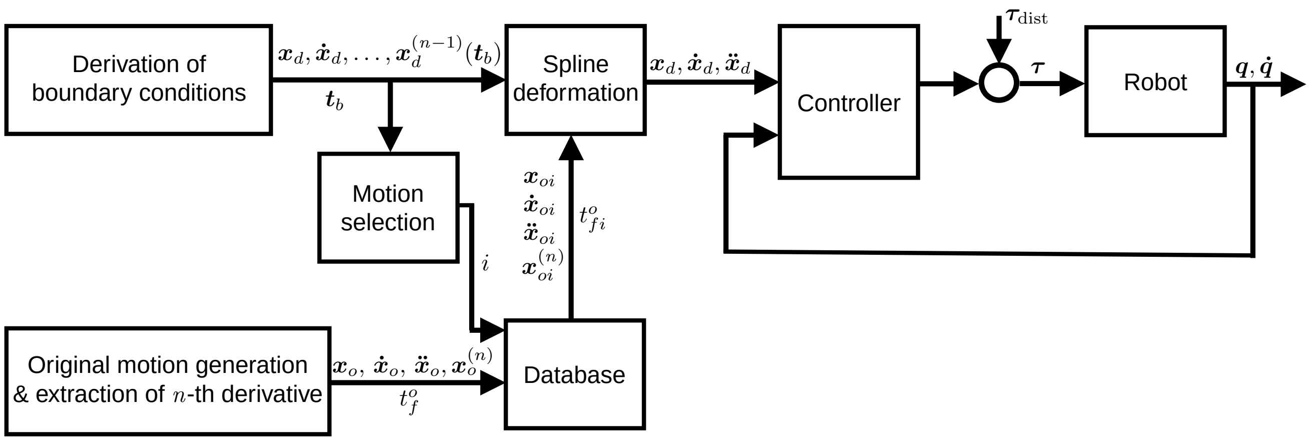

Derivation of boundary conditions
$\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d, \dots, \boldsymbol{x}_d^{(n-1)}(\boldsymbol{t}_b)$
$\boldsymbol{t}_b$
Spline deformation
$\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d, \ddot{\boldsymbol{x}}_d$
Controller
$\boldsymbol{\tau}_{\text{dist}}$
$\boldsymbol{\tau}$
Robot
$\boldsymbol{q}, \dot{\boldsymbol{q}}$
Motion selection
i
$\boldsymbol{x}_{oi}$ $\dot{\boldsymbol{x}}_{oi}$ $\ddot{\boldsymbol{x}}_{oi}$ $\boldsymbol{x}_{oi}^{(n)}$
t_{fi}^o
Original motion generation & extraction of n-th derivative
$\boldsymbol{x}_o, \dot{\boldsymbol{x}}_o, \ddot{\boldsymbol{x}}_o, \boldsymbol{x}_o^{(n)}$
t_f^o
Database

Figure 2: Overall motion planning and control scheme. New boundary and switching states $\boldsymbol{x}_d(\boldsymbol{t}_b)$, $\boldsymbol{x}_d^{(k)}(\boldsymbol{t}_b)$ and its timing $\boldsymbol{t}_b$ are based on the task goal. A motion selection algorithm chooses one original motion $\boldsymbol{x}_o, \dot{\boldsymbol{x}}_o, \ddot{\boldsymbol{x}}_o$ of time duration t_f^o from the database that fits the desired task best. Spline deformation of the original trajectory produces a new trajectory $\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d, \ddot{\boldsymbol{x}}_d$ that passes through new fixed states and preserves a desired n-th derivative of the selected original motion. If deformation is performed in task space coordinates, the trajectory is transformed into joint space coordinates before passing it to the controller. The controller follows this trajectory, given the external disturbance $\boldsymbol{\tau}_{\text{dist}}$.

The major steps for the application of the spline deformation method are detailed in the Algorithm 1. The more detailed explanation for separate steps will be given further in the paper.

Algorithm 1 Spline deformation and feasibility check

```
Generation of original motions and corresponding grids
Assignment of $\boldsymbol{K}_p, \boldsymbol{K}_v$ controller gains
Setting the desired derivatives to be conserved
Encoding derivatives of the original trajectories in $\boldsymbol{\Delta}$
Calculation of bounds for $\boldsymbol{x}_o, \dot{\boldsymbol{x}}_o, \ddot{\boldsymbol{x}}_o$
Sorting precalculated motions w.r.t. the task objective
isMotionFeasible ← $false$
$i \leftarrow 1$
**repeat**
    Initialization of boundary conditions $\boldsymbol{x}_d(\boldsymbol{t}_b)$, $\boldsymbol{x}_d^{(k)}(\boldsymbol{t}_b)$, $\boldsymbol{t}_b$
    Choice of the $i$-th motion from the database
    Spline deformation of the motion
    **if** bounds for $\tau$, $\dot{q}$ or $\ddot{q}$ are exceeded **or** $t \geq \Delta t_{\text{comp}}$ **then**
        i++
    **else**
        isMotionFeasible ← $true$
    **end if**
**until** isMotionFeasible **or** $i > i_{\max}$
**if** isMotionFeasible **then**
    Execute motion
**else**
    Stop motion
**end if**
```

The spline deformation algorithm works for original motions that should be at least C^2 smooth. These motions can be obtained by applying motion generation methods or through demonstration. Regardless of what motion generation method is chosen, the deformation procedure will preserve the chosen derivatives of motion profile of the original trajectory which possess essential behavior information for the task.

III. Spline Deformation Method

In order to deform trajectories, we design the method that keeps chosen local derivatives of motion profile, incorporates new boundary conditions and modifies the entire trajectory – not just a single time step. In this section, a solution for the problem (1) is derived, for now excluding feasibility guarantees for inequality constraints (1e).

A. Finite Difference Parametrization of Trajectory Derivatives

Local trajectory properties for trajectory deformation can be represented as a linear system of equations. Together with boundary constraints, we obtain an overdetermined system of equations that can be solved using least squares. The original trajectory is split into n_d equitemporally spaced sampling points

$$\begin{aligned} \boldsymbol{P} &= [\boldsymbol{p}(t_1), \boldsymbol{p}(t_2), \dots, \boldsymbol{p}(t_{n_d})]^T \in \mathbb{R}^{n_d \times m}, \\ t_{i+1} - t_i &= \Delta t = \frac{t_{n_d} - t_1}{n_d}, \ \forall i \in \{1, \dots, n_d - 1\}, \end{aligned} \tag{2}$$

where m is the number of task space coordinates, trajectory duration (1d) is fixed as $t_{n_d} = t_f$.

Using this representation, we discretize the original trajectory as $\boldsymbol{P}_o = [\boldsymbol{p}_1^o, \boldsymbol{p}_2^o, \dots, \boldsymbol{p}_{n_d}^o]^T$. Local derivative properties of original trajectory are encoded with the differential Laplace-Beltrami operator $\boldsymbol{\delta}$ [26]. We use local derivatives of motion profile that are calculated for each sampling point i through the finite difference. The velocity and the acceleration along trajectory $\boldsymbol{x}_o$ are derived as

$$\begin{aligned} \boldsymbol{\delta}_{v,i} &= \frac{\boldsymbol{p}_{i+1}^o - \boldsymbol{p}_{i-1}^o}{2\Delta t}, \quad i \in \{2, \dots, n_d - 1\}, \\ \boldsymbol{\delta}_{a,i} &= \frac{\boldsymbol{p}_{i+1}^o - 2\boldsymbol{p}_i^o + \boldsymbol{p}_{i-1}^o}{\Delta t^2}, \quad i \in \{2, \dots, n_d - 1\}, \\ \boldsymbol{\Delta} &= \begin{bmatrix} \boldsymbol{\delta}_2 & \boldsymbol{\delta}_3 & \dots & \boldsymbol{\delta}_{n_d-2} & \boldsymbol{\delta}_{n_d-1} \end{bmatrix}^T, \end{aligned} \tag{3}$$

where n_d is a number of sampling points. At these points we want to keep local properties of the trajectory. For higher order derivatives, we need to apply special finite differences for the border elements, e.g. the expression for jerk is as follows

$$\begin{aligned}\boldsymbol{\delta}_{j,i} &= \frac{\boldsymbol{p}^o_{i+2} - 2\boldsymbol{p}^o_{i+1} + 2\boldsymbol{p}^o_{i-1} - \boldsymbol{p}^o_{i-2}}{\Delta t^3}, \; i \in \{3,\dots,n_d-2\}\,,\\ \boldsymbol{\delta}_{j,i} &= \frac{\boldsymbol{p}^o_{i+2} - 3\boldsymbol{p}^o_{i+1} + 3\boldsymbol{p}^o_{i} - \boldsymbol{p}^o_{i-1}}{\Delta t^3}, \quad i = 2,\\ \boldsymbol{\delta}_{j,i} &= \frac{\boldsymbol{p}^o_{i+1} - 3\boldsymbol{p}^o_{i} + 3\boldsymbol{p}^o_{i-1} - \boldsymbol{p}^o_{i-2}}{\Delta t^3}, \quad i = n_d-1,\end{aligned} \tag{4}$$

where right-sided finite difference scheme is applied for $\boldsymbol{\delta}_{j,2}$ and left-sided finite difference scheme for $\boldsymbol{\delta}_{j,n_d-1}$.

Positional constraints of the form

$$\boldsymbol{p}_j = \boldsymbol{c}_j, \quad j \in \{1,2,\dots,n_d\} \tag{5}$$

pin a specific sampling point $\boldsymbol{p}_j$ to a desired position $\boldsymbol{c}_j$ with a weighting factor $w_j \in \mathbb{R}^+_0$ to prioritize the constraints. The matrix $\bar{\boldsymbol{C}}$ contains all boundary positions c_j at the sampling points.

The matrices $\bar{\boldsymbol{P}}$ and $\bar{\boldsymbol{C}}$ of dimension $p \times n_d$ assign p positional constraints as described in (5) with

$$\bar{\boldsymbol{P}}_{i,j} = \begin{cases} 1, & \text{if } \boldsymbol{p}_j = \boldsymbol{c}_i, \quad i \in \{1,\dots,p\} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

For instance, $p = 2$ for a single phase and matrix $\bar{\boldsymbol{P}}$ becomes

$$\bar{\boldsymbol{P}} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}_{p\times n_d}. \tag{7}$$

We minimize the discrete version of the cost function (1a) and set the boundary positions (1b) by rewriting (3), (4) and (5) in the following matrix form

$$\begin{bmatrix} \boldsymbol{L} \\ \bar{\boldsymbol{P}} \end{bmatrix} \boldsymbol{P} = \begin{bmatrix} \boldsymbol{\Delta} \\ \bar{\boldsymbol{C}} \end{bmatrix}, \tag{8}$$

with the following finite differences for velocity profile

$$\boldsymbol{L}_1 = \frac{1}{\Delta t} \begin{bmatrix} -\frac{1}{2} & 0 & \frac{1}{2} & & & & \\ & -\frac{1}{2} & 0 & \frac{1}{2} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -\frac{1}{2} & 0 & \frac{1}{2} & \\ & & & & -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}_{n_d-2\times n_d}, \tag{9}$$

for acceleration profile

$$\boldsymbol{L}_2 = \frac{1}{\Delta t^2} \begin{bmatrix} 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \end{bmatrix}_{n_d-2\times n_d}, \tag{10}$$

and for jerk profile

$$\boldsymbol{L}_3 = \frac{1}{\Delta t^3} \begin{bmatrix} -1 & 3 & -3 & 1 & & & & \\ -1 & 2 & 0 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & -1 & 2 & 0 & -2 & 1 & \\ & & & -1 & 3 & -3 & 1 & \end{bmatrix}_{n_d-2\times n_d}. \tag{11}$$

The higher derivatives for the discrete Laplace-Beltrami operator are derived analogously.

The discretized deformed trajectory $\boldsymbol{P}_s$ is obtained by solving the following system of equations

$$\boldsymbol{P}_s = \begin{bmatrix} \boldsymbol{L} \\ \bar{\boldsymbol{P}} \end{bmatrix}^+ \begin{bmatrix} \boldsymbol{\Delta} \\ \bar{\boldsymbol{C}} \end{bmatrix}. \tag{12}$$

Solving (12) requires pseudo-inversion of the matrix $\begin{bmatrix} \boldsymbol{L} & \bar{\boldsymbol{P}} \end{bmatrix}^T$. Thus, for a large number of sampling points and positional constraints this method becomes infeasible for real-time computation and hence an alternative is derived in the next section.

B. Spline Decomposition for a Single Derivative

We introduce an equivalent form for a single motion phase (12) by splitting the term for positional constraints $\bar{\boldsymbol{C}}$ into two parts and converting them into the continuous-time functions. With this decomposition we do not need to compute matrix pseudoinverse. The resulting continuous trajectory will be the sought deformed trajectory $\boldsymbol{x}_d$ from (1a)-(1d)

$$\boldsymbol{x}_d = \underbrace{\begin{bmatrix} \boldsymbol{L} \\ \bar{\boldsymbol{P}} \end{bmatrix}^+ \begin{bmatrix} \boldsymbol{\Delta} \\ \bar{\boldsymbol{C}}_1 \end{bmatrix}}_{\mathcal{A}} + \underbrace{\begin{bmatrix} \boldsymbol{L} \\ \bar{\boldsymbol{P}} \end{bmatrix}^+ \begin{bmatrix} \boldsymbol{0} \\ \bar{\boldsymbol{C}}_2 \end{bmatrix}}_{\mathcal{B}} = \underbrace{\boldsymbol{x}_o(t) + \boldsymbol{\delta x}_i}_{\mathcal{A}} + \underbrace{\boldsymbol{x}_s(t)}_{\mathcal{B}}.$$
$$\text{s.t. } \bar{\boldsymbol{C}}_1 + \bar{\boldsymbol{C}}_2 = \bar{\boldsymbol{C}}, \qquad \boldsymbol{\delta x}_i = \boldsymbol{x}_i - \boldsymbol{x}_o(0) \tag{13}$$

First, positional constraints in $\bar{\boldsymbol{C}}_1$ are chosen so that the term $\mathcal{A}$ results in a translation of the original trajectory $\boldsymbol{x}_o$ to the new initial state, see 1D example in Fig. 3.

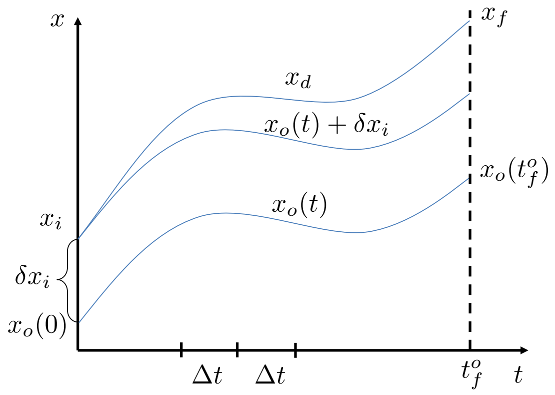


Figure 3: A 1-dimensional example of translation of the original trajectory with spline interpolation.

Second, $\bar{\boldsymbol{C}}_2$ is obtained as the difference $\bar{\boldsymbol{C}} - \bar{\boldsymbol{C}}_1$. Consequently, the term $\mathcal{B}$ minimizes n-th derivative of motion profile that is encoded in $\boldsymbol{\Delta}$ while fitting $\bar{\boldsymbol{C}}_2$. The proof for n-th derivative minimization is provided in Appendix A. The equivalent solution for $\mathcal{B}$ is a spline interpolation of the form

$$\boldsymbol{x}_s(t) = \sum_{k=0}^{2n-1} \boldsymbol{a}_k t^k. \tag{14}$$

Coefficients $\boldsymbol{a}_k$ are defined by positional constraints $\bar{\boldsymbol{C}}_2$ and additional specified boundary conditions (1c) - velocities, accelerations, etc. This way, the system of equations to be solved for all coordinates has only $2nm$ unknowns.

C. Spline Decomposition for a Weighted Sum of Derivatives

When the task requires simultaneous preservation of several derivatives of the original motion, the same methodology is applied. The optimization problem will have the following form

$$\begin{aligned} \underset{\boldsymbol{x}_d,\dot{\boldsymbol{x}}_d,\ddot{\boldsymbol{x}}_d}{\text{minimize}} \quad & J=\int_0^{t_f}\sum_{i=1}^{n}\beta_i\left\|\frac{\mathrm{d}^i\boldsymbol{x}_d}{\mathrm{d}t^i}-\frac{\mathrm{d}^i\boldsymbol{x}_o}{\mathrm{d}t^i}\right\|^2\mathrm{d}t\\ \text{subject to}\quad & \boldsymbol{x}_d(\boldsymbol{t}_b)=\boldsymbol{x}_{db}, \qquad (15)\\ & \boldsymbol{x}_d^{(k)}(\boldsymbol{t}_b)=\boldsymbol{x}_{db}^{(k)},\quad k\in\{1,\dots,n-1\},\\ & t_f=t_f^o, \end{aligned}$$

where β_i is a weighting coefficient for i-th derivative and n is a maximum derivative to be preserved.

For every preserved derivative we introduce the Laplacian $\boldsymbol{L}_i$ and finite difference matrix $\boldsymbol{\Delta}_i$, where the index i denote the corresponding derivative order.

$$\begin{bmatrix}\beta_1\boldsymbol{L}_1\\ \cdots\\ \beta_n\boldsymbol{L}_n\\ \bar{\boldsymbol{P}}\end{bmatrix}\boldsymbol{P}=\begin{bmatrix}\beta_1\boldsymbol{\Delta}_1\\ \cdots\\ \beta_n\boldsymbol{\Delta}_n\\ \bar{\boldsymbol{C}}\end{bmatrix}. \tag{16}$$

Therefore, the resulting solution for the deformed trajectory is found as

$$\boldsymbol{x}_d=\underbrace{\begin{bmatrix}\beta_1\boldsymbol{L}_1\\ \cdots\\ \beta_n\boldsymbol{L}_n\\ \bar{\boldsymbol{P}}\end{bmatrix}^+\begin{bmatrix}\beta_1\boldsymbol{\Delta}_1\\ \cdots\\ \beta_n\boldsymbol{\Delta}_n\\ \bar{\boldsymbol{C}}_1\end{bmatrix}}_{\mathcal{A}}+\underbrace{\begin{bmatrix}\beta_1\boldsymbol{L}_1\\ \cdots\\ \beta_n\boldsymbol{L}_n\\ \bar{\boldsymbol{P}}\end{bmatrix}^+\begin{bmatrix}\boldsymbol{0}\\ \cdots\\ \boldsymbol{0}\\ \bar{\boldsymbol{C}}_2\end{bmatrix}}_{\mathcal{B}}, \tag{17}$$
$$\text{s.t. } \bar{\boldsymbol{C}}_1+\bar{\boldsymbol{C}}_2=\bar{\boldsymbol{C}}$$

where element $\mathcal{A}$ is a translation of the original trajectory and $\mathcal{B}$ is a spline that minimizes corresponding derivatives. The equivalent for $\mathcal{B}$ in the continuous domain is a spline interpolation of the form

$$\begin{aligned}\boldsymbol{x}_s(t)&=\begin{bmatrix}\beta_1 & \dots & \beta_n\end{bmatrix}\begin{bmatrix}\sum\limits_{k=0}^{1}a_{1k}t^k\\ \cdots\\ \sum\limits_{k=0}^{2n-1}a_{nk}t^k\end{bmatrix} \qquad (18)\\ &=\sum_{k=0}^{2n-1}\left(\sum_{i=1+\lfloor\frac{k}{2}\rfloor}^{n}\beta_i a_{ik}\right)t^k.\end{aligned}$$

This spline interpolation consists of a sum of weighted polynomials of degrees $k\in[0,\dots,2n-1]$. That is by definition also a $(2n-1)$-st degree polynomial.

IV. Feasibility of Motion Execution

Once the deformed motion is derived, a safety procedure is required to check constraint satisfaction for the position, velocity, acceleration and joint torques for every single point of the deformed trajectory. This is done by performing a feasibility check not along the entire trajectory but only at segmentation points of the deformed trajectory. For that, the overall motion is split into smaller time intervals. Therefore, this section is dedicated to the derivation of the closed-form solutions for constraints $\boldsymbol{g}$ from (1e). This feasibility check allows detecting constraint violation which leads to further deformation with new conditions. We show how the time grid is obtained for feasibility check. On every segment of the superimposed time grid we derive 3 types of closed-form expressions for major motion components: analytic, infinity norm and mixed. These expressions are varying in complexity and tightness of the bounds.

A. Notations for Evaluation of Bounds

Here we explain the notations we will use for interval-based [36] bounds. We denote scalar interval as $[x]\equiv[\underline{x},\overline{x}]$, with the lower and upper bounds $\underline{x},\overline{x}\in\mathbb{R}$. The bounds of scalar operations are found as follows

$$\begin{aligned}&[a]+[b]=\left[\underline{a}+\underline{b},\,\overline{a}+\overline{b}\right],\\ &[a]-[b]=\left[\underline{a}-\overline{b},\,\overline{a}-\underline{b}\right],\\ &[a]\,[b]=\left[\min(s),\,\max(s)\right],s=\left\{\overline{a}\,\overline{b},\underline{a}\,\underline{b},\overline{a}\,\underline{b},\underline{a}\,\overline{b}\right\},\\ &\frac{[a]}{[b]}=[a]\,\frac{1}{[b]}=[\underline{a},\overline{a}]\left[\frac{1}{\overline{b}},\frac{1}{\underline{b}}\right],\ 0\notin[b].\end{aligned}$$

An *interval vector* is denoted as $[\boldsymbol{b}]=\left\{\boldsymbol{b}\mid\underline{\boldsymbol{b}}\le\boldsymbol{b}\le\overline{\boldsymbol{b}}\right\}$ for component-wise inequality with the bounds $\underline{\boldsymbol{b}},\overline{\boldsymbol{b}}\in\mathbb{R}^m$. An alternative representation of $[\boldsymbol{b}]$ is obtained with the center vector $\boldsymbol{b}_c=\frac{1}{2}(\underline{\boldsymbol{b}}+\overline{\boldsymbol{b}})$ and the non-negative radius vector $\boldsymbol{b}_\epsilon=\frac{1}{2}(\underline{\boldsymbol{b}}-\overline{\boldsymbol{b}})$ as follows $[\boldsymbol{b}]=\{\boldsymbol{b}\mid\boldsymbol{b}_c-\boldsymbol{b}_\epsilon\le\boldsymbol{b}\le\boldsymbol{b}_c+\boldsymbol{b}_\epsilon\}$.

An *interval matrix* is set as

$$[\boldsymbol{A}]=\left[\underline{\boldsymbol{A}},\overline{\boldsymbol{A}}\right]=\left\{\boldsymbol{A}\mid\underline{\boldsymbol{A}}\le\boldsymbol{A}\le\overline{\boldsymbol{A}}\right\}$$

with bounds $\underline{\boldsymbol{A}},\overline{\boldsymbol{A}}\in\mathbb{R}^{m\times m}$. It can be rewritten using the center matrix $\boldsymbol{A}_c=\frac{1}{2}(\underline{\boldsymbol{A}}+\overline{\boldsymbol{A}})$ and the radius matrix $\boldsymbol{A}_\epsilon=\frac{1}{2}(\underline{\boldsymbol{A}}-\overline{\boldsymbol{A}})$ as $[\boldsymbol{A}]=\{\boldsymbol{A}\mid\boldsymbol{A}_c-\boldsymbol{A}_\epsilon\le\boldsymbol{A}\le\boldsymbol{A}_c+\boldsymbol{A}_\epsilon\}$. Spectral radius of a matrix is $\rho(\boldsymbol{A})=\max\limits_{i=1,\dots,m}(|\lambda_i|)$, where λ_i are the eigenvalues of $\boldsymbol{A}$.

For a matrix-vector product between $\boldsymbol{A}\in\mathbb{R}^{m\times m}$ and $\boldsymbol{b}\in\mathbb{R}^m$ the following holds

$$[\boldsymbol{A}]\,[\boldsymbol{b}]=\begin{bmatrix}\sum_j[A(1,j)]\,[b(j)]\\ \cdots\\ \sum_j[A(m,j)]\,[b(j)]\end{bmatrix}.$$

An *interval function* is set as $[\boldsymbol{f}]([\boldsymbol{x}])=\{\boldsymbol{f}(\boldsymbol{x})\mid\boldsymbol{x}\in[\boldsymbol{x}]\}$. The infinity norm ($\infty$-norm) $\|\boldsymbol{f}(\boldsymbol{x})\|_\infty$ is derived as

$$\|\boldsymbol{f}(\boldsymbol{x})\|_\infty=\max_{\boldsymbol{x}}|\,\boldsymbol{f}(\boldsymbol{x})\,|\,.$$

With the help of these expressions we derive closed form bounds for components of the robot motion.

B. Time Grid for Feasibility Check

Boundaries for all main kinematic and dynamic variables need to be provided for every time segment of the motion. The size of segments influences not only the kinematic but also the dynamic properties of the task. Primarily, it affects bounds for the torque required to follow the deformed trajectory.

As the original trajectories are C^2 smooth, there exist boundaries for $\boldsymbol{x}_o$, $\dot{\boldsymbol{x}}_o$ and $\ddot{\boldsymbol{x}}_o$

$$\begin{aligned}[\boldsymbol{x}_o]([\boldsymbol{t}]) &= \{\boldsymbol{x}_o(\boldsymbol{t}) \mid \boldsymbol{t} \in [\boldsymbol{t}]\},\\ [\dot{\boldsymbol{x}}_o]([\boldsymbol{t}]) &= \{\dot{\boldsymbol{x}}_o(\boldsymbol{t}) \mid \boldsymbol{t} \in [\boldsymbol{t}]\},\\ [\ddot{\boldsymbol{x}}_o]([\boldsymbol{t}]) &= \{\ddot{\boldsymbol{x}}_o(\boldsymbol{t}) \mid \boldsymbol{t} \in [\boldsymbol{t}]\},\end{aligned} \tag{19}$$

on the predefined original temporary grid

$$[\boldsymbol{t}] = \left\{\boldsymbol{t} \mid t_1 \in [t_1^o, t_2^o], \ldots, t_{f-1} \in \left[t_{f-1}^o, t_f^o\right]\right\}. \tag{20}$$

In general case, the time grid of original motion (20) may not match with the deformation grid from (2). After deformation with $(n_d - 1)$ segments, the superimposed time grid in Fig. 4 allows putting together the closed-form bounds of the original and the deformation part from (13) and its derivatives.

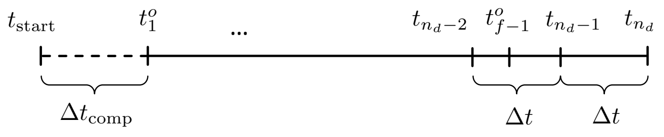


Figure 4: The superimposed grid consists of the fixed original time grid from (20) and the deformation time grid from (2). The time grid for spline deformation is aligned with the original trajectory at the time of fixed boundary state $t_{n_d} = t_f^o = t_f$. Interval Δt_{comp} is a time remaining for computation, t_{start} is a time instance when a request for motion is received.

Checking the required velocities, accelerations, controller errors and torques only at the segmentation points reduces the overall computations, allows finding particular intervals where the trajectory is infeasible and provides a flexible tool for controller parametrization.

The remaining time Δt_{comp} is defined by the task and calculated as the difference between overall time available for the motion $(t_{n_d} - t_{\text{start}})$ and time duration of the original motion $(t_f^o - t_1^0)$. I.e. for faster original motions we have higher bounds but more time for deformation and feasibility checks. Thus, it is useful to have several precalculated trajectories of various time duration t_f^o in the database.

C. Analytic Expressions for Feasibility Check

Analytic expressions allow finding tight representation for trajectory, its derivatives and corresponding torques on every segment of the superimposed grid. We consider a task executed by an open-chain manipulator with the following dynamic model

$$\begin{aligned}\boldsymbol{\tau} &= \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}),\\ \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{F}(\boldsymbol{q}, \dot{\boldsymbol{q}}),\end{aligned} \tag{21}$$

where the term $\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}$ denotes the configuration-dependent mass-inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ comprises the Coriolis and centrifugal forces, $\boldsymbol{G}(\boldsymbol{q})$ accounts for all gravitational terms and $\boldsymbol{F}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ includes the Coulomb and viscous frictional forces.

We use an internal computed-torque scheme together with an outer loop PD controller (CT + PD) to provide the desired control characteristics. For CT + PD controller, the following relationship holds for closed-loop error dynamics:

$$\begin{aligned}\ddot{\boldsymbol{e}} + \boldsymbol{K}_v\dot{\boldsymbol{e}} + \boldsymbol{K}_p\boldsymbol{e} &= \boldsymbol{M}^{-1}\boldsymbol{\tau}_{\text{dist}},\\ \boldsymbol{u} &= -\boldsymbol{K}_p\boldsymbol{e} - \boldsymbol{K}_v\dot{\boldsymbol{e}}\end{aligned}$$

where $\boldsymbol{u}$ is a control input function, the $\boldsymbol{K}_p = \operatorname{diag}(k_{pi})$ and $\boldsymbol{K}_v = \operatorname{diag}(2\sqrt{k_{pi}})$ are critically-damped PD control gains of the outer loop.

If the lower and upper bounds of the main system and control variables satisfy the inequality constraints $\boldsymbol{g}$ from (1e), the motion is ready to be executed.

1) Trajectory Feasibility: The superimposed time grid allows calculating upper bounds on the position, velocity and acceleration of the deformed trajectory in a closed form for each time interval independently. The spline-based deformation presented in Sec. III makes it possible to provide a closed form solution for the task-space bounds of $\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d$, $\ddot{\boldsymbol{x}}_d$ considering only the segmentation points. The deformed trajectory is split up into two independent parts $\mathcal{A}$ and $\mathcal{B}$ as in (13) or (17). Since the element $\mathcal{A}$ is solely a translation of the original trajectory $\boldsymbol{x}_o$. The additive deformation term $\mathcal{B}$ is a spline interpolation (14) or (18) that can be represented using Bézier splines for every interval as

$$\boldsymbol{x}_s = \sum_{k=0}^{l} \boldsymbol{c}_k \boldsymbol{B}_k^l(t), \tag{22}$$

where the weighting factors $\boldsymbol{c}_k$ denote control points of Bézier curves and $\boldsymbol{B}_k^l(t)$ represent Bernstein polynomials of degree $l = 2n - 1$ with

$$\begin{aligned}\boldsymbol{B}_k^l(t) &= \binom{l}{k}(1-t)^{l-k}\,t^k,\\ t &\in [0,1], \quad k = 0, \ldots, l.\end{aligned} \tag{23}$$

Further, we use partition of unity and non-negativity properties of a Bernstein polynomial together with the special expression of its derivative. Therefore, every point $\boldsymbol{p}(t)$ stays within the convex hull of its control points. The same holds for the derivatives of a $\boldsymbol{p}(t)$, since they are also Bernstein polynomials. These properties, as well as bounds for the trajectory and its corresponding derivatives, are provided in Appendix B.

Tight trajectory bounds are found using the interval analysis. Without loss of generality, we provide derivations for upper bounds, whereas lower bounds can be found analogously. We provide the estimations of $\boldsymbol{x}_d$, $\dot{\boldsymbol{x}}_d$, $\ddot{\boldsymbol{x}}_d$ that are bounded by analytic expression

$$\begin{aligned}[\boldsymbol{x}_d] &= [\boldsymbol{x}_o] + [\boldsymbol{x}_s],\\ [\dot{\boldsymbol{x}}_d] &= [\dot{\boldsymbol{x}}_o] + [\dot{\boldsymbol{x}}_s],\\ [\ddot{\boldsymbol{x}}_d] &= [\ddot{\boldsymbol{x}}_o] + [\ddot{\boldsymbol{x}}_s],\end{aligned} \tag{24}$$

where the original trajectory bounds $[\boldsymbol{x}_o]$, $[\dot{\boldsymbol{x}}_o]$, $[\ddot{\boldsymbol{x}}_o]$ are derived in (19), and the deformation parts are found online for the j-th trajectory interval as

$$\begin{aligned}
[\boldsymbol{x}_s]([t_j,t_{j+1}]) &= \left[l\min_{k=0,\dots,l}(c_{j,k}),\ l\max_{k=0,\dots,l}(c_{j,k})\right],\\
[\dot{\boldsymbol{x}}_s]([t_j,t_{j+1}]) &= \left[l\min_{k=0,\dots,l-1}(c_{j,k+1}-c_{j,k}),\ l\max_{k=0,\dots,l-1}(c_{j,k+1}-c_{j,k})\right],\\
[\ddot{\boldsymbol{x}}_s]([t_j,t_{j+1}]) &= \Big[l(l-1)\min_{k=0,\dots,l-2}(c_{j,k+2}-2c_{j,k+1}+c_{j,k}),\\
&\qquad l(l-1)\max_{k=0,\dots,l-2}(c_{j,k+2}-2c_{j,k+1}+c_{j,k})\Big],
\end{aligned} \tag{25}$$

based on the control points of the Bernstein polynomials.

Limiting values for deformed joint positions $[\boldsymbol{q}_d]$ are obtained through inverse kinematics mapping of $[\boldsymbol{x}_d]$. Furthermore, $[\dot{\boldsymbol{q}}_d]$ and $[\ddot{\boldsymbol{q}}_d]$ are obtained as follows:

$$\begin{aligned}
[\dot{\boldsymbol{q}}_d] &= \left[\boldsymbol{J}^+\right][\dot{\boldsymbol{x}}_d]\\
[\ddot{\boldsymbol{q}}_d] &= \left[\boldsymbol{J}^+\right][\ddot{\boldsymbol{x}}_d] + \left[\dot{\boldsymbol{J}}\right][\dot{\boldsymbol{q}}_d]
\end{aligned} \tag{26}$$

where $[\boldsymbol{J}^+]$ and $\left[\dot{\boldsymbol{J}}\right]$ are dependent on $[\boldsymbol{q}_d]$.

2) Bounds on Tracking Error and Imposed Disturbance: Bounds on tracking error are crucial when we want to assess the motion feasibility under model parametric uncertainty or under influence of external forces. In general, external forces are mapped to the joint torques as $\boldsymbol{\tau}_{\text{ext}} = \boldsymbol{J}^T\boldsymbol{F}_{ext}$. Bounds on $\boldsymbol{\tau}_{\text{ext}}$ could be found using analytic interval analysis.

$$\begin{aligned}
[\boldsymbol{\tau}_{\text{ext}}] &= \left\{\boldsymbol{\tau}_{\text{ext}} \mid \underline{\boldsymbol{\tau}_{\text{ext}}} \le \boldsymbol{\tau}_{\text{ext}} \le \overline{\boldsymbol{\tau}_{\text{ext}}}\right\} = \left[\boldsymbol{J}^T\right][\boldsymbol{F}_{\text{ext}}],\\
[\boldsymbol{\tau}_{\text{ext}}] &= \left\{\boldsymbol{\tau}_{\text{ext}} \mid \boldsymbol{\tau}_{\text{ext,c}} - \boldsymbol{\tau}_{\text{ext},\epsilon} \le \boldsymbol{\tau}_{\text{ext}} \le \boldsymbol{\tau}_{\text{ext,c}} + \boldsymbol{\tau}_{\text{ext},\epsilon}\right\},\\
\boldsymbol{\tau}_{\text{ext,c}} &= \frac{\overline{\boldsymbol{\tau}_{\text{ext}}} + \underline{\boldsymbol{\tau}_{\text{ext}}}}{2},\\
\boldsymbol{\tau}_{\text{ext},\epsilon} &= \frac{\overline{\boldsymbol{\tau}_{\text{ext}}} - \underline{\boldsymbol{\tau}_{\text{ext}}}}{2},
\end{aligned} \tag{27}$$

where $\left[\boldsymbol{J}^T\right]$ depends on $[\boldsymbol{q}_d]$ and $[\boldsymbol{F}_{\text{ext}}]$ represents a range of expected external forces for the contact interval, $\boldsymbol{\tau}_{\text{ext,c}}$ is the average (center) external torque that acts as a modeled external torque, and $\boldsymbol{\tau}_{\text{ext},\epsilon}$ is the maximum deviation (radius) from the average torque that is applied as the disturbance torque $[\boldsymbol{\tau}_{\text{ext,dist}}] = \{\boldsymbol{\tau}_{\text{ext,dist}} \mid -\boldsymbol{\tau}_{\text{ext},\epsilon} \le \boldsymbol{\tau}_{\text{ext,dist}} \le \boldsymbol{\tau}_{\text{ext},\epsilon}\}$.

Similarly, if the robot system matrices or their elements are uncertain but bounded, they can be mapped to the disturbance torque. For example, lets assume modeled parametric uncertainty $\Delta\boldsymbol{M}$ in mass matrix resulting in

$$[\boldsymbol{\tau}] + [\boldsymbol{\tau}_{\text{par,dist}}] = ([\boldsymbol{M}] + [\Delta\boldsymbol{M}])[\ddot{\boldsymbol{q}}] + [\boldsymbol{N}], \tag{28}$$

where $[\boldsymbol{\tau}_{\text{par,dist}}] = [\Delta\boldsymbol{M}][\ddot{\boldsymbol{q}}]$.

Disturbance parts of external torques from external forces and parametric uncertainty are combined as

$$[\boldsymbol{\tau}_{\text{dist}}] = [\boldsymbol{\tau}_{\text{par,dist}}] + [\boldsymbol{\tau}_{\text{ext,dist}}]. \tag{29}$$

Calculations for the control input function $\boldsymbol{u}$ require the calculation of boundaries for the inverse mass matrix $\boldsymbol{M}^{-1}$ and for the errors

$$\begin{aligned}
[\boldsymbol{e}] &= \left[\operatorname{diag}(\frac{1}{4k_{pi}})\right]\left[\boldsymbol{M}^{-1}\right][\boldsymbol{\tau}_{\text{dist}}],\\
[\dot{\boldsymbol{e}}] &= \left[\operatorname{diag}(\frac{1}{\sqrt{k_{pi}}e})\right]\left[\boldsymbol{M}^{-1}\right][\boldsymbol{\tau}_{\text{dist}}],\\
[\boldsymbol{M}][\boldsymbol{u}] &= -[\boldsymbol{M}][\boldsymbol{K}_p][\boldsymbol{e}] - [\boldsymbol{M}][\boldsymbol{K}_v][\dot{\boldsymbol{e}}],
\end{aligned} \tag{30}$$

where constant scalar e is the base of the natural logarithm, and the analytic expression for the inverse matrix

$$\left[\boldsymbol{M}^{-1}\right] = \left[\min\left(\frac{\operatorname{adj}(\boldsymbol{M})}{\det(\boldsymbol{M})}\right), \max\left(\frac{\operatorname{adj}(\boldsymbol{M})}{\det(\boldsymbol{M})}\right)\right], \tag{31}$$

where $\min$ and $\max$ operations are applied element-wise. This interval calculation requires the decomposition of $\operatorname{adj}(\boldsymbol{M})$ and $\det(\boldsymbol{M})$ into single scalar operations, which might become computationally demanding for multi-DoF robots.

3) Torque Feasibility: Correct execution of the motion highly depends on torque constraints. The goal is to find out whether the produced torques to follow the deformed trajectory stay within the constant interval $[-\tau_{\text{nom}}, \tau_{\text{nom}}]$ of minimum and maximum actuator torques at every time step. This is done by deriving lower and upper bounds for each factor involved in the torque calculation. A segment-wise torque feasibility check gives us the opportunity to judge whether the produced deformed trajectory leads to torque saturation or not, depending on the parameters of the feedback controller. The method produces upper bounds for the joint torques in a closed form that are easily verified. These bounds are based on linear error system properties of computed torque with PD control scheme and special properties of the Bernstein polynomials that allow taking into account only the segmentation points of the trajectory and its corresponding control points.

Tight bounds for the torque $[\boldsymbol{\tau}_{\text{bound}}]$ to follow the deformed trajectory can be obtained as follows

$$[\boldsymbol{\tau}_{\text{bound}}] = [\boldsymbol{M}][\ddot{\boldsymbol{q}}_d] - [\boldsymbol{u}] + [\boldsymbol{N}] + \boldsymbol{\tau}_{\text{ext,c}}, \tag{32}$$

by substituting the minimum and maximum interval values of the components from (26), (27) and (30) found on the closed intervals. The boundary values for $[\boldsymbol{M}]([\boldsymbol{q}])$ and $[\boldsymbol{N}]([\boldsymbol{q}],[\dot{\boldsymbol{q}}])$ have simple analytic expressions obtained by substitution, where the boundary values for $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are found as

$$\begin{aligned}
[\boldsymbol{q}] &= [\boldsymbol{q}_d] + [\boldsymbol{e}],\\
[\dot{\boldsymbol{q}}] &= [\dot{\boldsymbol{q}}_d] + [\dot{\boldsymbol{e}}].
\end{aligned} \tag{33}$$

The bounds found with analytic expressions based on interval analysis are tight but computationally not so efficient, mainly because this method requires finding $\min$ and $\max$ values for every operation between single elements of (32). This effect is particularly noticeable for the estimation of an analytic expression for the inverse mass matrix required to find tracking error and its derivative.

D. Infinity Norm Expressions for Feasibility Check

Torque bounds provided by the ∞-norm for the tracking of a deformed trajectory are easy to compute but offer rather conservative estimates. In [35] we derived bounds for ∞ norm for cubic spline deformation. These estimates could be generalized for the preservation of the n-th desired derivative and for the case of disturbance due to contacts and parametric uncertainty. Hence, the number of control points changes with the maximum derivative to be preserved.

For the 2-norm of external torque, the following relation will hold in case of the contact

$$\left\|\boldsymbol{\tau}_{\text{ext,dist}}\right\|_2 \le \left\|\boldsymbol{\tau}_{\text{ext,dist}}\right\|_\infty \le \left\|\boldsymbol{J}^T\right\|_\infty \left\|\boldsymbol{F}_{\text{ext}}\right\|_\infty, \tag{34}$$

and disturbance torque due to parametric uncertainty for $\boldsymbol{M}$

$$\begin{aligned}\|\boldsymbol{\tau}\|_\infty + \|\boldsymbol{\tau}_{\text{par,dist}}\|_\infty &= \|\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{N}(\boldsymbol{q},\dot{\boldsymbol{q}})\|_\infty + \|\Delta\boldsymbol{M}\ddot{\boldsymbol{q}}\|_\infty \\ \|\boldsymbol{\tau}_{\text{par,dist}}\|_2 \le \sqrt{m}\,\|\boldsymbol{\tau}_{\text{par,dist}}\|_\infty &\le \sqrt{m}\,\|\Delta\boldsymbol{M}\ddot{\boldsymbol{q}}\|_\infty\,. \end{aligned} \tag{35}$$

where m is the length of the vector $\boldsymbol{q}$. Therefore, bounds for ∞-norm of tracking error functions e_i and $\dot{e}_i$ are obtained using the transfer functions of disturbance to error and to the derivative of error in Laplace domain as follows

$$\begin{aligned}\|e_i(t)\|_\infty &\le \|G_i(s)\|_2 \left\|\boldsymbol{M}^{-1}\boldsymbol{\tau}_{\text{dist}}\right\|_2 \\ &\le \frac{1}{2k_{pi}^{-\frac{3}{4}}} \left\|\boldsymbol{M}^{-1}\right\|_2 \|\boldsymbol{\tau}_{\text{dist}}\|_2 \\ \|\dot{e}_i(t)\|_\infty &\le \|sG_i(s)\|_2 \left\|\boldsymbol{M}^{-1}\boldsymbol{\tau}_{\text{dist}}\right\|_2 \\ &\le \frac{1}{2k_{pi}^{-\frac{1}{4}}} \left\|\boldsymbol{M}^{-1}\right\|_2 \|\boldsymbol{\tau}_{\text{dist}}\|_2 \end{aligned} \tag{36}$$

with 2-norm of inverse mass matrix $\left\|\boldsymbol{M}^{-1}\right\|_2$ bounded by $\frac{1}{\lambda_{M,\min}}$, where $\lambda_{M,\min}$ is a minimum eigenvalue of the mass matrix and $\|\boldsymbol{\tau}_{\text{dist}}\|_2 \le \|\boldsymbol{\tau}_{\text{ext,dist}}\|_2 + \|\boldsymbol{\tau}_{\text{par,dist}}\|_2$.

Thus, the following torque bound is derived

$$\begin{aligned}\tau_{\text{bound},\infty} =&\bar{m}\,\|\ddot{\boldsymbol{q}}_d\|_\infty + \bar{m}(\overline{\boldsymbol{K}_v}\,\|\dot{\boldsymbol{e}}\|_\infty + \overline{\boldsymbol{K}_p}\,\|\boldsymbol{e}\|_\infty) + \\ &\bar{c}m(\|\dot{\boldsymbol{q}}_d\|_\infty + \|\dot{\boldsymbol{e}}\|_\infty)^2 + \bar{g} + \|\boldsymbol{\tau}_{\text{ext,dist}}\|_\infty\,,\end{aligned} \tag{37}$$

where $\bar{m}, \bar{c}, \bar{g}$ are ∞-norm bounds of system matrices that are derived using the expressions from Appendix C. Torque norms become tighter when calculated independently for each joint as $\tau_{i,\text{bound}}$.

E. Mixed Expressions for Feasibility Check

Mixed expressions help to find an appropriate trade-off between computational time and tightness of the bounds. This becomes particularly important for robotic tasks with limited reaction times and for the control methods where the analytic expressions for some interval elements, e.g. $\left[\boldsymbol{M}^{-1}\right] = \left\{\boldsymbol{M}^{-1} \mid \underline{\boldsymbol{M}^{-1}} \le \boldsymbol{M}^{-1} \le \overline{\boldsymbol{M}^{-1}}\right\}$ is difficult to obtain. Thus, one can substitute these expressions with analytic equivalent of the ∞ norms as

$$\begin{aligned}\left\|\boldsymbol{M}^{-1}\right\|_\infty &\le \sqrt{m_M}\left\|\boldsymbol{M}^{-1}\right\|_2 \le \frac{\sqrt{m_M}}{\lambda_{M,\min}}, \\ \overline{\boldsymbol{M}^{-1}} &\le \left\|\boldsymbol{M}^{-1}\right\|_\infty \boldsymbol{J}_{m_M} \le \frac{\sqrt{m_M}}{\lambda_{M,\min}}\boldsymbol{J}_{m_M}, \\ \underline{\boldsymbol{M}^{-1}} &\ge -\left\|\boldsymbol{M}^{-1}\right\|_\infty \boldsymbol{J}_{m_M} \ge -\frac{\sqrt{m_M}}{\lambda_{M,\min}}\boldsymbol{J}_{m_M},\end{aligned} \tag{38}$$

where m_M is a dimensionality of the matrix $\boldsymbol{M}$ and $\boldsymbol{J}_{m_M}$ is a square matrix of ones. Interval bounds for eigenvalues are derived using center matrices and spectral radius [37], [38]. We derive interval bounds for eigenvalues of symmetrical positive-definite mass-inertia matrix $\boldsymbol{M}$ as follows

$$\begin{aligned}[\lambda_M] &= [\lambda_{M,\min}, \lambda_{M,\max}] \\ &= [\lambda_{\min}(M_c) - \rho(M_\epsilon), \lambda_{\max}(M_c) + \rho(M_\epsilon)]\,.\end{aligned} \tag{39}$$

With this substitution, we speed up calculations and get rid of the necessity of calculating the large symbolic expression for inverse matrix. However, torque estimates become larger than for the pure analytic method. Replacing other elements of analytic expression with ∞ norms may lead to further reduction of computation time.

V. EXPERIMENTAL VERIFICATION: PLANAR VOLLEYBALL

The spline deformation can be applied to motion planning for a large variety of robotic tasks, especially where the time of computation, feasibility and ability to set new goals online are the key issues. In this section, we present the scenario to illustrate the spline deformation method and its main features. The objective is to play planar volleyball, that requires hitting the ball that must travel along the prescribed parabola. For such a scenario, hitting position, orientation, as well as linear and angular velocities, must be set for every hit. Thus, it means that 2-nd order boundary conditions must be set, i.e. $n \ge 2$ and at least cubic spline deformation is required to keep desired boundaries and to preserve desired motion profile $\boldsymbol{\Delta}$. Moreover, the robotic volleyball is a very fast robotic task that illustrates very well the importance of online deformations, feasibility checks and precise timing.

A. Task Description

We consider a problem of playing volleyball on the air-table as shown in Fig. 5, with a planar $\mathbf{3}$-DoF robot arm ($\mathbf{3}$R) against a human or another robot.

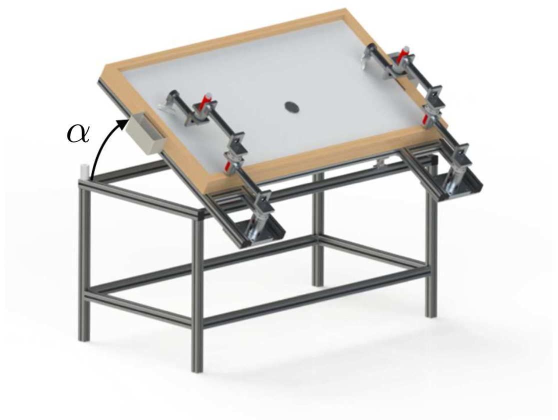


Figure 5: Air-table tilted by an angle α with attached robot manipulators as a testbed. The gravitational acceleration on the surface of the air-table is $g_m = g\sin\alpha$.

In this experiment, we apply spline deformation method with cubic polynomial interpolation that keeps minimum acceleration deviation to original trajectory and fixes boundary conditions for coordinates $\boldsymbol{x_h}$ and its derivatives $\dot{\boldsymbol{x}}_{\boldsymbol{h}}$. By minimizing the deviation of the acceleration we can preserve the curvature of the precomputed trajectory and also indirectly minimize torque.

For our volleyball task, a tracking system provides us with position, orientation and velocities of the ball. Using several subsequent frames, a parabolic trajectory of the object together with its angular velocity is predicted through filtering of the visual data and using puck dynamic equations. The hitting time t_h and state $\boldsymbol{x}_h = \begin{bmatrix} x_h & y_h & \phi_h \end{bmatrix}^T, \dot{\boldsymbol{x}}_{\boldsymbol{h}} = \begin{bmatrix} v_{xh} & v_{yh} & \omega_h \end{bmatrix}^T$ is found using an optimization procedure based on the chosen strategy, see Sec. V-B.

For the robot, the task consists of hitting an incoming volleyball such that the ball flies over the net (in orange) and lands in the opponent's field (in green). When modeling the impact of the ball, one has to find the full state vector of the end effector for the desired ball motion. The state is described by the hit position (x_h, y_h) on the table, the fixed end effector angle ϕ_h, and its translational (v_{xh}, v_{yh}) and angular (ω_h) velocities.

The overall scenario is illustrated in Fig. 6. We presented the ball - end effector collision model based on momentum conservation in [34]. The chosen point for the rebound is

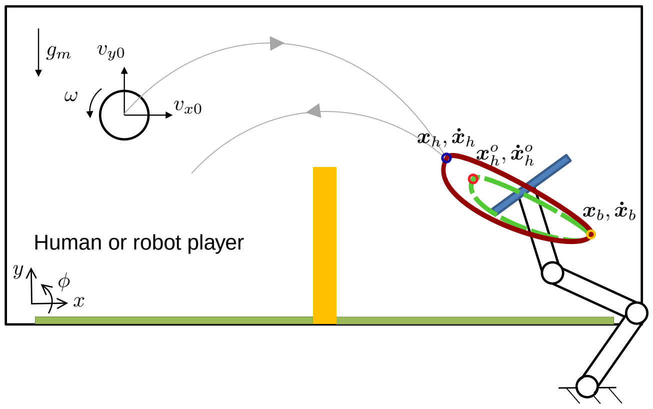

g_m
v_{y0}
ω
v_{x0}
$\boldsymbol{x}_h, \dot{\boldsymbol{x}}_h$
$\boldsymbol{x}_h^o, \dot{\boldsymbol{x}}_h^o$
$\boldsymbol{x}_b, \dot{\boldsymbol{x}}_b$
Human or robot player
y
ϕ
x

Figure 6: Scenario overview of the planar volleyball task. The robot rebounds the ball over the net (orange separation line) based on a prediction of a ball flight trajectory and desired return motion. Green dashed curve represents the original trajectory, while the dark red solid curve is a deformed trajectory. Yellow circle depicts the resting position of the end effector, the red circle is a hitting state of the original trajectory, and the blue circle is a hitting state of the deformed trajectory.

parametrized solely by the time of rebound t_h. This point is used further for optimization based on the chosen strategy to find the collision full state $\boldsymbol{x}_h$ and $\dot{\boldsymbol{x}}_h$.

In the absence of air friction and disturbances, the ball travels along the parabola after the rebound. Velocities (v'_x, v'_y) and ω' after the collision are assigned based on the desired return trajectory. This return trajectory can be uniquely defined by setting two waypoints along the parabola. Alternatively, a single waypoint with the corresponding velocity provides the same result.

The hitting orientation of the end effector ϕ_h varies to minimize $J = \min_{\phi_h}(\omega' - \omega_d)^2$, where ω_d is a desired angular velocity of the ball. For facilitation of the return stroke, minimization of the angular velocity is applied in the cooperative scenario $\omega_d = 0$. For maximization of the angular velocity in competitive scenario $\omega_d = \omega_{\max}$. Once the suitable orientation is found the corresponding end effector linear velocities v_{xh}, v_{yh} are calculated accordingly.

B. Volleyball Game Strategies

The return motion of the ball depends on the state of the object before the collision and the contact with the robot. The state of the object in free flight solely depends on the time t_h.

First, suitable hitting time and return velocities should be chosen based on a selected strategy. For instance, the cooperative strategy implies enhancement of the operation repeatability. It requires minimization of the linear and angular velocities $\dot{\boldsymbol{x}}_{h,\text{opp}}$ during the hit of the partner robot and maximization of manipulability $\omega_{\max} = \max_{t_h} \sqrt{\boldsymbol{J}\boldsymbol{J}^T}$ for the hitting point of opposite robot. With such a strategy representation, it is possible to manipulate more than one object at a time by assigning different non-intersecting orbits with a proper phase shift for the return ball trajectories. On the contrary, the competitive strategy tries to increase the linear and angular velocities and decrease manipulability of the opponent so that higher torques and accelerations are required. The choice of the hitting state is set as the optimization problem:

$$\underset{v'_x, v'_y, \omega', t_{n_d}}{\text{minimize}} \quad \dot{\boldsymbol{x}}_{h,\text{opp}}^T \boldsymbol{R} \dot{\boldsymbol{x}}_{h,\text{opp}} - \omega_{\max}^T \boldsymbol{K} \omega_{\max}, \tag{40}$$

where positive weighting matrices $\boldsymbol{R}$ and $\boldsymbol{K}$ are set for cooperative scenario. Similarly, negative weighting matrices $\boldsymbol{R}$ and $\boldsymbol{K}$ are applied for competitive scenarios. Matrix $\boldsymbol{K} = 0$ is used when the robot plays against a human opponent. In competitive mode, return parabolas with higher variation range are used against human participants.

Second, it is advantageous to use multiple precomputed trajectories and select the one that needs to be deformed least for hitting the ball. The direct collocation optimal control method DirCol [1] is used to obtain a set of original trajectories. For our experiment with strategies, we precomputed $n_p = 5$ original piece-wise spline trajectories depicted in Fig. 7.

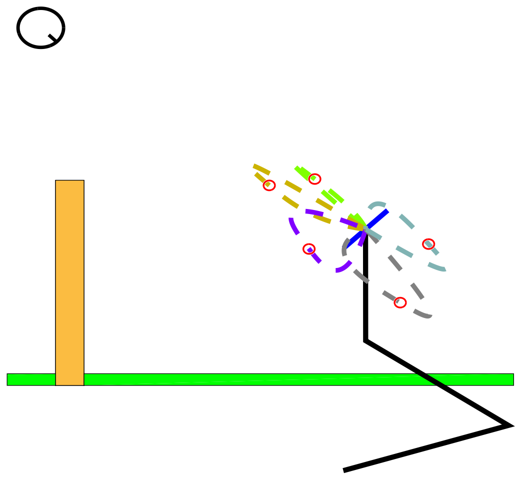


Figure 7: Original precomputed trajectories (dashed curves) with designed derivative profile, variable hitting states and time duration. Red circles represent hitting position for precomputed trajectories.

After the hitting state is chosen based on (40), one can select the best reference trajectory i^* and deform it as shown in Fig. 8. This selection allows choosing motions with desired derivative profile and proper time duration.

$$\begin{aligned} i^* = \underset{i \in [1..n_p]}{\text{argmin}} \quad & k_{c1} \|\dot{\boldsymbol{p}}_{pr} - \dot{\boldsymbol{p}}_i^o\|_2^2 + k_{c2} \|\boldsymbol{p}_{pr} - \boldsymbol{p}_i^o\|_2^2 + \\ & k_{c3}(t_{n_d} - t_{f,i}^o)^2, \end{aligned} \tag{41}$$

where n_p is a number of precomputed trajectories and k_{ci} are the weighting factors for each condition. It can be done using (41) by choosing a trajectory which has similar boundary conditions and motion time.

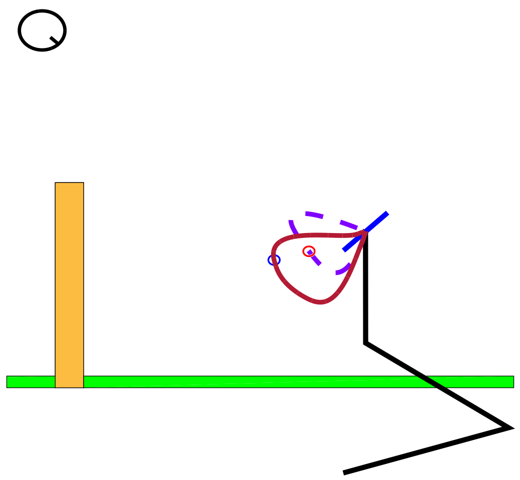


Figure 8: Deformed trajectory (solid line) is derived in order to fit new hitting positions (blue circle), orientation and velocities. The time to reach the hit point and the time back to the resting position are fixed both for original and deformed trajectories.

C. Experimental Evaluation

The experimental verification of the proposed spline deformation method has been conducted using an experimental setup consisting of two $\mathbf{3}$-DoF planar robot arms connected to the edges of the tilted air-table, see Fig. 5. The air-table is capable of partially compensating the gravitational component of the object that is perpendicular to the surface and thus increasing flight time of the objects. The balls (pucks) and flat end effectors are 3D printed rigid objects. Several markers placed on the puck are tracked during the experiment using a Qualisys motion capture system with a framerate of 250 Hz, whereas the robot control system operates at 1 kHz rate. The workspace is bounded on each side of the table by a board.

Figure 9 shows the trace of the real experimental trajectories of the puck and the end effector.

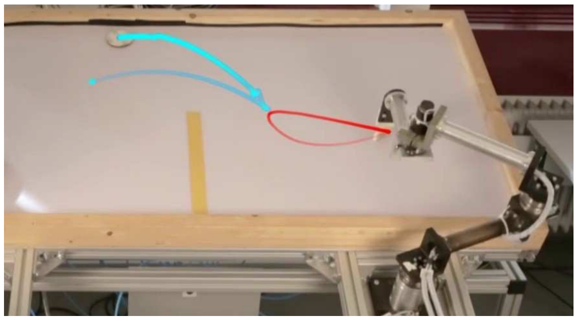


Figure 9: Trace of the central points of the end effector and the puck.

Figure 10 presents a first scenario where a human throws the ball and the robot hits it back over the net.

Another scenario is presented in Fig. 11, where two robots are juggling two balls over the net. These balls are traveling along two non-intersecting orbits: from the left to the right with a lower orbit and from the right to the left with a higher orbit. This experiment shows the reliability of the impact control and its timing and thus the accuracy of reaching fixed states.

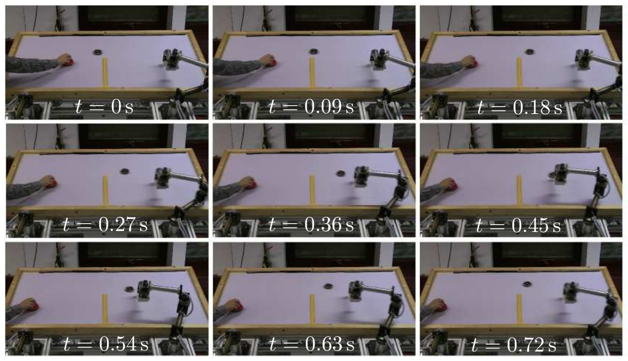


Figure 10: Frames of the planar volleyball experiment with a human playing against the robot. The puck (black circle) is hit by the human and then rebounded by the robot.

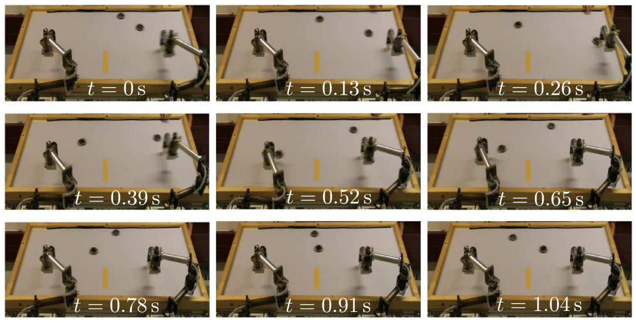


Figure 11: Frames of the experiment with two robots simultaneously juggling two pucks (black circles). First, the right robot rebounds one of the pucks while the second puck approaches the left robot. Second, the left robot rebounds the second puck while the first puck moves away from the right robot towards the left robot. After that, the cycle repeats again.

The spline deformation method provides important features for bounds and allows online computations. The advantage of the method in computational time with respect to the DMP and LTE techniques is shown in Fig. 12.

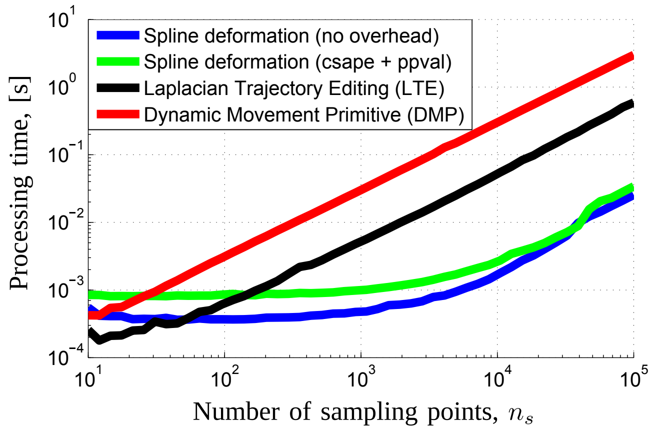


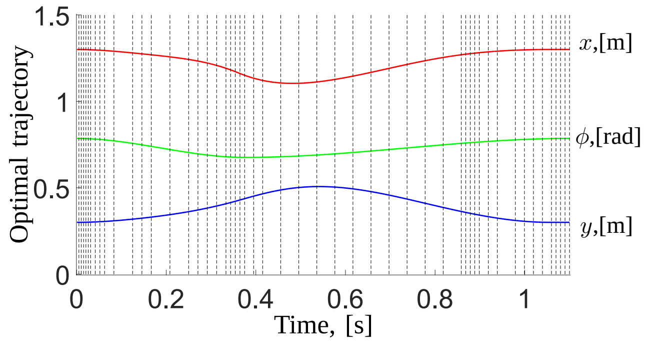
Figure 12: Comparison of the computational complexity of trajectory generation with spline deformation, DMP and LTE methods. All calculations are done with an Intel Core i3 -3120M CPU with 4GB RAM with MATLAB R2015a running under Windows7.

The first two graphs include spline interpolation and evaluation of the piece-wise polynomial with standard Matlab functions (Spline deformation (csape + ppval)) and with self-written optimized code (Spline deformation (no overhead)).

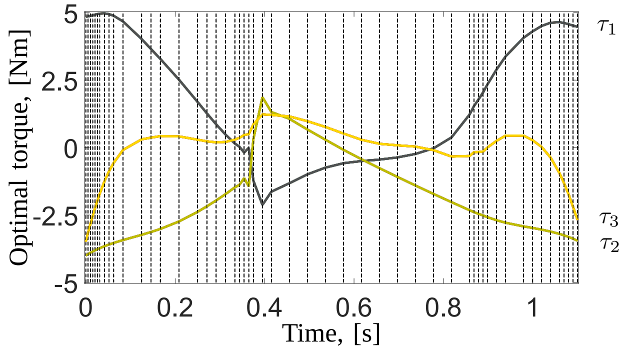
This comparison is done for a single trajectory deformation step for a varied number of sampling points with discretization time $1\,ms$. According to Fig. 12, for medium and large trajectories the spline representation is more than one magnitude faster than the LTE and two magnitudes faster than the DMP. In log-log plot, relationship of the form $y = ax^k$ appears as a straight line with the power and constant term corresponding to slope and intercept of the line. It can be also seen that for $n_s > 10^4$ the effect of computational overhead, due to the built-in MATLAB function for calculating splines (csape, ppval), significantly reduces.

D. Estimation of Torque Bounds

Due to the special properties of the Bernstein polynomials and the computed torque scheme, we derive torque bounds for three different estimation schemes. These bounds are calculated for each interval of the superposition of optimal and deformed grids, with PD gains $\boldsymbol{K}_p = \mathrm{diag}(1369,\,1089,\,841)$, $\boldsymbol{K}_v = \mathrm{diag}(74,\,66,\,58)$. Figure 13 shows the original precomputed trajectory with the corresponding grid. For every interval of this grid, tight bounds are calculated offline. In fact, we can recalculate this original grid with any desired subdivision offline.



(a) The optimal trajectory of the end effector in task space.

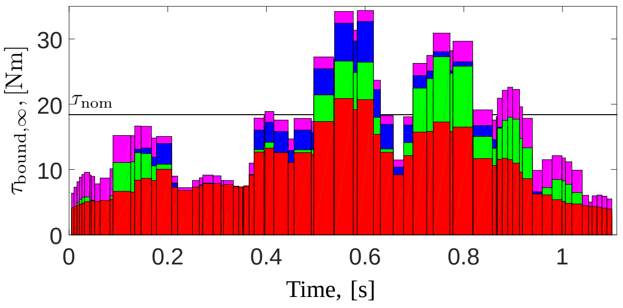


(b) Optimal joint torques.

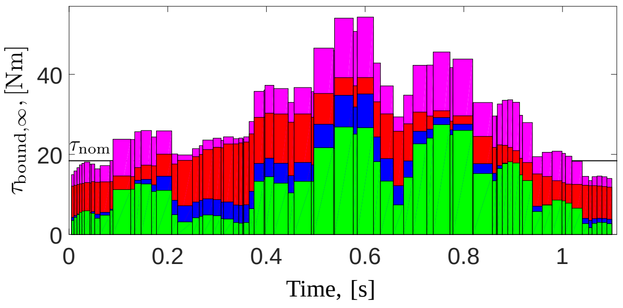
Figure 13: Optimal motion and time grid generated by DirCol. Black vertical dashed lines represent optimal time grid.

Figures 14-16 contain grids from original trajectory superimposed with the equitemporal grid used for deformation. We might use iterative subdivision, e.g. for motion re-deformation, thus it is preferable to keep 2 independent grids. An exemplary 2-phase optimal motion presented in Fig. 13a and Fig. 13b is computed offline with the cost function $J = \int_0^{t_f} \sum_i \tau_{o,i}^2 \mathrm{d}t$, equality constraints for initial and final points of the motion $\boldsymbol{x}_o(0) = \boldsymbol{x}_o(t_f) = [1.3\,\mathrm{m}, 0.3\,\mathrm{m}, \pi/4\,\mathrm{rad}]$, $\dot{\boldsymbol{x}}_o(0) = \dot{\boldsymbol{x}}_o(t_f) = [0\,\mathrm{m/s}, 0\,\mathrm{m/s}, 0\,\mathrm{rad/s}]$. The end state of the 1-st phase, i.e. hitting state, $\boldsymbol{x}_h^o = [1.1\,\mathrm{m}, 0.5\,\mathrm{m}, 0.64\,\mathrm{rad}]$, $\dot{\boldsymbol{x}}_h^o = [-1\,\mathrm{m/s}, 0.7\,\mathrm{m/s}, 0\,\mathrm{rad/s}]$ occurs at $t_h = 0.375\,\mathrm{s}$. The inequality constraints consist of a range of motor torques and kinematic constraints.

Three methods for torque estimation provide different results correspondent to their computational complexity. The first method is based on (37) and requires an ∞-norm upper bound for all components, see Fig. 14a and 14b. It is relatively fast as it mostly relies on precomputed values, but it is also the most conservative method and can be used for rather slow motions away from singularities. The magenta graph in Fig. 14 represents the ∞-norm for the matrix representation, i.e. for all joints at once, while the red, green and blue graphs present the ∞-norm for each joint independently. For our fast motion, the nominal torque $\tau_{\mathrm{nom}} = 18.4\,\mathrm{Nm}$ is exceeded for some intervals even with no external disturbance.



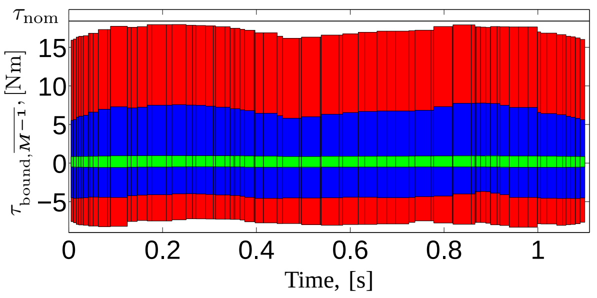
(a) Upper bounds with disturbance $\|\boldsymbol{\tau}_{\mathrm{dist}}\|_2 = 0\mathrm{Nm}$.



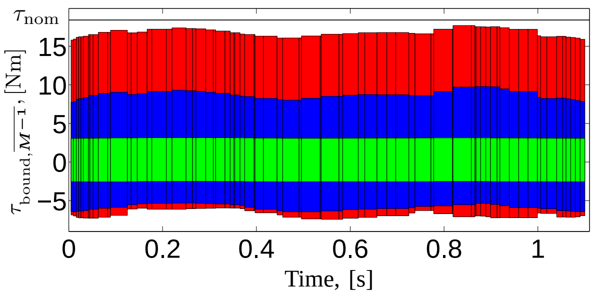
(b) Upper bounds with disturbance $\|\boldsymbol{\tau}_{\mathrm{dist}}\|_2 \leq 0.01\mathrm{Nm}$.

Figure 14: Upper bounds for ∞-norm of the torques for all joints τ_{bound} (magenta) and for every joint separately $\tau_{i,\mathrm{bound}}$ (red, green and blue) on the superimposed time grid.

The second method is based on (32) and produces tight bounds and can be used for fast motions with high disturbances. As a downside, analytic expressions for the bounds of the inverse mass matrix $\left[\boldsymbol{M}^{-1}\right]$ should be provided. This results in extra calculations depending on the size of this matrix, and thus on the number of DoFs. It can be seen from Fig. 15a and Fig. 15b that with an analytic solution, it is possible to have a disturbance torque $\boldsymbol{\tau}_{\mathrm{dist}}$ more than 100 times larger than for the ∞-norm of the inverse mass matrix.



(a) Torque bounds with disturbance $-0.01 \leq \tau_{\text{dist},i} \leq 0.01$[Nm].



(b) Torque bounds with disturbance $-1.3 \leq \tau_{\text{dist},i} \leq 1.3$[Nm].

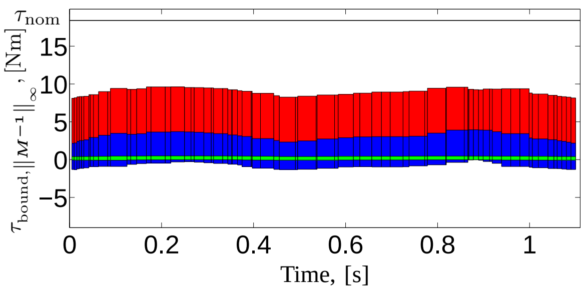
Figure 15: Torque bounds with the analytic expression for upper boundary of the inverse mass matrix and all other components. Upper and lower torque bounds are calculated for every joint separately $\tau_{\text{i,bound}}$ (red, green and blue) on the superimposed time grid.

The third approach provides an analytic expression for all terms except for the inverse mass matrix which is calculated with the ∞-norm, see Fig. 16a and 16b. One can see that the ∞-norm of the inverse mass matrix provides torque estimates inside the required limits, but the ratio between the norm of the inverse mass matrix and the maximum applied torque is higher than for pure analytical approach. The contacts with the ball are taken into account individually around the point of impact. It can be seen in Fig. 16c that the contact affects only the neighboring intervals with the disturbance torque as shown in (27).

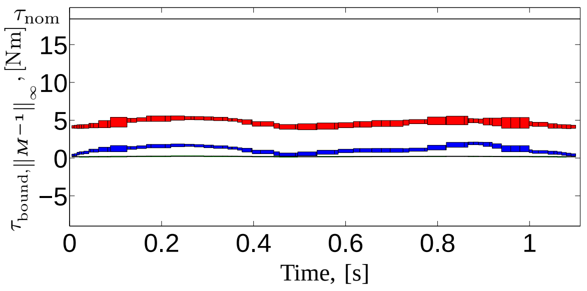
The statistical study in Fig. 17 provides the statistical analysis of the time required for torque evaluation and torque deviation for all the three presented methods.

For all presented torque estimation methods, closed-form symbolic equations for system matrices of the robot need to be known. For ∞-norm and for a mixture of the analytic method with ∞-norm of the inverse mass matrix, inversion of the mass matrix is not needed. Instead, the minimal eigenvalue of the mass matrix is required. Of course, the higher the number of DoFs, e.g. for humanoid robots, the higher number of interval subdivisions need to be performed to provide tighter bounds.

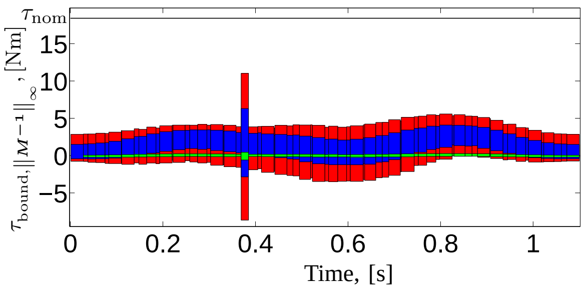
To sum up, results provided by torque estimation methods are correspondent to their computational complexity. Infinity norms provide the capability for real-time computation but perform better for conservative motions with small or no disturbance. Tight torque estimates with the analytic expressions are preferable for fast motions with high disturbances or parametric uncertainties. However, they are slower and require preliminary analysis of the kinematics of the robot and the resulting structure of equations for the inverse mass matrix.



(a) Torque bounds with disturbance $\tau_{\text{dist},i} = 0$Nm.



(b) Torque bounds with disturbance $-0.01 \leq \tau_{\text{dist},i} \leq 0.01$Nm.



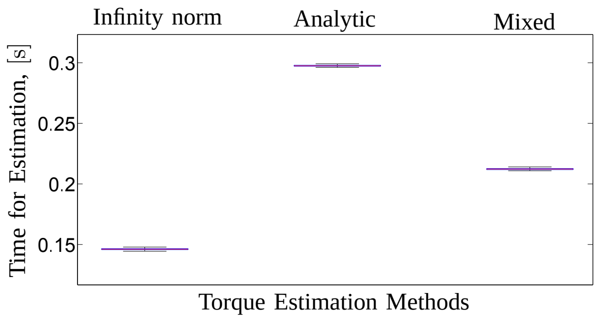
(c) Torque bounds with the short-time contact for average external torque $\tau_{\text{ext,av}} = 2$[Nm] and disturbance torque deviation $\overline{\tau_{\text{dist}}} = 0.065$[Nm], $\underline{\tau_{\text{dist}}} = -0.065$[Nm]

Figure 16: Torque bounds for the mixed torque estimation with ∞-norm for upper boundary of the inverse mass matrix and an analytic expression for all other components. Upper and lower torque bounds are calculated for every joint separately $\tau_{\text{i,bound}}$ (red, green and blue) on the superimposed time grid.

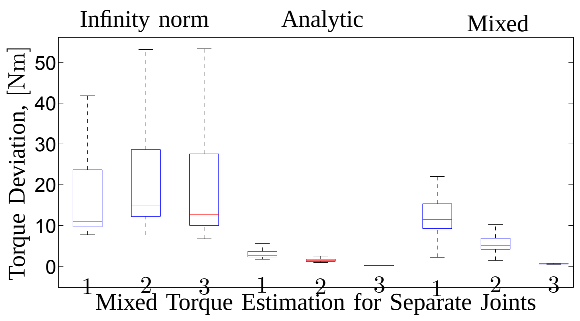
The mixture of analytic bounds with infinity norm performs well for fast motions, avoids complex computations for chosen elements and provides torque bounds that are suitable for real-world manipulators.

VI. Conclusion

In this article, we develop a novel method that deforms precalculated robot trajectories in response to changing task conditions. It allows preserving the features of original trajectories dependent on the desired derivatives of a motion. Such desirable task behavior is adopted with the computationally efficient algorithm that outperforms similar motion adaptation methods using the spline decomposition technique. The feasibility of the produced trajectory and its tracking control with real-world torque-limited actuators is thoroughly investigated. Guarantees for trajectory and torque boundedness are provided in presence of external disturbance and parametric uncertainties. This makes the generated motions dynamically consistent with the robot model, torque-limited actuators and



(a) Statistical evaluation of the time required for the torque estimation.



(b) Statistical evaluation of the torque deviation for the three torque estimation methods. The torque estimation deviations are shown for every single joint torque separately.

Figure 17: Statistical evaluation of three torque estimation methods. The analysis was performed by running simulation of each method 100 times with $|\tau_{\text{dist}}| \leq 0.01[\text{Nm}]$. The central line is the median, the edges of the box are the 25th and 75th percentiles. The evaluation is shown for the whole motion consisting of 2 segments and 68 subintervals. All calculations are done with an Intel Core i3-3120M CPU with 4GB RAM.

controller parametrization. An adaptive approach to a trade-off selection between the size of deformation, trajectory similarity and available computation time allows deriving physically plausible motions online.

Even though the proper choice of a motion from the database can provide the desired robot performance, the method can work even with a single or no precalculated trajectory. In the latter case, original trajectory is initialized with zeros and thus will minimize the desired derivatives of the motion profile.

The timing of the overall motion and of fixed points is of great importance as they remain unchanged after deformation. Deformation works for the situations when the motions in the database have shorter time duration than the time available for the real motion. i.e. $\Delta t_{\text{comp}} \geq 0$. Otherwise, one should cut out a part of the original trajectories or apply time scaling to them. In this case, the minimum derivative deviation will be provided with respect to the time scaled trajectories.

The experimental validation for the spline deformation is performed with several planar volleyball scenarios. These experiments clearly show the online capability, high accuracy, repeatability and robustness of the generated motions in presence of disturbances.

Future work will be focused on guaranteed feasible motions in a fixed amount of time and on extending the applicability of the method to other robotic tasks.

Appendix A
Minimization of the Desired Derivative of Motion

It is stated in Sec. III that the term

$$\begin{bmatrix}\mathbf{L}\\ \bar{\mathbf{P}}\end{bmatrix}^{+}\begin{bmatrix}\mathbf{0}\\ \bar{\mathbf{C}}_2\end{bmatrix}, \tag{42}$$

can be expressed in the continuous domain by a spline. As the matrix $\mathbf{L}$ consists of n-th order finite differences along the entire trajectory and the matrices $\bar{\mathbf{P}}, \bar{\mathbf{C}}_2$ specify waypoints, the term in (42) can then be interpreted as minimizing the n-th derivative along the trajectory while passing a set of waypoints. An optimal trajectory x_{opt} is then calculated as

$$x_{\text{opt}} = \min_x I(x) = \min_x \frac{1}{2}\int\limits_0^T \left(x^{(n)}(t)\right)^2 dt. \tag{43}$$

Through the calculus of variation we consider the disturbed trajectory $x(t) + \epsilon\eta$ with the scalar ϵ and η as an arbitrary function fulfilling the boundary conditions

$$\begin{aligned}\eta(0) = 0, &\quad \eta(T) = 0,\\ \dot{\eta}(0) = 0, &\quad \dot{\eta}(T) = 0,\\ \cdots &\quad \cdots\\ \eta^{(n-1)}(0) = 0, &\quad \eta^{(n-1)}(T) = 0.\end{aligned} \tag{44}$$

This results in

$$\begin{aligned}I(x+\epsilon\eta) &= \frac{1}{2}\int\limits_0^T (x^{(n)} + \epsilon\eta^{(n)})^2 dt,\\ \frac{dI(x+\epsilon\eta)}{d\epsilon} &= \int\limits_0^T (x^{(n)} + \epsilon\eta^{(n)})\eta^{(n)} dt.\end{aligned} \tag{45}$$

For x to minimize $I(x+\epsilon\eta)$, the following condition has to be fulfilled. In all other cases the trajectory x is not optimal.

$$\left.\frac{dI(x+\epsilon\eta)}{d\epsilon}\right|_{\epsilon=0} = 0 = \int\limits_0^T x^{(n)}\eta^{(n)} dt \tag{46}$$

Through partial integration of (46) $n-2$ times we obtain

$$\begin{aligned}\int\limits_0^T x^{(n)}\eta^{(n)} &= \underbrace{x^{(n)}\eta^{(n-1)}\Big|_0^T}_{=0} - \int\limits_0^T x^{(n+1)}\eta^{(n-1)} = \cdots\\ &= \int\limits_0^T x^{(2n)}\eta \equiv 0\end{aligned} \tag{47}$$

as the resulting condition that must hold for any function η. This is the case for every function fulfilling

$$x^{(2n)} = 0 \;\; \forall t \in [0, T]. \tag{48}$$

Any $(2n-1)$-th order polynomial of the form

$$x = \sum_{j=0}^{2n-1} \mathbf{a}_j t^j \tag{49}$$

fulfills this condition.

APPENDIX B
FEATURES OF BERNSTEIN POLYNOMIALS

A. Properties of Bernstein Polynomials

Partition of unity property:

$$\sum_{k=0}^{l-1} \boldsymbol{B}_k^{l-1} = 1.$$

Non-negativity property:

$$\boldsymbol{B}_i^l \geq 0.$$

The derivative of a Bernstein polynomial:

$$\dot{\boldsymbol{B}}_i^l = l(\boldsymbol{B}_{i-1}^{l-1} - \boldsymbol{B}_i^{l-1}),$$

where $\boldsymbol{B}_k^l = 0$ for $k < 0$ and $k > l$.

B. Derivation of Bounds for Bernstein Polynomials

Using these properties, we derive velocities and accelerations of deformation part. Since coefficients $\boldsymbol{c}_k$ from (22) are constant, the expression for the velocity is as follows

$$\dot{\boldsymbol{x}}_s = \sum_{k=0}^{l} \boldsymbol{c}_k \dot{\boldsymbol{B}}_k^l = \boldsymbol{c}_0 \dot{\boldsymbol{B}}_0^l + \boldsymbol{c}_1 \dot{\boldsymbol{B}}_1^l + \ldots + \boldsymbol{c}_l \dot{\boldsymbol{B}}_l^l. \tag{50}$$

Derivatives of the separate Bernstein polynomials

$$\begin{aligned} \dot{\boldsymbol{B}}_0^l &= l(\underbrace{\boldsymbol{B}_{-1}^{l-1}}_{=0} - \boldsymbol{B}_0^{l-1}), \\ \dot{\boldsymbol{B}}_1^l &= l(\boldsymbol{B}_0^{l-1} - \boldsymbol{B}_1^{l-1}), \\ &\ldots \\ \dot{\boldsymbol{B}}_l^l &= l(\boldsymbol{B}_{l-1}^{l-1} - \underbrace{\boldsymbol{B}_l^{l-1}}_{=0}). \end{aligned} \tag{51}$$

After substitution of (51) into (50) we get

$$\begin{aligned} \dot{\boldsymbol{x}}_s &= l((-\boldsymbol{c}_0 \boldsymbol{B}_0^{l-1} + \boldsymbol{c}_1 \boldsymbol{B}_0^{l-1}) + (-\boldsymbol{c}_1 \boldsymbol{B}_1^{l-1} + \boldsymbol{c}_2 \boldsymbol{B}_1^{l-1}) \\ &+ \ldots + (-\boldsymbol{c}_{l-1} \boldsymbol{B}_{l-1}^{l-1} + \boldsymbol{c}_l \boldsymbol{B}_{l-1}^{l-1})) = l \sum_{k=0}^{l-1} (\boldsymbol{c}_{k+1} - \boldsymbol{c}_k) \boldsymbol{B}_k^{l-1}. \end{aligned}$$

Analogously, for acceleration we get

$$\begin{aligned} \ddot{\boldsymbol{x}}_s &= l \sum_{k=0}^{l-1} (\boldsymbol{c}_{k+1} - \boldsymbol{c}_k) \dot{\boldsymbol{B}}_k^{l-1} \\ &= l(l-1) \sum_{k=0}^{l-2} (\boldsymbol{c}_k - 2\boldsymbol{c}_{k+1} + \boldsymbol{c}_{k+2}) \boldsymbol{B}_k^{l-2}. \end{aligned}$$

Therefore, velocities are bounded through the following chain of inequalities

$$\begin{aligned} \dot{\boldsymbol{x}}_s &= l \sum_{k=0}^{l-1} (\boldsymbol{c}_{k+1} - \boldsymbol{c}_k) \boldsymbol{B}_k^{l-1} \\ &= l((\boldsymbol{c}_1 - \boldsymbol{c}_0) \boldsymbol{B}_0^{l-1} + \ldots + (\boldsymbol{c}_l - \boldsymbol{c}_{l-1}) \boldsymbol{B}_{l-1}^{l-1}) \\ &\leq l(\max_{k=0,\ldots,l-1} (\boldsymbol{c}_{k+1} - \boldsymbol{c}_k) \boldsymbol{B}_0^{l-1} + \ldots \\ &+ \max_{k=0,\ldots,l-1} (\boldsymbol{c}_{k+1} - \boldsymbol{c}_k) \boldsymbol{B}_{l-1}^{l-1}) \\ &= l \max_{k=0,\ldots,l-1} (\boldsymbol{c}_{k+1} - \boldsymbol{c}_k)(\boldsymbol{B}_0^{l-1} + \boldsymbol{B}_1^{l-1} + \ldots + \boldsymbol{B}_{l-1}^{l-1}) \\ &= l \max_{k=0,\ldots,l-1} (\boldsymbol{c}_{k+1} - \boldsymbol{c}_k) = \overline{\dot{\boldsymbol{x}}_s}. \end{aligned}$$

Analogously, bounds are derived for acceleration

$$\ddot{\boldsymbol{x}}_s \leq l(l-1) \max_{k=0,\ldots,l-2} (\boldsymbol{c}_k - 2\boldsymbol{c}_{k+1} + \boldsymbol{c}_{k+2}) = \overline{\ddot{\boldsymbol{x}}_s}.$$

APPENDIX C
INFINITY NORM BOUNDS

The variables for bounds are derived using ∞-norms and interval analysis of continuous functions on closed intervals.

Upper bound for a single matrix element can be found as

$$\begin{aligned} \|a_{i,j}\|_\infty &= \max_{\boldsymbol{q},\dot{\boldsymbol{q}}} (|\sup(a_{i,j})|, |\inf(a_{i,j})|), \\ &\boldsymbol{q} \in [\underline{\boldsymbol{q}}, \overline{\boldsymbol{q}}], \\ &\dot{\boldsymbol{q}} \in [\underline{\dot{\boldsymbol{q}}}, \overline{\dot{\boldsymbol{q}}}]. \end{aligned}$$

Bounds for vector elements including $\bar{g}$ are calculated as analytic expressions in a vector form

$$\|\boldsymbol{a}\|_\infty = \max_i (|a_i|) = \max_i (\|a_i\|_\infty).$$

Matrix elements bounds including $\bar{m}$ and $\bar{c}$ are found as follows

$$\|\boldsymbol{A}\|_\infty = \max_i (\sum_{j=1}^{n} \|a_{i,j}\|_\infty).$$

The derivation of the $\|\boldsymbol{J}^+\|_\infty$ is dependent on the dimensionality of the robot Jacobian m_j and the minimum singular value $\sigma_{j,\min}$ of the Jacobian $\boldsymbol{J}$

$$\left\|\boldsymbol{J}^+\right\|_\infty \leq \sqrt{m_j} \left\|\boldsymbol{J}^+\right\|_2 \leq \frac{\sqrt{m_j}}{\sigma_{j,\min}}.$$

REFERENCES

[1] O. Von Stryk and M. Schlemmer, "Optimal control of the industrial robot manutec r3," *Computational optimal control, International series of Numerical Mathematics*, vol. 115, pp. 367–382, 1994.

[2] S. Lengagne, N. Ramdani, and P. Fraisse, "Planning and fast replanning safe motions for humanoid robots," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1095–1106, 2011.

[3] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of whole-body optimal dynamic multi-contact motions," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1104–1119, 2013.

[4] S. Quinlan, *Real-time modification of collision-free paths*. PhD thesis, Stanford University, 1994.

[5] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *IEEE Int. Conf. on Robotics and Automation*, pp. 489–494, 2009.

[6] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization.," in *Robotics: Science and Systems*, vol. 9, pp. 1–10, Citeseer, 2013.

[7] A. Pekarovskiy, F. Stockmann, M. Okada, and M. Buss, "Hierarchical robustness approach for nonprehensile catching of rigid objects," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014.

[8] S. Patil, J. Burgner, R. J. Webster, and R. Alterovitz, "Needle steering in 3-d via rapid replanning," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 853–864, 2014.

[9] W. Sun, S. Patil, and R. Alterovitz, "High-frequency replanning under uncertainty using parallel sampling-based motion planning," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 104–116, 2015.

[10] J. E. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985.

[11] K. G. Shin and N. D. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, no. 6, pp. 531–541, 1985.

[12] O. Dahl and L. Nielsen, "Torque-limited path following by online trajectory time scaling," *IEEE Transactions on Robotics and Automation*, vol. 6, pp. 554–561, Oct 1990.

[13] O. Gerelli and C. Bianco, "Real-time path-tracking control of robotic manipulators with bounded torques and torque-derivatives," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 532–537, Sept 2008.

[14] X. Broquere, D. Sidobre, and I. Herrera-Aguilar, "Soft motion trajectory planner for service manipulator robot," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2808–2813, IEEE, 2008.

[15] T. Kröger and F. M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 94–111, 2010.

[16] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *IEEE Int. Conf. on Robotics and Automation*, pp. 1398–1403, 2002.

[17] J. Kober, K. Muelling, O. Kroemer, C. H. Lampert, B. Scholkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *IEEE Int. Conf. on Robotics and Automation*, pp. 853–858, 2010.

[18] D.-H. Park, P. Pastor, S. Schaal, *et al.*, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *8th IEEE-RAS International Conference on Humanoid Robots. Humanoids*, pp. 91–98, 2008.

[19] S. Calinon, E. Sauser, A. Billard, and D. Caldwell, "Evaluation of a probabilistic approach to learn and reproduce gestures by imitation," in *IEEE Int. Conf. on Robotics and Automation*, pp. 2381–2388, 2010.

[20] K. Muelling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[21] M. Gleicher, "Retargetting motion to new characters," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 33–42, ACM, 1998.

[22] J. Lee and S. Y. Shin, "A hierarchical approach to interactive motion editing for human-like figures," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 39–48, ACM, 1999.

[23] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Proceedings of the 16th International Symposium on Robotics Research (ISRR)*, 2013.

[24] Q.-C. Pham and Y. Nakamura, "A new trajectory deformation algorithm based on affine transformations," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 1054–1063, 2015.

[25] T. Nierhoff and S. Hirche, "Fast trajectory replanning using laplacian mesh optimization," in *IEEE Int. Conf. on Control, Automation, Robotics and Vision*, 2012.

[26] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 175–184, 2004.

[27] Ö. Arslan and U. Saranli, "Reactive planning and control of planar spring–mass running on rough terrain," *IEEE Transactions on Robotics*, vol. 28, no. 3, pp. 567–579, 2012.

[28] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *The International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.

[29] J. Kieffer, A. J. Cahill, and M. R. James, "Robust and accurate time-optimal path-tracking control for robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 6, pp. 880–890, 1997.

[30] F. Ghorbel, B. Srinivasan, and M. W. Spong, "On the uniform boundedness of the inertia matrix of serial robot manipulators," *Journal of Robotic Systems*, vol. 15, no. 1, pp. 17–28, 1998.

[31] J. Craig, P. Hsu, and S. Sastry, "Adaptive control of mechanical manipulators," in *IEEE Int. Conf. on Robotics and Automation*, vol. 3, pp. 190–195, Apr 1986.

[32] J. Mulero-Martinez, "Uniform bounds of the coriolis/centripetal matrix of serial robot manipulators," *IEEE Transactions on Robotics*, vol. 23, pp. 1083–1089, Oct 2007.

[33] F. L. Lewis, C. T. Abdallah, and D. M. Dawson, *Control of robot manipulators*, vol. 236. Macmillan New York, 1993.

[34] A. Pekarovskiy, T. Nierhoff, J. Schenek, Y. Nakamura, S. Hirche, and M. Buss, "Online deformation of optimal trajectories for constrained nonprehensile manipulation," in *IEEE Int. Conf. on Robotics and Automation*, 2015.

[35] A. Pekarovskiy, T. Nierhoff, S. Hirche, and M. Buss, "Spline deformation of locally optimal trajectories: Feasibility and upper bound on control inputs," in *Proceedings of the 54th IEEE Conference on Decision and Control*, 2015.

[36] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to interval analysis*. Siam, 2009.

[37] J. Rohn, "Bounds on eigenvalues of interval matrices," *ZAMM-Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 78, no. 3, 1998.

[38] M. Hladík, D. Daney, and E. Tsigaridas, "Bounds on real eigenvalues and singular values of interval matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 4, pp. 2116–2129, 2010.

Alexander Pekarovskiy is currently pursuing the PhD degree in Robotics at the Chair of Automatic Control Engineering, Technical University of Munich. He received his Bachelor's and Master's in Control Systems from Saint-Petersburg Electrotechnical University "LETI", Saint-Petersburg, Russia in 2009 and 2011, respectively. His research interests include numerical optimal control, sampling-based optimization, dynamic manipulation, mobile manipulation and real-time reactive motion generation.

Thomas Nierhoff is currently working for Robert Bosch GmbH as a Solution Engineer. He received the Diploma Engineering degree in Robotics in 2011 and the Doctor of Engineering degree in electrical engineering from Technical University of Munich (TUM), Munich, Germany, in 2015. His research interests include Laplacian trajectory editing, full body motion adaption and pool robotics.

Sandra Hirche received the Diploma degree in mechanical engineering and transport systems from Technical University Berlin, Berlin, Germany, in 2002 and the Doctor of Engineering degree in electrical engineering and information technology from Technical University of Munich (TUM), Munich, Germany, in 2005. From 2005 to 2007 she was a Postdoctoral Researcher with the Japanese Society for the Promotion of Science, Fujita Laboratory, Tokyo Institute of Technology, Tokyo, Japan. From 2008 to 2012 she was an Associate Professor with TUM. Since 2013 she has been the TUM Liesel Beckmann Distinguished Professor and holds the Chair of Information-oriented Control with the Department of Electrical Engineering and Information Technology, TUM. Prof. Hirche has served on the Editorial Boards of IEEE Transactions on Control Systems Technology and IEEE Transactions on Haptics. Her main research interests include cooperative, distributed, and networked control with applications in human-robot interaction; multirobot systems; and general robotics.

Martin Buss received the Diploma Engineering degree in electrical engineering from the Technische Universität Darmstadt, Darmstadt, Germany, in 1990, and the Doctor of Engineering degree in electrical engineering from The University of Tokyo, Tokyo, Japan, in 1994. In 1988, he was a research student for one year with the Science University of Tokyo. From 1994–1995, he was a postdoctoral researcher with the Department of Systems Engineering, Australian National University, Canberra, Australia. From 1995–2000, he was a senior research assistant and lecturer with the Chair of Automatic Control Engineering, Department of Electrical Engineering and Information Technology, Technical University of Munich, Munich, Germany. From 2000–2003, he was full professor, the head of the control systems group, and the deputy director of the Institute of Energy and Automation Technology, Faculty IV , Electrical Engineering and Computer Science, Technical University Berlin, Berlin, Germany. Since 2003 he has been full professor (chair) with the Chair of Automatic Control Engineering, Faculty of Electrical Engineering and Information Technology, Technical University of Munich, where he has been with the Medical Faculty since 2008. Since 2006, he has also been the coordinator of the Deutsche Forschungsgemeinschaft cluster of excellence "Cognition for Technical Systems (CoTeSys)". His research interests include automatic control, mechatronics, multimodal human system interfaces, optimization, nonlinear, and hybrid discrete-continuous systems.