2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Real-Time Predictive Kinematics Control of Redundancy: a Benchmark of Optimal Control Approaches

Jonas Wittmann[1,2], Arian Kist[1] and Daniel J. Rixen[1]

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981675

***Abstract*— Modern collaborative manipulators operate in unknown environments and share the work space with human coworkers. To ensure flexibility, their kinematic design is redundant which increases the solution space of the inverse kinematics (IK). We propose a real-time capable *Predictive Kinematics Controller (PKC)* that tracks task space trajectories as a first priority and computes optimal joint trajectories w.r.t. secondary objectives based on model predictive control (MPC). Therefor, the *PKC* solves a MPC problem in the nullspace of the task space trajectory. We benchmark a direct shooting, a direct collocation and an indirect gradient method in simulation and we identify the direct shooting method as the most efficient. We demonstrate the superior performance of the *PKC* compared to state-of-the-art local redundancy resolution approaches. In experiments, we show the real-time capability of our implementation.**

I. Introduction

Robotic automation solutions usually define tasks in Cartesian space or a subspace of it. Inverse Kinematics (IK) computes the corresponding joint reference values for the robot control. On position level, an analytical solution for the nonlinear mapping between task space and configuration space ($\mathcal{C}$-space) only exists in special cases. Therefore, iterative methods like *Newton-Raphson* are generally applied. In contrast to that, on velocity and acceleration level, i.e. in differential kinematics, the relationship between task space and $\mathcal{C}$-space is linear and a closed-form IK solution is defined [1]. For redundant kinematic structures, the differential IK has an infinite number of solutions and the definition of additional secondary objectives, e.g. joint limit avoidance, is used to obtain a unique solution. [2] proposes the state-of-the-art IK scheme for local redundancy resolution, the *Automatic Supervisory Control (ASC)*, which may be applied either on velocity or acceleration level:

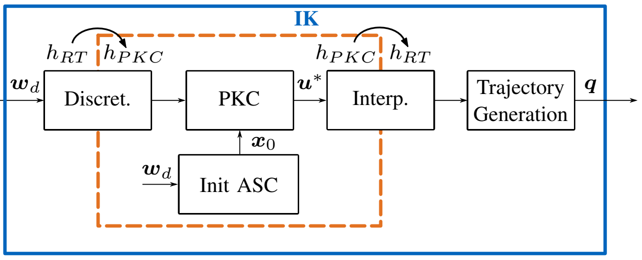
$$\dot{\boldsymbol{q}}_d = \boldsymbol{J}^{\#}\underbrace{(\dot{\boldsymbol{w}}_d + \boldsymbol{K}(\boldsymbol{w}_d - \boldsymbol{w}))}_{\dot{\boldsymbol{w}}_{d,eff}} + \underbrace{(\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J})}_{\boldsymbol{N}}\ \boldsymbol{u} \tag{1}$$

$$\ddot{\boldsymbol{q}}_d = \boldsymbol{J}^{\#}(\ddot{\boldsymbol{w}}_{d,eff} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + \boldsymbol{N}\,(\boldsymbol{u} - k\dot{\boldsymbol{q}}) \tag{2}$$

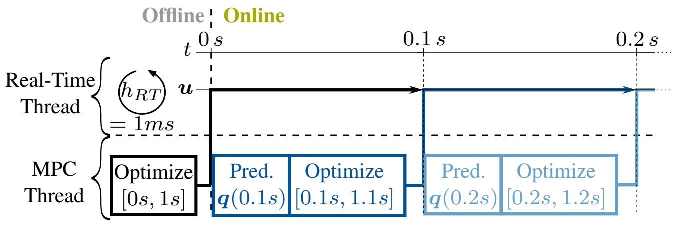
The $\mathcal{C}$-space and task space trajectories are defined by $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ and $\boldsymbol{w}$, $\dot{\boldsymbol{w}}$, $\ddot{\boldsymbol{w}}$ respectively. Trajectory values with subscript d denote desired reference values and trajectory values without subscript denote the measured trajectory profiles executed by the robot. $\boldsymbol{u}$ controls secondary objectives based on gradient descent. E.g. joint limit avoidance can be

[1]Technical University of Munich, TUM School of Engineering and Design, Department of Mechanical Engineering, Munich Institute of Robotics and Machine Intelligence (MIRMI), Chair of Applied Mechanics, Boltzmannstraße 15, 85748 Garching, Germany

[2]Corresponding author; e-mail: jonas.wittmann@tum.de



(a) Two different cycle times h_{RT} and h_{PKC} are used for real-time control and *PKC* respectively.



(b) Multi-threaded implementation of the *PKC* with $h_{RT} = 1$ ms and $h_{PKC} = 100$ ms. The time horizon length is 1 s.

Fig. 1. The low-level real-time control and the *PKC* run in different threads and use two different cycle times, linked by discretization and interpolation. The *PKC* computes the optimal nullspace control $\boldsymbol{u}^\star$ at a lower rate than the one of the real-time control and therefore $\boldsymbol{u}^\star$ is interpolated before applying it to the robot. The *Real-Time Thread* executes the interpolation of the first horizon time step while the *MPC Thread* computes the next time horizon. Thereby, in each time horizon, the *PKC* is initialized with the *ASC* solution (see Section III). This architecture implies a cascaded control in which we use a inner control loop to solve the IK and an outer control loop to solve the MPC. Our multi-threaded implementation ensures real-time capability (see Section IV-E).

expressed as an optimization objective l_{Jla}, that is controlled by $\boldsymbol{u} = \left(-\frac{\partial l_{Jla}}{\partial \boldsymbol{q}}\right)^T$. $\boldsymbol{J}^{\#} = \boldsymbol{W}^{-1}\boldsymbol{J}_w^T(\boldsymbol{J}_w\boldsymbol{W}^{-1}\boldsymbol{J}_w^T)^{-1}$ is the *Moore-Penrose pseudoinverse* and we set $\boldsymbol{W} = \boldsymbol{I}$. To deal with numerical drift that is introduced by differential kinematics, the *effective* desired task space reference values $\star_{d,eff}$ compensate the task space error [1]. $\ddot{\boldsymbol{w}}_{d,eff}$ additionally compensates the drift on velocity level. The control inputs for secondary tasks $\boldsymbol{u}$ must not influence the execution of the primary task and therefore $\boldsymbol{N}$ is used for nullspace mapping. In second-order redundancy resolution schemes, instability problems occur that result in nullspace drift of the joint variables: if the nullspace acceleration due to second-order objectives vanishes in (2), the nullspace velocity still remains constant [3]. Therefore, the damping $k\dot{\boldsymbol{q}}$ is introduced.

(1) and (2) are implemented in the lower real-time control of the robot. In case the task space trajectory is adapted

online or in case the environment changes, the local approaches react instantaneously, however only locally, e.g. when already in proximity to obstacles. Global redundancy resolution approaches are on the other side of the spectrum and compute globally optimized joint reference values for the whole trajectory, however, they are not reactive and not real-time capable [4], [5].

We propose an IK resolution scheme in between local and global approaches that combines the best of both worlds: optimal solutions and real-time. Our contributions are:

- We propose a real-time capable *PKC* for MPC of redundant robot kinematics that we validate in experiments on a *FRANKA EMIKA* robot. Related work often simplifies the problem to quadratic programming (QP). Our approach includes nonlinear objectives and constraints which is required for involved secondary objectives, e.g. collision avoidance. Fig. 1 shows our *PKC* implementation that we detail in Section IV.
- To ensure efficiency, we benchmark three numerical optimal control approaches and choose the best among them: direct collocation, direct shooting and indirect gradient method. Comparing these state-of-the-art approaches in kinematics control has not been published yet and, if the problem is not simplified to QP, related work most often uses the indirect gradient method to solve the optimal control. However, we show that the direct shooting method is more efficient.
- We derive the analytical gradients for the direct shooting and direct collocation method. Compared to numerically approximated gradients, this increases the computational efficiency significantly.
- We show that MPC, applied on velocity level, as done in most related work, generates jerky trajectory profiles. Therefore, we propose a *PKC* implementation on acceleration level to smooth the control signals.

II. Related Work

[6] proposes a global redundancy control that minimizes an integral performance index subject to differential IK. They derive the first-order optimality conditions of the optimal control problem and use an indirect shooting method to solve the resulting two-point boundary value problem. [7] adopts the approach in [6] and decouples the canonical differential equations of the first-order optimality conditions. They apply *Pontryagin's minimum principle*, to boil down the optimal control problem to an unconstrained minimization of the Hamiltonian. [7] uses a conjugate gradient approach to solve the latter. They demonstrate the advantages of the global approach compared to *ASC*. For long trajectories and robots with a high number of degrees of freedom (DoF), the computational effort to solve the MPC is not negligible. To ensure real-time capability, [8] defines the maximum time δ_{max} to solve the optimization. A MPC cycle starting at time t solves the optimization and the optimal control input computed in the previous cycle is used for the real-time control cycle until $t+\delta_{max}$. [9] enhances the approach by a feedback controller that compensates uncertainties at a faster sampling rate than the MPC cycle for a more robust control. [7] proposes to apply their optimization formulation within MPC. To avoid discontinuous nullspace velocities, they propose a double integrator for $\boldsymbol{u}$. They define self-collision avoidance as the secondary objective and validate the implementation on a manipulator with nine DoF. The efficiency is limited, with a real-time control cycle of 10 ms and a MPC horizon of 0.15 s with only one iteration per optimization. [10] extends the approach and uses different sample times for the MPC and low-level control in a multi-threaded architecture. They use third-order polynomials to interpolate the solution of the former before applying it to a humanoid. [11] replaces the differential IK and directly sets $\dot{\boldsymbol{q}} = \boldsymbol{u}$ which violates the primary goal to track the task space trajectory. Therefore, the cost function, besides collision avoidance, incorporates the task space error on position level. They use a direct method to solve the MPC, a sampling rate of 0.1 s and a time horizon of 1.0 s in an application on a mobile manipulator with ten DoF. [5], similar to [7], implements a second-order integrator from $\boldsymbol{u}$ to $\boldsymbol{q}$ to smooth the signals. They neglect the nullspace projection in (1) and instead ensure task space trajectory tracking in the cost function. A QP problem results that is solved in 0.45 ms for a kinematic with four DoF. [12] enhances this approach and uses B-splines and Newmark time integration to compute smooth trajectories within the MPC horizon. [13] proposes a QP-based predictive IK approach that replaces the differential kinematics by double integrators that directly apply $\boldsymbol{u}$ as reference joint accelerations. The task space error on velocity level is incorporated in the cost function. The advantage of the presented QP-based predictive IK approaches is that they are real-time capable. However, they implement task space trajectory tracking within the cost function which limits accuracy. Further, QP implementations prevent more general nonlinear objectives, e.g. required for collision avoidance.

III. Proposed Method

Our *PKC* uses the MPC formulation given in (3), which is based on [6] extended by inequality constraints for the joint limits that we set symmetric for simpler notification. Further, the joint states at the beginning of a time horizon t_0 with given horizon duration T_{PKC} are known according to (3c). $\mathbf{f}_{\text{IK}}$ are the first- or second-order differential IK according to (1) or (2) respectively.

$$\min_{\boldsymbol{u}(t)} \qquad L(\boldsymbol{q},\boldsymbol{u}) = \int_{t_0}^{t_0+T_{PKC}} l(\boldsymbol{q},\boldsymbol{u})\,dt \tag{3a}$$

$$\text{s. t.:} \qquad \mathbf{f}_{\text{IK}} : \boldsymbol{w}, \dot{\boldsymbol{w}}, \boldsymbol{w}_d, \dot{\boldsymbol{w}}_d, \ddot{\boldsymbol{w}}_d, \boldsymbol{u}, \boldsymbol{q} \mapsto \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} \tag{3b}$$

$$\boldsymbol{q}(t_0) = \boldsymbol{q}_1 \tag{3c}$$

$$|\boldsymbol{q}| \leq \boldsymbol{q}_{max}; |\dot{\boldsymbol{q}}| \leq \dot{\boldsymbol{q}}_{max}; |\ddot{\boldsymbol{q}}| \leq \ddot{\boldsymbol{q}}_{max} \tag{3d}$$

To derive a most efficient *PKC* implementation, we benchmark three state-of-the-art numerical optimal control approaches and thereby derive their analytical gradients: direct collocation, direct shooting and indirect gradient method. The two former transcribe the infinite dimensional problem in (3) to the finite dimensional nonlinear program (NLP) in (4) for which we use state-of-the-art NLP solvers. With a trapezoidal integration, we approximate L with a time step size h_{PKC} that discretizes T_{PKC} into N_{PKC} time steps. The constraints (4b-4c) and the optimization variable $\boldsymbol{x}$ depend on the transcription method and are outlined in the following.

$$\min_{\boldsymbol{x}} \quad \phi(\boldsymbol{x}) = \frac{1}{2}\, h_{PKC} \sum_{k=1}^{N_{PKC}-1} (l_k + l_{k+1}) \tag{4a}$$

$$\text{s. t.:} \quad \boldsymbol{c}(\boldsymbol{x}) = \mathbf{0} \tag{4b}$$

$$\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0} \tag{4c}$$

Note that regardless of the transcription method, we initialize the optimization variable $\boldsymbol{x}$ based on the *ASC* solution.

A. Direct Transcription on Velocity Level

In this subsection, we present direct methods to solve (3) on velocity-level. That is, we use (1) in (3b).

a) Direct Collocation with Analytical Gradients: In collocation methods both the state and the control are fully discretized and used as optimization variables. Applying this approach on velocity level and considering (3c), gives the following optimization vector:

$$\boldsymbol{x} = [\underbrace{\boldsymbol{u}_1^T \ \cdots \ \boldsymbol{u}_{N_{PKC}}^T}_{\hat{\boldsymbol{u}}^T} \ \underbrace{\boldsymbol{q}_2^T \ \cdots \ \boldsymbol{q}_{N_{PKC}}^T}_{\hat{\boldsymbol{q}}^T}]^T \tag{5}$$

Due to the collocation approach $l_k = l(\boldsymbol{u}_k, \boldsymbol{q}_k)$ is only a function of the current control input and joint configuration and independent of other time steps than k. Thus, the following derivation of the analytical gradient of the optimization function holds:

$$\frac{\partial \phi(\boldsymbol{x})}{\partial(\hat{\boldsymbol{u}}, \hat{\boldsymbol{q}})} = \frac{h_{PKC}}{2} \sum_{k=1}^{N_{PKC}-1} \left(\frac{\partial l_k}{\partial(\hat{\boldsymbol{u}}, \hat{\boldsymbol{q}})} + \frac{\partial l_{k+1}}{\partial(\hat{\boldsymbol{u}}, \hat{\boldsymbol{q}})} \right) \tag{6a}$$

$$\frac{\partial l_k}{\partial(\hat{\boldsymbol{u}}, \hat{\boldsymbol{q}})} = \begin{bmatrix} \mathbf{0}^T & \cdots & \frac{\partial l_k}{\partial(\boldsymbol{u}_k, \boldsymbol{q}_k)} & \cdots & \mathbf{0}^T \end{bmatrix} \tag{6b}$$

The respective derivatives of l w.r.t. $\boldsymbol{q}_k$ and $\boldsymbol{u}_k$ depend on the chosen secondary objectives, which are given in Section IV. Similar to the discretization of the cost integral in (4a), we use a trapezoidal rule for the collocation constraints that incorporate the differential IK of (3b). Thus, (4b) is defined as follows:

$$\boldsymbol{c}(\boldsymbol{x}) = \begin{bmatrix} \cdots \\ \boldsymbol{q}_k - \boldsymbol{q}_{k+1} + \frac{1}{2} h_{PKC} (\mathbf{f}_{\text{IK,k}} + \mathbf{f}_{\text{IK,k+1}}) \\ \cdots \end{bmatrix} \tag{7}$$

Similar to the optimization function, $\boldsymbol{c}_k(\boldsymbol{u}_k, \boldsymbol{q}_k, \boldsymbol{u}_{k+1}, \boldsymbol{q}_{k+1})$ only depends on the current and next time step and a sparse Jacobian w.r.t. $\boldsymbol{x}$ results. On velocity level, considering (1), the analytical Jacobian of the collocation constraints (7) at time step k are defined as follows:

$$\frac{\partial \boldsymbol{c}_k}{\partial \boldsymbol{u}_k} = \frac{1}{2} h_{PKC} \frac{\partial \mathbf{f}_{\text{IK,k}}}{\partial \boldsymbol{u}_k} \quad \text{with} \quad \frac{\partial \mathbf{f}_{\text{IK,k}}}{\partial \boldsymbol{u}_k} = \boldsymbol{N}(\boldsymbol{q}_k) \tag{8}$$

$$\frac{\partial \boldsymbol{c}_k}{\partial \boldsymbol{q}_k} = \boldsymbol{I} + \frac{1}{2} h_{PKC} \frac{\partial \mathbf{f}_{\text{IK,k}}}{\partial \boldsymbol{q}_k} \tag{9}$$

The local derivatives $\frac{\partial \mathbf{f}_{\text{IK,k}}}{\partial \boldsymbol{q}_k} = \begin{bmatrix} \frac{\partial \mathbf{f}_{\text{IK,k}}}{\partial q_{k,1}} & \cdots & \frac{\partial \mathbf{f}_{\text{IK,k}}}{\partial q_{k,N_q}} \end{bmatrix}$ in (9) are computed column-wise for a robot with N_q joints (note that the subscript k, indicating the time step, is dropped for readability) [7]:

$$\frac{\partial \mathbf{f}_{\text{IK}}}{\partial q_i} = \frac{\partial \boldsymbol{J}_w^{\#}}{\partial q_i} \dot{\boldsymbol{w}}_{d,eff} + \boldsymbol{J}_w^{\#} \frac{\partial \dot{\boldsymbol{w}}_{d,eff}}{\partial q_i} + \frac{\partial \boldsymbol{N}}{\partial q_i} \boldsymbol{u} \tag{10a}$$

$$\frac{\partial \boldsymbol{J}_w^{\#}}{\partial q_i} = \left(\frac{\partial \boldsymbol{J}_w}{\partial q_i} \right)^T \left(\boldsymbol{J}_w \boldsymbol{J}_w^T \right)^{-1} - \boldsymbol{J}_w^T \Bigg[\left(\boldsymbol{J}_w \boldsymbol{J}_w^T \right)^{-1} \left(\frac{\partial \boldsymbol{J}_w}{\partial q_i} \boldsymbol{J}_w^T + \boldsymbol{J}_w \left(\frac{\partial \boldsymbol{J}_w}{\partial q_i} \right)^T \right) \left(\boldsymbol{J}_w \boldsymbol{J}_w^T \right)^{-1} \Bigg] \tag{10b}$$

$$\frac{\partial \boldsymbol{N}}{\partial q_i} = - \left(\frac{\partial \boldsymbol{J}_w^{\#}}{\partial q_i} \boldsymbol{J}_w + \boldsymbol{J}_w^{\#} \frac{\partial \boldsymbol{J}_w}{\partial q_i} \right) \tag{10c}$$

We use a finite forward difference approximation for $\frac{\partial \dot{\boldsymbol{w}}_{d,eff}}{\partial \boldsymbol{q}}$ in (10a) since the incorporated drift compensation in $\dot{\boldsymbol{w}}_{d,eff}$ includes the task space error on position level. The rotational component is formulated with quaternions and there is no analytical gradient derivation for this term.

(4c) ensures the actuator limits according to (3d). The Jacobian of the position level constraint is straightforward as it directly depends on the optimization vector, i.e. $\hat{\boldsymbol{q}}$. The Jacobian of the velocity level constraint has already been derived in (8) and (9) as $\dot{\boldsymbol{q}}_k = \mathbf{f}_{\text{IK}}(\boldsymbol{w}_k, \dot{\boldsymbol{w}}_k, \boldsymbol{u}_k, \boldsymbol{q}_k)$. The joint accelerations are obtained using central differences for interior points and forward/backward differences on the edges and the corresponding Jacobian is straightforward by exploiting the velocity constraint Jacobian.

The derived gradient information is involved and incorrect implementations badly influence the solution of the nonlinear optimization [14]. Therefore, we compared our analytical derived gradients to a centered finite difference approximation with a step size of 1e−6. A relative error of less than 8e−8 for the optimization derivative and of less than 6e−7 for the constraint derivatives proves the correct formulation and implementation of the gradients.

Note that h_{PKC} is different to the time step of the real-time low-level control h_{RT} (see Fig. 1). Therefore, we use a similar method to [15] to interpolate the NLP solution and to obtain the control values within the time $t \in [t_k, t_k + h_{PKC}]$:

$$\boldsymbol{u}(t) \approx \boldsymbol{u}_k + \frac{\boldsymbol{u}_{k+1} - \boldsymbol{u}_k}{t_{k+1} - t_k}(t - t_k) \tag{11a}$$

$$\dot{\boldsymbol{q}}(t) \approx \mathbf{f}_{\text{IK,k}} + \frac{\mathbf{f}_{\text{IK,k+1}} - \mathbf{f}_{\text{IK,k}}}{t_{k+1} - t_k}(t - t_k) \tag{11b}$$

$$\boldsymbol{q}(t) \approx \boldsymbol{q}_k + \mathbf{f}_{\text{IK,k}}(t - t_k) + \frac{\mathbf{f}_{\text{IK,k+1}} - \mathbf{f}_{\text{IK,k}}}{2(t_{k+1} - t_k)}(t - t_k)^2 \tag{11c}$$

b) Direct Shooting with Analytical Gradients: In contrast to the collocation method, the optimization variables for the shooting method consists of the nullspace controls only, i.e. $\boldsymbol{x} = \hat{\boldsymbol{u}}$. The joint positions $\hat{\boldsymbol{q}}$ are no independent optimization variables, but depend on $\hat{\boldsymbol{u}}$. We use an Euler forward integration to simulate the system:

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + h_{PKC}\ \mathbf{f}_{\text{IK,k}}(\boldsymbol{q}_k, \boldsymbol{u}_k) \tag{12}$$

(12) implicitly fulfills the collocation constraint in (7), which eliminates (4b). Every $\boldsymbol{u}_k$ influences the consecutive joint configurations $\boldsymbol{q}_{k+1} \dots \boldsymbol{q}_{N_{PKC}}$. To compute the analytical gradients of the optimization objective and the constraints, the following gradient is required:

$$\begin{aligned}\frac{\partial \boldsymbol{q}_k}{\partial \hat{\boldsymbol{u}}} &= \begin{bmatrix} \frac{\partial \boldsymbol{q}_k}{\partial \boldsymbol{u}_1} & \cdots & \frac{\partial \boldsymbol{q}_k}{\partial \boldsymbol{u}_{k-1}} & \frac{\partial \boldsymbol{q}_k}{\partial \boldsymbol{u}_k} & \cdots & \frac{\partial \boldsymbol{q}_k}{\partial \boldsymbol{u}_{N_{PKC}}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial \boldsymbol{q}_k}{\partial \boldsymbol{u}_1} & \cdots & \frac{\partial \boldsymbol{q}_k}{\partial \boldsymbol{u}_{k-1}} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}\end{aligned} \tag{13}$$

According to [16], we use the *sensivity differential equation* method to compute ${}_{q}\mathbf{S} = \frac{\partial \hat{\boldsymbol{q}}}{\partial \hat{\boldsymbol{u}}}$, i.e. the sensivity of $\hat{\boldsymbol{q}}$ w.r.t. $\hat{\boldsymbol{u}}$. A direct relation between $\boldsymbol{q}$ and $\boldsymbol{u}$ is not there, however, $\mathbf{f}_{\text{IK}}$ defines the relation between $\dot{\boldsymbol{q}}$ and $\boldsymbol{u}$ and thus we are able to compute the sensivity of $\dot{\boldsymbol{q}}$ w.r.t. $\boldsymbol{u}$, i.e. ${}_{\dot{q}}\mathbf{S} = \frac{\partial \hat{\dot{\boldsymbol{q}}}}{\partial \hat{\boldsymbol{u}}} = \frac{\partial \mathbf{f}_{\text{IK}}}{\partial \hat{\boldsymbol{u}}}$. According to (12), the sensivity of a time step k, ${}_{q}\mathbf{S}_k$, is determined by a forward integration of ${}_{\dot{q}}\mathbf{S}_k$ as given in (14). The initial value ${}_{q}\boldsymbol{S}_1 = \frac{\partial \boldsymbol{q}_1}{\partial \hat{\boldsymbol{u}}}$ is $\mathbf{0}$.

$$\begin{aligned}{}_{q}\boldsymbol{S}_{k+1} &= {}_{q}\boldsymbol{S}_k + h_{PCK}\ {}_{\dot{q}}\boldsymbol{S}_k \\ &= {}_{q}\boldsymbol{S}_k + h_{PCK}\ \left[\frac{\partial \mathbf{f}_{\text{IK}}}{\partial \boldsymbol{q}_k}\ {}_{q}\boldsymbol{S}_k + \frac{\partial \mathbf{f}_{\text{IK}}}{\partial \boldsymbol{u}_k} \frac{\partial \boldsymbol{u}_k}{\partial \hat{\boldsymbol{u}}}\right]\end{aligned} \tag{14}$$

The gradients of $\frac{\partial \mathbf{f}_{\text{IK}}}{\partial \boldsymbol{q}_k}$ are given in (10a) and $\frac{\partial \boldsymbol{u}_k}{\partial \hat{\boldsymbol{u}}}$ is straightforward:

$$\begin{aligned}\frac{\partial \boldsymbol{u}_k}{\partial \hat{\boldsymbol{u}}} &= \begin{bmatrix} \frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{u}_1} & \cdots & \frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{u}_k} & \cdots & \frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{u}_{N_{PKC}}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{0} & \cdots & \boldsymbol{I} & \cdots & \mathbf{0} \end{bmatrix}\end{aligned} \tag{15}$$

The derived sensivity ${}_{q}\boldsymbol{S}_k$ simplifies the formulation of the gradient of the objective function. (6a) still holds, however, the partial derivative w.r.t. $\hat{\boldsymbol{q}}$ vanishes and $\frac{\partial l_k}{\partial \boldsymbol{u}_k}$ is given by:

$$\frac{\partial l_k}{\partial \hat{\boldsymbol{u}}} = \frac{\partial l_k}{\partial \boldsymbol{q}_k}\ {}_{q}\boldsymbol{S}_k + \frac{\partial l_k}{\partial \boldsymbol{u}_k} \frac{\partial \boldsymbol{u}_k}{\partial \hat{\boldsymbol{u}}} \tag{16}$$

The actuator constraints (4c) directly depend on $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ and therefore $\frac{\partial \boldsymbol{g}}{\partial \hat{\boldsymbol{u}}}$ is composed of ${}_{q}\boldsymbol{S}_k$ for the position constraint and ${}_{\dot{q}}\boldsymbol{S}_k$ for the velocity constraint. The acceleration constraint is computed identically to the collocation method. Again, the gradient formulation is straightforward.

B. Direct Transcription on Acceleration Level

The computation on acceleration level, i.e. using (2) in (3b), is similar to the computation on velocity level. However, instead of the trapezoidal integration, we use the second-order Newmark scheme to discretize the differential IK on acceleration level. Note that for the direct collocation on acceleration level, the optimization vector is given by $\boldsymbol{x} = [\hat{\boldsymbol{u}}^T \quad \hat{\boldsymbol{q}}^T \quad \hat{\dot{\boldsymbol{q}}}^T]^T$. For the direct shooting method on acceleration level, the sensitivities are derived as:

$${}_{\ddot{q}}\boldsymbol{S}_k = \frac{\partial \mathbf{f}_{\text{IK}}}{\partial \boldsymbol{q}_k}\ {}_{q}\boldsymbol{S}_k + \frac{\partial \mathbf{f}_{\text{IK}}}{\partial \dot{\boldsymbol{q}}_k}\ {}_{\dot{q}}\boldsymbol{S}_k + \frac{\partial \mathbf{f}_{\text{IK,k}}}{\partial \boldsymbol{u}_k} \frac{\partial \boldsymbol{u}_k}{\partial \hat{\boldsymbol{u}}} \tag{17a}$$

$${}_{\dot{q}}\boldsymbol{S}_{k+1} = {}_{\dot{q}}\boldsymbol{S}_k + h_{PKC}\ {}_{\ddot{q}}\boldsymbol{S}_k \tag{17b}$$

$${}_{q}\boldsymbol{S}_{k+1} = {}_{q}\boldsymbol{S}_k + h_{PKC}\ {}_{\dot{q}}\boldsymbol{S}_k + \frac{1}{2}\ h_{PKC}^2\ {}_{\ddot{q}}\boldsymbol{S}_k \tag{17c}$$

C. Indirect Gradient Method

With the derived analytical gradients, we are able to implement the indirect conjugate gradient approach of [7]. For details, we refer to [7] and we only outline the main steps here. In contrast to the direct approaches, the first-order optimality conditions for (3) are derived instead of transcribing it to a NLP. Therefore, the *Hamiltonian* $H(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{\lambda})$ is formulated in which $\boldsymbol{\lambda}$ are the Lagrangian multipliers that describe the costate variables. With boundary condition (3c) and assuming a free final state for $\boldsymbol{q}$, i.e. $\boldsymbol{\lambda}(t_0 + T_{PKC}) = \mathbf{0}$, the first-order optimality conditions are decoupled:

$$H(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{\lambda}) = l(\boldsymbol{q}, \boldsymbol{u}) + \boldsymbol{\lambda}^T \mathbf{f}_{\text{IK}}(\boldsymbol{q}, \boldsymbol{u}) \tag{18}$$

$$\dot{\boldsymbol{q}} = \left(\frac{\partial H}{\partial \boldsymbol{\lambda}}\right)^T = \mathbf{f}_{\text{IK}}(\boldsymbol{q}, \boldsymbol{u}) \tag{19}$$

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial H}{\partial \boldsymbol{q}}\right)^T = -\left(\frac{\partial l}{\partial \boldsymbol{q}}\right)^T - \left(\frac{\partial \mathbf{f}_{\text{IK}}}{\partial \boldsymbol{q}}\right)^T \boldsymbol{\lambda} \tag{20}$$

Since the optimal control trajectory should minimize the Hamiltonian and $\boldsymbol{u}$ is unconstrained, the following additional condition must hold:

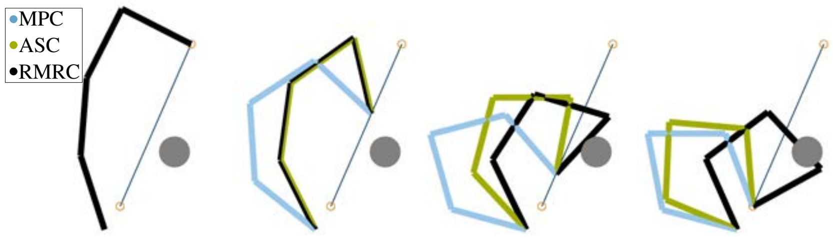
$$\mathbf{0} \overset{!}{=} \left(\frac{\partial H}{\partial \boldsymbol{u}}\right)^T = \left(\frac{\partial l}{\partial \boldsymbol{u}}\right)^T + \left(\frac{\partial \mathbf{f}_{\text{IK}}}{\partial \boldsymbol{u}}\right)^T \boldsymbol{\lambda} \tag{21}$$

Starting from an initial guess for $\boldsymbol{u}$, $\frac{\partial H}{\partial \boldsymbol{u}}$ is used to determine the search direction for the update of $\boldsymbol{u}$ to iteratively minimize the Hamiltonian. We discretize the problem formulation with the same time step as for the direct methods [16], i.e. h_{PKC}. To summarize, the optimal nullspace controls $\boldsymbol{u}$ are obtained by the following iterative approach:

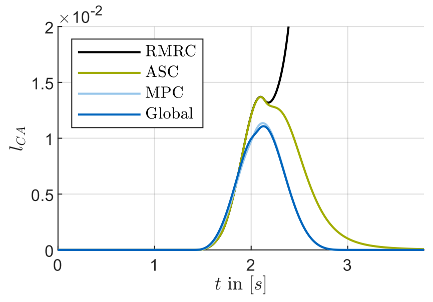
1) Initial guess $\boldsymbol{u}^0$.
2) Euler forward integration of $\boldsymbol{q}$ using (19) and (3c).
3) Euler backward integration of $\boldsymbol{\lambda}$ using (20) and $\boldsymbol{\lambda}(t_0 + T_{PKC}) = \mathbf{0}$.
4) Evaluation of the termination condition, i.e. the cost decrease of $L(\boldsymbol{q}, \boldsymbol{u})$ w.r.t. the previous iteration. Terminate, if condition fulfilled.
5) Calculate step direction $\boldsymbol{s}$ for the update of $\boldsymbol{u}$ with (21) and the conjugate gradient approach.
6) Calculate step size α for the update of $\boldsymbol{u}$ with the *Armijo* rule.
7) Update: $\boldsymbol{u} \mapsto \boldsymbol{u} + \alpha \boldsymbol{s}$.

IV. Results

Before implementing the *PKC* in our C++ framework for the *FRANKA EMIKA* robot, we did a simulation benchmark of the approaches proposed in Section III to derive the most efficient approach. We implemented a planar manipulator with four DoF (see Fig. 2) in *MATLAB* that we simulate on a 64-bit *Windows 10* PC with an *Intel i7-9700K* CPU at 3.6 GHz and 16 GB RAM. The task space trajectory $\boldsymbol{w}_d(t)$ is a point-to-point trajectory with quintic time scaling. We use the interior-point solver of *MATLAB* for the NLP resulting from the direct shooting and direct collocation approaches. In a preliminary study the interior-point solver performed better than the *MATLAB* implementation of a sequential quadratic programming solver w.r.t. computation time. However, we



(a) $t = 0$ s. (b) $t = 1.75$ s. (c) $t = 2.55$ s. (d) $t = 3.75$ s.



(e) Collision avoidance cost.

Fig. 2. Collision avoidance evaluation scenario. Fig. 2a-2d show the motion from top to down computed with different IK schemes. The *global* solution and the *MPC* solution result in the same motion and only the *MPC* approach is shown. Fig. 2e shows the collision avoidance cost over time for the four approaches.

TABLE I
AVERAGE COMPUTATION TIME AND STANDARD DEVIATION FOR THE DIRECT TRANSCRIPTION METHODS ON VELOCITY LEVEL

	Analytical	Forward Differences
Direct Shooting	10.96 ± 0.05 s	691.01 ± 5.66 s
Direct Collocation	20.57 ± 0.06 s	1366.20 ± 7.16 s

did not investigate or optimize the performance of the solvers for specific problem formulations. There might be specifically suited nonlinear optimization solvers for collocation methods that result in big sparse matrices or for shooting methods that result in small dense matrices.

As mentioned, h_{PKC} is larger than the time step for the simulation and later the real-time control that we both denote with h_{RT} (see Fig. 1a). The *PKC* may be applied for the whole trajectory, i.e. T_{PCK} is defined as the trajectory duration, or in the sense of MPC, i.e. T_{PCK} is a predefined horizon length. In the following, we denote the former approach with *global* and the latter with *MPC*. In the case of *MPC*, we enforce continuity by introducing an additional constraint, that forces the first control of the current horizon to be equal to the last control of the previous horizon.

A. Efficiency due to Analytical Gradients

Table I shows the average computation time and its standard deviation for the direct transcription methods on velocity level to solve the scenario in Fig. 2. Thereby, we use the *global* approach and we compare an implementation that uses the analytically derived gradients of Section III with an implementation that approximates the gradients numerically with forward differences. The collision avoidance cost l_{CA} is given in the Appendix and we refer to [7] for its derivation. We did five simulation runs each and the number of iterations is set to five. The discretized NLP has $N_{PKC} = 39$ time steps. Solving the NLP with the analytically derived gradients in Section III decreases the computation by 98.41 % for the shooting method and 98.49 % for the collocation method which demonstrates the requirement for analytical gradients in real-time applications.

Note that despite the analytically derived gradients the computation times are still considerably long for the given scenario. This is due to the implementation of the collision detection module for which we use the *MATLAB Robotics System Toolbox*. It implements the *Gilbert-Johnson-Keerthi (GJK)* distance algorithm that is less efficient than other approaches e.g. based on simplified geometries like swept sphere volumes.

B. Benchmark of Optimal Control Approaches

We determine the most efficient optimal control approach in a benchmark on velocity level. Therefore, we generate 50 random task space trajectories. The optimization objective incorporates joint limit avoidance l_{Jla}, comfort pose maintenance l_{Cmf} and a control penalty l_u with the formula given in the Appendix: $l(\boldsymbol{q}, \boldsymbol{u}) = 4l_{Jla} + 2l_{Cmf} + l_u$. Note that the indirect gradient method is unconstrained. Thus, to provide a fair comparison, for this scenario (3d) is also removed for the direct methods and instead l_{Jla} is added to the cost function. We compute the *global* solution for each trajectory and evaluate the cost improvement relative to the local redundancy resolution *Resolved Motion Rate Control (RMRC)* proposed in [17]. This method is standard for redundant robots and applications without secondary objectives: $\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{w}}_{d,eff}$:

$$L_{impr} = \frac{L_{PKC} - L_{RMRC}}{L_{RMRC}} \tag{22}$$

Further, we define the following relative trajectory computation time T_{rel}, as long trajectories require more computation time than short ones:

$$T_{rel} = \frac{\text{Computation Time}}{\text{Trajectory Duration}} \tag{23}$$

Table II shows the average and standard deviation for L_{impr} and T_{rel} for the 50 task space trajectories. All methods lead to similar results for L_{impr}, which is expected, as all method should provide the optimal solution. However, w.r.t. T_{rel} the shooting method shows the best performance. In $58\,\%$ of the trajectories, the shooting method was the fastest, the collocation method in $26\,\%$ and the indirect method in $16\,\%$. The investigations show the superior performance of the direct shooting method that we choose for the implementation of the *PKC*.

TABLE II
COMPARISON OF GLOBAL OPTIMIZATION WITH DIRECT AND INDIRECT METHODS ON VELOCITY LEVEL ACCORDING TO (22) AND (23)

	Collocation	Shooting	Indirect
L_{impr} [%]	-23.4 ± 19.35	-23.6 ± 19.49	-23.0 ± 19.32
T_{rel} [-]	3.06 ± 1.68	2.57 ± 1.01	4.24 ± 1.96

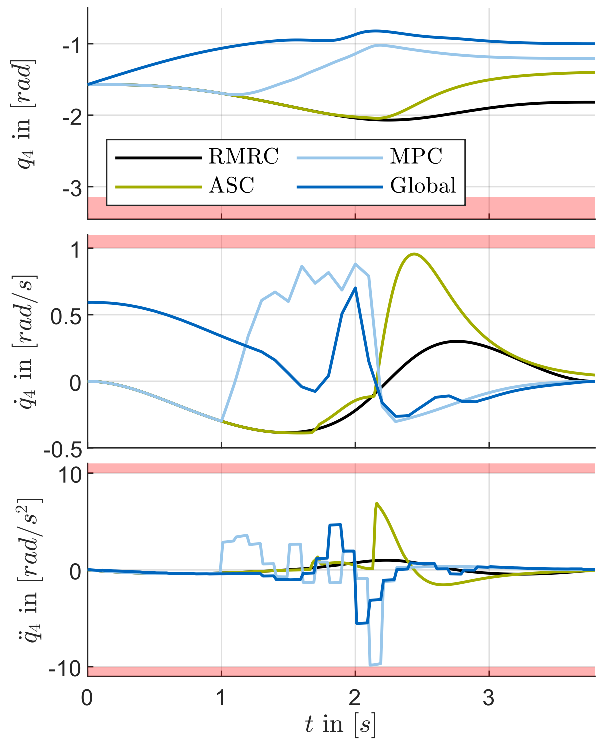


Fig. 3. Joint trajectories of the fourth joint for different IK schemes. Red patches visualize the symmetrically defined actuator limits.

C. Cost Improvement with the PKC

We demonstrate the benefit of our *PKC* in a benchmark with the local redundancy resolution schemes *RMRC* [17] and *ASC* [2] (see Section I) on velocity level. We use the same scenario as in Section IV-A. We compute a *global* solution with *PKC* and a *MPC* solution. In the latter case, we set $T_{PKC} = 1$ s, $h_{PKC} = 0.1$ s and $h_{RT} = 0.01$ s. The results are given in Fig. 2. Since *RMRC* does not include a secondary objective for collision avoidance, it passes through the obstacle. *ASC* can avoid the obstacle, but only reacts locally, when the manipulator is already near the obstacle. *PKC* detects the approaching obstacle early and therefore avoids it with great distance. Both PKC approaches (*global* and *MPC*) give similar results and both outperform *ASC*, which has higher costs.

Fig. 3 shows the corresponding joint trajectories for the fourth joint. The profiles show the early reaction of *PKC*. The *global PKC* approach reacts from the beginning. The *MPC* approach reacts from approximately $t = 1$ s on, i.e. when the obstacle is detected within the prediction horizon. The local *ASC* approach leads to a late reaction with discontinuous

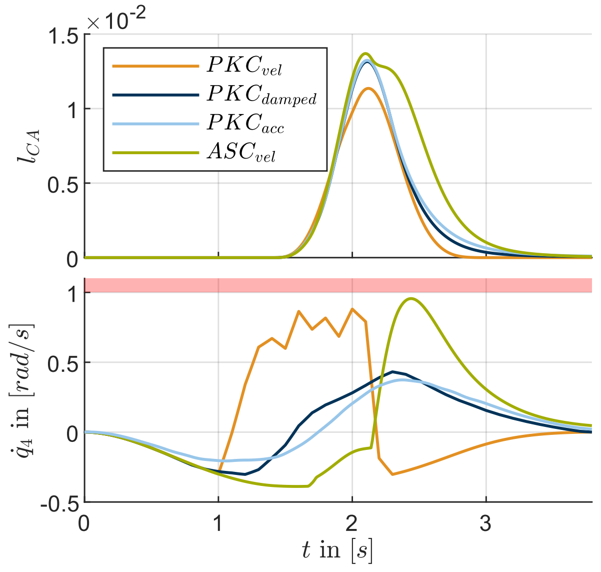


Fig. 4. Comparison of smoothing approaches. Collision avoidance cost is shown at the top and joint velocity profile of the fourth joint at the bottom.

velocity and acceleration. Though superior to *ASC*, *PKC* encounters high joint velocity values to avoid the obstacle and is not smooth which is discussed in the following.

D. Smooth Trajectories with Acceleration Level PKC

The limited smoothness of *PKC* is a consequence of the linear interpolation of $\boldsymbol{u}$ when computing the reference control values in (11). We investigated two approaches to smooth the trajectory. The first approach, the *damped velocity* approach, adds additional damping l_u to the optimization objective with the formula given in the Appendix: $l(\boldsymbol{q}, \boldsymbol{u}) = 20 l_{CA} + 0.1 l_u$. The second approach applies *PKC* on acceleration level as proposed in Section III. Fig. 4 shows the trajectory profiles for q_4 for the three approaches. With the *damped velocity* approach and the *PKC* on acceleration level, the robot keeps a smaller distance to the obstacle, since a further term is considered in the cost. Nevertheless, the behavior is still superior to *ASC*. The damped approach reduces the velocity over the whole trajectory, even though, the linear interpolated control still shows limited smoothness. Applying *PKC* on acceleration level gives the best result.

E. Experiments on a FRANKA EMIKA Robot

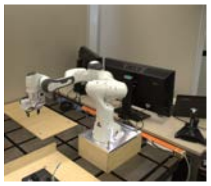
We implemented the *PKC* in our C++ control framework for a *FRANKA EMIKA* robot that runs on 64-bit *Ubuntu 18.04* using a 12-core *Intel i7-7800K* CPU at 3.7 GHz with 32 GB RAM. We use the *NLopt* library [14] to solve the nonlinear optimizations. However, *NLopt* does not provide an interface for a standard interior-point solver and therefore, in contrast to the *MATLAB* simulation, we use a sequential quadratic programming solver for the C++ framework.

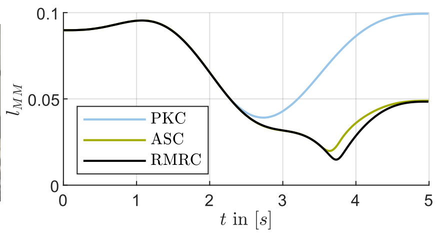
We use the multi-threaded implementation shown in Fig. 1b to compute the *PKC* horizon and the real-time control in parallel. Similar to [9], the respective control sequence is

(a) Initial Config.

(b) *RMRC*

(c) *ASC*


(d) *PKC*



(e) Manipulability measure.

Fig. 5. Fig. 5a shows the initial configuration, which is the same for all three IK-scheme. Fig. 5e shows the manipulability measure over time.

optimized one *PKC* cycle ahead in a different thread than the real-time control. In order to initialize the optimization a prediction of its initial state has to be made. This is done by measuring the current state and performing an Euler forward integration using the currently applied control.

We use the maximization of the manipulablity according to [18] as the application scenario. The formula for the cost $l = -l_{MM}$ is given in the Appendix. We use the *PKC* on velocity level with the *MPC* approach. Thereby, we solve a time horizon of $T_{PKC} = 1.5$ s with a step size of $h_{PKC} = 0.1$ s. In all MPC cycles, the optimization was finished before the next cycle started which satisfies the real-time requirement. Fig. 5e compares *PKC*, *RMRC* and *ASC* for the given scenario. Between $t = 3.5 - 4$ s, *RMRC* encounters a point with low manipulability, i.e. close to a singular configuration. *ASC* reacts locally, which allows only a slight improvement of the manipulablity. *PKC* is able to detect the point of low manipulablity as soon as it is within its receding horizon, i.e. 1.5 s beforehand. Thus, *PKC* increases the manipulablity measure significantly, outperforming the local schemes. The improvement of the manipulabilty is achieved by bending the manipulator's elbow. The videos of the evaluation scenarios in Fig. 2 and 5 are given at: https://youtu.be/8_OktWSjm68.

V. Conclusions

We derived an efficient implementation for our *PKC* based on a benchmark of three optimal control approaches - direct shoothing, direct collocation and indirect gradient method - and based on the derivation of the analytical gradients. Our implementation is a generic formulation for MPC of redundant kinematic structures. We showed the applicability to a wide range of optimization functions given in the Appendix. As for other nullspace projection approaches, objectives like collision avoidance are only optimized in the nullspace of an already computed task space trajectory. Therefore, infeasible solutions might be computed e.g. when the end-effector reference trajectory intersects an obstacle. Other techniques like generalized hierarchical control with priority transitions and task insertions/deletions as proposed in [19] might be used to overcome this issue.

In contrast to related work, we do not rely on a QP-based approach and thus, we are able to solve nonlinear objectives like collision avoidance or manipulability. Further, related work uses double integrators to smooth the obtained trajectory profiles. We derive the problem formulation also on acceleration level and demonstrate its benefit. We present the direct shooting method as the most efficient optimization approach in redundancy resolution. For future work, it might be interesting to investigate the performance of a multiple shooting method for the *PKC*. They are often proposed superior to single shooting in literature as changes early in the trajectory do not propagate to the end of the trajectory and parallel computing might speed up the computation time.

Our proposed *PKC* for redundancy resolution outperforms state-of-the-art local IK schemes w.r.t. secondary objectives and smoothness. Real-time capability is a key aspect in MPC. With our proposed multi-threaded computation approach, we achieve real-time capability which we demonstrate in experiments with a horizon length of 1.5 s and a step size of 0.1 s.

Appendix

Optimization objectives used for the evaluation of the *PKC*. The gradients of l_{Jla}, l_{Cmf} and l_u w.r.t. $\boldsymbol{q}$ and $\boldsymbol{u}$, required for the derivations in Section III, are straightforward. Note that we only give l_{Jla} for the upper limit and l_{soft} defines a soft joint limit. The lower limits are accordingly. $d_{i,j}$ is the distance between robot link i and obstacle j. For the gradients of l_{CA} and l_{MM}, we refer to [20] and [21] respectively.

$$l_{Jla} = \sum_{i=1}^{N_q} \frac{1}{2} \frac{(q_i - \overbrace{(q_{max,i} - q_{soft})}^{q_a})^2}{q_{soft}^2} \quad \text{if } q_i > q_a \quad (24)$$

$$l_{Cmf} = \frac{1}{2}(\boldsymbol{q} - \boldsymbol{q}_{Cmf})^T(\boldsymbol{q} - \boldsymbol{q}_{Cmf}) \quad (25)$$

$$l_{CA} = \sum_{i=1}^{N_b}\sum_{j=1}^{N_o} (d_a - d_{i,j}(\boldsymbol{q}))^3 \quad \text{if } d_{i,j}(\boldsymbol{q}) < d_a \quad (26)$$

$$l_u = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{u} \quad (27)$$

$$l_{MM} = \sqrt{det(\boldsymbol{J}\boldsymbol{J}^T)} \quad (28)$$

References

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, planning and control*, ser. Advanced textbooks in control and signal processing. London: Springer, 2010.

[2] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[3] F. Flacco and A. de Luca, “Discrete-time redundancy resolution at the velocity level with acceleration/torque optimization properties,” *Robotics and Autonomous Systems*, vol. 70, pp. 191–201, 2015.

[4] Y. Nakamura, *Advanced robotics: Redundancy and optimization*, ser. Addison-Wesley series in electrical and computer engineering: Control engineering. Reading, Mass.: Addison-Wesley, 1991.

[5] M. Faroni, M. Beschi, L. M. Tosatti, and A. Visioli, “A predictive approach to redundancy resolution for robot manipulators,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8975–8980, 2017.

[6] Y. Nakamura and H. Hanafusa, “Optimal redundancy control of robot manipulators,” *The International Journal of Robotics Research*, vol. 6, no. 1, pp. 32–42, 1987.

[7] C. Schütz *et al.*, “Predictive online inverse kinematics for redundant manipulators,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 5056–5061.

[8] R. Findeisen *et al.*, “Computational delay in nonlinear model predictive control,” *IFAC Proceedings Volumes*, vol. 37, no. 1, 2004.

[9] Y. Su, K. K. Tan, and T. H. Lee, “Computation delay compensation for real time implementation of robust model predictive control,” in *10th IEEE International Conference on Industrial Informatics (INDIN), 2012*. Piscataway, NJ: IEEE, 2012, pp. 242–247.

[10] A.-C. Hildebrandt *et al.*, “Kinematic optimization for bipedal robots: a framework for real-time collision avoidance,” *Autonomous Robots*, vol. 43, no. 5, pp. 1187–1205, 2019.

[11] A. Zube, “Cartesian nonlinear model predictive control of redundant manipulators considering obstacles,” in *2015 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2015, pp. 137–142.

[12] K. Ayusawa *et al.*, “Predictive inverse kinematics: optimizing future trajectory through implicit time integration and future jacobian estimation,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 566–573.

[13] H. Cao *et al.*, “Predictive damped inverse kinematics for redundant and underactuated robotic systems,” in *2020 IEEE 16th International Conference on Control & Automation (ICCA)*. IEEE, 2020.

[14] S. G. Johnson. (2021) Nlopt. [Online]. Available: https://nlopt.readthedocs.io/en/latest/NLopt_Introduction/

[15] M. Kelly, “An introduction to trajectory optimization: How to do your own direct collocation,” *SIAM Review*, vol. 59, no. 4, 2017.

[16] M. Gerdts, *Optimal Control of ODEs and DAEs*, 1st ed., ser. De Gruyter textbook. s.l.: Walter de Gruyter GmbH Co.KG, 2012.

[17] D. Whitney, “Resolved motion rate control of manipulators and human prostheses,” *IEEE Transactions on Man Machine Systems*, vol. 10, no. 2, pp. 47–53, 1969.

[18] T. Yoshikawa, “Manipulability of robotic mechanisms,” *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[19] M. Liu, Y. Tan, and V. Padois, “Generalized hierarchical control,” *Autonomous Robots*, vol. 40, no. 1, pp. 17–31, Jan 2016. [Online]. Available: https://doi.org/10.1007/s10514-015-9436-1

[20] C. Schütz, “Trajectory planning for redundant manipulators,” Ph.D. Thesis, Technical University of Munich, Munich, 2017.

[21] J. Baur, J. Pfaff, H. Ulbrich, and T. Villgrattner, “Design and development of a redundant modular multipurpose agricultural manipulator,” in *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2012, pp. 823–830.