Received February 10, 2022, accepted April 8, 2022, date of publication April 14, 2022, date of current version April 21, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3167403

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, 00-665 Warsaw, Poland

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

This work was supported in part by the National Science Centre, Poland, under Grant 2018/29/B/ST8/00374; and in part by the Warsaw University of Technology (WUT) Internal Grant for Research in Automation, Electronic, and Electrical Engineering.

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 fulfilment 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,

The associate editor coordinating the review of this manuscript and approving it for publication was Sunith Bandaru.

an enhancement consisting of modification of the Jacobian matrix singular values so that it never loses rank is proposed. To guarantee the continuity of the solution, the pseudoinverse operator from [5], [10] is utilized. Another approach for avoiding joint limits, based on a weighted least norm solution and minimizing unnecessary self-motion, is presented in [11]. In [12], an iterative algorithm, exploiting constraints prioritization, is used to clamp the joint space motion in the vicinity of the joint limits. Another iterative method is proposed in [13] to construct a null space projection operator, which—after ensuring its continuity by introducing activation factors—forms a basis for obtaining joint velocities.

The aforementioned methods are more complex than the basic pseudoinverse-based IK. Even though they account for the joint position limits, the higher order constraints are not straightforward to include. To remedy this problem, a method based on a quadratic programming (QP) formulation is proposed in this paper. Although the QP formulation of IK is known—it can be used to derive the pseudoinverse-based IK [2], [14]—this paper presents an important enhancement. The scientific novelty of this work is the proposition of a velocity-level IK method that allows the fulfilment of the joint acceleration constraints together with the velocity- and position-level constraints. The elements of the goal function, the Hessian matrix and other necessary quantities, are formulated in the form that uses accelerations instead of the usual velocities, as in [2], [14]. The discretization of the joint state equations allows to use the joint accelerations as the decision variables in the QP-based IK.

The remainder of this paper is organized as follows. In Section II, classic methods of kinematic redundancy resolution are briefly recapitulated. Section III presents a novel method of enforcing joint limits on acceleration, velocity and position levels within the framework of the QP approach to the IK solution. Then, in Section IV, the performance of the proposed method is verified and compared with the classic approach. Finally, the conclusions and outlook of further works are given in Section V.

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 $\mathbf{x} \in \mathbb{R}^m$ are related to the joint space variables $\mathbf{q} \in \mathbb{R}^n$ by the equation:

$$\mathbf{x} = \mathbf{f}(\mathbf{q}), \tag{1}$$

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

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \boldsymbol{\phi} \end{bmatrix}, \tag{2}$$

whereas the forward kinematics function $\mathbf{f}(\mathbf{q})$ can be written as:

$$\mathbf{f}(\mathbf{q}) = \begin{bmatrix} \mathbf{f}_{\mathbf{r}}(\mathbf{q}) \\ \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{q}) \end{bmatrix}, \tag{3}$$

where $\mathbf{f}_{\mathbf{r}}(\mathbf{q})$ describes the position and $\mathbf{f}_{\boldsymbol{\phi}}(\mathbf{q})$ 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 $\mathbf{r}$ and $\boldsymbol{\phi}$ might be necessary in $\mathbf{x}$. Accordingly, not all elements of $\mathbf{f}(\mathbf{q})$ given by (3) would be necessary in such a case.

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

$$\dot{\mathbf{x}} = \mathbf{J}_a\dot{\mathbf{q}}, \tag{4}$$

where:

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

The analytical task Jacobian matrix $\mathbf{J}_a$ is computed as [1]:

$$\mathbf{J}_a(\mathbf{q}) = \frac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}}. \tag{5}$$

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

$$\mathbf{v} = \mathbf{J}\dot{\mathbf{q}}, \tag{6}$$

instead of relationship between $\dot{\mathbf{q}}$ and $\dot{\mathbf{x}}$.

Again, for a full 6 degree-of-freedom task, the vector $\mathbf{v} \in \mathbb{R}^6$ can consist of the end effector's translational velocity $\dot{\mathbf{r}} \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$:

$$\mathbf{v} = \begin{bmatrix} \dot{\mathbf{r}} \\ \boldsymbol{\omega} \end{bmatrix}, \tag{7}$$

and the angular velocity $\boldsymbol{\omega}$ can be expressed as:

$$\boldsymbol{\omega} = \mathbf{T}(\boldsymbol{\phi})\dot{\boldsymbol{\phi}}, \tag{8}$$

where $\mathbf{T}(\boldsymbol{\phi})$ 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 $\dot{\mathbf{r}}$ and $\boldsymbol{\omega}$ might be necessary in $\mathbf{v}$. For example, in a pure translational end effector task, the task variables vector $\mathbf{x}$ and the task velocity vector $\mathbf{v}$ are given by $\mathbf{x} = \mathbf{r}$ and $\mathbf{v} = \dot{\mathbf{r}}$, respectively. It is worth noting that in such a case the analytical and geometric Jacobians are identical: $\mathbf{J}_a = \mathbf{J}$. On the other hand, when $m = 6$, there exists a following relationship between $\mathbf{J}$ and $\mathbf{J}_a$:

$$\mathbf{J} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}(\boldsymbol{\phi}) \end{bmatrix} \mathbf{J}_a. \tag{9}$$

The well-known recursive algorithms for computing the geometric Jacobian $\mathbf{J}$ can be found in [1], [2].

Moreover:

- $\mathcal{R}(\mathbf{J}) \subseteq \mathbb{R}^m$ is the range space of $\mathbf{J}(\mathbf{q})$, and it contains the end effector task velocities $\mathbf{v}$ that can be generated by the joint velocities $\dot{\mathbf{q}}$ for a given configuration $\mathbf{q}$,
- $\mathcal{N}(\mathbf{J}) \subseteq \mathbb{R}^n$ is the null space of $\mathbf{J}(\mathbf{q})$, and it contains the joint velocities $\dot{\mathbf{q}}$ 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, $\dim(\mathcal{N}(\mathbf{J})) > 0$,
- $\dim(\mathcal{R}(\mathbf{J})) + \dim(\mathcal{N}(\mathbf{J})) = n$.

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

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}\mathbf{v} \tag{10}$$

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

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}\mathbf{v}. \tag{11}$$

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 $\|\dot{\mathbf{q}}\|$. 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]:

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}\mathbf{v} + \dot{\mathbf{q}}^{NS}, \tag{12}$$

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

$$\mathbf{J}\dot{\mathbf{q}}^{NS} = \mathbf{0}. \tag{13}$$

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

$$\dot{\mathbf{q}}^{NS} = \mathbf{P}\dot{\mathbf{q}}^{JS}, \tag{14}$$

where $\dot{\mathbf{q}}^{JS}$ 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 $\dot{\mathbf{q}}^{JS}$ onto the null space of the Jacobian $\mathcal{N}(\mathbf{J})$ [1], [2], [17]:

$$\mathbf{P} = \mathbf{I}_n - \mathbf{J}^{\#}\mathbf{J}, \tag{15}$$

and $\mathbf{JP} = \mathbf{0}$.

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

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}\mathbf{v} + \left(\mathbf{I} - \mathbf{J}^{\#}\mathbf{J}\right)\dot{\mathbf{q}}^{JS}, \tag{16}$$

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 $\dot{\mathbf{q}}^{JS}$ for the solution (16) is to employ the local optimization [1], [2], [16], [18]:

$$\dot{\mathbf{q}}^{JS} = k_H \nabla_{\mathbf{q}} H(\mathbf{q}), \tag{17}$$

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:

$$\mathbf{v}^i = \mathbf{J}^i\dot{\mathbf{q}}, \quad i = 1, \ldots, l, \tag{18}$$

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):

$$\dot{\mathbf{q}}^i = \dot{\mathbf{q}}^{i-1} + \left(\mathbf{J}^i\mathbf{P}_A^{i-1}\right)^{\#}\left(\mathbf{v}^i - \mathbf{J}^i\dot{\mathbf{q}}^{i-1}\right), \tag{19}$$

for $i = 1, \ldots, l$, and where $\dot{\mathbf{q}}^l$ is the solution accounting for all the tasks, while $\dot{\mathbf{q}}^0 = \mathbf{0}$ and $\mathbf{P}_A^0 = \mathbf{I}$. The matrix $\mathbf{P}_A^i$ [3], [6]:

$$\mathbf{P}_A^i = \mathbf{I} - \mathbf{J}_A^{i\,\#}\mathbf{J}_A^i = \mathbf{P}_A^{i-1} - \left(\mathbf{J}^i\mathbf{P}_A^{i-1}\right)^{\#}\mathbf{J}^i\mathbf{P}_A^{i-1} \tag{20}$$

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

$$\mathbf{J}_A^i = \begin{bmatrix} \mathbf{J}^1 \\ \mathbf{J}^2 \\ \vdots \\ \mathbf{J}^i \end{bmatrix}. \tag{21}$$

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):

$$\mathbf{q}(t) = \mathbf{q}(t_0) + \int_{t_0}^{t} \dot{\mathbf{q}}(\tau)d\tau. \tag{22}$$

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:

$$\mathbf{q}_{k+1} = \mathbf{q}_0 + \sum_{h=0}^{k} \dot{\mathbf{q}}_h \Delta t, \tag{23}$$

which can be expressed recursively as:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_k \Delta t, \tag{24}$$

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, Δt is the time step, and $k = 1, 2, \ldots, N$. The joint velocity $\dot{\mathbf{q}}_k$ is just a discrete form of (16):

$$\dot{\mathbf{q}}_k = \mathbf{J}^{\#}(\mathbf{q}_k)\mathbf{v}_k + \mathbf{P}(\mathbf{q}_k)\dot{\mathbf{q}}_k^{JS}, \tag{25}$$

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

$$\mathbf{P}(\mathbf{q}_k) = \mathbf{I} - \mathbf{J}^{\#}(\mathbf{q}_k)\mathbf{J}(\mathbf{q}_k). \tag{26}$$

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:

$$\mathbf{v}_k = \mathbf{v}_{d,k}. \tag{27}$$

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

$$\mathbf{v}_k = \mathbf{v}_{d,k} + \mathbf{K}\mathbf{e}_k, \tag{28}$$

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

$$\mathbf{e}_k = \begin{bmatrix} \mathbf{r}_{d,k} - \mathbf{f}_{\mathbf{r}}(\mathbf{q}_k) \\ \mathbf{e}_{\boldsymbol{\phi},k} \end{bmatrix}, \tag{29}$$

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

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

$$\mathbf{v}_k = \mathbf{v}_{d,k} + K\mathbf{e}_k, \tag{30}$$

where $K > 0$ 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 K:

$$K < \frac{2}{\Delta t}. \tag{31}$$

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:

$$\begin{cases} \mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_k \Delta t + \frac{1}{2}\ddot{\mathbf{q}}_k (\Delta t)^2 \\ \dot{\mathbf{q}}_{k+1} = \dot{\mathbf{q}}_k + \ddot{\mathbf{q}}_k \Delta t. \end{cases} \tag{32}$$

Equation (32) can be written as:

$$\mathbf{z}_{k+1} = \mathbf{A}\mathbf{z}_k + \mathbf{B}\mathbf{u}_k, \tag{33}$$

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

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{q}_k \\ \dot{\mathbf{q}}_k \end{bmatrix}, \tag{34}$$

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

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_n & \Delta t \mathbf{I}_n \\ \mathbf{0}_{n\times n} & \mathbf{I}_n \end{bmatrix}, \tag{35}$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \mathbf{I}_n \\ \Delta t \mathbf{I}_n \end{bmatrix}, \tag{36}$$

and are constant.

Equation (33) can be divided into two parts:

$$\begin{cases} \mathbf{q}_{k+1} = \mathbf{A}_0\mathbf{z}_k + \mathbf{B}_0\mathbf{u}_k \\ \dot{\mathbf{q}}_{k+1} = \mathbf{A}_1\mathbf{z}_k + \mathbf{B}_1\mathbf{u}_k \end{cases} \tag{37}$$

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

$$\begin{aligned} \mathbf{A}_0 &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{A}, \\ \mathbf{B}_0 &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{B}, \\ \mathbf{A}_1 &= \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{A}, \\ \mathbf{B}_1 &= \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{B}, \end{aligned} \tag{38}$$

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

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

$$\mathbf{J}(\mathbf{q}_{k+1}) \cdot \dot{\mathbf{q}}_{k+1} = \mathbf{v}_{k+1}. \tag{39}$$

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

$$\mathbf{J}_{k+1} \cdot (\mathbf{A}_1\mathbf{z}_k + \mathbf{B}_1\mathbf{u}_k) = \mathbf{v}_{k+1}, \tag{40}$$

where:

$$\mathbf{J}_{k+1} = \mathbf{J}(\mathbf{q}_{k+1}) = \mathbf{J}\left(\mathbf{A}_0\mathbf{z}_k + \mathbf{B}_0\mathbf{u}_k\right). \tag{41}$$

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 non-linear 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 $\hat{\mathbf{q}}_{k+1}$ of the future joint position $\mathbf{q}_{k+1}$ can be computed as:

$$\hat{\mathbf{q}}_{k+1} = \mathbf{A}_0\mathbf{z}_k + \mathbf{B}_0\hat{\mathbf{u}}_k, \tag{42}$$

where $\hat{\mathbf{u}}_k = \mathbf{u}^*_{k-1}$ is the solution from the previous step. Then, by using (42), $\mathbf{J}_{k+1}$ can be approximated as $\hat{\mathbf{J}}_{k+1}$:

$$\hat{\mathbf{J}}_{k+1} = \mathbf{J}(\hat{\mathbf{q}}_{k+1}). \tag{43}$$

Using (43) in (40), a linear equation in terms of $\mathbf{u}_k$ is obtained:

$$\hat{\mathbf{J}}_{k+1} \cdot (\mathbf{A}_1\mathbf{z}_k + \mathbf{B}_1\mathbf{u}_k) = \mathbf{v}_{k+1}. \tag{44}$$

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 ($\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$), velocity ($\dot{\mathbf{q}}_{\min}$ and $\dot{\mathbf{q}}_{\max}$) and acceleration limits ($\ddot{\mathbf{q}}_{\min}$ and $\ddot{\mathbf{q}}_{\max}$). 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:

$$\mathbf{q}_{\min} \leq \mathbf{q}_k + \dot{\mathbf{q}}_k\Delta t + \frac{1}{2}\ddot{\mathbf{q}}_k(\Delta t)^2 \leq \mathbf{q}_{\max}, \tag{45}$$

where the operator $\leq$ used with vectors means an element-wise operation. The joint velocity bounds are:

$$\dot{\mathbf{q}}_{\min} \leq \dot{\mathbf{q}}_k + \ddot{\mathbf{q}}_k\Delta t \leq \dot{\mathbf{q}}_{\max}, \tag{46}$$

and the acceleration limits can be included as:

$$\ddot{\mathbf{q}}_{\min} \leq \ddot{\mathbf{q}}_k \leq \ddot{\mathbf{q}}_{\max}. \tag{47}$$

Another fact to bear in mind is that the robot joints cannot stop in an instant. Applying the maximum deceleration $\ddot{q}_{\min,j} < 0$ to the j-th joint for some time $t \geq t_k$ results in the following position and velocity:

$$\begin{cases} q_j(t) &= q_{k,j} + \dot{q}_{k,j}(t-t_k) + \frac{1}{2}\ddot{q}_{\min,j}(t-t_k)^2 \\ \dot{q}_j(t) &= \dot{q}_{k,j} + \ddot{q}_{\min,j}(t-t_k). \end{cases} \tag{48}$$

When the j-th joint reaches its limit $q_{\max,j}$, its velocity should be less or equal to zero. Suppose this event happens at some $t = t^* > t_k$:

$$\begin{cases} q_j(t^*) = q_{k,j} + \dot{q}_{k,j}(t^*-t_k) + \frac{1}{2}\ddot{q}_{\min,j}(t^*-t_k)^2 = q_{\max,j} \\ \dot{q}_j(t^*) = \dot{q}_{k,j} + \ddot{q}_{\min,j}(t^*-t_k) \leq 0. \end{cases} \tag{49}$$

From (49), it follows that the j-th joint velocity at t_k is upper bounded by [14], [37], [38]:

$$\dot{q}_{k,j} \leq \sqrt{-2\ddot{q}_{\min,j}(q_{\max,j} - q_{k,j})}. \tag{50}$$

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

$$\dot{q}_{k,j} \geq -\sqrt{2\ddot{q}_{\max,j}(q_{k,j} - q_{\min,j})}. \tag{51}$$

The joint constraints (45)–(51) can also be transformed into a form dependent on $\mathbf{u}_k$. In that regard, the joint position limits (45) become:

$$\mathbf{q}_{\min} - \mathbf{A}_0\mathbf{z}_k \leq \mathbf{B}_0\mathbf{u}_k \leq \mathbf{q}_{\max} - \mathbf{A}_0\mathbf{z}_k, \tag{52}$$

while the joint velocity bounds (46) are now:

$$\dot{\mathbf{q}}_{\min} - \mathbf{A}_1\mathbf{z}_k \leq \mathbf{B}_1\mathbf{u}_k \leq \dot{\mathbf{q}}_{\max} - \mathbf{A}_1\mathbf{z}_k, \tag{53}$$

and the acceleration limits (47) can be included as:

$$\ddot{\mathbf{q}}_{\min} \leq \mathbf{u}_k \leq \ddot{\mathbf{q}}_{\max}. \tag{54}$$

Finally, the constraints (50) and (51) for $\dot{\mathbf{q}}_{k+1}$ can be transformed, by using (32) and (42), into:

$$\boldsymbol{\rho}_{\min} \leq \mathbf{u}_k \leq \boldsymbol{\rho}_{\max}, \tag{55}$$

where:

$$\begin{aligned} \rho_{\min,j} &= \frac{-\sqrt{2\ddot{q}_{\max,j}(\hat{q}_{k+1,j} - q_{\min,j})} - \dot{q}_{k,j}}{\Delta t} \\ \rho_{\max,j} &= \frac{\sqrt{-2\ddot{q}_{\min,j}(q_{\max,j} - \hat{q}_{k+1,j})} - \dot{q}_{k,j}}{\Delta t} \end{aligned} \tag{56}$$

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:

$$\min_{\dot{\mathbf{q}}} \frac{1}{2}\sum_{i=1}^{l}\left(\mathbf{J}^i\dot{\mathbf{q}} - \mathbf{v}^i\right)^T \mathbf{W}_i \left(\mathbf{J}^i\dot{\mathbf{q}} - \mathbf{v}^i\right), \tag{57}$$

where l is the number of null space tasks, $\mathbf{W}_i \in \mathbb{R}^{m_i \times m_i}$ 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 $\mathbf{J}^i \in \mathbb{R}^{m_i \times n}$ and task velocity $\mathbf{v}^i \in \mathbb{R}^{m_i}$ 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 $\mathbf{q}$. An example of such a task is the obstacle avoidance [21], [22]. The joint space tasks can be implemented by setting $\mathbf{J}^i = \mathbf{I}$ and $\mathbf{v}^i = \dot{\mathbf{q}}^{JS,i}$, where $\dot{\mathbf{q}}^{JS,i}$ is given by (17).

As described above, the secondary tasks can be defined in different ways, each having different units—for example, joint space task velocity will have units of $\frac{rad}{s}$, whereas the Cartesian space: $\frac{m}{s}$. Since in (57) the terms are all summed, the units shall be consistent. Therefore, the role of $\mathbf{W}_i$ 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 $\mathbf{W}_i$ 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 $\mathbf{u}_k$, the goal function in (57) also has to be redefined. A new minimization problem—with the joint acceleration vector replacing the joint velocity as the optimization variable—can be written as:

$$\min_{\mathbf{u}_k} \frac{1}{2}\sum_{i=1}^{l}\left(\hat{\mathbf{J}}^i_{k+1}\cdot(\mathbf{A}_1\mathbf{z}_k+\mathbf{B}_1\mathbf{u}_k)-\hat{\mathbf{v}}^i_{k+1}\right)^T\mathbf{W}_i \cdot\left(\hat{\mathbf{J}}^i_{k+1}\cdot(\mathbf{A}_1\mathbf{z}_k+\mathbf{B}_1\mathbf{u}_k)-\hat{\mathbf{v}}^i_{k+1}\right)+\frac{1}{2}\gamma_{l+1}\mathbf{u}_k^T\mathbf{u}_k, \tag{58}$$

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

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

Equation (58) can be further regrouped into:

$$\min_{\mathbf{u}_k} \frac{1}{2}\sum_{i=1}^{l}\mathbf{u}_k^T\mathbf{B}_1^T(\hat{\mathbf{J}}^i_{k+1})^T\mathbf{W}_i\hat{\mathbf{J}}^i_{k+1}\mathbf{B}_1\mathbf{u}_k+\frac{1}{2}\gamma_{l+1}\mathbf{u}_k^T\mathbf{u}_k +\sum_{i=1}^{l}\mathbf{u}_k^T\mathbf{B}_1^T(\hat{\mathbf{J}}^i_{k+1})^T\mathbf{W}_i\left(\hat{\mathbf{J}}^i_{k+1}\mathbf{A}_1\mathbf{z}_k-\hat{\mathbf{v}}^i_{k+1}\right). \tag{59}$$

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:

$$\min_{\boldsymbol{\xi}} \frac{1}{2}\boldsymbol{\xi}^T\mathbf{H}\boldsymbol{\xi}+\boldsymbol{\xi}^T\mathbf{h}$$
$$\text{s. t. } \boldsymbol{\Phi}^{\text{eq}}\boldsymbol{\xi}=\mathbf{b}_{\text{eq}},\quad \mathbf{b}^{\text{iq}}_{\min}\le\boldsymbol{\Phi}^{\text{iq}}\boldsymbol{\xi}\le\mathbf{b}^{\text{iq}}_{\max}, \tag{60}$$

where the vector of optimization variables $\boldsymbol{\xi}\in\mathbb{R}^n$ is:

$$\boldsymbol{\xi}=\mathbf{u}_k \tag{61}$$

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)):

$$\mathbf{H}=\sum_{i=1}^{l}\mathbf{B}_1^T(\hat{\mathbf{J}}^i_{k+1})^T\mathbf{W}_i\hat{\mathbf{J}}^i_{k+1}\mathbf{B}_1+\gamma_{l+1}\mathbf{I}, \tag{62}$$

and:

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

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 $\boldsymbol{\Phi}^{\text{eq}}\in\mathbb{R}^{m\times n}$ is:

$$\boldsymbol{\Phi}^{\text{eq}}=\hat{\mathbf{J}}_{k+1}\mathbf{B}_1, \tag{64}$$

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

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

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

$$\boldsymbol{\Phi}^{\text{iq}}=\begin{bmatrix}\mathbf{B}_0\\ \mathbf{B}_1\\ \mathbf{I}_n\\ \mathbf{I}_n\end{bmatrix}, \tag{66}$$

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

$$\mathbf{b}^{\text{iq}}_{\min}=\begin{bmatrix}\mathbf{q}_{\min}-\mathbf{A}_0\mathbf{z}_k\\ \dot{\mathbf{q}}_{\min}-\mathbf{A}_1\mathbf{z}_k\\ \ddot{\mathbf{q}}_{\min}\\ \boldsymbol{\rho}_{\min}\end{bmatrix}, \tag{67}$$

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

$$\mathbf{b}^{\text{iq}}_{\max}=\begin{bmatrix}\mathbf{q}_{\max}-\mathbf{A}_0\mathbf{z}_k\\ \dot{\mathbf{q}}_{\max}-\mathbf{A}_1\mathbf{z}_k\\ \ddot{\mathbf{q}}_{\max}\\ \boldsymbol{\rho}_{\max}\end{bmatrix}. \tag{68}$$

The solution of the IK problem is obtained for the time interval $\langle t_0, t_{end}\rangle$, with a given time step Δt, 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):

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

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

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

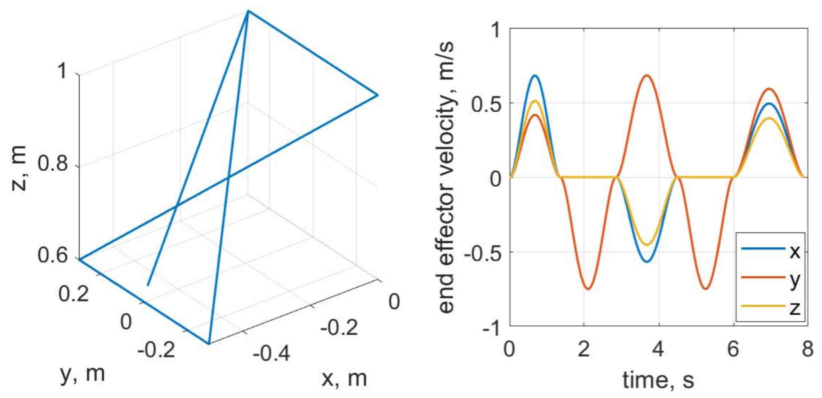

FIGURE 1. Scenario 1A: Desired end effector path (left) and velocity (right).

```
Algorithm 1 QP IK Solver
  (Initialization)
  Set $t := t_0$, $\hat{\mathbf{u}}_k := \mathbf{0}$, $\mathbf{z}_k := \left[\mathbf{q}(t_0)^T \;\; \dot{\mathbf{q}}(t_0)^T\right]^T$
  Set the joint limits $\mathbf{q}_{\min}$, $\mathbf{q}_{\max}$, $\dot{\mathbf{q}}_{\min}$, $\dot{\mathbf{q}}_{\max}$, $\ddot{\mathbf{q}}_{\min}$, $\ddot{\mathbf{q}}_{\max}$
  Compute constant matrices $\mathbf{A}$ (35), $\mathbf{B}$ (36), and $\mathbf{A}_0$, $\mathbf{B}_0$, $\mathbf{A}_1$, $\mathbf{B}_1$ (38)
  (Main loop)
  while $t < t_{end}$ do
      Compute $\hat{\mathbf{q}}_{k+1}$ (42), $\hat{\mathbf{J}}_{k+1}$ (43)
      Obtain $\mathbf{v}_{k+1}$ from the end effector trajectory generator
      If secondary tasks are present, obtain $\hat{\mathbf{J}}^i_{k+1}$ and $\hat{\mathbf{v}}^i_{k+1}$
      Formulate the QP problem (60)
      Set the initial guess $\boldsymbol{\xi}_{init} := \hat{\mathbf{u}}_k$
      Solve the QP problem (60) and obtain $\boldsymbol{\xi}^*$
      Set $\mathbf{u}_k := \boldsymbol{\xi}^*$
      Compute new state $\mathbf{z}_{k+1} := \mathbf{A}\mathbf{z}_k + \mathbf{B}\mathbf{u}_k$ (33)
      Set $\hat{\mathbf{u}}_{k+1} := \mathbf{u}_k$
      Set $t := t + \Delta t$
      Set $k := k + 1$
  end while
```

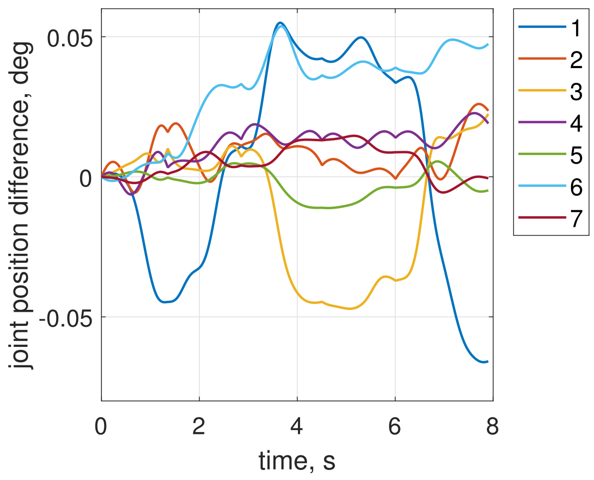

FIGURE 2. Scenario 1A: Joint position difference between the classic IK solution and the proposed method.

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 $\frac{1}{2}\dot{\mathbf{q}}^T\dot{\mathbf{q}}$ 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 $\dot{\mathbf{q}}_p$ is obtained from (6) where the Jacobian $\mathbf{J}$ is replaced with a square matrix $\mathbf{J}_p$ with only linearly independent columns of $\mathbf{J}$. Then, the QP problem is formulated to obtain the free variables of the joint velocity which are necessary to compute the full vector $\dot{\mathbf{q}}$; conveniently, the equality constraint (6) is no longer needed, the unequality 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 $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$. 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

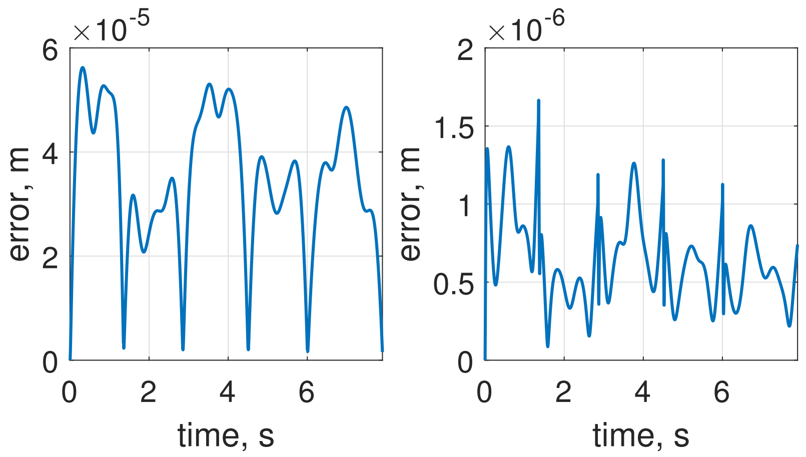

FIGURE 3. Scenario 1A: End effector error for the classic IK (left) and the proposed method (right).

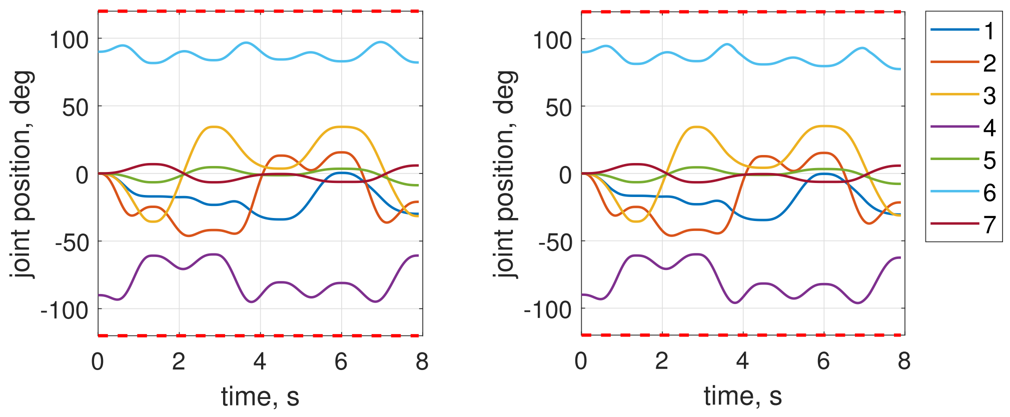

FIGURE 4. Scenario 1B: Joint position for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

joint acceleration $\ddot{\mathbf{q}}$ (or $\mathbf{u}$) 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.

TABLE 1. LWR 4+ modified Denavit-Hartenberg parameters.

j	$\alpha_j\ (rad)$	$a_j\ (m)$	$q_j\ (rad)$	$d_j\ (m)$
1	0	0	q_1	0.31
2	0.5π	0	q_2	0
3	-0.5π	0	q_3	0.4
4	-0.5π	0	q_4	0
5	0.5π	0	q_5	0.39
6	0.5π	0	q_6	0
7	-0.5π	0	q_7	0

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: $\mathbf{r}_{EE}^{(7)} = \left[0.1\ 0\ 0.078\right]^T\ m$.

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.

TABLE 2. Scenario 1A&B: Points of the path.

Point	$x,\quad y,\quad z\ (m)$
$\mathbf{X}_0$	$-0.49,\quad 0,\quad 0.632$
$\mathbf{X}_1$	$0,\quad 0.3,\quad 1$
$\mathbf{X}_2$	$0,\quad -0.3,\quad 1$
$\mathbf{X}_3$	$-0.5,\quad 0.3,\quad 0.6$
$\mathbf{X}_4$	$-0.5,\quad -0.3,\quad 0.6$

TABLE 3. Scenario 2: Points of the path.

Point	$x,\quad y,\quad z\ (m)$
$\mathbf{X}_0$	$0,\quad -0.29,\quad 0.632$
$\mathbf{X}_1$	$-0.35,\quad 0.3,\quad 1$
$\mathbf{X}_2$	$-0.35,\quad -0.3,\quad 1$
$\mathbf{X}_3$	$-0.35,\quad -0.3,\quad 0.6$
$\mathbf{X}_4$	$-0.35,\quad 0.3,\quad 0.6$

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 perfromed 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 *quadprog*, utilizing the *active-set* algorithm with default settings [49].

B. SCENARIO 1A

The start configuration was equal to:

$$\mathbf{q}_0 = \begin{bmatrix}0^\circ & 0^\circ & 0^\circ & -45^\circ & 0^\circ & 45^\circ & 0^\circ\end{bmatrix}^T,$$

corresponding to the end effector position $\mathbf{X}_0$, and then the path continued through the points $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_4$, and again $\mathbf{X}_1$, 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 to $T_1 = 1.35\ s$, $T_2 = 1.5\ s$, $T_3 = 1.65\ s$, $T_4 = 1.5\ s$, and $T_5 = 1.9\ s$, resulting in total motion time $t_{end} = 7.9$ seconds. 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.

To account for the joint velocity minimization in the proposed method, there was one additional task ($l = 1$) defined in (57) as $\mathbf{J}^1 = \mathbf{I}$ and $\mathbf{v}^1 = \mathbf{0}$ with the weight matrix $\mathbf{W}_1 = \gamma_1\mathbf{I}$ and $\gamma_1 = 10^6 s^2$. Correspondingly, the necessary components of (62) and (63) were set to $\hat{\mathbf{J}}^i_{k+1} = \mathbf{I}$ and $\hat{\mathbf{v}}^i_{k+1} = \mathbf{0}$ for each simulation step k. There was no joint acceleration minimization task, and therefore $\gamma_2 = 0$. No joint constraints were set.

As can be seen in Fig. 2, both solutions are very close—the maximum difference is in the order of a hundredth of a degree. The norm of the end effector error, computed as:

$$e = ||\mathbf{r}_d - \mathbf{f}_\mathbf{r}(\mathbf{q})||, \tag{71}$$

where $\mathbf{r}_d$ is the desired end effector trajectory and $\mathbf{f}_\mathbf{r}(\mathbf{q})$ describes the forward kinematics, is shown in Fig. 3. The maximum error (71) for the classic method is $e = 5.62 \cdot 10^{-5}\ m$, whereas for the proposed method it is $e = 1.67 \cdot 10^{-6}\ m$. The average computation time in each step is 0.14 *ms* for the classic IK and 1.5 *ms* for the proposed method.

The results show that the proposed method behaves correctly and provides the solution similar to the pseudoinverse-based approach.

C. SCENARIO 1B

In this simulation, the setup from Scenario 1A (Sec. IV-B) was repeated. This time, however, the existence of the following joint constraints was considered: $\mathbf{q}_{\min} = -120\mathbf{I}^\circ_{7\times1}$, $\mathbf{q}_{\max} = 120\mathbf{I}^\circ_{7\times1}$, $\dot{\mathbf{q}}_{\min} = -150\mathbf{I}_{7\times1}\frac{\circ}{s}$, $\dot{\mathbf{q}}_{\max} = 150\mathbf{I}_{7\times1}\frac{\circ}{s}$, $\ddot{\mathbf{q}}_{\min} = -250\mathbf{I}_{7\times1}\frac{\circ}{s^2}$, $\ddot{\mathbf{q}}_{\max} = 250\mathbf{I}_{7\times1}\frac{\circ}{s^2}$.

Additionaly, the joint acceleration minimization task was included in the proposed method with a weight set to $\gamma_2 = 10\ s^4$. The joint velocity minimimization was still active, with the same weight as in Sec. IV-B, namely $\gamma_1 = 10^6 s^2$.

As can be seen in Fig. 4, the joint positions are within their bounds in both solutions. The same is true for joint velocities, shown in Fig. 5. However, at the acceleration level, the constraints for the 2nd and 4th joint are violated in the classic IK solution, as pictured in Fig. 6. At the same time, the proposed method fulfills the constraints—it is clearly visible in Fig. 6 that the accelerations saturate at their limits.

The maximum end effector error (71) for the proposed method is the same as in the case without the joint limits (i.e., Scenario 1A, Sec. IV-B), namely $e = 1.67 \cdot 10^{-6}\ m$. 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 $\mathbf{q}_0$ was chosen as:

$$\mathbf{q}_0 = \begin{bmatrix}-45^\circ & 0^\circ & 0^\circ & 45^\circ & 0^\circ & -45^\circ & 0^\circ\end{bmatrix}^T,$$

corresponding to the end effector position $\mathbf{X}_0$. From $\mathbf{X}_0$, the path continued through the points $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_4$, and again $\mathbf{X}_1$, shown in Table 3.

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 $t_{end} = 7.1$ seconds. 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.

The joint limits were set as follows: $\mathbf{q}_{\min} = -100\mathbf{I}^\circ_{7\times1}$, $\mathbf{q}_{\max} = 100\mathbf{I}^\circ_{7\times1}$, $\dot{\mathbf{q}}_{\min} = -150\mathbf{I}_{7\times1}\frac{\circ}{s}$, $\dot{\mathbf{q}}_{\max} = 150\mathbf{I}_{7\times1}\frac{\circ}{s}$, $\ddot{\mathbf{q}}_{\min} = -350\mathbf{I}_{7\times1}\frac{\circ}{s^2}$, $\ddot{\mathbf{q}}_{\max} = 350\mathbf{I}_{7\times1}\frac{\circ}{s^2}$.

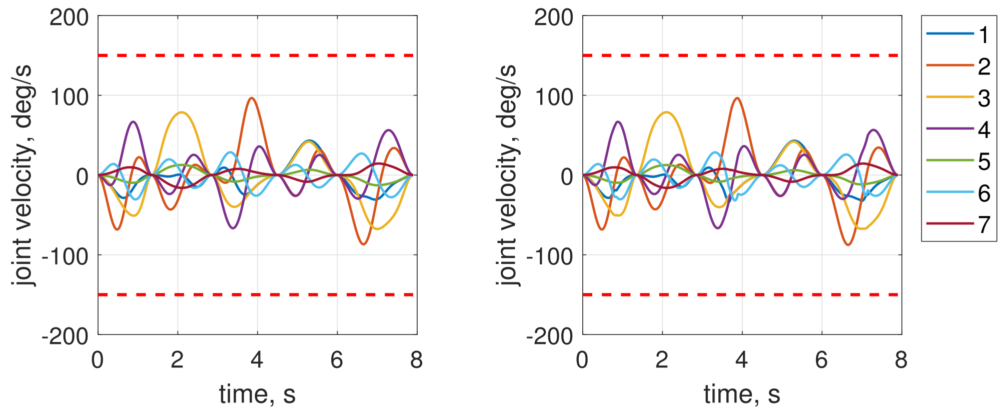

FIGURE 5. Scenario 1B: Joint velocity for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

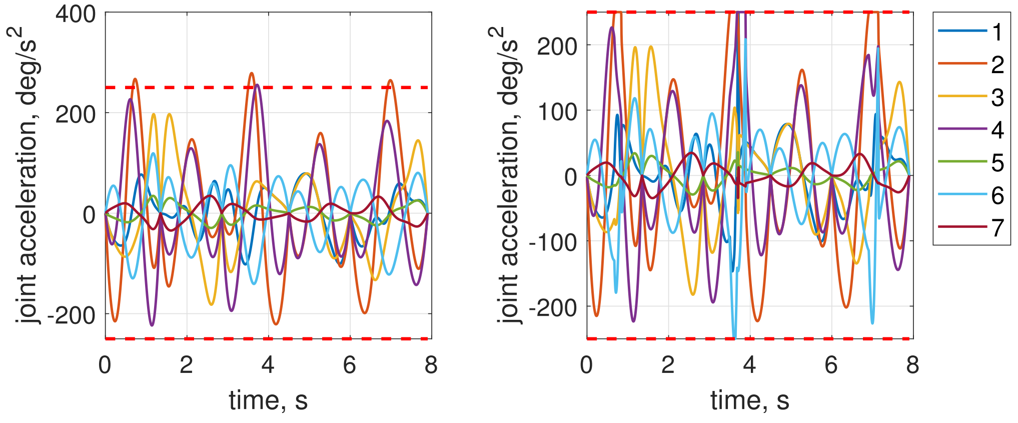

FIGURE 6. Scenario 1B: Joint acceleration for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

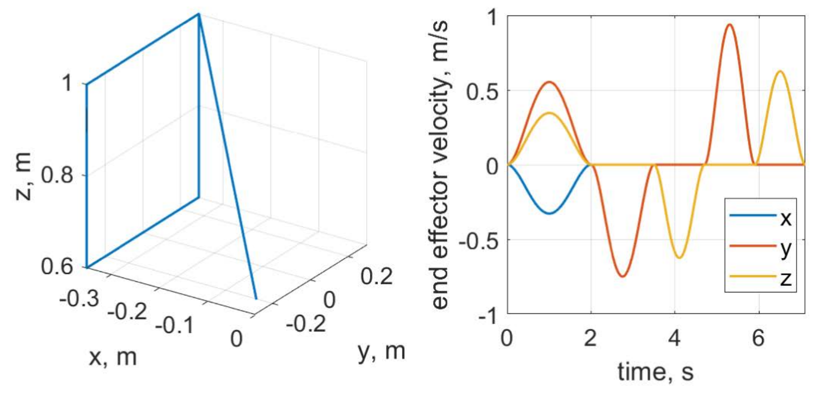

FIGURE 7. Scenario 2: Desired end effector path (left) and velocity (right).

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 $\gamma_2 = 10\, s^4$.

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 $e = 1.95 \cdot 10^{-6}$.

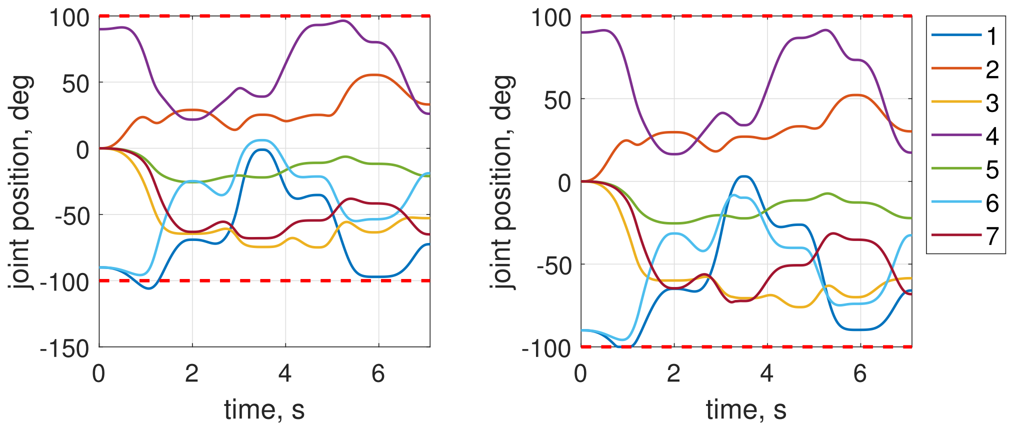

FIGURE 8. Scenario 2: Joint position for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

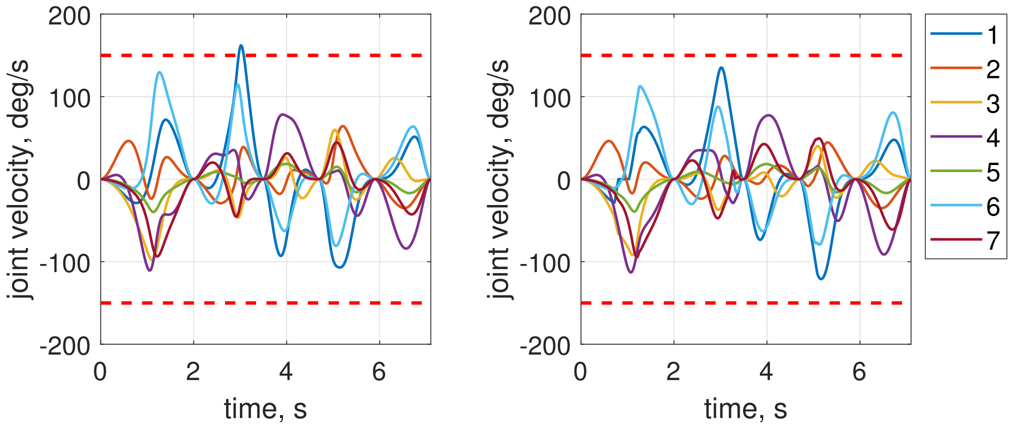

FIGURE 9. Scenario 2: Joint velocity for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

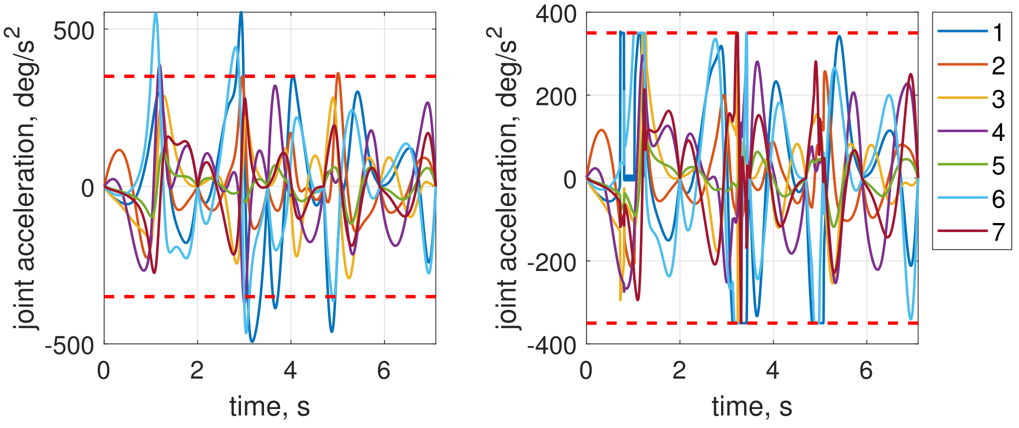

FIGURE 10. Scenario 2: Joint acceleration for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

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 $\mathbf{q}_0$ was:

$$\mathbf{q}_0 = \begin{bmatrix} 0^\circ & 0^\circ & 0^\circ & -45^\circ & 0^\circ & 45^\circ & 0^\circ \end{bmatrix}^T,$$

corresponding to the end effector position $\mathbf{X}_0$, and then the path continued through the points $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_4$, and again $\mathbf{X}_1$, 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

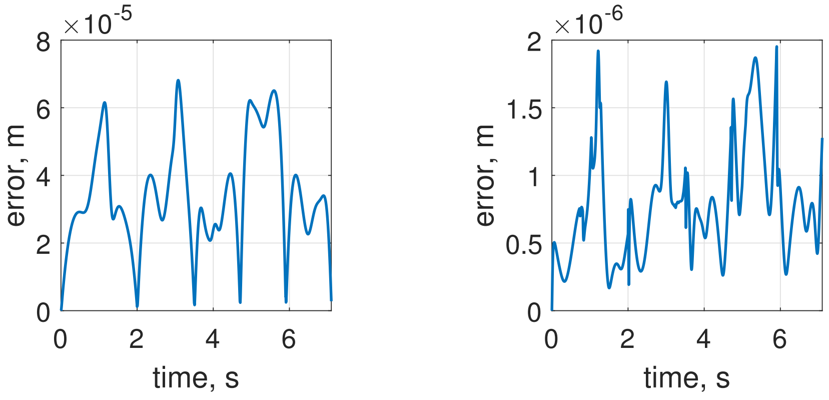

FIGURE 11. Scenario 2: End effector error for the classic IK (left) and the proposed method (right).

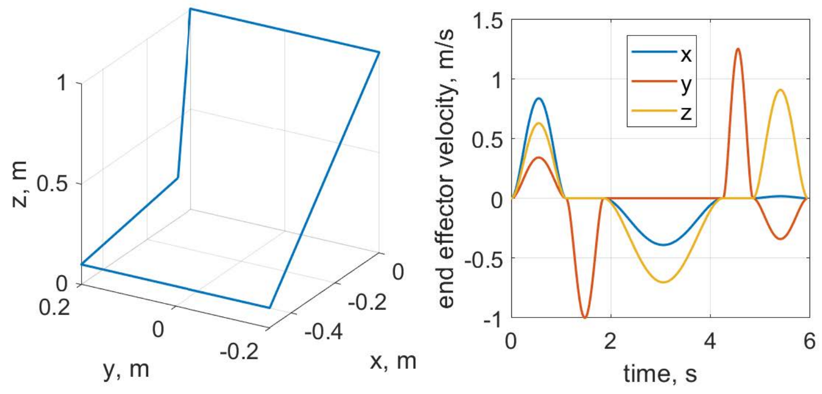

FIGURE 12. Scenario 3: Desired end effector path (left) and velocity (right).

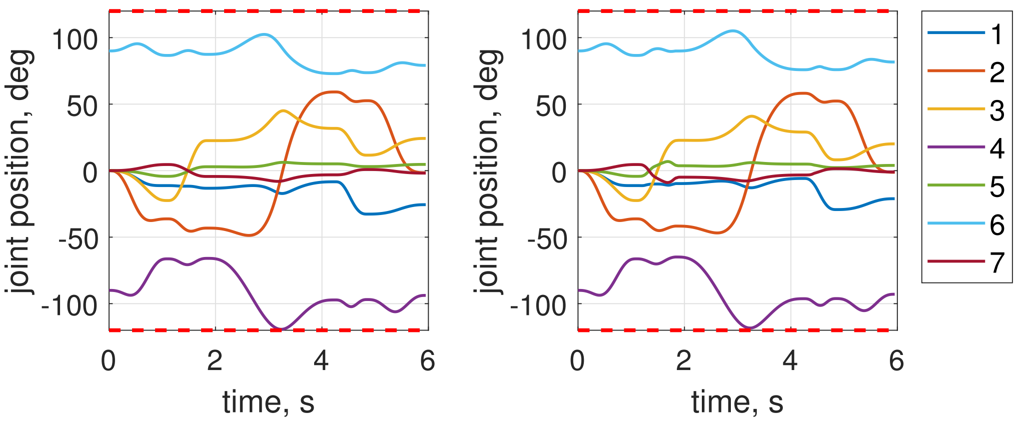

FIGURE 13. Scenario 3: Joint position for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

$T_5 = 1.1\ s$, resulting in total motion time $t_{end} = 5.95$ seconds. The trajectory generation is described in detail in Appendix.

The joint limits were set to: $\mathbf{q}_{\min} = -120\mathbf{I}^{\circ}_{7\times1}$, $\mathbf{q}_{\max} = 120\mathbf{I}^{\circ}_{7\times1}$, $\dot{\mathbf{q}}_{\min} = -150\mathbf{I}_{7\times1}\frac{\circ}{s}$, $\dot{\mathbf{q}}_{\max} = 150\mathbf{I}_{7\times1}\frac{\circ}{s}$, $\ddot{\mathbf{q}}_{\min} = -350\mathbf{I}_{7\times1}\frac{\circ}{s^2}$, $\ddot{\mathbf{q}}_{\max} = 350\mathbf{I}_{7\times1}\frac{\circ}{s^2}$.

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.

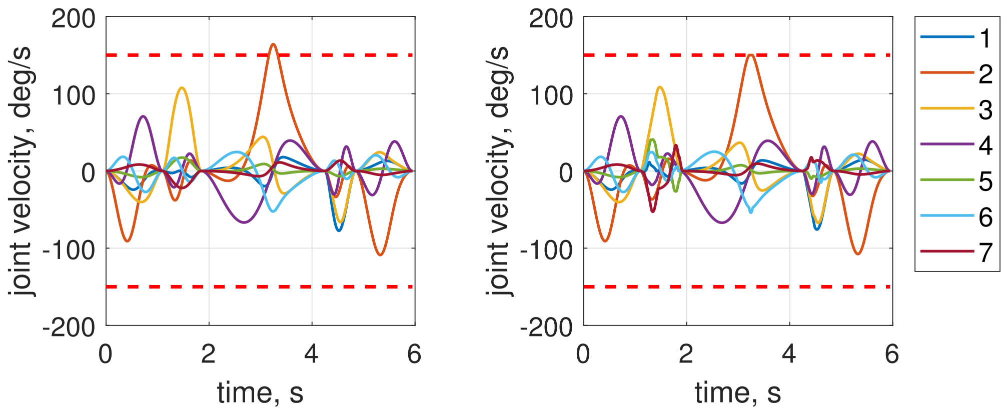

FIGURE 14. Scenario 3: Joint velocity for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

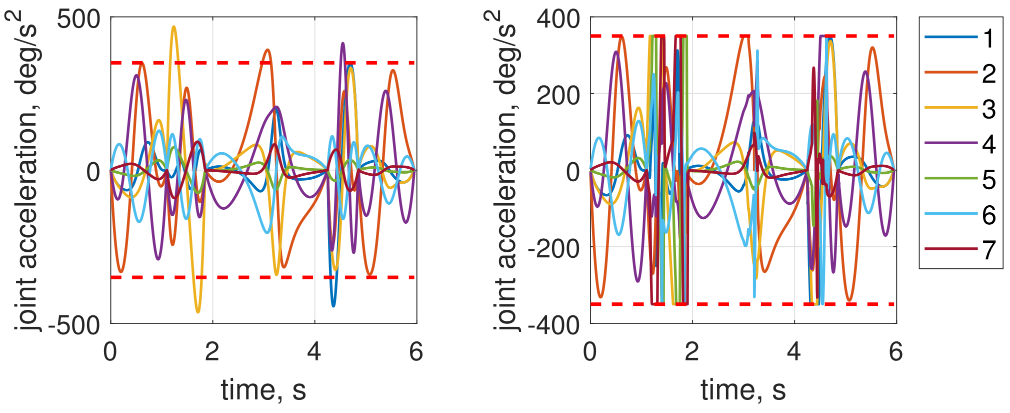

FIGURE 15. Scenario 3: Joint acceleration for the classic IK (left) and the proposed method (right). The red dashed lines represent the joint limits.

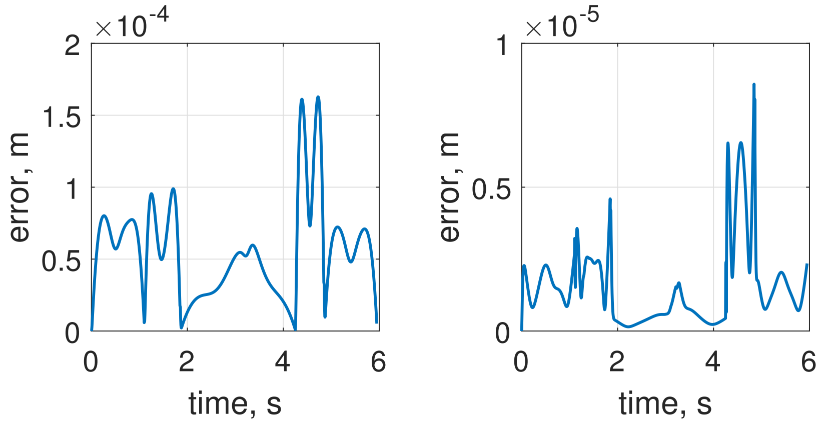

FIGURE 16. Scenario 3: End effector error for the classic IK (left) and the proposed method (right).

However, as can be seen in Fig. 14, the joint velocity for the 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 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.

TABLE 4. Scenario 3: Points of the path.

Point	$x, \quad y, \quad z\,(m)$
$\mathbf{X}_0$	$-0.49, \quad 0, \quad 0.632$
$\mathbf{X}_1$	$0, \quad 0.2, \quad 1$
$\mathbf{X}_2$	$0, \quad -0.2, \quad 1$
$\mathbf{X}_3$	$-0.5, \quad -0.2, \quad 0.1$
$\mathbf{X}_4$	$-0.5, \quad 0.2, \quad 0.1$

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
TRAJECTORY GENERATION

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

$$\begin{cases} \mathbf{r}(t) &= \mathbf{X}_{i-1} + \boldsymbol{\mu}_i s_i(\varphi) \\ \mathbf{v}(t) &= \boldsymbol{\mu}_i \dot{s}_i(\varphi), \end{cases} \tag{72}$$

where the unit vector $\boldsymbol{\mu}_i$ along the direction of motion is:

$$\boldsymbol{\mu}_i = \frac{\mathbf{X}_i - \mathbf{X}_{i-1}}{||\mathbf{X}_i - \mathbf{X}_{i-1}||}, \tag{73}$$

whereas the time law $s_i(\varphi)$ is defined as a fifth-order polynomial that smoothly increases from 0 to 1 for φ increasing from 0 to T_i:

$$s_i(\varphi) = \frac{6}{T_i^5}\varphi^5 - \frac{15}{T_i^4}\varphi^4 + \frac{10}{T_i^3}\varphi^3, \tag{74}$$

and the local time φ, measured from the start to end of the segment, is:

$$\varphi(t) = \begin{cases} t & \text{for } t \in \langle 0,\ T_1) \\ t - \sum_{k=1}^{i-1} T_k & \text{for } t \in \langle T_1,\ t_{end} \rangle\,, \end{cases} \tag{75}$$

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*. Cham, Switzerland: Springer, 2016, pp. 221–242.
[2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, "Differential kinematics and statics," in *Robotics*. London, U.K.: Springer, 2009, pp. 105–160.
[3] B. Siciliano and J. J. E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. 5th Int. Conf. Adv. Robot. (ICAR)*, vol. 2, Jun. 1991, pp. 1211–1216.
[4] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Trans. Robot. Autom.*, vol. 13, no. 3, pp. 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 Trans. Robot.*, vol. 25, no. 3, pp. 670–685, Jun. 2009.
[6] F. Flacco and A. D. Luca, "A reverse priority approach to multi-task control of redundant robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 2421–2427.
[7] S. Moe, G. Antonelli, A. R. Teel, K. Y. Pettersen, and J. Schrimpf, "Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results," *Frontiers Robot. AI*, vol. 3, p. 16, Apr. 2016.
[8] A. Colomé and C. Torras, "Redundant inverse kinematics: Experimental comparative review and two enhancements," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5333–5340.
[9] A. Colomé and C. Torras, "Closed-loop inverse kinematics for redundant robots: Comparative assessment and two enhancements," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 2, pp. 944–955, Apr. 2015.
[10] N. Mansard, A. Remazeilles, and F. Chaumette, "Continuity of varying-feature-set control laws," *IEEE Trans. Autom. Control*, vol. 54, no. 11, pp. 2493–2505, Nov. 2009.
[11] T. F. Chan and R. V. Dubey, "A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators," *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 286–292, Apr. 1995.
[12] D. Raunhardt and R. Boulic, "Progressive clamping," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 4414–4419.
[13] P. Jiang, S. Huang, J. Xiang, and M. Z. Q. Chen, "Iteratively successive projection: A novel continuous approach for the task-based control of redundant robots," *IEEE Access*, vol. 7, pp. 25347–25358, 2019.
[14] F. Flacco, A. De Luca, and O. Khatib, "Control of redundant robots under hard joint constraints: Saturation in the null space," *IEEE Trans. Robot.*, vol. 31, no. 3, pp. 637–654, Jun. 2015.
[15] A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. SMC-7, no. 12, pp. 868–871, Dec. 1977.
[16] H. Zghal, R. V. Dubey, and J. A. Euler, "Efficient gradient projection optimization for manipulators with multiple degrees of redundancy," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, May 1990, pp. 1006–1011.

[17] A. Ben Israel and T. N. E. Greville, *Generalized Inverses: Theory and Applications*. New York, NY, USA: Springer-Verlag, 2003, doi: 10.1007/b97366.

[18] D. N. Nenchev, "Redundancy resolution through local optimization: A review," *J. Robot. Syst.*, vol. 6, no. 6, pp. 769–798, Dec. 1989.

[19] T. Yoshikawa, "Dynamic manipulability of robot manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Mar. 1985, pp. 1033–1038.

[20] A. Zube, "Cartesian nonlinear model predictive control of redundant manipulators considering obstacles," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Mar. 2015, pp. 137–142.

[21] A. A. Maciejewski and C. A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 109–117, Sep. 1985.

[22] T. Petrič, A. Gams, N. Likar, and L. Žlajpah, "Obstacle avoidance with industrial robots," in *Motion and Operation Planning of Robotic Systems*, G. Carbone and F. Gomez-Bravo, Eds. Cham, Switzerland: Springer, 2015, pp. 113–145.

[23] F. Flacco and A. De Luca, "Unilateral constraints in the reverse priority redundancy resolution method," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 2564–2571.

[24] D. A. Drexler, "Solution of the closed-loop inverse kinematics algorithm using the Crank–Nicolson method," in *Proc. IEEE 14th Int. Symp. Appl. Mach. Intell. Informat. (SAMI)*, Jan. 2016, pp. 351–356.

[25] D. A. Drexler and L. Kovács, "Second-order and implicit methods in numerical integration improve tracking performance of the closed-loop inverse kinematics algorithm," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 003362–003367.

[26] D. Drexler, "Closed-loop inverse kinematics algorithm with implicit numerical integration," *Acta Polytech. Hungarica*, vol. 14, no. 1, pp. 147–161, Jan. 2017.

[27] P. Falco and C. Natale, "On the stability of closed-loop inverse kinematics algorithms for redundant robots," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 780–784, Aug. 2011.

[28] G. Antonelli, "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems," *IEEE Trans. Robot.*, vol. 25, no. 5, pp. 985–994, Oct. 2009.

[29] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *J. Graph. Tools*, vol. 10, no. 3, pp. 37–49, Jan. 2005.

[30] D. D. Vito, C. Natale, and G. Antonelli, "A comparison of damped least squares algorithms for inverse kinematics of robot manipulators," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6869–6874, 2017.

[31] A. Reiter, A. Müller, and H. Gattringer, "Inverse kinematics in minimum-time trajectory planning for kinematically redundant manipulators," in *Proc. 42nd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2016, pp. 6873–6878.

[32] A. Reiter, A. Müller, and H. Gattringer, "On higher order inverse kinematics methods in time-optimal trajectory planning for kinematically redundant manipulators," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1681–1690, Apr. 2018.

[33] M. Faroni, M. Beschi, C. G. L. Bianco, and A. Visioli, "Predictive joint trajectory scaling for manipulators with kinodynamic constraints," *Control Eng. Pract.*, vol. 95, Feb. 2020, Art. no. 104264.

[34] T. Kröger, A. Tomiczek, and F. Wahl, "Towards on-line trajectory computation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 736–741.

[35] X. Broquere, D. Sidobre, and I. Herrera-Aguilar, "Soft motion trajectory planner for service manipulator robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 2808–2813.

[36] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2008, pp. 3248–3253.

[37] F. Flacco and A. De Luca, "Optimal redundancy resolution with task scaling under hard bounds in the robot joint space," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 3969–3975.

[38] A. Del Prete, "Joint position and velocity bounds in discrete-time acceleration/torque control of robot manipulators," *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 281–288, Jan. 2018.

[39] J. Branke, K. Deb, K. Miettinen, and R. Slowiński, *Multiobjective Optimizatio*. Berlin, Germany: Springer-Verlag, 2008.

[40] F.-T. Cheng, T.-H. Chen, and Y.-Y. Sun, "Efficient algorithm for resolving manipulator redundancy—The compact QP method," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, Jan. 1992, pp. 508–513.

[41] Z. Zhang and Y. Zhang, "Variable joint-velocity limits of redundant robot manipulators handled by quadratic programming," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 2, pp. 674–686, Apr. 2013.

[42] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 785–792, Aug. 2011.

[43] D. Khudher and R. Powell, "Quadratic programming for inverse kinematics control of a hexapod robot with inequality constraints," in *Proc. Int. Conf. Robot., Current Trends Future Challenges (RCTFC)*, 2016, pp. 1–5.

[44] S. Cocuzza, I. Pretto, and S. Debei, "Least-squares-based reaction control of space manipulators," *J. Guid., Control, Dyn.*, vol. 35, no. 3, pp. 976–986, May 2012.

[45] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, 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 *Proc. 41st Int. Symp. Robot. (ISR), 6th German Conf. Robot. (ROBOTIK)*, Jun. 2010, pp. 1–8.

[46] *Lightweight Robot 4+ Specification, Version: Spez LBR 4+ V2en*, KUKA, Augsburg, Germany, 2010.

[47] L. Woliński and M. Wojtyra, "Comparison of dynamic properties of two LWR 4+ robots," in *Proc. 21st CISM-IFToMM Symp. Robot Design, Dyn. Control (ROMANSY)*, vol. 569, Jun. 2016, pp. 413–420.

[48] K. J. Waldron and J. Schmiedeler, *Kinematics*. Cham, Switzerland: Springer, 2016, pp. 11–36.

[49] *MATLAB Documentation: Quadprog*, Mathworks, Natick, MA, USA, 2020.

[50] L. S. Blackford, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, and M. Heroux, "An updated set of basic linear algebra subprograms (BLAS)," *ACM Trans. Math. Softw.*, vol. 28, no. 2, pp. 135–151, Jun. 2002.

[51] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1999.

[52] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *Int. J. Robust Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[53] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Math. Program. Comput.*, vol. 6, no. 4, pp. 327–363, 2014.

[54] H. J. Ferreau, A. Potschka, and C. Kirches. (2017). *qpOASES*. [Online]. Available: http://www.qpOASES.org/

[55] M. Wojtyra and L. Woliński, "Proposition of on-line velocity scaling algorithm for task space trajectories," in *ROMANSY 23—Robot Design, Dynamics and Control*, G. Venture, J. Solis, Y. Takeda, and A. Konno, Eds. Cham, Switzerland: Springer, 2021, pp. 423–431.

ŁUKASZ WOLIŃSKI received the B.S. and M.S. degrees in robotics and control from the Warsaw University of Technology, Warsaw, Poland, in 2012 and 2013, respectively, where he is currently pursuing the Ph.D. degree in automation, electronic, and electrical engineering.

In 2015, he completed a research visit at JKU Linz. In 2018, he worked for RT Robotics Company, 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 the Warsaw University of Technology (WUT), 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 has published a book and more than 80 journal articles and conference papers. His research interests include multibody systems and robotics with its applications.

Dr. Wojtyra serves as a reviewer for several journals.