2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Predictive Inverse Kinematics: Optimizing Future Trajectory through Implicit Time Integration and Future Jacobian Estimation

Ko Ayusawa, Wael Suleiman and Eiichi Yoshida

Abstract— **This paper presents an inverse kinematics (IK) method which can control future velocities and accelerations for multi-body systems. The proposed IK method is formulated as a quadratic programing (QP) that optimizes future joint trajectories. The features of the proposed IK are: (1) the evaluation of accelerations at future time instances, (2) the trajectory representation that can implicitly integrate the time integral formula into QP, (3) the computation of future Jacobian matrices based on the comprehensive theory of differential kinematics proposed in our previous work. Those features enable a stable and fast IK computation while evaluating the future accelerations. We also conducted thorough numerical studies to show the efficiency of the proposed method.**

I. Introduction

The inverse kinematics (IK) is one of the most important mathematical foundations in robotics. The classical IK of a robot computes the joint angles that realize the desired position and orientation of the end-effector or a specific link. Since the symbolic solution is difficult to be obtained except for some special structures, the numerical approaches with solving the differential kinematics equation have been studied [1], [2]. The numerical approaches usually require the computation of the Jacobian matrix about the kinematic equation. Its efficient computation has been already established [3].

One important issue of the numerical approaches is how to overcome the numerical instability due to the singularity of the Jacobian matrix. Several algorithms have been developed to provide the solution with numerical stability [4], [5], [6]. Another focus of researches is how to simultaneously handle various types of prioritized constraints including inequality constraints. Since the closed form solution of the differential equation under inequality constraints is difficult to be obtained, some efficient and robust approaches utilize the technique of nonlinear optimization [7], [8], [9], [10]. Those IK algorithms are now widely applied to motion generation for not only industrial manipulators but also various types of multi-body systems [11], [12], [13], [14].

The main focus of the paper is how to evaluate high-order differential information such as accelerations or jerks during the IK computation, while most IK methods compute only the time-series of joint angles. The integration of such evaluation is motivated by several reasons, for instance, huge accelerations can cause damage or accelerating the wear of the actuators of a robot, since the accelerations in the generated trajectory are related to the exerted torques [10]. Moreover, minimizing the jerk reduces the excitation of the resonant frequencies of an industrial manipulator [15], [16], this is especially true for most of collaborative industrial robots (cobots), which are equipped with compliant actuators, such as Series-Elastic Actuators (SEA). The minimization of the jerk is also regarded as an important factor when generating human-like motion by a robot [17], [18], which is indicated by the minimum jerk theory relating to human hand trajectory [19].

The paper presents an IK method that can evaluate high-order differential information by predicting future joint trajectories. The predictive IK is formulated as the quadratic programing (QP) problem about the future trajectory. The problem evaluates the accelerations at future time instances, which implicitly evaluates high-order differential information. Though the optimization including accelerations or jerks has been proposed in the related work [17], it often faces the problem of inaccurate solutions as well as numerical instability, which will be shown in this paper. The highlight of the proposed method is that the time integral formula is implicitly integrated into the QP problem. This implicit time integration enables the optimization of the variables at future time instances. Another highlight is that the method also computes the Jacobian matrices at future time instances. The future Jacobian matrices are required when handling the constraints on the future instances of the desired trajectory. An efficient and simple computation of those Jacobian matrices is carried out by using the comprehensive theory of the differential kinematics [20], [21]. The proposed method has been thoroughly validated and analyzed in simulation using a planer robot and a humanoid robot, those simulation results pointed out the efficiency of the proposed method as well as solving several numerical instability issues that conventional IK methods suffer from.

*This work was partly supported by JSPS Grant-in-Aid for Scientific Research (A) Number 17H00768, and by JSPS Grant-in-Aid for Scientific Research (B) Number 18H03315..

K. Ayusawa and E. Yoshida are with CNRS-AIST JRL (Joint Robotics Laboratory), UMI3218/RL Intelligent Systems Research Institute, National Institute of Advanced Industrial Science and Technology (IS-AIST), Japan. {k.ayusawa, e.yoshida}@aist.go.jp
W. Suleiman is with Electrical and Computer Engineering Department, Faculty of Engineering, University of Sherbrooke, Canada. {Wael.Suleiman}@USherbrooke.ca

II. Predictive inverse kinematics

A. QP formulation for inverse kinematics

The inverse kinematics problem of a robot can be formulated as quadratic programing (QP):

$$\begin{aligned}\min_{\dot{\boldsymbol{q}}_t} \quad & \frac{1}{2}\dot{\boldsymbol{q}}_t^T \boldsymbol{Q}\dot{\boldsymbol{q}}_t \\ \text{subject to} \quad & \boldsymbol{J}_t\dot{\boldsymbol{q}}_t = \dot{\boldsymbol{r}}_t \\ & \dot{\boldsymbol{q}}^- \leq \dot{\boldsymbol{q}}_t \leq \dot{\boldsymbol{q}}^+\end{aligned} \tag{1}$$

where, $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the joint angle velocity, n is the number of degrees of freedom (DoF) of the robot, the subscription t means the value at t-th discretized time instance, $\boldsymbol{Q}$ is a positive semi-definite matrix, $\dot{\boldsymbol{r}} \in \mathbb{R}^6$ is the linear and angular velocity of the end-effector, $\boldsymbol{J}$ is the Jacobian matrix of the end-effector, and $\dot{\boldsymbol{q}}^+$ and $\dot{\boldsymbol{q}}^-$ are the upper and lower limit of the joint velocity. For convenience of explanation, the formulations in the following sections only show the constraint about the end-effector and the limits of joint velocities, however, note that other linear equality or inequality constraints can also be considered.

If there is no inequality constraint, the problem can be efficiently solved by using the pseudo-inverse of the equality constraint [1]. On the other hand, the inequality constraints makes it difficult to get the closed-form expression of the problem. Therefore, it is usually solved by using a QP solver [7], [8], [9], [10].

After solving **Eq.**(1), the joint angle at the next time instance can be computed by the following time integration:

$$\boldsymbol{q}_{t+1} = \boldsymbol{q}_t + \dot{\boldsymbol{q}}_t \Delta t \tag{2}$$

The motivation of the paper is to evaluate not only velocities but also accelerations of the robot. Note that the optimization in **Eq.**(1) and the time integration in **Eq.**(2) are separated; the optimization evaluates the velocities at the current time instance, and the time integration computes the joint angles at the next time instance. This inconsistency about the time instances leads the issues when evaluating the accelerations, which will be shown later in Section IV showing the numerical experiments.

B. Mathematical notations for future trajectory

This section shows the preliminary mathematical notations for the trajectory at future time instances that will be used in the sequel of the paper.

The future trajectory is represented by the discretized samples of joint angles:

$$\widetilde{\boldsymbol{q}}_N \triangleq \begin{bmatrix}\boldsymbol{q}_0{}^T & \boldsymbol{q}_1{}^T & \cdots & \boldsymbol{q}_N{}^T\end{bmatrix}^T \tag{3}$$

where, $\boldsymbol{q}_i$ indicates the joint angle at time instance t_i.

The derivatives of $\widetilde{\boldsymbol{q}}_N$ are also defined as follows.

$$\widetilde{\dot{\boldsymbol{q}}}_N = \begin{bmatrix}\dot{\boldsymbol{q}}_0{}^T & \dot{\boldsymbol{q}}_1{}^T & \cdots & \dot{\boldsymbol{q}}_N{}^T\end{bmatrix}^T \tag{4}$$

$$\widetilde{\ddot{\boldsymbol{q}}}_N = \begin{bmatrix}\ddot{\boldsymbol{q}}_0{}^T & \ddot{\boldsymbol{q}}_1{}^T & \cdots & \ddot{\boldsymbol{q}}_N{}^T\end{bmatrix}^T \tag{5}$$

In order to handle the future trajectory in the QP formulation as shown in the previous section, we introduce the following concepts: *1)* the linear trajectory parameterization representing time integration, *2)* the future Jacobian matrix for future constraints, and *3)* the tracking error from the future desired trajectory.

1) Linear trajectory parameterization representing time integration : Let us assume that the joint angle trajectories and their derivatives can be formulated with the following linear form:

$$\begin{bmatrix}\widetilde{\boldsymbol{q}}_N \\ \widetilde{\dot{\boldsymbol{q}}}_N \\ \widetilde{\ddot{\boldsymbol{q}}}_N\end{bmatrix} = \begin{bmatrix}\boldsymbol{D}_q \\ \boldsymbol{D}_{\dot{q}} \\ \boldsymbol{D}_{\ddot{q}}\end{bmatrix}\boldsymbol{\alpha} + \begin{bmatrix}\boldsymbol{d}_q \\ \boldsymbol{d}_{\dot{q}} \\ \boldsymbol{d}_{\ddot{q}}\end{bmatrix} \tag{6}$$

where, $\boldsymbol{\alpha}$ is the trajectory parameter vector. The time integration such as **Eq.**(2) can be also formulated as the above linear form. Some trajectory interpolation techniques such as B-spline interpolation are also included in the above form. The detail of the implementation of $\boldsymbol{D}_q$, $\boldsymbol{D}_{\dot{q}}$, and $\boldsymbol{D}_{\ddot{q}}$ is introduced in section II-D.

2) Future Jacobian matrix for future constraints: The constraints at future time instances require the computation of the corresponding future Jacobian matrices. In order to handle the future constraints in the QP framework, let us approximate the Jacobian matrix $\boldsymbol{J}_t$ at future time instance t_t by the following Taylor expansion:

$$\boldsymbol{J}_t \approx \widehat{\boldsymbol{J}}_t \triangleq \boldsymbol{J}_0 + (t_t - t_0)\dot{\boldsymbol{J}}_0 + \frac{1}{2}(t_t - t_0)^2\ddot{\boldsymbol{J}}_0 \tag{7}$$

where, $\widehat{\boldsymbol{J}}_t$ is the approximated matrix, and $\boldsymbol{J}_0$, $\dot{\boldsymbol{J}}_0$, and $\ddot{\boldsymbol{J}}_0$ are available at the current time instance t_0. When computing **Eq.**(7), the computation of $\dot{\boldsymbol{J}}_0$ and $\ddot{\boldsymbol{J}}_0$ is needed. This paper also introduces a simple and efficient computation of them. The details will be explained in section III.

3) Tracking error from future desired trajectory: Let us assume the following dynamics for the tracking error:

$$\begin{aligned}\dot{\boldsymbol{e}} &= -\boldsymbol{K}\boldsymbol{e} \\ \boldsymbol{e} &\triangleq \begin{bmatrix}\boldsymbol{R}f_\alpha(\widehat{\boldsymbol{R}}^T\boldsymbol{R}]) \\ \widehat{\boldsymbol{p}} - \boldsymbol{p}\end{bmatrix}\end{aligned} \tag{8}$$

where, $\boldsymbol{p}$ and $\boldsymbol{R}$ are the position and orientation of the end-effector, and $\widehat{\boldsymbol{p}}$ and $\widehat{\boldsymbol{R}}$ are the desired ones, $\boldsymbol{K}$ is a diagonal and positive semi-definite matrix, and the operator f_α computes the angle axis vector from the rotation matrix.

In this paper, let us assume that the angle axis in the piecewise trajectory is constant and the error dynamics can be represented as the following closed form:

$$\widehat{\boldsymbol{e}}_t = e^{\boldsymbol{K}(t_t - t_0)}\boldsymbol{e}_{t_0} \tag{9}$$

The time derivative can be written by:

$$\dot{\widehat{\boldsymbol{e}}}_t = \boldsymbol{K}e^{\boldsymbol{K}(t_t - t_0)}\boldsymbol{e}_{t_0} \tag{10}$$

This dynamics prevents the sudden change of the joint angle in the optimization.

C. QP formulation for predictive IK

The proposed method solves the following QP problem:

$$\min_{\boldsymbol{\alpha}, \forall t\ \boldsymbol{s}_t} \frac{1}{2}\widetilde{\dot{\boldsymbol{q}}}_N{}^T\boldsymbol{Q}_q\widetilde{\dot{\boldsymbol{q}}}_N + \frac{1}{2}\widetilde{\ddot{\boldsymbol{q}}}_N{}^T\boldsymbol{Q}_{\dot{q}}\widetilde{\ddot{\boldsymbol{q}}}_N + \frac{\lambda}{2}\sum_t \boldsymbol{s}_t{}^T\boldsymbol{s}_t \tag{11}$$

$$\text{subject to } \forall t \quad \hat{\boldsymbol{J}}_t \dot{\boldsymbol{q}}_t = \dot{\hat{\boldsymbol{r}}}_t + \dot{\hat{\boldsymbol{e}}}_t + \boldsymbol{s}_t$$
$$\dot{\boldsymbol{q}}^- \le \dot{\boldsymbol{q}}_t \le \dot{\boldsymbol{q}}^+$$
$$\begin{bmatrix} \boldsymbol{q}_0 \\ \dot{\boldsymbol{q}}_0 \\ \ddot{\boldsymbol{q}}_0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_0^{init} \\ \dot{\boldsymbol{q}}_0^{init} \\ \ddot{\boldsymbol{q}}_0^{init} \end{bmatrix}$$

where, $\boldsymbol{s}_t$ is the slack variable in order to relax the equality constraint about the tracking error, $\dot{\hat{\boldsymbol{r}}}_t$ is the desired linear and angular velocity of the end-effector, and $\boldsymbol{Q}_q$ and $\boldsymbol{Q}_{\dot{q}}$ are positive semi-definite matrices.

The QP problem in **Eq.**(11) considers the boundary conditions about the joint angle, velocity, and acceleration at the current time instance t_0 whose values have to be equal to $\boldsymbol{q}_0^{init}$, $\dot{\boldsymbol{q}}_0^{init}$, and $\ddot{\boldsymbol{q}}_0^{init}$, respectively. Though the boundary constraints have to be satisfied strictly, the other equality constraints are relaxed by using the slack variables in order to make **Eq.**(11) solvable.

D. Trajectory representation with implicit time integration

The implementation of the trajectory parameterization shown in **Eq.**(6) affects the performance as well as the computational speed of the inverse kinematics problem in **Eq.**(11). This section introduces the two methods for trajectory parameterization.

1) Parameterization based on B-splines: Let us define the following cubic B-spline basis function $b_i(\tau)$:

$$b_{pos}(\tau) = \begin{cases} \frac{1}{6}\{(2-|\tau|)^3 - 4(1-|\tau|)^3\} & (0 \le |\tau| \le 1) \\ \frac{1}{6}(2-|\tau|)^3 & (1 \le |\tau| \le 2) \\ 0 & (otherwise) \end{cases} \tag{12}$$

Let the joint angle at time instance t_t be modeled by N_B set of cubic B-spline bases as follows:

$$\boldsymbol{q}_t = \sum_{i=1}^{N_B} b_{pos}\left(\tfrac{t_t}{h} - i + 2\right) \boldsymbol{c}_i \tag{13}$$
$$h \triangleq \frac{t_N - t_0}{N_B - 3}$$

where, $\boldsymbol{c}_i \in \mathbb{R}^n$ is the coefficient vector of i-th basis. Each basis $b_i(\tau)$ is aligned at regular intervals, and h is the interval between two bases. The overview of the alignment is shown in **Fig.** 1.

In this case, the parameters $\boldsymbol{\alpha}$, $\boldsymbol{D}_q$ and $\boldsymbol{d}_q$ in **Eq.**(6) can be formulated as follows:

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{c}_1{}^T & \boldsymbol{c}_2{}^T & \cdots & \boldsymbol{c}_{N_B}{}^T \end{bmatrix}^T \tag{14}$$
$$\boldsymbol{D}_q{}^{(t,i)} = b_{pos}\left(\tfrac{t_t}{h} - i + 2\right) \boldsymbol{I} \tag{15}$$
$$\boldsymbol{d}_q = \boldsymbol{0} \tag{16}$$

where, $\boldsymbol{I}$ is an identity matrix, and $\boldsymbol{D}_q{}^{(t,i)} \in \mathbb{R}^{n \times n}$ is the submatrix in the t-th row and j-th column block matrix of $\boldsymbol{D}_q$ As can be seen from **Eq.**(12), $\boldsymbol{D}_q$ has a sparse structure.

The first and second derivatives of **Eq.**(13) can be obtained by replacing the basis function b_{pos} with the following

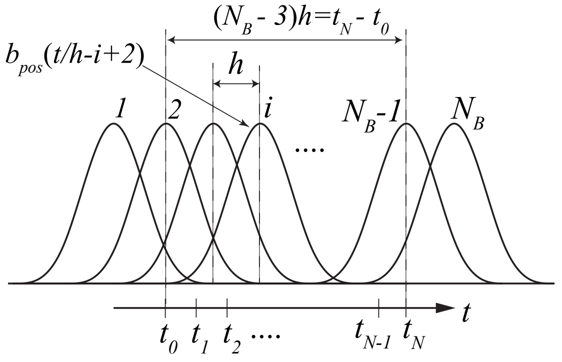

Fig. 1. Relationship between the discretized time instances and the location of cubic B-spline bases.

functions:

$$b_{vel}(\tau) = \begin{cases} \frac{sign(\tau)}{2}\{(2-|\tau|)^2 - 4(1-|\tau|)^2\} & (0 \le |\tau| \le 1) \\ \frac{sign(\tau)}{2}(2-|\tau|)^2 & (1 \le |\tau| \le 2) \\ 0 & (otherwise) \end{cases} \tag{17}$$

$$b_{acc}(\tau) = \begin{cases} (2-|\tau|) - 4(1-|\tau|) & (0 \le |\tau| \le 1) \\ (2-|\tau|) & (1 \le |\tau| \le 2) \\ 0 & (otherwise) \end{cases} \tag{18}$$

Therefore, we can have:

$$\boldsymbol{D}_{\dot{q}}{}^{(t,i)} = b_{vel}\left(\tfrac{t_t}{h} - i + 2\right) \boldsymbol{I} \tag{19}$$
$$\boldsymbol{D}_{\ddot{q}}{}^{(t,i)} = b_{acc}\left(\tfrac{t_t}{h} - i + 2\right) \boldsymbol{I} \tag{20}$$
$$\boldsymbol{d}_{\dot{q}} = \boldsymbol{d}_{\ddot{q}} = \boldsymbol{0} \tag{21}$$

The main feature of this implementation is that the trajectory in the more distance future can be considered by increasing the number of N_B, however, also increasing the computational cost.

2) Parameterization based on Newmark-β method: The joint angle and the derivatives at the next time instance t_1 can be computed from the information at the current time instance t_0 by the Newmark-β method [22] as follows:

$$\begin{bmatrix} \boldsymbol{q}_1 \\ \dot{\boldsymbol{q}}_1 \\ \ddot{\boldsymbol{q}}_1 \end{bmatrix} = \begin{bmatrix} \beta \Delta t^2 \boldsymbol{I} \\ \gamma \Delta t \boldsymbol{I} \\ \boldsymbol{I} \end{bmatrix} \ddot{\boldsymbol{q}}_1 + \begin{bmatrix} \boldsymbol{d}_\beta \\ \boldsymbol{d}_\gamma \\ \boldsymbol{0} \end{bmatrix} \tag{22}$$

where, $\boldsymbol{d}_\beta$ and $\boldsymbol{d}_\gamma$ are defined as:

$$\boldsymbol{d}_\gamma = \boldsymbol{q}_0 + \Delta t \dot{\boldsymbol{q}}_0 + \frac{1}{2}(1 - 2\beta)\Delta t^2 \ddot{\boldsymbol{q}}_0$$
$$\boldsymbol{d}_\beta = \dot{\boldsymbol{q}}_0 + (1 - \gamma)\Delta t \ddot{\boldsymbol{q}}_0$$

Unlike the case of using B-splines parameterization, **Eq.**(22) is formulated in the closed form with respect to $\boldsymbol{q}_0$, $\dot{\boldsymbol{q}}_0$, and $\ddot{\boldsymbol{q}}_0$. The boundary conditions in **Eq.**(11) can be eliminated by substituting $\boldsymbol{q}_0 = \boldsymbol{q}_0^{init}$, $\dot{\boldsymbol{q}}_0 = \dot{\boldsymbol{q}}_0^{init}$ and $\ddot{\boldsymbol{q}}_0 = \ddot{\boldsymbol{q}}_0^{init}$ into **Eq.**(22). The optimization problem can be finally simplified as follows:

$$\min_{\ddot{q}_1, \boldsymbol{s}_1} \ \ddot{\boldsymbol{q}}_1 \boldsymbol{Q}_{\ddot{q}} \ddot{\boldsymbol{q}}_1 + \frac{1}{2}(\gamma \ddot{\boldsymbol{q}}_1 + \boldsymbol{d}_\gamma)^T \boldsymbol{Q}_{\dot{q}} (\gamma \ddot{\boldsymbol{q}}_1 + \boldsymbol{d}_\gamma) + \frac{\lambda}{2} \boldsymbol{s}_1{}^T \boldsymbol{s}_1 \tag{23}$$

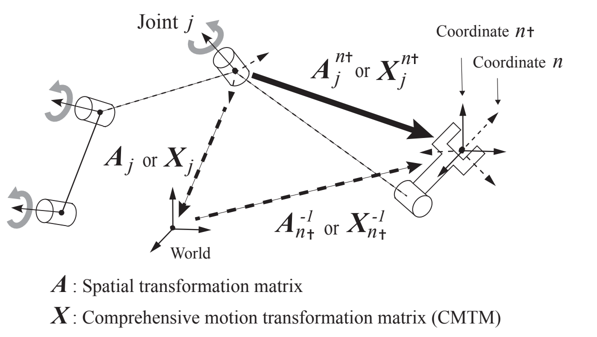
Joint j
Coordinate $n\dagger$
Coordinate n
$A_j^{n\dagger}$ or $X_j^{n\dagger}$
A_j or X_j
$A_{n\dagger}^{-1}$ or $X_{n\dagger}^{-1}$
World
A : Spatial transformation matrix
X : Comprehensive motion transformation matrix (CMTM)

Fig. 2. Overview of coordinate transformation in the Jacobian matrix computation.

$$\text{subject to} \quad \widehat{J}_1(\gamma\ddot{q}_1 + d_\gamma) = \dot{\hat{r}}_1 + \dot{\hat{e}}_1 + s_1$$
$$\dot{q}^- \le \gamma\ddot{q}_1 + d_\gamma \le \dot{q}^+$$

The merit of the formulation in **Eq.**(23) is that the number of optimized variables is the same as that in the case of the standard IK with slack variables, which is useful for applications that require fast computation.

III. Computation of Jacobian derivatives

A. Computation of Jacobian matrix

At first, this section introduces the computation of the Jacobian matrix according to the spatial algebra [23]. Let us consider the end-effector's Jacobian matrix of a serial manipulator with rotational joints for convenience of explanation, though the following formulation can be generalized for any link's Jacobian matrix of a multi-body system.

The Jacobian matrix of the global position and orientation of the end-effector can be represented by:

$$J = \begin{bmatrix} J_1 & \cdots & J_n \end{bmatrix} \tag{24}$$

where, $J_j \in \mathbb{R}^{6\times1}$ is the block matrix corresponding j-th joint.

Each block matrix J_j can be computed as follows:

$$J_j = A_j^{n\dagger} K_j \tag{25}$$

- A_j is the spatial transformation matrix of the coordinate system of joint j defined as:

$$A_j \triangleq \begin{bmatrix} R_j & O_3 \\ [p_j\times_3]\,R_j & R_j \end{bmatrix} \tag{26}$$

  where, p_j and R_j are the position and orientation of the coordinate system of joint j, respectively.
- The skew operator is represented as follows:

$$[x\times_3] \triangleq \begin{bmatrix} 0 & -x_{(3)} & x_{(2)} \\ x_{(3)} & 0 & -x_{(1)} \\ -x_{(2)} & x_{(1)} & 0 \end{bmatrix}$$

- The inverse of A_j can be computed by the inverse spatial transformation:

$$A_j^{-1} = \begin{bmatrix} R_j^T & O_3 \\ -R_j^T[p_j\times_3] & R_j^T \end{bmatrix} \tag{27}$$

- A_k^j means the relative spatial transformation matrix from coordinate j to k defined as:

$$A_k^j \triangleq A_j^{-1} A_k \tag{28}$$

- The coordinate $n\dagger$ means the system whose origin is the same as the joint coordinate n but whose orientation is equal to the global frame (i.e. $R_n = I$), as shown in **Fig.** 2.
- K_j represents the constant matrix defined according to the type of joint j. If joint j is rotational, K_j is given by:

$$K_j \triangleq \begin{bmatrix} a_j^T & 0^T \end{bmatrix}^T \tag{29}$$

  where, $a_j \in \mathbb{R}^3$ designs the joint axis direction.

After computing the forward kinematics of the robot, each block matrix J_j in J can be directly computed from **Eq**.(25). Though **Eq**.(25) is formulated by using spatial formulations in this paper, it is essentially equivalent to the formulation of the basic Jacobian matrix [3].

B. Computation of the derivatives of Jacobian matrix

The time derivatives of the Jacobian matrix can be simply computed by using 18 dimensional transformation matrix that is called the *comprehensive motion transformation matrix (CMTM)* [20], [21]. CMTM is the extended form of the spatial transformation matrix A in order to transform not only position and orientation but also linear and angular velocities and accelerations. CMTM has many similar mathematical features to those of A, and the simple chain product of CMTMs leads to the forward kinematics computation including velocities and accelerations. The relationship between CMTM and forward kinematics including differential kinematics is detailed in [21].

CMTM of the coordinate system of joint j is defined as follows:

$$X_h \triangleq \begin{bmatrix} A_j & O_6 & O_6 \\ A_j\,[v_j\times_6] & A_j & O_6 \\ \frac{1}{2}A_j\left([\dot{v}_j\times_6] + [v_j\times_6]^2\right) & A_j\,[v_j\times_6] & A_j \end{bmatrix} \tag{30}$$

where, v_j is the spatial velocity of the coordinate of joint j, and $[*\times_6]$ indicates the matrix form of spatial cross product:

$$[v\times_6] \triangleq \begin{bmatrix} [\omega\times_3] & O_3 \\ [\nu\times_3] & [\omega\times_3] \end{bmatrix}$$

where,

$$v \triangleq \begin{bmatrix} \omega^T & \nu^T \end{bmatrix}^T$$

By using CMTMs, J_j and its derivatives (i.e. $\dot{J}_j$ and $\ddot{J}_j$) can be computed in the similar manner to **Eq**.(25). Let us consider the following matrix.

$$\widetilde{J}_j \triangleq \begin{bmatrix} J_j & O & O \\ \dot{J}_j & J_j & O \\ \frac{1}{2}\ddot{J}_j & \dot{J}_j & J_j \end{bmatrix} \tag{31}$$

It can be simply computed by:

$$\widetilde{\boldsymbol{J}}_j = \boldsymbol{X}_j^{n\dagger}\boldsymbol{G}_j \tag{32}$$

where,

- $\boldsymbol{X}_k^j$ means the relative CMTM from coordinate j to k defined as:

$$\boldsymbol{X}_k^j \triangleq \boldsymbol{X}_j{}^{-1}\boldsymbol{X}_k \tag{33}$$

- The inverse of $\boldsymbol{X}_j$ is computed by the inverse transformation without computing the inverse matrix directly.

$$\boldsymbol{X}_j{}^{-1} = \begin{bmatrix} \boldsymbol{A}_j{}^{-1} & \boldsymbol{O}_6 & \boldsymbol{O}_6 \\ -[\boldsymbol{v}_j\times_6]\boldsymbol{A}_j{}^{-1} & \boldsymbol{A}_j{}^{-1} & \boldsymbol{O}_6 \\ -\frac{1}{2}\left([\dot{\boldsymbol{v}}_j\times_6]-[\boldsymbol{v}_j\times_6]^2\right)\boldsymbol{A}_j{}^{-1} & -[\boldsymbol{v}_j\times_6]\boldsymbol{A}_j{}^{-1} & \boldsymbol{A}_j{}^{-1} \end{bmatrix} \tag{34}$$

- $\boldsymbol{G}_j$ represents the constant matrix defined according to the type of joint j such as:

$$\boldsymbol{G}_j = \begin{bmatrix} \boldsymbol{K}_j & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{K}_j & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{K}_j \end{bmatrix} \tag{35}$$

Each block matrix $\boldsymbol{J}_j$ in $\boldsymbol{J}$ and its derivatives ($\dot{\boldsymbol{J}}_j$ and $\ddot{\boldsymbol{J}}_j$) are obtained as the components of $\widetilde{\boldsymbol{J}}_j$ that is simply computed from **Eq.**(32). After computing the forward kinematics including velocities and accelerations, the derivatives are directly obtained from the variables of joint j and n.

It should be noted that the Jacobian matrices in the paper are represented with respect to the global coordinates (i.e. coordinate $n\dagger$); on the other hand, the formulations in [21] are with respect to the local coordinates (i.e. coordinate n).

Though several works introduce the derivatives of the kinematics computation according to the recursive formula about the partial derivatives [17], [24], the computation shown in **Eq.**(32) can simply and directly compute each block matrix of $\boldsymbol{J}$, $\dot{\boldsymbol{J}}$ and $\ddot{\boldsymbol{J}}$ similarly to the computation of basic Jacobian.

CMTM is useful as a mathematical tool that simplifies the differential formulations. On the other hand, the direct computation of 18×18 matrix products should be avoided in the actual implementation by utilizing the sparse and symmetric structure of CMTM (similarily to the case when computing the products of spatial transformation matrices).

The computational complexity of computing $\boldsymbol{J}$, $\dot{\boldsymbol{J}}$ and $\ddot{\boldsymbol{J}}$ is $O(N_J)$ respectively, where N_J is the number of joints. Their computational time is usually small with respect to the time of solving QP shown in **Eq.**(11) or **Eq.**(23).

IV. Numerical Evaluation

A. IK scenario with planner manipulator

This section presents a specific IK scenario of a planner manipulator when the standard method (i.e. **Eq.**(1)) faces the issues due to the limits of joint velocity. The planner manipulator has 4 rotational joints and the length of each segment is 1 m as shown in **Fig.** 3. The initial joint angles of the manipulator are:

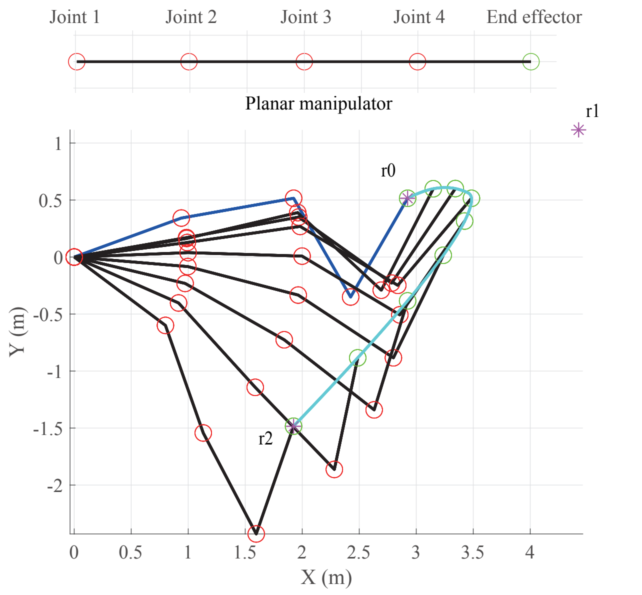

Fig. 3. Planar 4DOF manipulator. The upper figure shows the manipulator when all the joint angles are set to zero. The lower figure illustrates the reference trajectories which are defined by the three control points.

$$\boldsymbol{q}_0 = \begin{bmatrix}20 & -10 & -70 & 120\end{bmatrix} (deg)$$

The position of end-effector at the initial joint angles is defined as $\boldsymbol{r}_0$. The desired trajectory of the end-effector $\boldsymbol{r}(t)$ is given as the Bezier curve defined by three control points $\boldsymbol{r}_0$, $\boldsymbol{r}_1$, and $\boldsymbol{r}_2$:

$$\boldsymbol{r}(t) = \frac{1}{{T_e}^2}((1-t)^2\boldsymbol{r}_0 + 2t(1-t)\boldsymbol{r}_1 + t^2\boldsymbol{r}_2) \quad (0 \le t \le T_e)$$
$$\boldsymbol{r}_1 = \boldsymbol{r}_0 + \begin{bmatrix}1.5 & 0.6 & 0\end{bmatrix}^T \ (m)$$
$$\boldsymbol{r}_2 = \boldsymbol{r}_0 - \begin{bmatrix}1 & 2 & 0\end{bmatrix}^T \ (m)$$

where, $T_e (= 4\ \mathrm{s})$ is the end time of the trajectory. The trajectory is discretized with time-step $\Delta T = 0.005\ s$. The overview of the desired trajectory is illustrated in **Fig.** 3. The upper and lower limits about joint velocities are also considered:

$$\dot{\boldsymbol{q}}^+ = -\dot{\boldsymbol{q}}^- = 0.5\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}^T$$

In this IK scenario, the robot could not follow the desired trajectory due to its velocity limits, which can be seen from **Fig.** 4 where the reference trajectories and the generated ones are shown. The results of the tracking errors and the joint velocities with the standard method formulated in **Eq.**(1) are given in **Fig.** 5. Though the standard method could solve QP at each time instance, the generated trajectories of joint velocities are vibrating when the desired position cannot be tracked due to the velocity limits.

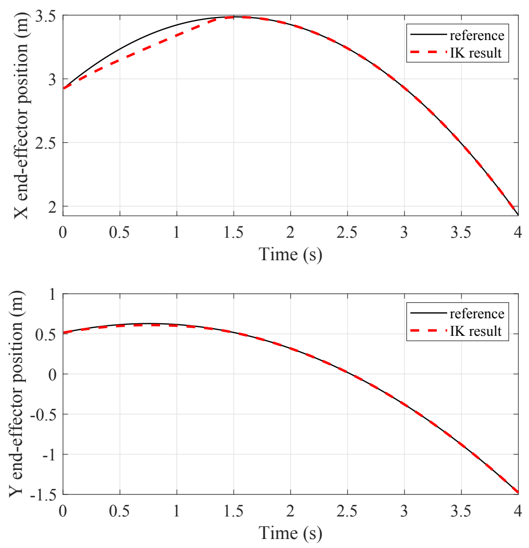

Fig. 4. Reference trajectories and the trajectories generated by standard IK.

B. Comparative analysis with planner manipulator

This section presents a comparison of IK methods in order to clarify the features of the proposed method. The reason of the velocity vibrations shown in **Fig.** 5 is obvious, this is because the standard method cannot evaluate the accelerations. The IK approach with evaluating accelerations is expected to generate smoother joint velocity trajectories. On the other hand, the prediction of future trajectories is important to generate not only smooth but also accurate results.

In order to clarify this issue, the proposed method was compared to the IK method shown in [10], which is most similar to the proposed method. Though the method in [10] is formulated as jerk optimization, it is actually equivalent to the optimization about the future acceleration at next time instance. One main difference from the proposed method is that the method in [10] uses the Jacobian matrices at a current time instance for constraints at the next time instance. Therefore, the method cannot consider the implicit time integration like the Newmark-β method, which leads to poor prediction of future accelerations. The explicit time integration leads the poor prediction of future accelerations.

The results using the method in [10] are shown in **Fig.** 6. The smooth joint velocities trajectories were generated unlike the case shown in **Fig.** 5. However, **Fig.** 6 shows the large tracking errors of the end-effector position after $2s$ even though the trajectory could be tracked in the case of **Fig.** 5. The results by using the proposed method with the Newmark-β method ($\beta = 1/2, \gamma = 11/12$) in **Fig.** 7 shows that the end-effector could track the desired trajectory after $2s$. Therefore, the prediction of future Jacobian matrices is important for the accuracy of the IK results.

We also tested the trajectories interpolation using B-splines, we compare the results to the case of the Newmark-β method which is shown in **Fig.** 7. The number of B-spline bases was $N_B = 5$ and the timespan between the two bases was $h = 0.02\ s$. The bases are used to estimate $(N =)8$ future time steps (i.e. $0.005, 0.01, ..., 0.04$ s in the future). The results using B-splines are shown in **Fig.** 8, one can observe smaller accelerations than the case of the Newmark-β method. It is because the Newmark-β only predicts one future time step, whereas 8 future time steps were predicted in the case of **Fig.** 8. On the other hand, the number of variables of the QP problem in the case of **Fig.** 8 is 5 times larger than that of **Fig.** 7.Therefore, the Newmark-β method will be effective especially for on-line applications, while the prediction using B-splines will be useful for precise off-line applications.

C. Motion retargeting on humanoid

The proposed method was also tested on the motion retargeting: the conversion from human motion to humanoid motion. Since the measured human motion data contains noises, the standard IK leads large accelerations. This section evaluates whether the proposed method can generate smoother joint trajectories.

The measured postures of human hands during a valve opening task were converted to the joint trajectories of the upper-body of humanoid robot HRP-4 [25]. The human motion data was recorded in advance and the conversion was performed by off-line computation. The inequality constraints about the upper and lower limits of joint angles and velocities of the robot were also considered in the IK computation.

The snapshots of the generated postures are shown in **Fig.** 9. The generated trajectories of joint velocities are also shown in **Fig.** 10. The upper graph in **Fig.** 10 shows the results of joint velocities in the case of the standard IK, and the lower graph shows those of the proposed IK with the Newmark-β method. As can be seen from the figures, the proposed method could generate smoother trajectories thanks to the optimization of future accelerations.

V. Conclusion

In this paper, we proposed a novel method for inverse kinematics that can evaluate accelerations at future time instances. The proposed method is formulated as QP problem of optimizing the future joint angles to track the reference trajectory. The highlighted features of the proposed method are:

- The prediction can generate smooth trajectories even under the inequality constraints.
- The time integral formula for joint angles is incorporated into the QP problem by using the trajectory parameterization. The implicit time integration like the Newmark-β method can be utilized in the QP, which leads to numerically stable future trajectories.
- The Jacobian matrices at future time instances are predicted by the Taylor expansion using the Jacobian derivatives. A simple and efficient computation of those

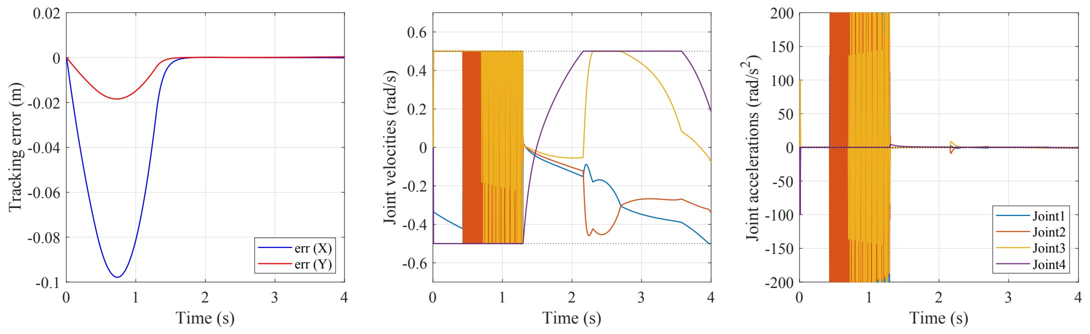

Fig. 5. Results of standard IK. The left figure shows the Cartesian tracking error from the reference trajectory. The middle and right ones show the generated joint velocities and accelerations, respectively.

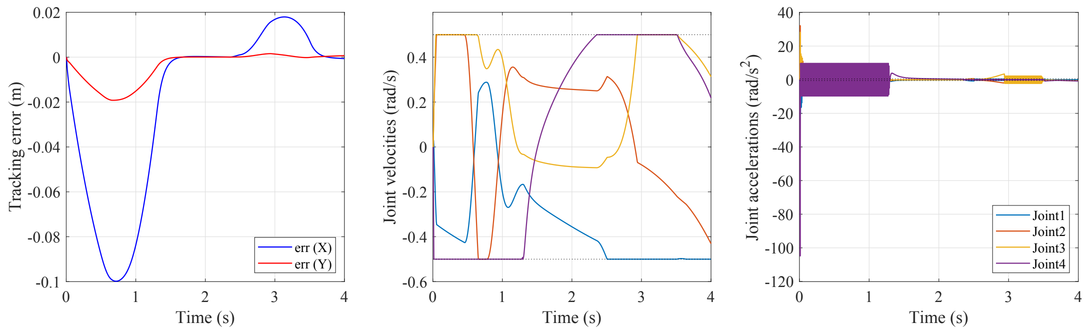

Fig. 6. Results of the IK method shown in [10]. The graphs and line types are the same as those in **Fig.** 5.

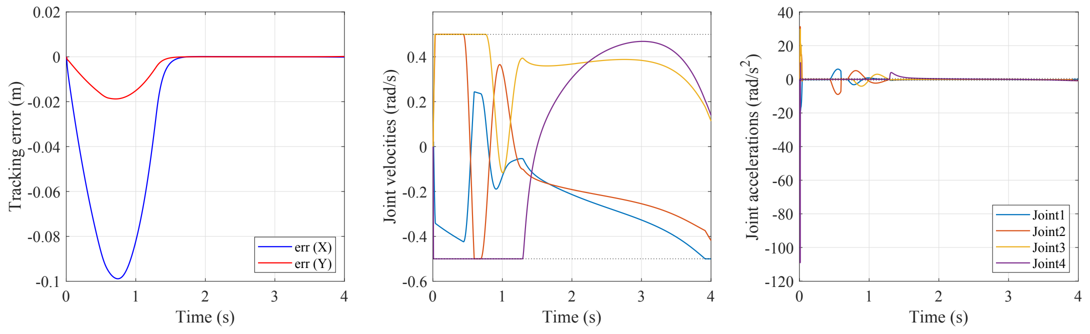

Fig. 7. Results of the proposed IK with Newmark-β integration. The graphs and line types are the same as those in **Fig.** 5.

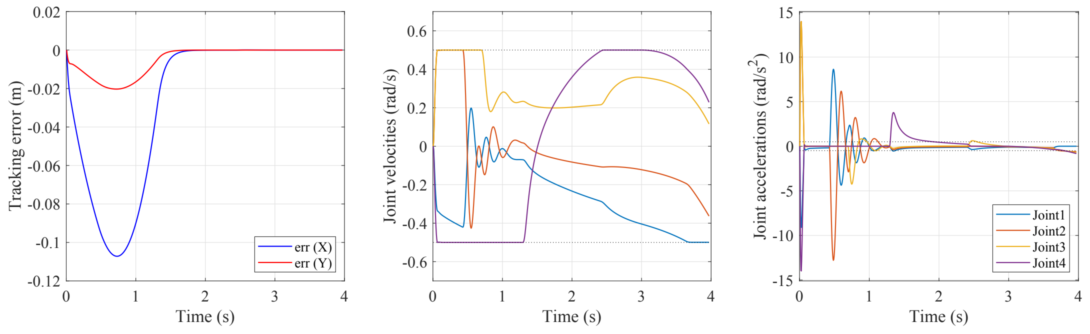

Fig. 8. Results of the proposed IK with trajectory parameterization by B-splines. The graphs and line types are the same as those in **Fig.** 5.

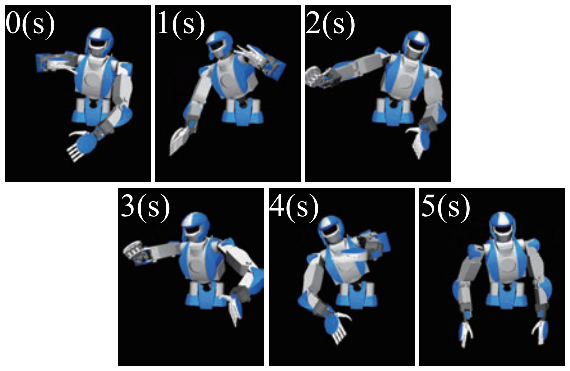

Fig. 9. Snapshots of the generated motion of humanoid HRP-4.

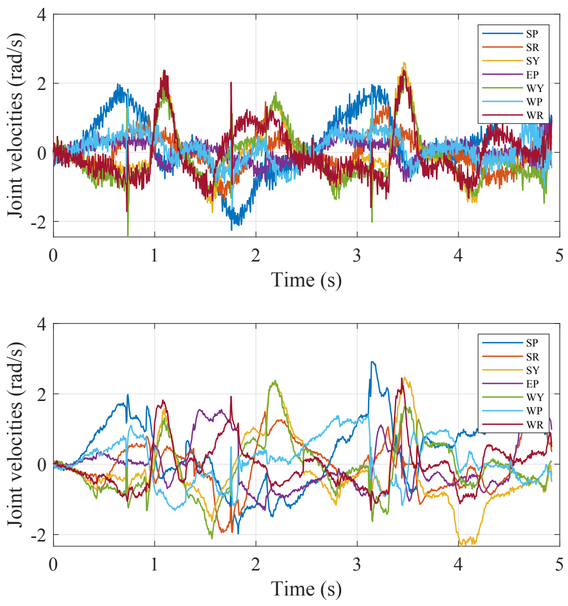

Fig. 10. Comparison of the trajectories of joint velocities between the standard IK and the proposed IK. The upper graph shows the result of the standard IK, and the lower one shows those of the proposed IK.

derivatives are introduced using the comprehensive motion transformation [21].

The proposed method was compared to several inverse kinematics methods by numerical tests using a planner manipulator and a humanoid robot. The proposed method showed better tracking performance as well as smoother joint trajectories.

References

[1] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Addison Wesley Publishing Company, 1991.

[2] D. Whiteny, "The mathematics of coordinated control of prosthetic arms and manipulators," *Trans. of the ASME, Journal of Dynamic Systems, Measurement, and Control*, vol. 94, no. 4, pp. 303–309, 1972.

[3] O. Khatib, "A unified approach for motion and force control of robotic manipulators: the operational space formulation." *IEEE J. Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[4] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *ASME J. of Dynamic Systems, Measurement, and Control*, vol. 108, no. 4, pp. 163–171, 1986.

[5] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[6] T. Sugihara, "Solvability-unconcerned inverse kinematics by the levenberg–marquardt method," *IEEE Trans. on Robotics*, vol. 27, no. 5, pp. 984–991, 2011.

[7] O. Kanoun, F. Lamiraux, P.-B. Wieber, F. Kanehiro, E. Yoshida, and J.-P. Laumond, "Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 2939–2944.

[8] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond, "A local collision avoidance method for non-strictly convex polyhedra," in *Proc. of Robotics: Science and Systems*, 2009.

[9] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Int. J. of Robotic Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[10] W. Suleiman, "On inverse kinematics with inequality constraints: new insights into minimum jerk trajectory generation," *Advanced Robotics*, vol. 30, no. 17-18, pp. 1164–1172, 2016.

[11] Y. Umetani and K. Yoshida, "Resolved motion rate control of space manipulators with generalized jacobian matrix," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 3, pp. 303–314, 1989.

[12] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Resolved momentum control: humanoid motion planning based on the linear and angular momentum," in *Proc. of the 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2003, pp. 1644–1650.

[13] K. Yamane and Y. Nakamura, "Natural motion animation through constraining and deconstraining at will," *IEEE Trans. on Visualization and Computer Graphics*, vol. 9, no. 3, pp. 352–360, 2003.

[14] K. Ayusawa and Y. Nakamura, "Fast inverse kinematics algorithm for large DOF system with decomposed computation of gradient and its application to musculoskeletal model," in *Proc. of the 17th Robotics Symposia*, 2012, pp. 148–155.

[15] J. J. Craig, *Introduction to Robotics*. New York: Addison-Wesley, 1989.

[16] A. Piazzi and A. Visioli, "Global minimum-jerk trajectory planning of robot manipulators," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 1, pp. 140–149, 2000.

[17] W. Suleiman, E. Yoshida, F. Kanehiro, J.-P. Laumond, and A. Monin, "Human motion imitation by humanoid robot," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2008, pp. 2967–2704.

[18] K. Ayusawa and E. Yoshida, "Motion retargeting for humanoid robots based on simultaneous morphing parameter identification and motion optimization," *IEEE Trans. on Robotics*, vol. 33, no. 6, pp. 1343–1357, 2017.

[19] N. Hogan, "An organizing principle for a class of voluntary movements," *Journal of Neuroscience*, vol. 4, no. 11, pp. 2745–2754, 1984.

[20] K. Ayusawa and E. Yoshida, "Comprehensive theory of differential kinematics and dynamics for motion optimization," in *Proc. of Robotics: Science and Systems*, 2017.

[21] ——, "Comprehensive theory of differential kinematics and dynamics towards extensive motion optimization framework," *Int. J. of Robotics Research*, vol. 37, no. 13-14, pp. 1554–1572, 2018.

[22] N. Newmark, "A method of computation for structural dynamics," *ASCE Journal of Engineering Mechanics Division*, vol. 85, no. 3, pp. 67–94, 1959.

[23] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer, 2008.

[24] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Proc. of Robotics: Science and Systems*, 2018.

[25] K. Kaneko, F. Kanehiro, M. Morisawa, K. Akachi, G. Miyamori, A. Hayashi, and N. Kanehira, "Humanoid robot HRP-4 - humanoid robotics platform with lightweight and slim body," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2011, pp. 4400–4407.