A novel QP-based kinematic redundancy resolution method with joint constraints satisfaction

ŁUKASZ WOLIŃSKI, and MAREK WOJTYRA

Institute of Aeronautics and Applied Mechanics, Warsaw University of Technology, Warsaw, Nowowiejska 24, 00-665, Poland (e-mail: lwolinski@meil.pw.edu.pl, mwojtyra@meil.pw.edu.pl)

Corresponding author: Łukasz Woliński (e-mail: lwolinski@meil.pw.edu.pl).

This research was supported by the National Science Centre (Poland) grant no. 2018/29/B/ST8/00374.

ABSTRACT Kinematically redundant manipulators are advantageous for their increased dexterity and ability to fulfill some secondary requirements along with their primary task to follow the prescribed trajectory. The redundancy results in a non-trivial inverse kinematics problem (IK). Standard methods of redundancy resolution are based on the pseudoinverse of the Jacobian matrix of the manipulator. The excessive degrees of freedom are utilized to perform secondary tasks that are projected onto the null space of the Jacobian matrix. However, the joint constraints satisfaction cannot be easily ensured in this way—even though methods that account for the joint position limits are known, the constraints for joint velocities and especially accelerations are not straightforward to include. In this paper, a novel redundancy resolution method based on a less common quadratic programming (QP) approach is described. The proposed velocity-level IK method allows fulfillment of the joint constraints at the position, velocity, and acceleration levels. In the derived formulas, accelerations instead of the usual velocities are used—the discretized joint state equations allow the use of joint accelerations as decision variables in the QP problem. The developed algorithm is investigated in a series of numerical tests in which the kinematics of the KUKA LWR4+ redundant 7-DOF manipulator is exploited. The newly elaborated QP-based IK method is firstly compared with the classic pseudoinverse-based approach and then tested for its ability to keep the joint accelerations within the prescribed bounds. The prospects of the proposed approach are discussed in the concluding section.

INDEX TERMS inverse kinematics, optimization, quadratic programming, redundant manipulators, redundant robots

I. INTRODUCTION

A TYPICAL industrial robot has as many degrees of freedom as necessary for a task it was chosen for. However, to work in human-centered unstructured environments, a robotic manipulator shall have a versatile design. As can be observed in a human arm, a redundant structure increases dexterity. Similarly, redundant manipulators can easily handle singularities, avoid obstacles or keep their joints away from the limits [1], [2].

Usually, the desired end effector trajectory is planned in the Cartesian space, whereas the robot’s controller requires the joint space trajectory. Therefore, the inverse kinematics problem has to be solved. Since redundant robots have more degrees of freedom than the end effector trajectory, the solution of inverse kinematics (IK) is not trivial. Standard methods are based on the pseudoinverse of the Jacobian matrix of the manipulator [1], [2]. The redundant degrees of freedom can be utilized to perform tasks additional to the end effector trajectory tracking. These secondary tasks are projected onto the null space of the Jacobian matrix [3]–[7].

The classic methods of redundancy resolution are discussed in more detail in Sec. II of this article, in order to establish a ground for their comparison with the newly proposed approach.

However, the joint constraints satisfaction cannot be ensured in the pseudoinverse-based IK—attaining this goal needs additional effort. In [8], [9], fulfilment of the joint position limits is treated as the main task; moreover, an enhancement consisting of modification of the Jacobian matrix singular values so that it never loses rank is proposed. To
II. CLASSIC INVERSE KINEMATICS SOLUTION

A. PRELIMINARIES

Let \( n \) be the dimension of the manipulator’s joint space, and \( m \) be the dimension of the task space. The task space variables in the vector \( x \in \mathbb{R}^m \) are related to the joint space variables \( q \in \mathbb{R}^n \) by the equation:

\[
x = f(q),
\]

(1)

where \( f(q) \to \mathbb{R}^m \) is a function describing the forward kinematics of the manipulator. For a full 6 degree-of-freedom end effector task, the vector \( x \in \mathbb{R}^6 \) can consist of the end effector’s position \( r \in \mathbb{R}^3 \) and orientation \( \phi \in \mathbb{R}^3 \) (given by example by Euler angles):

\[
x = \begin{bmatrix} r \\ \phi \end{bmatrix},
\]

(2)

whereas the forward kinematics function \( f(q) \) can be written as:

\[
f(q) = \begin{bmatrix} f_r(q) \\ f_\phi(q) \end{bmatrix},
\]

(3)

where \( f_r(q) \) describes the position and \( f_\phi(q) \) describes the orientation of the end effector.

In a general case, the end effector task might have less degrees of freedom \((m < 6)\), and not all elements of \( r \) and \( \phi \) might be necessary in \( x \). Accordingly, not all elements of \( f(q) \) given by (3) would be necessary in such a case.

Differentiating (1) with respect to time leads to the following equation:

\[
\dot{x} = J_a \dot{q},
\]

(4)

where:

- \( J_a = J_a(q) \in \mathbb{R}^{m \times n} \) is the analytical task Jacobian of the manipulator,
- \( m < n \) for the manipulator to be redundant for a given task of the size \( m \),
- \( \dot{q} \in \mathbb{R}^n \) is the vector of joint velocities,
- \( x \in \mathbb{R}^m \) is the vector of time derivatives of the task variables \( x \).

The analytical task Jacobian matrix \( J_a \) is computed as [1]:

\[
J_a(q) = \frac{\partial f(q)}{\partial q}.
\]

(5)

In practice, it is often more convenient to use the geometric Jacobian \( J = J(q) \in \mathbb{R}^{m \times n} \), since it describes the relationship between the joint velocities \( \dot{q} \) and task velocities \( v \in \mathbb{R}^m \):

\[
v = J \dot{q},
\]

(6)

instead of relationship between \( \dot{q} \) and \( \dot{x} \). Again, for a full 6 degree-of-freedom task, the vector \( v \in \mathbb{R}^6 \) can consist of the end effector’s translational velocity \( \dot{r} \in \mathbb{R}^3 \) and angular velocity \( \omega \in \mathbb{R}^3 \):

\[
v = \begin{bmatrix} \dot{r} \\ \omega \end{bmatrix},
\]

(7)

and the angular velocity \( \omega \) can be expressed as:

\[
\omega = T(\phi) \dot{\phi},
\]

(8)

where \( T(\phi) \) is a transformation matrix that corresponds to the used method of parameterization of orientation.

However, in a case of \( m < 6 \) not all elements of \( \dot{r} \) and \( \omega \) might be necessary in \( v \). For example, in a pure translational end effector task, the task variables vector \( x \) and the task velocity vector \( v \) are given by \( x = r \) and \( v = \dot{r} \), respectively. It is worth noting that in such a case the analytical and geometric Jacobians are identical: \( J_a = J \). On the other hand, when \( m = 6 \), there exists a following relationship between \( J \) and \( J_a \):

\[
J = \begin{bmatrix} I & 0 \\ 0 & T(\phi) \end{bmatrix} J_a.
\]

(9)

The well-known recursive algorithms for computing the geometric Jacobian \( J \) can be found in [1], [2].

Moreover:

- \( \mathbb{R}(J) \subseteq \mathbb{R}^m \) is the range space of \( J(q) \), and it contains the end effector task velocities \( v \) that can be generated by the joint velocities \( \dot{q} \) for a given configuration \( q \).
\( \mathcal{N}(\mathbf{J}) \subseteq \mathbb{R}^n \) is the null space of \( \mathbf{J}(\mathbf{q}) \), and it contains the joint velocities \( \mathbf{q} \) that do not produce any task velocity, for a given configuration \( \mathbf{q} \).

- \( \rho \leq m \) is the rank of \( \mathbf{J} \),
- \( \dim (\mathcal{R}(\mathbf{J})) = \rho \),
- \( \dim (\mathcal{N}(\mathbf{J})) = n - \rho \), and, because of the redundancy, \( \dim (\mathcal{N}(\mathbf{J})) \geq 0 \),
- \( \dim (\mathcal{R}(\mathbf{J})) + \dim (\mathcal{N}(\mathbf{J})) = n \).

For redundant manipulators, the Jacobian \( \mathbf{J}(\mathbf{q}) \) is not square (because \( m < n \)), and the straightforward solution to the inverse kinematics problem:

\[
\mathbf{q} = \mathbf{J}^{-1} \mathbf{v}
\]

is not available. Instead, to solve Eq. (6) with respect to \( \mathbf{q} \), a Moore-Penrose pseudoinverse \( \mathbf{J}^{\#} \) of the Jacobian matrix \( \mathbf{J} \) might be used [1], [2]:

\[
\mathbf{q} = \mathbf{J}^{\#} \mathbf{v}.
\]

Assuming that all of the manipulator joints are of the same type (e.g., rotational or translational), the solution (11) minimizes the Euclidean norm of the joint velocity vector \( \| \mathbf{q} \| \). However, other solutions are possible. In particular, the ability of the redundant manipulator to perform self motions—enabled by the existence of \( \mathcal{N}(\mathbf{J}) \)—might be utilized to achieve additional objectives along the main task, given by \( \mathbf{v} \).

Therefore, the minimum norm solution (11) can be expanded to the following form [1], [2], [15], [16]:

\[
\mathbf{q} = \mathbf{J}^{\#} \mathbf{v} + \mathbf{q}^{NS},
\]

where \( \mathbf{q}^{NS} \in \mathcal{N}(\mathbf{J}) \) is the null space velocity. In other words, \( \mathbf{q}^{NS} \) does not affect the end effector task velocity \( \mathbf{v} \), since:

\[
\mathbf{J} \mathbf{q}^{NS} = \mathbf{0}.
\]

Usually, the null space velocity is expressed as [1], [2]:

\[
\mathbf{q}^{NS} = \mathbf{P} \mathbf{q}^{JS},
\]

where \( \mathbf{q}^{JS} \) is an arbitrarily chosen vector that represents the constraint of the additional task specified directly in the joint space, and \( \mathbf{P} \) is the matrix which projects the vector \( \mathbf{q}^{JS} \) onto the null space of the Jacobian \( \mathcal{N}(\mathbf{J}) \) [1], [2], [17]:

\[
\mathbf{P} = \mathbf{I}_n - \mathbf{J}^{\#} \mathbf{J},
\]

and \( \mathbf{JP} = \mathbf{0} \).

Substituting (15) and (14) into (12) results in:

\[
\mathbf{q} = \mathbf{J}^{\#} \mathbf{v} + (\mathbf{I} - \mathbf{J}^{\#} \mathbf{J}) \mathbf{q}^{JS},
\]

which is the solution to the inverse kinematics problem, known as the redundancy resolution at the velocity level.

### B. SECONDARY TASKS

One of the methods to select the vector \( \mathbf{q}^{JS} \) for the solution (16) is to employ the local optimization [1], [2], [16], [18]:

\[
\mathbf{q}^{JS} = k_H \nabla_\mathbf{q} H(\mathbf{q}),
\]

where \( H(\mathbf{q}) \) is a scalar configuration-dependent objective function, \( \nabla_\mathbf{q} H(\mathbf{q}) \) is its gradient and \( k_H \) is a scalar gain coefficient. Since (17) is projected onto the null space of the Jacobian, this approach is called the projected gradient method. It can be used for singularity avoidance [1], [2], increasing the dynamic manipulability [19], obstacle avoidance [2], [20], or keeping the joints close to the middle of their range [1], [2].

The ability of the redundant manipulator to avoid the collisions with obstacles in the workspace can be also utilized by adding a task defined by its Jacobian matrix and velocity vector [21], [22]. However, specifying multiple additional tasks alongside the main one might result in conflicting task situations. Therefore, a task priority strategy is needed to ensure that each lower priority task is satisfied only in the null space of the higher priority tasks [3], [4], [6], [22], [23].

The multiple tasks are usually defined as:

\[
\mathbf{v}^i = \mathbf{J}^i \mathbf{q}, \quad i = 1, \ldots, l,
\]

where \( \mathbf{v}^i \in \mathbb{R}^{m_i} \) is the \( i \)-th task velocity, \( \mathbf{J}^i \in \mathbb{R}^{m_i \times n} \) is the \( i \)-th task Jacobian, \( l \) is the number of tasks, \( m_i \) is the \( i \)-th task dimension, and \( \sum_{i=1}^l m_i \leq n \). The order of task priority is decreasing, i.e., the task \( i + 1 \) has a lower priority than the task \( i \).

A recursive solution for (18) is given in [3] (called in [6] a standard method):

\[
\mathbf{q}^i = \mathbf{q}^{i-1} + (\mathbf{J}^i \mathbf{P}_A^{i-1})^\# (\mathbf{v}^i - \mathbf{J}^i \mathbf{q}^{i-1}),
\]

for \( i = 1, \ldots, l \), and where \( \mathbf{q}^0 = \mathbf{0} \) and \( \mathbf{P}_A^0 = \mathbf{I} \). The matrix \( \mathbf{P}_A^i \) [3], [6]:

\[
\mathbf{P}_A^i = \mathbf{I} - \mathbf{J}^i \mathbf{J}^i\mathbf{P}_A^{i-1} = \mathbf{P}_A^{i-1} - (\mathbf{J}^i \mathbf{P}_A^{i-1}) \mathbf{J}^i \mathbf{P}_A^{i-1},
\]

is the projector onto the null space of the augmented Jacobian of the first \( i \) tasks:

\[
\mathbf{J}_A^i = \begin{bmatrix} \mathbf{J}_1^i \\ \vdots \\ \mathbf{J}_l^i \end{bmatrix}.
\]

### C. JOINT SPACE SOLUTION

To obtain the joint coordinates in any given moment \( t \), the solution (16) has to be integrated over time (starting from \( t_0 \)):

\[
\mathbf{q}(t) = \mathbf{q}(t_0) + \int_{t_0}^t \mathbf{\dot{q}}(\tau) d\tau.
\]

However, for implementation in the control system, usually equations in discrete form are required. The integral (22)
can be approximated by using the forward rectangular rule, resulting in the explicit Euler method:

\[ \mathbf{q}_{k+1} = \mathbf{q}_0 + \sum_{h=0}^{k} \dot{\mathbf{q}}_h \Delta t, \]  

(23)

which can be expressed recursively as:

\[ \mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_k \Delta t, \]  

(24)

where \( \mathbf{q}_0 = \mathbf{q}(t_0) \) is the initial joint configuration, \( \mathbf{q}_k = \mathbf{q}(t_k) \) is the sample at time instant \( t_k \), \( \Delta t \) is the time step, and \( k = 1, 2, \ldots, N \). The joint velocity \( \dot{\mathbf{q}}_k \) is just a discrete form of (16):

\[ \dot{\mathbf{q}}_k = \mathbf{J}^\#(\mathbf{q}_k) \mathbf{v}_k + \mathbf{P}(\mathbf{q}_k) \dot{\mathbf{q}}^{JS}_k, \]  

(25)

where \( \dot{\mathbf{q}}_k = \dot{\mathbf{q}}(t_k), \mathbf{v}_k = \mathbf{v}(t_k), \dot{\mathbf{q}}^{JS}_k = \dot{\mathbf{q}}^{JS}(t_k), \) and \( \mathbf{P}(\mathbf{q}_k) \) (recall (15)):

\[ \mathbf{P}(\mathbf{q}_k) = \mathbf{I} - \mathbf{J}^\#(\mathbf{q}_k) \mathbf{J}(\mathbf{q}_k). \]  

(26)

It should be noted that the numerical integration introduces some error in each step which accumulates over time. The closed-loop inverse kinematics (CLIK) approach overcomes this problem by treating the inverse kinematics as a feedback control problem and including the end effector error in the solution [1], [2], [9], [24]–[30].

Up to this point, the end effector velocity \( \mathbf{v}_k \) was treated as the desired velocity:

\[ \mathbf{v}_k = \mathbf{v}_{d,k}. \]  

(27)

However, in the CLIK solution, the end effector velocity \( \mathbf{v}_k \) takes the following form:

\[ \mathbf{v}_k = \mathbf{v}_{d,k} + \mathbf{K}\mathbf{e}_k, \]  

(28)

where \( \mathbf{K} \in \mathbb{R}^{m \times n} \) is a positive definite gain matrix, and \( \mathbf{e}_k \) is the end effector trajectory error in the \( k \)-th step:

\[ \mathbf{e}_k = \begin{bmatrix} \mathbf{r}_{d,k} - \mathbf{f}_r(\mathbf{q}_k) \\ \mathbf{e}_{\phi,k} \end{bmatrix}, \]  

(29)

where \( \mathbf{r}_{d,k} \) is the desired end effector position in the \( k \)-th step, and \( \mathbf{f}_r(\mathbf{q}_k) \) is the position part of the forward kinematics function \( \mathbf{f}(\mathbf{q}_k) \) (given by (3)) in the \( k \)-th step. The orientation error \( \mathbf{e}_{\phi,k} \) is much harder to define and depends on the chosen orientation representation. A couple of ways of expressing the orientation error \( \mathbf{e}_{\phi,k} \) are presented in [2].

It should also be noted that in many works (28) is simplified to a form [9], [10], [24]–[27], [29]:

\[ \mathbf{v}_k = \mathbf{v}_{d,k} + \mathbf{K}\mathbf{e}_k, \]  

(30)

where \( \mathbf{K} > 0 \) is a scalar gain coefficient. Stability of the CLIK algorithms based on (30) and using the explicit integration methods was investigated in [27], leading to the following condition for \( \mathbf{K} \):

\[ K < \frac{2}{\Delta t}. \]  

(31)

III. PROPOSED METHOD

A. JOINT STATE EQUATIONS

As can be seen in Sec. II, the pseudoinverse-based IK does not account for joint constraints—in particular for velocity and acceleration bounds. To directly bound the acceleration \( \ddot{\mathbf{q}}_k \) at each time \( t_k \), the joint state equations have to be formulated at the acceleration level:

\[
\begin{aligned}
q_{k+1} &= q_k + q\Delta t + \frac{1}{2}\ddot{q}(\Delta t)^2, \\
\ddot{q}_{k+1} &= \ddot{q}_k + \dddot{q}(\Delta t).
\end{aligned}
\]  

(32)

Equation (32) can be written as:

\[ \mathbf{z}_{k+1} = \mathbf{A}\mathbf{z}_k + \mathbf{B}\mathbf{u}_k, \]  

(33)

where the state \( \mathbf{z}_k \in \mathbb{R}^{2n} \) is:

\[ \mathbf{z}_k = \begin{bmatrix} \mathbf{q}_k \\ \dot{\mathbf{q}}_k \end{bmatrix}, \]  

(34)

and the joint acceleration is the control vector: \( \mathbf{u}_k = \dddot{\mathbf{q}}_k \). The state matrices \( \mathbf{A} \in \mathbb{R}^{2n \times 2n} \) and \( \mathbf{B} \in \mathbb{R}^{2n \times n} \) are defined as:

\[ \mathbf{A} = \begin{bmatrix} \mathbf{I}_n & \Delta \mathbf{I}\mathbf{I}_n \\ 0_{n \times n} & \mathbf{I}_n \end{bmatrix}, \]  

(35)

\[ \mathbf{B} = \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \mathbf{I}_n \\ \Delta \mathbf{I}\mathbf{I}_n \end{bmatrix}, \]  

(36)

and are constant.

Equation (33) can be divided into two parts:

\[
\begin{aligned}
\begin{bmatrix} \mathbf{q}_{k+1} \\ \dot{\mathbf{q}}_{k+1} \end{bmatrix} &= \begin{bmatrix} \mathbf{A}_0 & \mathbf{B}_0 \end{bmatrix} \begin{bmatrix} \mathbf{z}_k \\ \mathbf{u}_k \end{bmatrix}, \\
\end{aligned}
\]  

(37)

where the matrices \( \mathbf{A}_0, \mathbf{B}_0, \mathbf{A}_1 \) and \( \mathbf{B}_1 \) are:

\[ \mathbf{A}_0 = \begin{bmatrix} \mathbf{I} & 0 \end{bmatrix} \mathbf{A}, \]  

(38)

\[ \mathbf{B}_0 = \begin{bmatrix} \mathbf{I} & 0 \end{bmatrix} \mathbf{B}, \]  

\[ \mathbf{A}_1 = \begin{bmatrix} 0 & \mathbf{I} \end{bmatrix} \mathbf{A}, \]  

\[ \mathbf{B}_1 = \begin{bmatrix} 0 & \mathbf{I} \end{bmatrix} \mathbf{B}, \]  

and \( \mathbf{I} \in \mathbb{R}^{n \times n} \) and \( 0 \in \mathbb{R}^{n \times n} \).

The relationship between the joint and end effector velocity vectors—given by (6)—can be written for step \( k + 1 \) as:

\[ \mathbf{J}(\mathbf{q}_{k+1}) : \dot{\mathbf{q}}_{k+1} = \mathbf{v}_{k+1}. \]  

(39)

Combining (39) with (37), the following result is obtained:

\[ \mathbf{J}_{k+1} : (\mathbf{A}_1\mathbf{z}_k + \mathbf{B}_1\mathbf{u}_k) = \mathbf{v}_{k+1}, \]  

(40)

where:

\[ \mathbf{J}_{k+1} = \mathbf{J}(\mathbf{q}_{k+1}) = \mathbf{J}(\mathbf{A}_0\mathbf{z}_k + \mathbf{B}_0\mathbf{u}_k). \]  

(41)

Equation (40) is nonlinear in terms of \( \mathbf{u}_k \)—because of \( \mathbf{J}_{k+1} \) given by (41). It can be used to formulate and solve a nonlinear optimization problem; some examples of nonlinear optimization utilization in solving IK include [20], [31], [32]. On the other hand, an approximation of \( \mathbf{J}_{k+1} \) based on the solution from the previous step can be used to formulate a linear kinematic constraint. In that approach, an
approximation \( \dot{q}_{k+1} \) of the future joint position \( q_{k+1} \) can be computed as:
\[
\dot{q}_{k+1} = A_0 \ddot{z}_k + B_0 \ddot{u}_k,
\]
where \( \ddot{u}_k = u^*_{k-1} \) is the solution from the previous step. Then, by using (42), \( J_{k+1} \) can be approximated as \( \dot{J}_{k+1} \):
\[
\dot{J}_{k+1} = J(\dot{q}_{k+1}).
\]

Using (43) in (40), a linear equation in terms of \( u_k \) is obtained:
\[
\dot{J}_{k+1} \cdot (A_1 \ddot{z}_k + B_1 u_k) = v_{k+1}.
\]

In [33], a similar approach—utilizing the previous step solution to simplify the kinematic equations—was successfully applied to the joint space trajectory generation.

B. JOINT CONSTRAINTS

The equality constraint (44) describes the end effector trajectory tracking task. The novel IK formulation requires also the inequality constraints to represent the joint position (\( q_{\text{min}} \) and \( q_{\text{max}} \)), velocity (\( \dot{q}_{\text{min}} \) and \( \dot{q}_{\text{max}} \)) and acceleration limits (\( \ddot{q}_{\text{min}} \) and \( \ddot{q}_{\text{max}} \)). Although the constant acceleration bounds represent purely kinematic limits, they are often used in literature as approximation of torque constraints [14], [34]–[38].

No bounds on acceleration might cause the discontinuities in joint velocities, as observed in [14]. However, in [14], the joint acceleration constraints are included indirectly in additional velocity bounds.

The joint position constraints can be written as:
\[
q_{\text{min}} \leq q_k + \dot{q}_k \Delta t + \frac{1}{2} \ddot{q}_k (\Delta t)^2 \leq q_{\text{max}},
\]
where the operator \( \leq \) used with vectors means an element-wise operation. The joint velocity bounds are:
\[
\dot{q}_{\text{min}} \leq \dot{q}_k \leq \dot{q}_{\text{max}},
\]
and the acceleration limits can be included as:
\[
\ddot{q}_{\text{min}} \leq \ddot{q}_k \leq \ddot{q}_{\text{max}}.
\]

Another fact to bear in mind is that the robot joints cannot stop in an instant. Applying the maximum deceleration \( \ddot{q}_{\text{max},j} < 0 \) to the \( j \)-th joint for some time \( t \geq t_k \) results in the following position and velocity:
\[
\begin{align*}
q_j(t) &= q_{j,k} + \dot{q}_{j,k}(t - t_k) + \frac{1}{2} \ddot{q}_{\text{min},j}(t - t_k)^2, \\
\dot{q}_j(t) &= \dot{q}_{j,k} + \ddot{q}_{\text{min},j}(t - t_k)
\end{align*}
\]

When the \( j \)-th joint reaches its limit \( q_{\text{max},j} \), its velocity should be less or equal to zero. Suppose this event happens at some \( t = t^* > t_k \):
\[
\begin{align*}
q_j(t^*) &= q_{j,k} + \dot{q}_{j,k}(t^* - t_k) + \frac{1}{2} \ddot{q}_{\text{min},j}(t^* - t_k)^2 = q_{\text{max},j}, \\
\dot{q}_j(t^*) &= \dot{q}_{j,k} + \ddot{q}_{\text{min},j}(t^* - t_k) \leq 0
\end{align*}
\]

From (49), it follows that the \( j \)-th joint velocity at \( t_k \) is upper bounded by [14], [37], [38]:
\[
\dot{q}_{j,k} \leq \sqrt{-2 \ddot{q}_{\text{min},j} (q_{\text{max},j} - q_{j,k})}.
\]

Similarly, from the case of applying the maximum acceleration \( \ddot{q}_{\text{max},j} > 0 \) to the \( j \)-th joint to avoid crossing the \( q_{\text{min},j} \) limit, it follows that the lower velocity bound at \( t_k \) is [14], [37], [38]:
\[
\dot{q}_{j,k} \geq -\sqrt{2 \ddot{q}_{\text{max},j} (q_{j,k} - q_{\text{min},j})}.
\]

The joint constraints (45)–(51) can also be transformed into a form dependent on \( u_k \). In that regard, the joint position limits (45) become:
\[
q_{\text{min}} - A_0 \ddot{z}_k \leq B_0 u_k \leq q_{\text{max}} - A_0 \ddot{z}_k,
\]
while the joint velocity bounds (46) are now:
\[
\dot{q}_{\text{min}} - A_1 \ddot{z}_k \leq B_1 u_k \leq \dot{q}_{\text{max}} - A_1 \ddot{z}_k,
\]
and the acceleration limits (47) can be included as:
\[
\ddot{q}_{\text{min}} \leq u_k \leq \ddot{q}_{\text{max}}.
\]

Finally, the constraints (50) and (51) for \( \dot{q}_{k+1} \) can be transformed, by using (32) and (42), into:
\[
\rho_{\text{min}} \leq u_k \leq \rho_{\text{max}},
\]
where:
\[
\begin{align*}
\rho_{\text{min},j} &= -\frac{2 \ddot{q}_{\text{max},j} (q_{k+1} - q_{\text{min},j}) - \dot{q}_{j,k}}{\Delta t}, \\
\rho_{\text{max},j} &= \frac{2 \ddot{q}_{\text{min},j} (q_{\text{max},j} - q_{k+1,j}) - \dot{q}_{j,k}}{\Delta t}
\end{align*}
\]
for \( j = 1, \ldots, n \).

C. SECONDARY TASKS IN THE GOAL FUNCTION

As described in section II-B, a redundant manipulator can handle multiple tasks thanks to the utilization of the null space projection. In this section, a different approach is presented, based on a multiobjective optimization formalism [39].

The secondary tasks can be thought of as parts of a multi-objective goal function to be minimized:
\[
\min_q \frac{1}{\tau} \sum_{i=1}^{l} (J^i q - v^i)^T W_i (J^i q - v^i),
\]
where \( l \) is the number of null space tasks, \( W_i \in \mathbb{R}^{m_i \times m_i} \) are positive-definite weight matrices, and \( m_i \) is the size of the \( i \)-th task. By setting the different values of weights, the task hierarchy can be defined, with the most important task having the highest weight [39]. In general, deciding on the hierarchy of tasks of various kinds might not be trivial.

The task Jacobian \( J^i \in \mathbb{R}^{m \times m} \) and task velocity \( v^i \in \mathbb{R}^m \) can represent any secondary task defined as (18). It is worth pointing out that these tasks usually do not directly depend on time but rather on the joint configuration \( q \). An example of such a task is the obstacle avoidance [21], [22].

The joint space tasks can be implemented by setting \( J^i = I \) and \( v^i = q^{I_S,i} \), where \( q^{I_S,i} \) is given by (17).

As described above, the secondary tasks can be defined in different ways, each having different units—for example,
joint task space velocity will have units of rad/s, whereas the Cartesian space: m/s. Since in (57) the terms are all summed, the units shall be consistent. Therefore, the role of \( W_i \) matrices is not only to set the task priorities with weights but also to ensure the consistency of the units. To this end, the units of elements of \( W_i \) can be chosen in such a way that each term in the sum in (57) becomes unitless.

Since the equality constraint (44) and inequality constraints (52)–(55) are formulated in terms of \( u_k \), the goal function in (57) also has to be redefined. A new minimization problem—with the joint velocity vector replacing the joint velocity as the optimization variable—can be written as:

\[
\min_{\mathbf{u}_k} \frac{1}{2} \sum_{i=1}^{l} \left( \mathbf{J}_{k+1}^i \cdot (\mathbf{A}_1 \mathbf{z}_k + \mathbf{B}_1 \mathbf{u}_k) - \mathbf{v}_k^i \right)^T W_i \cdot \mathbf{J}_{k+1}^i \cdot (\mathbf{A}_1 \mathbf{z}_k + \mathbf{B}_1 \mathbf{u}_k) - \mathbf{v}_k^i \right) + \frac{1}{2} \gamma_{t+1} \mathbf{u}_k^T \mathbf{u}_k,
\]

where the joint velocity seen in (57) was replaced by \( \hat{q}_{k+1} \) from (37). The terms \( \mathbf{J}_{k+1}^i \) and \( \mathbf{v}_k^i \) are the approximations of the \( i \)-th task Jacobian and velocity, respectively, computed for step \( k + 1 \) with the use of \( \hat{q}_{k+1} \), given by (42). If the task velocity \( \mathbf{v}_k^i \) does not depend on the joint configuration \( \mathbf{q} \) but rather on time, then \( \mathbf{v}_k^i = \mathbf{v}_k^{i+1} = \mathbf{v}(t_{k+1}) \).

The cost function in (58) includes the additional goal of minimizing the control action \( \mathbf{u}_k \), introduced with the weight \( \gamma_{t+1} > 0 \). It should be pointed out that the joint velocity minimization task is still possible by defining the \( i \)-th task as: \( \hat{q}_{k+1}^i = \mathbf{I}, \mathbf{v}_k = \mathbf{0} \) and \( \mathbf{W}_i = \gamma_{t+1} \mathbf{I} \), where \( \gamma_{t+1} > 0 \).

Equation (58) can be further regrouped into:

\[
\min_{\mathbf{u}_k} \frac{1}{2} \sum_{i=1}^{l} \mathbf{u}_k^T \mathbf{B}_1^T \mathbf{J}_{k+1}^i \mathbf{J}_{k+1}^i T \mathbf{W}_i \mathbf{J}_{k+1}^i T \mathbf{B}_1 \mathbf{u}_k + \frac{1}{2} \gamma_{t+1} \mathbf{u}_k^T \mathbf{u}_k + \sum_{i=1}^{l} \mathbf{u}_k^T \mathbf{B}_1^T \mathbf{J}_{k+1}^i \mathbf{J}_{k+1}^i T \mathbf{W}_i \left( \mathbf{J}_{k+1}^i \mathbf{A}_1 \mathbf{z}_k - \mathbf{v}_k^i \right).
\]

The constant terms of (58) were omitted in (59), because they do not affect the optimization process.

### D. QUADRATIC PROGRAMMING FORMULATION OF THE INVERSE KINEMATICS

Gathering the equations derived in sections III-A, III-B, and III-C, the QP formulation of IK can be expressed as:

\[
\min_{\mathbf{u}_k} \frac{1}{2} \mathbf{\xi}^T \mathbf{H} \mathbf{\xi} + \mathbf{\xi}^T \mathbf{h}
\]

s.t. \( \mathbf{\Phi}_{\text{eq}} \mathbf{\xi} = \mathbf{b}_{\text{eq}}, \mathbf{b}_{\text{min}}^\mathbf{\xi} \leq \mathbf{\xi} \leq \mathbf{b}_{\text{max}}^\mathbf{\xi}, \)

where the vector of optimization variables \( \mathbf{\xi} \in \mathbb{R}^n \) is:

\[
\mathbf{\xi} = \mathbf{u}_k
\]

whereas the components of the Hessian matrix \( \mathbf{H} \in \mathbb{R}^{n \times n} \) and gradient vector \( \mathbf{h} \in \mathbb{R}^n \) of the goal function are, respectively (compare with (59)):

\[
\mathbf{H} = \sum_{i=1}^{l} \mathbf{B}_1^T \left( \mathbf{J}_{k+1}^i \right)^T \mathbf{W}_i \left( \mathbf{J}_{k+1}^i \right)^T \mathbf{B}_1 + \gamma_{t+1} \mathbf{I},
\]

and:

\[
\mathbf{h} = \sum_{i=1}^{l} \mathbf{B}_1^T \left( \mathbf{J}_{k+1}^i \right)^T \mathbf{W}_i \left( \mathbf{J}_{k+1}^i \mathbf{A}_1 \mathbf{z}_k - \mathbf{v}_k^i \right). \tag{63}
\]

It should be pointed out that due to the joint acceleration being the decision variable (61), the matrices \( \mathbf{H} \) and \( \mathbf{h} \), given by (62) and (63), respectively, take a different form than in [2], [14].

Based on (44), the matrix of the equality constraints set \( \mathbf{\Phi}_{\text{eq}} \in \mathbb{R}^{m \times n} \) is:

\[
\mathbf{\Phi}_{\text{eq}} = \mathbf{J}_{k+1} \mathbf{B}_1, \tag{64}
\]

whereas the equality constraints vector \( \mathbf{b}_{\text{eq}} \in \mathbb{R}^m \) is:

\[
\mathbf{b}_{\text{eq}} = \mathbf{v}_{k+1} - \mathbf{J}_{k+1} \mathbf{A}_1 \mathbf{z}_k. \tag{65}
\]

Based on (52)–(55), the matrix of the inequality constraints set \( \mathbf{\Phi}_{\text{eq}} \in \mathbb{R}^{4n \times n} \) is:

\[
\mathbf{\Phi} = \begin{bmatrix} \mathbf{B}_0 \\ \mathbf{B}_1 \\ \mathbf{I}_n \\ \mathbf{I}_n \end{bmatrix}, \tag{66}
\]

whereas the vector of the lower bounds \( \mathbf{b}_{\text{min}}^\mathbf{\xi} \in \mathbb{R}^{4n} \) is:

\[
\mathbf{b}_{\text{min}}^\mathbf{\xi} = \begin{bmatrix} \mathbf{q}_{\text{min}} - \mathbf{A}_0 \mathbf{z}_k \\ \mathbf{q}_{\text{min}} - \mathbf{A}_1 \mathbf{z}_k \\ \mathbf{q}_{\text{min}} \\ \rho_{\text{min}} \end{bmatrix}, \tag{67}
\]

and the vector of the upper bounds \( \mathbf{b}_{\text{max}}^\mathbf{\xi} \in \mathbb{R}^{4n} \) is:

\[
\mathbf{b}_{\text{max}}^\mathbf{\xi} = \begin{bmatrix} \mathbf{q}_{\text{max}} - \mathbf{A}_0 \mathbf{z}_k \\ \mathbf{q}_{\text{max}} - \mathbf{A}_1 \mathbf{z}_k \\ \mathbf{q}_{\text{max}} \\ \rho_{\text{max}} \end{bmatrix}. \tag{68}
\]

The solution of the IK problem is obtained for the time interval \( \langle t_0, t_{\text{end}} \rangle \), with a given time step \( \Delta t \), using Algorithm 1.

Lastly, to include the CLIK in the QP formulation (60), the end effector velocity \( \mathbf{v}_{k+1} \) can be defined similarly as in (30):

\[
\mathbf{v}_{k+1} = \mathbf{v}_{d,k+1} + \mathbf{K} \hat{\mathbf{e}}_{k+1}, \tag{69}
\]

where the end effector error \( \hat{\mathbf{e}}_{k+1} \) is a function of \( \mathbf{x}_{d,k+1} \) and \( \mathbf{f}(\hat{q}_{k+1}) \), similarly as in (29). For a pure translational task, it is:

\[
\hat{\mathbf{e}}_{k+1} = \mathbf{r}_{d,k+1} - \mathbf{f}(\hat{q}_{k+1}). \tag{70}
\]

### E. DISCUSSION

Obviously, the idea of the QP formulation of the IK problem is not new. The pseudoinverse-based solution (11) can be obtained analytically by minimizing the quadratic goal function \( \frac{1}{2} \mathbf{q}^T \mathbf{q} \) with an equality constraint (6), and no constraints on joint positions, velocities or accelerations [2].

An efficient QP-based method is proposed in [40]. It can be divided into two steps. First, the reduced joint velocity \( \hat{q}_p \) is obtained from (6) where the Jacobian \( \mathbf{J} \) is replaced with a square matrix \( \mathbf{J}_p \) with only linearly independent columns.
Algorithm 1 QP IK solver

(Initialization)
Set \( t := t_0 \), \( u_k := 0 \), \( z_k := [q(t_0)^T \ q'(t_0)^T]^T \)
Set the joint limits \( q_{\text{min}}, q_{\text{max}}, \dot{q}_{\text{min}}, \dot{q}_{\text{max}}, \ddot{q}_{\text{min}}, \ddot{q}_{\text{max}} \)
Compute constant matrices \( A \) (35), \( B \) (36), and \( A_0, B_0, A_1, B_1 \) (38)

(Main loop)
while \( t < t_{\text{end}} \) do
  Compute \( \dot{q}_{k+1} \) (42), \( \dot{J}_{k+1} \) (43)
  Obtain \( \dot{v}_{k+1} \) from the end effector trajectory generator
  If secondary tasks are present, obtain \( \dot{J}_{k+1}^* \) and \( \ddot{v}_{k+1}^* \)
  Formulate the QP problem (60)
  Set the initial guess \( \dot{\xi}_{\text{init}} := \dot{u}_k \)
  Solve the QP problem (60) and obtain \( \dot{\xi}^* \)
  Set \( \dot{u}_k := \dot{\xi}^* \)
  Compute new state \( z_{k+1} := A z_k + B u_k \) (33)
  Set \( u_{k+1} := \dot{u}_k \)
  Set \( t := t + \Delta t \)
  Set \( k := k + 1 \)
end while

of \( J \). Then, the QP problem is formulated to obtain the free variables of the joint velocity which are necessary to compute the full vector \( \dot{q} \); conveniently, the equality constraint (6) is no longer needed, the inequality constraints for joint positions and velocities are included, and the number of the optimization variables is reduced to the number of redundant degrees of freedom \( n - m \). However, calculating the partial solution to (6) and combining it with the optimal solution to a reduced QP problem might no longer be necessary—due to the abundance of computing power of modern computers, a full QP problem for a typical 7-degree-of-freedom manipulator can be easily solved.

In [41], the variable joint velocity limits in the QP IK problem for redundant robots are explored. It should be noted, however, that these constraints are derived for one specific manipulator. Meanwhile, the constraints (50) and (51) (or (55) and (56)) are general for manipulators with revolute joints.

The QP formulation of IK problem is not limited to manipulators. It can also be used in mobile robots such as humanoids [42] or hexapods [43].

All of the discussed methods implement constraints only on the joint positions \( q \) and velocities \( \dot{q} \). The main advantage of the proposed method is that it directly includes the joint acceleration constraints. A QP-based method that constrains the joint accelerations is also proposed in [44]; however, the inverse kinematics problem is formulated at the acceleration level. On the other hand, in our method, even though the joint acceleration \( \ddot{q} \) (or \( u \)) is the optimization variable in (60), the manipulator kinematics is defined at the velocity level, represented by the constraint (6).

The different subtasks are weighted in the goal function which simplifies the algorithm; however, in such a formulation, a tradeoff between secondary tasks might occur. In [42], a sequential QP (SQP) formulation is proposed that also handles inequality subtasks (e.g. keeping a part of the manipulator below or above a specified height). A modification of Algorithm 1 will be explored in the future to allow for prioritizing the subtasks in the SQP framework.

IV. RESULTS

A. SIMULATION SETUP

To compare the IK solver proposed in Sec. III with the classic pseudoinverse solution described in Sec. II, several numerical simulations were performed. The test subject was a kinematic model of the KUKA LWR 4+ 7-degree-of-freedom manipulator [45]–[47]. The results of four simulations are presented in the next sections, whereas this section is devoted to the description of the simulation setup.

The local coordinate frames were attached to the links of the LWR 4+ manipulator using the modified Denavit-Hartenberg parameters [48], shown in Tab. 1, whereas the end effector position in the last link frame was set as: 

\[ \begin{bmatrix} 0.1 & 0 & 0.078 \end{bmatrix}^T m. \]

In each scenario, the main task consisted of tracking the desired end effector position, i.e., a 3-degree-of-freedom task. Each trajectory consisted of several linear segments and was generated with the method described in detail in Appendix A.

The time step of the simulation was set as \( \Delta t = 0.005 \) s. The classic IK solution was obtained with (24) and (25), and the gain for the CLIK in (30) was \( K = 100 \frac{1}{s} \). In the proposed approach, the Algorithm 1 was used, and the CLIK gain in (69) was set to \( K = 50 \frac{1}{s} \).

The simulations were performed in MATLAB R2020b on a computer with the Intel(R) Core(TM) i7-6500U CPU with two 2.6 GHz cores and 16 GB RAM. At each time step, the solution to (60) was obtained by MATLAB function \texttt{quadprog}, utilizing the \texttt{active-set} algorithm with default settings [49].

B. SCENARIO 1A

The start configuration was equal to:

\[ q_0 = [0^\circ \ 0^\circ \ 0^\circ \ -45^\circ \ 0^\circ \ 45^\circ \ 0^\circ]^T, \]

corresponding to the end effector position \( X_0 \), and then the path continued through the points \( X_1, X_2, X_3, X_4 \), and again \( X_1 \), shown in Table 2.

The resulting path consisted of 5 linear segments, as shown in Fig. 1. The time to traverse each segment was set equal
The average computation time in each step is 0.14 ms for the classic IK and 1.49 ms for the proposed method.

D. SCENARIO 2

The start configuration $q_0$ was chosen as:

$$q_0 = \begin{bmatrix} -45^\circ & 0^\circ & 0^\circ & 45^\circ & 0^\circ & -45^\circ & 0^\circ \end{bmatrix}^T,$$

corresponding to the end effector position $X_0$. From $X_0$, the path continued through the points $X_1, X_2, X_3, X_4$, and again $X_1$, shown in Table 3.

| TABLE 3. Scenario 2: Points of the path |
|-----------------------------------------|
| $x$ | $y$ | $z$ (m) |
|---|---|---|
| $X_0$ | 0.49 | 0.632 |
| $X_1$ | 0.3 | 0.1 |
| $X_2$ | 0.3 | 0.1 |
| $X_3$ | 0.3 | 0.1 |
| $X_4$ | 0.3 | 0.1 |

The resulting path consisted of 5 linear segments, as shown in Fig. 7. The time to traverse each segment was set equal to $T_1 = 2$ s, $T_2 = 1.5$ s, $T_3 = 1.2$ s, $T_4 = 1.2$ s, and $T_5 = 1.2$ s, resulting in total motion time $t_{end} = 7.1$ s. The segment times were chosen in such a way that shows the limits of the classic IK approach. The trajectory generation is described in detail in Appendix A.

The joint limits were set as follows: $q_{\text{min}} = -100I_{7 \times 1}$, $q_{\text{max}} = 100I_{7 \times 1}$, $\dot{q}_{\text{min}} = -150I_{7 \times 1}$, $\dot{q}_{\text{max}} = 150I_{7 \times 1}$, $\ddot{q}_{\text{min}} = -350I_{7 \times 1}$, and $\ddot{q}_{\text{max}} = 350I_{7 \times 1}$.

For the proposed method, the joint velocity minimization and joint acceleration minimization tasks were active with the same weights as in Sec. IV-C: $\gamma_1 = 10^6 s^2$ and $\gamma_2 = 10^4 s^4$.

The joint positions are shown in Fig. 8. In the classic IK solution, the 1st joint exceeds its limit, whereas the proposed method satisfies the position constraints. The same happens at the velocity level, shown in Fig. 9. The joint accelerations are shown in Fig. 10. Again, the classic IK method results in the constraint violation, whereas in the proposed method the constraints are satisfied.

The end effector error (71) is shown in Fig. 11. The maximum error of the classic IK is $e = 6.82 \cdot 10^{-5}$ m, and the maximum error of the proposed method is $e = 1.95 \cdot 10^{-6}$ m. The average computation time of the classic IK is 0.11 ms and of the proposed method 1.29 ms.

E. SCENARIO 3

In this scenario, the start configuration $q_0$ was:

$$q_0 = \begin{bmatrix} 0^\circ & 0^\circ & 0^\circ & -45^\circ & 0^\circ & 45^\circ & 0^\circ \end{bmatrix}^T,$$

corresponding to the end effector position $X_0$, and then the path continued through the points $X_1, X_2, X_3, X_4$, and again $X_1$, shown in Table 4.

The resulting path consisted of 5 linear segments, as shown in Fig. 12. The time to traverse each segment was set equal to $T_1 = 1.1$ s, $T_2 = 0.75$ s, $T_3 = 2.4$ s, $T_4 = 0.6$ s, and $T_5 = 1.13 s$.
FIGURE 1. Scenario 1A: Desired end effector path (left) and velocity (right).

FIGURE 2. Scenario 1A: Joint position difference between the classic IK solution and the proposed method.

FIGURE 3. Scenario 1A: End effector error for the classic IK (left) and the proposed method (right).

1.1 s, resulting in total motion time $t_{end} = 5.95$ seconds. The trajectory generation is described in detail in Appendix A.

The joint limits were set to: $q_{\text{min}} = -120 \mathbf{I}_{7 \times 1}$, $q_{\text{max}} = 120 \mathbf{I}_{7 \times 1}$, $q_{\text{min}} = -150 \mathbf{I}_{7 \times 1}$, $q_{\text{max}} = 150 \mathbf{I}_{7 \times 1}$. 

Again, similarly as in Sec. IV-C and IV-D, the weights of the joint velocity and acceleration minimization tasks in the proposed method were set to $\gamma_1 = 10^6 s^2$ and $\gamma_2 = 10 s^4$, respectively.

The joint positions are within their bounds in both the classic IK and the proposed method, as shown in Fig. 13. However, as can be seen in Fig. 14, the joint velocity for the

---

**FIGURE 4.** Scenario 1B: Joint position for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

**FIGURE 5.** Scenario 1B: Joint velocity for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

**FIGURE 6.** Scenario 1B: Joint acceleration for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.
2nd axis exceeds its limit in the classic IK solution, whereas in the proposed method it just touches its bound. At the acceleration level, pictured in Fig. 15, the joint constraints violations are again the domain of the classic IK, whereas the proposed method satisfies the limit.

The end effector error (71) is shown in Fig. 16. The
Woliński, Wojtyra: A novel QP-based kinematic redundancy resolution method with joint constraints satisfaction

FIGURE 10. Scenario 2: Joint acceleration for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

FIGURE 11. Scenario 2: End effector error for the classic IK (left) and the proposed method (right).

FIGURE 12. Scenario 3: Desired end effector path (left) and velocity (right).

maximum error of the classic IK is $e = 1.63 \cdot 10^{-4} \ m$, and the maximum error of the proposed method is $e = 8.58 \cdot 10^{-6}$. The average computation time of the classic IK is $0.09 \ ms$ and of the proposed method $1.18 \ ms$.

V. CONCLUSION

A novel QP-based redundancy resolution method developed and tested in this work has proven to be capable of keeping the joint accelerations within the prescribed limits, which
distinguishes it from the previously proposed methods of this kind, being useful in maintaining constraints up to the velocity level only. The QP approach itself is an attractive alternative to classic pseudoinverse-based methods, as it offers greater flexibility in the fulfilment of secondary tasks.

During the simulations, to emphasize the practicability of the proposed algorithm, the kinematics of a 7-DOF industrial robot was exploited. Even though the computation time is an
order of magnitude bigger than for the classic IK method, it is still—on average—under 2 ms in the tested scenarios. A low-level C/C++ implementation utilizing fast algebraic libraries (like BLAS and LAPACK [50], [51]) and an efficient QP solver (for example qpOASES [52]–[54]), should be able to easily solve the IK problem for a typical redundant manipulator under real-time operation conditions, using a medium performance computer.

Of course, the newly proposed method is not a faultless approach that provides an ultimate solution to all redundancy resolution problems. Clearly, a constrained optimization problem might not always be feasible—the required motion may be beyond the robot’s capabilities, and a solution that fulfills the constraints may be non-existent. In other words, if the desired trajectory becomes too demanding, it cannot be achieved within the imposed joint limits. In such a case, it might be necessary to relax requirements for the end effector velocities or accelerations while maintaining the desired shape of the trajectory. Technically, a solution to that problem can come in the form of trajectory scaling—the end effector will stay on the path, but the motion duration will be extended [55]. Therefore, a natural continuation of the work presented here is to focus on extending the QP-based approach to trajectory scaling problems. Our preliminary research on this problem indicates that the QP-based framework of redundancy resolution is well suited for trajectory scaling applications.

APPENDIX A TRAJECTORY GENERATION

For each $i$-th segment, the end effector position $r(t)$ and velocity $v(t)$ are defined as:

$$
\begin{align*}
    r(t) &= X_{i-1} + \mu_i s_i(\varphi) \\
    v(t) &= \mu_i s_i(\varphi)
\end{align*}
$$

(72)

where the unit vector $\mu_i$ along the direction of motion is:

$$
\mu_i = \frac{X_i - X_{i-1}}{\| X_i - X_{i-1} \|},
$$

(73)

whereas the time law $s_i(\varphi)$ is defined as:

$$
s_i(\varphi) = \frac{6}{T_i^2} \varphi^5 - \frac{15}{T_i^3} \varphi^4 + \frac{10}{T_i^4} \varphi^3,
$$

(74)

and the local time $\varphi$, measured from the start to the end of the segment, is:

$$
\varphi(t) = \begin{cases} 
    t & \text{for } t \in (0, T_1) \\
    t - \sum_{k=1}^{i} T_k & \text{for } t \in (T_1, t_{end})
\end{cases},
$$

(75)

where the total duration of motion $t_{end} = \sum_{i=1}^{N_T} T_i$, whereas $T_i$ is the duration of the $i$-th segment, and $N_T$ is the number of segments.

REFERENCES

[1] S. Chiaverini, G. Oriolo, and A. A. Maciejewski. Redundant Robots, pages 221–242. Springer International Publishing, Cham, 2016.
[2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. Differential Kinematics and Statics, pages 105–160. Springer London, London, 2009.
[3] B. Siciliano and J. J. E. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In Advanced Robotics, 1991. ‘Robots in Unstructured Environments’. 91 ICAR., Fifth International Conference on, volume 2, pages 1211–1216, June 1991.
[4] S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. IEEE Transactions on Robotics and Automation, 13(3):398–410, Jun 1997.
[5] N. Mansard, O. Khatib, and A. Kheddar. A Unified Approach to Integrate Unilateral Constraints in the Stack of Tasks. IEEE Transactions on Robotics, 25(3):670–685, June 2009.
[6] F. Flacco and A. De Luca. A reverse priority approach to multi-task control of redundant robots. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2421–2427, Sept 2014.
S. Moe, G. Antonelli, A. R. Teel, K. Y. Pettersen, and J. Schirmpf. Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results. Frontiers in Robotics and AI, 7, 2020.

A. Colomé and C. Torres. Redundant inverse kinematics: Experimental comparative review and two enhancements. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5333–5340, Oct 2012.

A. Colomé and C. Torres. Closed-loop inverse kinematics for redundant robots: Comparative assessment and Two Enhancements. IEEE/ASME Transactions on Mechatronics, 20(2):944–955, Apr 2015.

N. Mansard, A. Remazeilles, and F. Chaumette. Continuity of varying-feature-set control laws. IEEE Transactions on Automatic Control, 54(11):2493–2505, Nov 2009.

T. F. Chan and R. V. Dubey. A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. IEEE Transactions on Robotics and Automation, 11(2):286–292, Apr 1995.

D. Raunhardt and R. Boulic. Progressive clamping. In Proceedings 2007 IEEE International Conference on Robotics and Automation, pages 4414–4419, April 2007.

P. Jiang, S. Huang, J. Xiang, and M. Z. Q. Chen. Iterative successive projection: A novel continuous approach for the task-based control of redundant robots. IEEE Access, 7:25347–25358, 2019.

F. Flacco, A. De Luca, and O. Khatib. Control of redundant robots under hard joint constraints: Saturation in the null space. IEEE Transactions on Robotics, 31(6):637–656, June 2015.

A. Liégeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. IEEE Transactions on Systems, Man, and Cybernetics, 7(12):868–871, Dec 1977.

H. Zghal, R. V. Dubey, and J. A. Euler. Efficient gradient projection optimization for manipulators with multiple degrees of redundancy. In Proceedings., IEEE International Conference on Robotics and Automation, pages 1006–1011 vol.2, May 1990.

A. Ben-Israel and T. N. E. Greville. Generalized inverses: Theory and applications. Springer-Verlag New York, New York, NY, 2003.

D. N. Nemchev. Redundancy resolution through local optimization: A review. Journal of Robotic Systems, 6(6):769–798, 1989.

T. Yoshikawa. Dynamic manipulability of robot manipulators. In Proceedings., IEEE International Conference on Robotics and Automation, pages 1006–1011, May 1990.

A. Colomé and C. Torres. Closed-loop inverse kinematics for redundant robots: Comparative assessment and two enhancements. IEEE/ASME Transactions on Mechatronics, 20(2):944–955, April 2015.

D. Di Vito, C. Natale, and G. Antonelli. A comparison ofamped least squares algorithms for inverse kinematics of robot manipulators. IFAC-PapersOnLine, 50(1):6869 – 6874, 2017. 29th IFAC World Congress.

A. Reiter, A. Müller, and H. Gattringer. Inverse kinematics in minimum-time trajectory planning for kinematically redundant manipulators. In IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, pages 6873–6878, Oct 2016.

A. Reiter, A. Müller, and H. Gattringer. On higher order inverse kinematics methods in time-optimal trajectory planning for kinematically redundant manipulators. IEEE Transactions on Industrial Informatics, 14(4):1681–1690, 2018.

M. Faroni, M. Beschi, C. Guarino Lo Bianco, and A. Visioli. Predictive joint trajectory scaling for manipulators with kinodynamic constraints. Control Engineering Practice, 95:104264, 2020.

T. Kröger, A. Tomièczek, and F. M. Wahl. Towards On-Line Trajectory Computation. In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 736–741, Oct 2006.

X. Broqueire, D. Sidobre, and I. Herrera-Aguilar. Soft motion trajectory planning for service manipulator robots. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2808–2809, Sept 2008.

F. Flacco and A. De Luca. Optimal redundancy resolution with task scaling under hard bounds. In 2013 IEEE International Conference on Robotics and Automation, pages 3969–3975, May 2013.

A. D. Prete. Joint position and velocity bounds in discrete-time acceleration/torque control of robot manipulators. IEEE Robotics and Automation Letters, 3(1):281–288, Jan 2018.

J. Branke, K. Deb, K. Miettinen, and R. Slowinski. Multiobjective optimization. Springer-Verlag Berlin Heidelberg, 2008.

Y. G. Kim, T. H. Chou, and M. X. Cheng. A dynamic programming algorithm for resolving manipulator redundancy-the compact QP method. In Proceedings 1992 IEEE International Conference on Robotics and Automation, volume 1, pages 508–513, 1992.

Z. Zhang and Y. Zhang. Variable joint-velocity limits of redundant robot manipulators handled by quadratic programming. IEEE/ASME Transactions on Mechatronics, 18(2):674–686, 2013.

O. Kanoun, F. Lamiraux, and P.-B. Wieber. Kinematic control of redundant manipulators: Guaranteeing the task-priority framework to inequality task. IEEE Transactions on Robotics, 27(4):785–792, 2011.

D. Khudre and R. Powell. Quadratic programming for inverse kinematics control of a hexapod robot with inequality constraints. In 2016 International Conference on Robotics: Current Trends and Future Challenges (RCTFC), pages 1–5, 2016.

S. Cocuzza, I. Pretto, and S. Debei. Least-Squares-Based Reaction Control of a Space Place Manipulator. Journal of Guidance, Control, and Dynamics, 35(3):976–986, 2012.

R. Bischoff, J. Kurth, G. Schreiber, R. Koepp, A. Albu-Schaeffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger. The KUKA-DLR Lightweight Robot arm—a new reference platform for robotics research and manufacturing. In ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), pages 1–8, June 2010.

KUKA. Lightweight Robot 4+ Specification, Version: Spez LBR 4+ V2en, 2010.

A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger. The KUKA-DLR Lightweight Robot arm—a new reference platform for robotics research and manufacturing. In ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), pages 1–8, June 2010.

KUKA. Lightweight Robot 4+ Specification, Version: Spez LBR 4+ V2en, 2010.

A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger. The KUKA-DLR Lightweight Robot arm—a new reference platform for robotics research and manufacturing. In ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), pages 1–8, June 2010.

KUKA. Lightweight Robot 4+ Specification, Version: Spez LBR 4+ V2en, 2010.

A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger. The KUKA-DLR Lightweight Robot arm—a new reference platform for robotics research and manufacturing. In ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), pages 1–8, June 2010.

KUKA. Lightweight Robot 4+ Specification, Version: Spez LBR 4+ V2en, 2010.

A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger. The KUKA-DLR Lightweight Robot arm—a new reference platform for robotics research and manufacturing. In ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), pages 1–8, June 2010.

KUKA. Lightweight Robot 4+ Specification, Version: Spez LBR 4+ V2en, 2010.
ŁUKASZ WOLIŃSKI received the B.S. and M.S. degrees in robotics and control from Warsaw University of Technology, Warsaw, Poland, in 2012 and 2013, respectively. He is currently pursuing the Ph.D. degree in automation, electronic and electrical engineering at Warsaw University of Technology, Warsaw, Poland.

In 2015, he completed a research visit at JKU Linz. In 2018, he worked for the company RT Robotics in Warsaw. Since 2020, he has been a research and teaching assistant with the Faculty of Power and Aeronautical Engineering, Warsaw University of Technology.

MAREK WOJTYRA received the Ph.D. degree in mechanics and the D.Sc. degree in automation and robotics from Warsaw University of Technology, Warsaw, Poland, in 2000 and 2013, respectively.

He is currently an Associate Professor with the Institute of Aeronautics and Applied Mechanics, WUT, where he is also the Head of the Division of Theory of Machines and Robots. He published a book and more than 80 journal and conference articles. His research interests include multibody systems and robotics with its applications.

Dr. Wojtyra serves as a reviewer for several journals.

---

[53] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. qpOASES: a parametric active-set algorithm for quadratic programming. Mathematical Programming Computation, 6(4):327–363, 2014.

[54] H. J. Ferreau, A. Potschka, and C. Kirches. qpOASES. http://www.qpOASES.org/, 2007–2017.

[55] M. Wojtyra and Ł. Woliński. Proposition of on-line velocity scaling algorithm for task space trajectories. In Gentiane Venture, Jorge Solis, Yukio Takeda, and Atsushi Konno, editors, ROMANSY 23 - Robot Design, Dynamics and Control, pages 423–431, Cham, 2021. Springer International Publishing.