Fast Inverse Kinematics Based on Pseudo-Forward Dynamics Computation: Application to Musculoskeletal Inverse Kinematics

Ko Ayusawa, *Member, IEEE*, Akihiko Murai, Ryusuke Sagawa, *Member, IEEE*, and Eiichi Yoshida, *Fellow, IEEE*

***Abstract*—Recently, fast and practical inverse kinematics (IK) methods for complicated human models have gained considerable interest owing to the spread of convenient motion-capture or human-augmentation technologies. Although the IK algorithms developed in robotics can also be applied to humans, they experience computational speed issues, especially in real-time applications.**

This letter presents a new IK algorithm based on the Levenberg–Marquardt (LM) method, LM-PFD (Pseudo-Forward Dynamics), which is remarkably effective particularly in systems with a large degree of freedom (DoF). In the proposed method, the $O(N)$ forward dynamics algorithm is utilized by introducing a virtual dynamical system derived from damping or weighing factors used in the LM method. The letter firstly introduces the basic implementation of LM-PFD for open kinematic chains. Subsequently, an enhanced implementation is presented to address closed kinematic chains, specifically focusing on wire-driven systems. The proposed method was tested on the IK of musculoskeletal models. The computational time of the model with approximately 150 DoF and 300 wires was within 5 ms.

***Index Terms*—Kinematics, human and humanoid motion analysis and synthesis, dynamics.**

I. Introduction

INVERSE kinematics (IK) is one of the most important mathematical foundations in the field of robotics [1], [2], [3], [4], [5]. In classical IK, when controlling the motion of a robotic manipulator, the joint angle of the robot is calculated to achieve the desired position of its end effector. IK computation can be applied to various redundant and multiple degrees of freedom (DoF) systems that are not limited to robotic manipulators. As it is especially effective for articulated body systems, IK computation is often used for controlling humanoid robots [6], [7] or the motion analysis of humans and animals [8], [9]. For estimating human joint movements from motion capture, for example, in sports or rehabilitation, the IK of a complex human model, including musculoskeletal models, has been studied [10], [11], and efficient algorithms have been developed in robotics attract attention.

Recently, the demand for fast and practical IK algorithms for complex human models has been increasing. The emergence of convenient motion-capturing technologies such as video motion capture has resulted in the collection of large amounts of human motion data [12], [13], [14], [15]. While AI technologies are expected to be applied for various applications related to human musculoskeletal motion, they usually require a large amount of learning dataset on musculoskeletal motion, indicating the massive process of IK computation according to a motion capture dataset. The upcoming development of human augmentation technologies such as robotic assistive devices [16], [17] also requires fast IK algorithms. As robotics devices are often controlled when measuring and estimating human movements, real-time IK computation for the human body has a key role in this field. There are several examples of computing IK of musculoskeletal models in real-time for visual feedback [18]. However, in terms of intervention or assistance toward human motion, the computational speed of these approaches must still be improved to perform real-time IK, for example, within the control loop of robotic devices.

Robotics IK computation usually requires solving nonlinear equations of the generalized coordinates of the kinematic chain, thus achieving a desired position and orientation of the target links. The nonlinear equations can be solved by several numerical methods, such as the Newton–Raphson method, while computing a basic Jacobian matrix [19] derived from the differential kinematics. However, in the case of a large-DoF system, such as humanoids or human models, the IK computation often faces computational issues at the singular posture of the system and when an equation does not admit a solution owing to practical matters, such as measurement noises or infeasible references. For computational stability, the problem of solving nonlinear equations is often treated as a nonlinear optimization problem

Manuscript received 10 March 2023; accepted 21 June 2023. Date of publication 31 July 2023; date of current version 8 August 2023. This letter was recommended for publication by Associate Editor Y. She and Editor L. Pallottino upon evaluation of the reviewers' comments. This work was supported in part by JSPS Grant-in-Aid for Scientific Research under Grants (S) 22H05002 and (A) 22H00545, in part by JST PRESTO Grant Number JPMJPR19JA, in part by JST Moonshot R&D under Grant JPMJMS2239, and in part by a New Energy and Industrial Technology Development Organization (NEDO) under Project JPNP20006. *(Corresponding author: Ko Ayusawa.)*

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the institutional review board of the National Institute of Advanced Industrial Science and Technology.

Ko Ayusawa is with the Human Augmentation Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Kashiwa, Chiba 2270082, Japan, and also with the CNRS-AIST JRL (Joint Robotics Laboratory), UMI3218/RL, Tsukuba, Ibaraki 3058560, Japan (e-mail: k.ayusawa@aist.go.jp).

Akihiko Murai is with the Human Augmentation Research Center, AIST, Kashiwa, Chiba 2770882, Japan (e-mail: a.murai@aist.go.jp).

Ryusuke Sagawa is with the Artificial Intelligence Research Center, AIST, Tsukuba, Ibaraki 305-8560, Japan (e-mail: ryusuke.sagawa@aist.go.jp).

Eiichi Yoshida is with the Faculty of Advanced Engineering, Department of Medical and Robotic Engineering Design, Tokyo University of Science, Katsushika-ku, Tokyo 1258585, Japan (e-mail: eiichi.yoshida@rs.tus.ac.jp).

Digital Object Identifier 10.1109/LRA.2023.3300207

that minimizes the residual of the nonlinear equations. Several IK methods utilizing optimization algorithms have been proposed [20], [21], [22], [23], [24].

In general, the merits or demerits of the algorithms used in nonlinear optimization depend on the size of the optimization problem. For example, the Levenberg–Marquardt (LM) method [25], which is popular in robotics [4], [5], [21], [23], [24], [26], usually requires the construction of the Jacobian matrix and the solving of linear equations at each iterative computation. The computational complexity of each iteration generally increases in proportion to the third power of the size of the problem. Therefore, in the case of large-sized problems, the gradient-based method, such as the conjugate gradient method or the quasi-Newton method, with appropriate line search algorithms is more convenient owing to the low computational cost of each iteration. The IK approaches based on the gradient-based methods for a large-DoF human model have been proposed [27]. However, the convergence speed at each iterative computation of those gradient methods is usually slow with respect to that of the LM method. In the case of complex human models, further improvements in IK algorithms are demanded to perform IK within a limited time.

This letter presents a new efficient IK algorithm based on LM method, LM-PFD (Pseudo-Forward Dynamics), which is remarkably effective particularly in systems with a large DoF. In the method, the problem of solving IK is treated as the pseudo problem of solving the forward dynamics of a virtual dynamical system. Several studies have made assumptions on the virtual dynamical systems [24], [27] where an assumption is utilized for gradient-based algorithms by introducing virtual springs. The proposed method is mainly aimed at replacing the problem of solving differential IK equations with the forward dynamics (FD) computation [28], [29], [30] for the virtual dynamical systems whose inertial properties correspond to the weighing matrices and damping factors used in the LM method. This replacement allows the utilization of effective O(N) algorithms developed in the field of FD computation in robotics, thus greatly reducing the computational time per iteration in the LM method.

Based on the concept of LM-PFD, the letter initially introduces the basic implementation utilizing the FD algorithm for open kinematic chains [28]. On the other hand, the motivation of this letter is to achieve rapid IK for human musculoskeletal models, while the method described above can handle only open kinematic chains. Secondly, the enhanced implementation method of LM-PFD for addressing close kinematic chains, particularly for wire-driven systems, is also provided. The efficacy of the enhanced method is demonstrated through solving the IK problem in human musculoskeletal models.

The remainder of this letter is organized as follows. Section II presents the basic concept and implementation of the proposed method. Section III introduces and explains the implementation to a closed kinematic chain, and Section IV presents its implementation in a wire-driven multibody system.

II. IK Based on Pseudo FD Computation

A. Basic Formulation of IK

Let N_L be the number of links in the target link system, N_J be the number of joints, and $\boldsymbol{q} \triangleq [\boldsymbol{q}_1{}^T \cdots \boldsymbol{q}_{N_L}{}^T]^T \in \mathbb{R}^N$ be the generalized coordinates, and $\boldsymbol{q}_j \in \mathbb{R}^{n_j}$ represents the generalized coordinates corresponding to each joint.

Let us define $\boldsymbol{p}_j(\boldsymbol{q}) \in \mathbb{R}^3$ and $\boldsymbol{R}_j(\boldsymbol{q}) \in SO(3)$ as the position and orientation of link j. The IK problem involves finding coordinates $\boldsymbol{q}$ that archive the target position and orientation of link j. Let us consider the following error

$$\boldsymbol{e}_j(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{\alpha}(\boldsymbol{R}_j^d \boldsymbol{R}(\boldsymbol{q})_j{}^T) \\ \boldsymbol{p}_j^d - \boldsymbol{p}_j(\boldsymbol{q}) \end{bmatrix} \tag{1}$$

where, $\boldsymbol{p}_j^d \in \mathbb{R}^3$ and $\boldsymbol{R}_j^d \in SO(3)$ represent the target position and orientation of link j, and $\boldsymbol{\alpha}(\boldsymbol{R}) \in \mathbb{R}^3$ indicates the conversion from a rotation matrix to an angle-axis vector.

In a typical IK problem, to make the error in (1) zero, the nonlinear equation (i.e., $\boldsymbol{e}_j(\boldsymbol{q}) = \boldsymbol{0}$) is solved with respect to $\boldsymbol{q}$. When solving the equation, the following linear equations about the translational and angular velocities $\dot{\boldsymbol{p}}_j$ and $\boldsymbol{\omega}_j \in \mathbb{R}^3$ can be utilized such that

$$\begin{bmatrix} \boldsymbol{\omega}_j{}^T & \dot{\boldsymbol{p}}_j{}^T \end{bmatrix}^T = \boldsymbol{J}_j \dot{\boldsymbol{q}} \tag{2}$$

where $\boldsymbol{J}_j$ is the basic Jacobian matrix [19].

The nonlinear equation cannot always be solved in general, and there exist several problems related to numerical stability. Owing to those issues, the problem of solving $\boldsymbol{e}_j(\boldsymbol{q}) = \boldsymbol{0}$ is often replaced with the optimization problem of minimizing the error in (1):

$$\min_{\boldsymbol{q}} f(\boldsymbol{q}) \triangleq \sum_{j=1}^{N_J} \frac{1}{2} \boldsymbol{e}_j{}^T \boldsymbol{\Sigma}_{L,j} \boldsymbol{e}_j = \frac{1}{2} \boldsymbol{e}^T \boldsymbol{\Sigma}_L \boldsymbol{e} \tag{3}$$

where $\boldsymbol{\Sigma}_{L,j} \in \mathbb{R}^{6\times 6}$ is the weight matrix for the error of link j and semi-positive definite. If there is no constraint related to link j, matrix $\boldsymbol{\Sigma}_{L,j}$ can be set to zero. The errors and weight matrices of all links are concatenated as the single vector/matrix forms $\boldsymbol{e}_L$ and $\boldsymbol{\Sigma}_L$, respectively:

$$\boldsymbol{e}_L \triangleq \begin{bmatrix} \boldsymbol{e}_1{}^T & \cdots & \boldsymbol{e}_{N_L}{}^T \end{bmatrix}^T \tag{4}$$

$$\boldsymbol{\Sigma}_L \triangleq \mathrm{diag}(\boldsymbol{\Sigma}_{L,1}, \cdots, \boldsymbol{\Sigma}_{L,N_L}) \tag{5}$$

where, notation $\mathrm{diag}(\cdots)$ indicates the block diagonal matrix constructed by the given matrix entries.

The optimization problem (3) can be solved using the following iterative method:

$$\boldsymbol{q}^{(k)} = \boldsymbol{q}^{(k-1)} + \alpha \dot{\boldsymbol{q}}^{(k)} \tag{6}$$

$$\dot{\boldsymbol{q}}^{(k)} = \boldsymbol{H}^{(k)} \boldsymbol{g}^{(k)} \tag{7}$$

where, $\dot{\boldsymbol{q}}^{(k)}$ is treated as the direction vector used in the k-th iteration, and $\alpha \in \mathbb{R}$ is the step size used in each update.

The gradient vector, $\boldsymbol{g} \in \mathbb{R}^{N_J}$, of the cost function can be computed as

$$\boldsymbol{g} \triangleq \boldsymbol{J}^T \boldsymbol{\Sigma}_L \boldsymbol{e}_L \tag{8}$$

$$\boldsymbol{J} \triangleq \begin{bmatrix} \boldsymbol{J}_1{}^T & \cdots & \boldsymbol{J}_{N_L}{}^T \end{bmatrix}^T \tag{9}$$

Matrix $\boldsymbol{H}^{(k)}$ can be determined using various methods, such as the steepest gradient method, conjugate gradient method, and quasi-Newton method. Most methods can guarantee global convergence when combined with a line search algorithm that appropriately finds α. The convergence speed and computational time of each iteration depend on the DoF of the system as well as the number of constraints of IK.

Here, let us utilize the Levenberg-Marquardt (LM) method [25]. The update rule of $\boldsymbol{H}$ can be written as

$$\boldsymbol{H}(\boldsymbol{q}) = \boldsymbol{B}(\boldsymbol{q})^{-1} \tag{10}$$

$$\boldsymbol{B}(\boldsymbol{q}) \triangleq \boldsymbol{J}^T \boldsymbol{\Sigma}_L \boldsymbol{J} + \boldsymbol{\Sigma}_Q \tag{11}$$

This form of the inverse matrix is also known as the singular robust inverse or damping inverse [4], [5]. There are several design ways about the damping factor $\boldsymbol{\Sigma}_Q \in \mathbb{R}^{N \times N}$. Let us consider the following positive definite matrix:

$$\boldsymbol{\Sigma}_Q \triangleq \mathrm{diag}\left(\boldsymbol{\Sigma}_{Q,1}, \cdots, \boldsymbol{\Sigma}_{Q,N_J}\right) \tag{12}$$

$$\boldsymbol{\Sigma}_{Q,i} \triangleq \mathrm{diag}\left(\lambda_{Q,i,1}, \cdots, \lambda_{Q,i,n_j}\right) \tag{13}$$

where $\lambda_{Q,i,j}$ is a positive scalar. In general, instead of computing $\boldsymbol{B}^{-1}$, the following linear equations are solved directly

$$\boldsymbol{B}\dot{\boldsymbol{q}} = \boldsymbol{g} \tag{14}$$

Although the LM method is effective in terms of convergence speed and numerical stability, the computational complexity of each iteration can be $\mathrm{O}(N^3)$, resulting in the computational cost, especially in the case of the IK problem of the large-DoF system. This study was aimed at speeding up the computation of solving the linear equation, (14), by utilizing the efficient computation algorithm of robotics, to maximize the characteristics of the LM method and realize a robust and fast IK computation.

B. *Equivalent Problem Using the Virtual Dynamical System*

Translational velocity $\dot{\boldsymbol{p}}_j$ and angular velocity $\boldsymbol{\omega}_j$ of link j are represented in the Cartesian coordinates (or the task space). Subsequently, let us consider spatial velocity $\boldsymbol{\nu}_j$ which is represented in the local link coordinates such that

$$\boldsymbol{\nu}_j = \boldsymbol{A}_j \begin{bmatrix} \boldsymbol{\omega}_j \\ \dot{\boldsymbol{p}}_j \end{bmatrix} = \boldsymbol{A}_j \boldsymbol{J}_j \dot{\boldsymbol{q}} \tag{15}$$

where $\boldsymbol{A}_j$ is the spatial transformation matrix defined as

$$\boldsymbol{A}_j \triangleq \begin{bmatrix} \boldsymbol{R}_j{}^T & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_j{}^T \end{bmatrix} \tag{16}$$

Let us assume the following virtual dynamical system.

- Let $\overline{\boldsymbol{\Sigma}}_{L,l} \times \mathbb{R}^{6\times 6}$ be the inertia tensor of each link l of the virtual system defined as

$$\overline{\boldsymbol{\Sigma}}_{L,l} = \boldsymbol{A}_j{}^{-T} \boldsymbol{\Sigma}_{L,l} \boldsymbol{A}_j{}^{-1} \tag{17}$$

- Each joint has the virtual actuator with inertia $\boldsymbol{\Sigma}_{Q,j}$.
- There exists no gravity in the system.

Note that (17) establishes the mapping between the weighting matrix for IK, denoted as $\boldsymbol{\Sigma}_{L,l}$, in the task space, and the virtual inertia matrix, denoted as $\overline{\boldsymbol{\Sigma}}_{L,l}$, in the local link coordinates of the dynamical system.

The Lagrangian of the above system can be obtained as

$$L_1 = \frac{1}{2}\boldsymbol{\nu}^T \overline{\boldsymbol{\Sigma}}_L \boldsymbol{\nu} + \frac{1}{2}\dot{\boldsymbol{q}} \boldsymbol{\Sigma}_Q \dot{\boldsymbol{q}} \tag{18}$$

where

$$\boldsymbol{\nu} \triangleq \begin{bmatrix} \boldsymbol{\nu}_1{}^T & \cdots & \boldsymbol{\nu}_{N_L}{}^T \end{bmatrix}^T \tag{19}$$

$$\overline{\boldsymbol{\Sigma}}_L \triangleq \mathrm{diag}(\overline{\boldsymbol{\Sigma}}_{L,1}, \cdots, \overline{\boldsymbol{\Sigma}}_{L,N_L}) \tag{20}$$

As the first term in (18) is represented in the local link coordinate frame, let (18) be transformed by the mapping in (17) with (15) as follows:

$$L_1 = \frac{1}{2}\dot{\boldsymbol{q}}^T (\boldsymbol{J}^T \boldsymbol{\Sigma}_L \boldsymbol{J} + \boldsymbol{\Sigma}_Q)\dot{\boldsymbol{q}} \tag{21}$$

The generalized momentum of this virtual system can be written as

$$\frac{\partial L_1}{\partial \dot{\boldsymbol{q}}^T} = (\boldsymbol{J}^T \boldsymbol{\Sigma}_L \boldsymbol{J} + \boldsymbol{\Sigma}_Q)\dot{\boldsymbol{q}} = \boldsymbol{\rho}_J \tag{22}$$

The inertia matrix in (22) is the same form as updating matrix $\boldsymbol{B}$ in (11). Therefore, the problem of solving (14) can be replaced by the problem of solving (22), assuming the generalized momentum such that $\boldsymbol{\rho}_J = \boldsymbol{g}$. The aforementioned replacement, substituting LM-based IK problem with the pseudo FD problem, is the fundamental concept of LM-PFD.

When solving (22), the fast algorithms of the FD computation [28], [29], [30] can be used. Especially, in the case of open kinematic chains, the Articulated-Body Algorithm (ABA) [28] is very efficient and can solve the problem with computational complexity $\mathrm{O}(N)$. The actual formulations of solving (22) by ABA will be shown in Appendix A.

C. *Summary*

In the iterative process of the IK computation, (14) can be replaced with the problem of solving the generalized momentum equations in (22) with respect to the generalized velocity, $\dot{\boldsymbol{q}}$. The processes of iteration k in the proposed IK are summarized as follows.

1) Compute gradient vector $\boldsymbol{g}^{(k)}$ of the cost function in (3), and assume the generalized momentum such that $\boldsymbol{\rho}_J = \boldsymbol{g}^{(k)}$. Though the gradient can be obtained directly from (8), it can be computed by utilizing the fast recursive algorithms [31], as shown in [27].
2) Update inertia tensors $\overline{\boldsymbol{\Sigma}}_{L,l}$ of the virtual system according to (17).
3) Solve (22) with respect to $\dot{\boldsymbol{q}}^{(k)}$ by utilizing the FD algorithms [28] without solving the linear equations directly.
4) Update $\boldsymbol{q}^{(k)}$ according to (6).

The whole procedure of IK computation is the same as that of the standard iterative methods.

There are several notes on the proposed method.

Firstly, weighting matrix $\boldsymbol{\Sigma}_{L,l}$ in (3) is expressed in the Cartesian space, whereas virtual inertia tensor $\overline{\boldsymbol{\Sigma}}_{L,l}$ is represented in the local link coordinates. As a result of the mapping from the Cartesian space to the local link coordinates, $\overline{\boldsymbol{\Sigma}}_{L,l}$ undergoes updates at each iteration based on (17). It should be noted that (22) remains valid only during the corresponding iteration. Hence, in contrast to motion generation problems such as trajectory planning or motion optimization [32], [33], the trajectory resulting from the temporal integration of the differential system, as denoted by (22), lacks mechanical significance.

Secondly, it is assumed that virtual inertia tensor $\overline{\boldsymbol{\Sigma}}_{L,l}$ (or weighting matrix $\boldsymbol{\Sigma}_{L,l}$) is not strictly positive-definite but rather positive semi-definite. For example, in the case of IK, where solely the desired end-effector posture is provided, $\boldsymbol{\Sigma}_{L,l} = \boldsymbol{O}$ holds except for the end-effector link. Even in such cases, (22) remains solvable if matrix $\boldsymbol{\Sigma}_Q$ is positive-definite. Additional details can be found in Appendix A.

Finally, in the usual IK problems, the computational time depends on the number of constraints (i.e., $\boldsymbol{e}_j(\boldsymbol{q}) = \boldsymbol{0}$). Owing to the introduction of weight matrices $\boldsymbol{\Sigma}_{L,l}$ for all links, the computational complexity of solving (22) can be $\mathrm{O}(N)$, and this does not change without depending on the number of constraints, except for the process of updating the virtual inertia tensors. The proposed method is effective especially when the number of constraints is large.

III. Implementation Method for Close Kinematic Chains

In the FD algorithms [29], [30] for closed kinematic chains, the equations of motion are represented in the link coordinate systems instead of the joint coordinate systems, while the constraints of joint motion are explicitly introduced as the equality constraints.

Let us consider joint j, which connects link p_j and c_j, and let $\boldsymbol{\nu}^j_{p_j,c_j} \in \mathbb{R}^6$ be the relative spatial velocity of links c_j from p_j such that

$$\boldsymbol{\nu}^j_{p_j,c_j} \triangleq -\boldsymbol{S}_{j,p_j}\boldsymbol{\nu}_{p_j} + \boldsymbol{S}_{j,p_l}\boldsymbol{\nu}_{p_l} \tag{23}$$

where matrix $\boldsymbol{S}_{j,k}$ is a spatial coordinate transformation matrix that maps the spatial velocity from coordinate k to coordinate j. The joint constraints about the relative velocity can be written as

$$\boldsymbol{C}_j\boldsymbol{\nu}^j_{p_j,c_j} = \boldsymbol{0} \tag{24}$$

where matrix $\boldsymbol{C}_j \in \mathbb{R}^{6-n_j\times 6}$ consists of orthogonal bases and extracts the constraint components from the spatial velocity. For ease of explanation, let us transform (24) with respect to (23) into the following form

$$\boldsymbol{C}_j\boldsymbol{S}_j\boldsymbol{\nu} = \boldsymbol{0} \tag{25}$$

where $\boldsymbol{S}_j$ has two nonzero block matrices corresponding link p_j and c_j as follows

$$\boldsymbol{S}_j \triangleq \begin{bmatrix}\boldsymbol{O} & \cdots & \boldsymbol{S}_{j,p_j} & \cdots & \boldsymbol{S}_{j,c_j} & \cdots & \boldsymbol{O}\end{bmatrix} \tag{26}$$

The following relationship between joint velocity $\dot{\boldsymbol{q}}_j$ and relative velocity $\boldsymbol{\nu}^j_{p_j,c_j}$ also holds

$$\dot{\boldsymbol{q}}_j = \boldsymbol{K}_j\boldsymbol{\nu}^j_{p_j,c_j} = \boldsymbol{K}_j\boldsymbol{S}_j\boldsymbol{\nu} \tag{27}$$

where matrix $\boldsymbol{K}_j \in \mathbb{R}^{n_j\times 6}$ consists of the complementary orthogonal bases of $\boldsymbol{C}_j$ and extracts the joint velocity components from the spatial velocity. The complementary relationships of $\boldsymbol{K}_j$ and $\boldsymbol{C}_j$ are denoted as

$$\boldsymbol{C}_j^T\boldsymbol{K}_j = \boldsymbol{O} \tag{28}$$

$$\boldsymbol{K}_j{\boldsymbol{K}_j}^T + \boldsymbol{C}_j{\boldsymbol{C}_j}^T = \boldsymbol{I} \tag{29}$$

where $\boldsymbol{I}$ is an identify matrix.

Now, let us consider the augmented Lagrangian according to the equality constraint in (25) as follows

$$\widehat{L}_1 = L_1 + \sum_{j=1}^{N_J}\boldsymbol{\lambda}_j^T(\boldsymbol{C}_j\boldsymbol{S}_j\boldsymbol{\nu}) \tag{30}$$

where $\boldsymbol{\lambda}_j \in \mathbb{R}^{n_j}$ is the Lagrange multiplier corresponding to the j-th joint constraint.

From (12), (18), (19), (27), and (30), $\widehat{L}_1$ can be transformed to

$$\widehat{L}_1 = \frac{1}{2}\boldsymbol{\nu}^T\left(\overline{\boldsymbol{\Sigma}}_L + \boldsymbol{S}^T\boldsymbol{K}^T\boldsymbol{\Sigma}_Q\boldsymbol{K}\boldsymbol{S}\right)\boldsymbol{\nu} + \boldsymbol{\lambda}^T(\boldsymbol{C}\boldsymbol{S}\boldsymbol{\nu}) \tag{31}$$

where

$$\overline{\boldsymbol{\Sigma}}_L \triangleq \mathrm{diag}\left(\overline{\boldsymbol{\Sigma}}_{L,1}, \cdots, \overline{\boldsymbol{\Sigma}}_{L,N_L}\right) \tag{32}$$

$$\boldsymbol{C} \triangleq \mathrm{diag}\left(\boldsymbol{C}_1, \cdots, \boldsymbol{C}_{N_J}\right) \tag{33}$$

$$\boldsymbol{K} \triangleq \mathrm{diag}\left(\boldsymbol{K}_1, \cdots, \boldsymbol{K}_{N_J}\right) \tag{34}$$

$$\boldsymbol{S} \triangleq \begin{bmatrix}{\boldsymbol{S}_1}^T & \cdots & {\boldsymbol{S}_{N_J}}^T\end{bmatrix}^T \tag{35}$$

$$\boldsymbol{\lambda} \triangleq \begin{bmatrix}{\boldsymbol{\lambda}_1}^T & \cdots & {\boldsymbol{\lambda}_{N_J}}^T\end{bmatrix}^T \tag{36}$$

Therefore, the equations about the link-space momentum, $\boldsymbol{\rho}_L \in \mathbb{R}^{6N_L}$, and the Lagrange multipliers take the following form

$$\frac{\partial\hat{L}_1}{\partial[\boldsymbol{\nu}^T\,\boldsymbol{\lambda}^T]} = \begin{bmatrix}\boldsymbol{\Lambda} & \boldsymbol{S}^T\boldsymbol{C}^T\\ \boldsymbol{C}\boldsymbol{S} & \boldsymbol{O}\end{bmatrix}\begin{bmatrix}\boldsymbol{\nu}\\ \boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\rho}_L\\ \boldsymbol{0}\end{bmatrix} \tag{37}$$

where $\boldsymbol{\Lambda} \in \mathbb{R}^{6N_L\times 6N_L}$ is the inertia matrix corresponding to the link equations as follows

$$\boldsymbol{\Lambda} \triangleq \overline{\boldsymbol{\Sigma}}_L + \boldsymbol{S}^T\boldsymbol{K}^T\boldsymbol{\Sigma}_Q\boldsymbol{K}\boldsymbol{S} \tag{38}$$

The relationships between the variables represented in the link space and those in the joint space are summarized as

$$\dot{\boldsymbol{q}} = \boldsymbol{K}\boldsymbol{S}\boldsymbol{\nu} \tag{39}$$

$$\boldsymbol{\rho}_L = \boldsymbol{S}^T\boldsymbol{K}^T\boldsymbol{\rho}_J \tag{40}$$

The flow of IK does not considerably differ from that introduced in Section II. In Step 3), the FD algorithms for closed kinematic chains [29], [30] are applied to solve (37), while utilizing (39) and (40) in the process.

As the matrices $\overline{\boldsymbol{\Sigma}}_L$, $\boldsymbol{\Sigma}_Q$, $\boldsymbol{K}$, $\boldsymbol{C}$, and $\boldsymbol{S}$ are sparse, (37) can also be directly solved by utilizing a sparse linear solver. When solving sparse linear equations, the ordering process is important for permuting the rows and columns of the sparse matrix before applying matrix factorization. The FD algorithms in robotics are special implementations of the ordering process utilizing the joint connectivity of the kinematic chain. Nevertheless, general ordering algorithms, such as the approximate minimum degree ordering [34], can also be used, and they are easily available in standard sparse solvers.

IV. Implementation Method for Wire-Driven Kinematic Chains

This section presents the further implementation method for a wire-driven kinematic chain. First, let us formulate the IK problem in the case of a wire-driven system.

A. IK of Wire-Driven Kinematic Chain

Let N_W be the total number of wires in the link system. The w-th wire ($1 \leq w \leq N_W$) passes through several via-points, where each via-point is located on a link, as illustrated in Fig. 1. Let $n_w(\geq 2)$ is the number of via-points of the w-th wire, $\mathrm{b}(w,k)$ indicates the index of the link where the k-th via-point

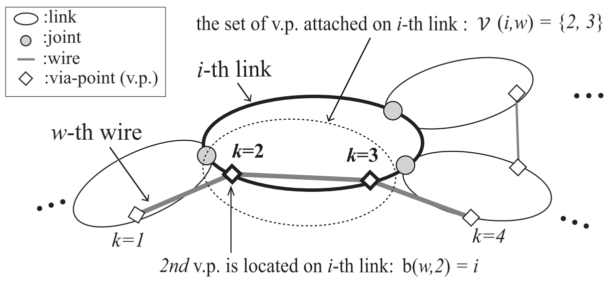
:link
:joint
:wire
:via-point (v.p.)
the set of v.p. attached on i-th link : 𝒱 (i,w) = {2, 3}
i-th link
w-th wire
k=2
k=3
k=1
k=4
2nd v.p. is located on i-th link: b(w,2) = i

Fig. 1. Schematic diagram of wire-driven kinematic chain. A wire passes through several via-points located on corresponding links. Symbol $\mathrm{b}(w,k)$ indicates the index of the link where the k-th via-point of the w-th wire is located, and $\mathcal{V}(i,w)$ is the index set of w-th via-points attached on i-th link.

of the w-th wire is located, and $\boldsymbol{r}_{w,k}$ be the global position of k-th via-point of w-th wire.

Length $l_w \in \mathcal{R}$ of the wire is computed as

$$l_w = \sum_{k=1}^{n_w-1} ||\boldsymbol{r}_{w,k} - \boldsymbol{r}_{w,k+1}|| \tag{41}$$

$$\boldsymbol{r}_{w,k} = \boldsymbol{p}_{\mathrm{b}(w,k)} + \boldsymbol{R}_{\mathrm{b}(w,k)}\widehat{\boldsymbol{r}}_{w,k} \tag{42}$$

where, $\widehat{\boldsymbol{r}}_{w,k}$ indicates the relative position of the k-th via-point of the w-th wire with respect to the origin of the coordinate system of link $\boldsymbol{r}_{w,k}$.

Let $\mathcal{V}(i,w)$ be the index set of w-th wire's via-points attached on i-th link of the kinematic chain. Velocity $\dot{l}_w$ of the wire length can be computed as [10]

$$\dot{l}_w = \boldsymbol{L}_w\boldsymbol{J}\dot{\boldsymbol{q}} \tag{43}$$

where

$$\boldsymbol{L}_w \triangleq \begin{bmatrix}\boldsymbol{L}_{w,1} & \boldsymbol{L}_{w,2} & \cdots & \boldsymbol{L}_{w,N_L}\end{bmatrix} \tag{44}$$

$$\boldsymbol{L}_{w,i} \triangleq \sum_{k\in\mathcal{E}(i,w)} \boldsymbol{d}_{w,k} - \sum_{k\in\mathcal{S}(i,w)} \overline{\boldsymbol{d}}_{w,k} \tag{45}$$

$$\boldsymbol{d}_{w,k} \triangleq \frac{(\boldsymbol{r}_{w,k}-\boldsymbol{r}_{w,k+1})^T}{||\boldsymbol{r}_{w,k}-\boldsymbol{r}_{w,k+1}||}\begin{bmatrix}[(\boldsymbol{R}_{\mathrm{b}(w,k)}\widehat{\boldsymbol{r}}_{w,k})\times]^T & \boldsymbol{I}\end{bmatrix} \tag{46}$$

$$\overline{\boldsymbol{d}}_{w,k} \triangleq \frac{(\boldsymbol{r}_{w,k-1}-\boldsymbol{r}_{w,k})^T}{||\boldsymbol{r}_{w,k-1}-\boldsymbol{r}_{w,k}||}\begin{bmatrix}[(\boldsymbol{R}_{\mathrm{b}(w,k)}\widehat{\boldsymbol{r}}_{w,k})\times]^T & \boldsymbol{I}\end{bmatrix} \tag{47}$$

where, $\mathcal{S}(i,w)(\triangleq \{k \in \mathcal{V}(i,w)\,|\,k \neq 1\})$ indicates the set of via-points of w-th wire attached on i-th link excluding the first via-point, and $\mathcal{E}(i,w) \triangleq \{k \in \mathcal{V}(i,w)\,|\,k \neq n_w\}$ means the set excluding the last (or n_w-th) via-point.

The IK of the wire-driven multibody system can be formulated as follows

$$\min_{\boldsymbol{q}} \hat{f}(\boldsymbol{q}) \triangleq \frac{1}{2}\boldsymbol{e}^T\boldsymbol{\Sigma}_L\boldsymbol{e} + \frac{1}{2}\boldsymbol{e}_W{}^T\boldsymbol{\Sigma}_W\boldsymbol{e}_W \tag{48}$$

where $\boldsymbol{\Sigma}_W \in \mathbb{R}^{N_W\times N_W}$ and $\boldsymbol{e}_W \in \mathbb{R}^{N_W}$ are the concatenated form defined as

$$\boldsymbol{\Sigma}_W \triangleq \mathrm{diag}(\sigma_1, \cdots, \sigma_{N_W}) \tag{49}$$

$$\boldsymbol{e}_W \triangleq \begin{bmatrix}d_1 & \cdots & d_{N_W}\end{bmatrix}^T \tag{50}$$

$$d_w \triangleq \min(l_w^d - l_w, 0) \tag{51}$$

Each wire has a constraint such that $l_w \le l_w^d$, where $l_w^d \in \mathbb{R}$ is the natural length of the wire. The error about the inequality constraint of w-th wire is represented as $d_w \in \mathbb{R}$, and the corresponding weighting factor is $\sigma_w \in \mathbb{R}$.

Then, the gradient of cost function $\hat{f}$ can be computed as follows

$$\boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{J}^T(\boldsymbol{\Sigma}_L\boldsymbol{e}_L + \boldsymbol{L}^T\boldsymbol{\Sigma}_W\boldsymbol{e}_W) \tag{52}$$

$$\boldsymbol{L} \triangleq \begin{bmatrix}\boldsymbol{L}_1{}^T & \cdots & \boldsymbol{L}_{N_W}{}^T\end{bmatrix}^T \tag{53}$$

When solving the optimization problem in (48), the updating matrix, $\boldsymbol{B}$, used in the LM method is written as

$$\boldsymbol{B}(\boldsymbol{q}) = \boldsymbol{J}^T(\boldsymbol{\Sigma}_L + \boldsymbol{L}^T\boldsymbol{\Sigma}_W\boldsymbol{L})\boldsymbol{J} + \boldsymbol{\Sigma}_Q \tag{54}$$

In the IK of the wire-driven system, $\boldsymbol{B}$ and $\boldsymbol{g}$ can be obtained from (52) and (54) to construct the linear (14).

B. Virtual Dynamics Problem Equivalent of a Wired System

Let us assume the following Lagrangian for the wired kinematic chain

$$L_2 = L_1 + \frac{1}{2}\sum_{w=1}^{N_W}\sigma_w\dot{l}_w{}^2 = L_1 + \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{J}^T\boldsymbol{L}^T\boldsymbol{\Sigma}_W\boldsymbol{L}\boldsymbol{J}\dot{\boldsymbol{q}} \tag{55}$$

where the first term is equivalent to (18) and the second is related to the wire kinematics of (43). The generalized momentum of the above-mentioned virtual system can be written as

$$\frac{\partial L_2}{\partial\dot{\boldsymbol{q}}^T} = \widehat{\boldsymbol{H}}\dot{\boldsymbol{q}} = \boldsymbol{\rho}_J \tag{56}$$

where $\widehat{\boldsymbol{H}}$ is the inertia matrix of the virtual dynamical system and is defined as

$$\widehat{\boldsymbol{H}} \triangleq \boldsymbol{J}^T(\boldsymbol{\Sigma}_L + \boldsymbol{L}^T\boldsymbol{\Sigma}_W\boldsymbol{L})\boldsymbol{J} + \boldsymbol{\Sigma}_Q \tag{57}$$

From (54) and (57), inertia matrix $\widehat{\boldsymbol{H}}$ can be of the same form as the updating matrix, $\boldsymbol{B}$, used in the LM method. Hence, the problem of solving (14) can be replaced by that of solving (56) when assuming the generalized momentum such that $\boldsymbol{\rho}_J = \boldsymbol{g}$.

In the next step, let us represent the formulation of (56) with respect to the link coordinates, being similar to the discussion in Section III.

From (15) and (43), wire velocity $\dot{l}_w$ can be written by using spatial velocities $\boldsymbol{\nu}$ as follows

$$\dot{l}_w = \widehat{\boldsymbol{L}}_w\boldsymbol{\nu} \tag{58}$$

where $\widehat{\boldsymbol{L}}_w \in \mathbb{R}^{N_W\times 6N_L}$ can be denoted as

$$\widehat{\boldsymbol{L}}_w \triangleq \begin{bmatrix}\boldsymbol{L}_{w,1}\boldsymbol{A}_1^{-1} & \cdots & \boldsymbol{L}_{w,N_L}\boldsymbol{A}_{N_L}^{-1}\end{bmatrix} \tag{59}$$

Similar to the discussion of (18) and (30), the augmented Lagrangian corresponding to (55) under the joint constraints can be derived from (58) as follows

$$\hat{L}_2 = \hat{L}_1 + \frac{1}{2}\boldsymbol{\nu}^T\widehat{\boldsymbol{L}}^T\boldsymbol{\Sigma}_W\widehat{\boldsymbol{L}}\boldsymbol{\nu} \tag{60}$$

$$\widehat{\boldsymbol{L}} \triangleq \begin{bmatrix}\widehat{\boldsymbol{L}}_1{}^T & \cdots & \widehat{\boldsymbol{L}}_{N_W}{}^T\end{bmatrix}^T \tag{61}$$

The equations about the link-space momentum and Lagrange multipliers are denoted as

$$\frac{\partial \hat{L}_2}{\partial [\boldsymbol{\nu}^T \boldsymbol{\lambda}^T]} = \begin{bmatrix} \widehat{\boldsymbol{\Lambda}} & \boldsymbol{S}^T\boldsymbol{C}^T \\ \boldsymbol{C}\boldsymbol{S} & \boldsymbol{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\rho}_L \\ \boldsymbol{0} \end{bmatrix} \tag{62}$$

where

$$\widehat{\boldsymbol{\Lambda}} \triangleq \boldsymbol{\Lambda} + \widehat{\boldsymbol{L}}^T \boldsymbol{\Sigma}_w \widehat{\boldsymbol{L}} \tag{63}$$

Therefore, the problem of solving (14) under (52) and (54) can be replaced with that of solving (62).

C. Approximation for Faster Computation

Since (62) for closed kinematic chains includes explicit formulas related to joint constraints, additional computations are required to solve the Lagrange multipliers by solving the entire set of complex equations, unlike in the case of (22) for open kinematic chains. On the other hand, as the LM method originally approximates the inverse of the Hessian matrix by using the damping inverse of the Jacobian matrix in its iterative computations, (62) does not need to be solved in an exact manner. In order to reduce the computation time, let us approximate (62) by adding a new damping factor, $\mu (> 0) \in \mathbb{R}$, such that

$$\begin{bmatrix} \widehat{\boldsymbol{\Lambda}} & \boldsymbol{S}^T\boldsymbol{C}^T \\ \boldsymbol{C}\boldsymbol{S} & -\mu \boldsymbol{E} \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\rho}_L \\ \boldsymbol{0} \end{bmatrix} \tag{64}$$

The solution of $\boldsymbol{\nu}$ for (64) can be written as

$$\boldsymbol{\nu} = \tilde{\boldsymbol{\Lambda}}^{-1} \boldsymbol{\rho}_L \tag{65}$$

$$\tilde{\boldsymbol{\Lambda}} \triangleq \overline{\boldsymbol{\Sigma}}_L + \boldsymbol{S}^T \boldsymbol{K}^T (\boldsymbol{\Sigma}_Q + \tfrac{1}{\mu}\boldsymbol{I}) \boldsymbol{K}\boldsymbol{S} + \widehat{\boldsymbol{L}} \boldsymbol{\Sigma}_w \widehat{\boldsymbol{L}} \tag{66}$$

In (64), $\boldsymbol{\nu}$ can be directly obtained without computing Lagrange multipliers $\boldsymbol{\lambda}$. The sparse structure of $\tilde{\boldsymbol{\Lambda}}$ depends on the connectivity of not only the joints but also wires. Therefore, in the case of wire-driven systems, the general FD algorithms cannot be applied. In this study, the approximate minimum degree method [34] is adopted for matrix ordering. Note that during the iterative computation, the structure of $\tilde{\boldsymbol{\Lambda}}$ does not change. Hence, the ordering process and the memory allocation for the factorization process are required to be computed only once before computing IK.

To be summarized, the procedure for the k-th iteration is as follows:

1) Compute $\boldsymbol{g}^{(k)}$ from (52), where its computation can be accelerated [27], and assume $\boldsymbol{\rho}_J = \boldsymbol{g}^{(k)}$.
2) Update the values of $\tilde{\boldsymbol{\Lambda}}$ defined in (66), while the matrix structure does not change.
3) Compute $\boldsymbol{\rho}_L$ from (40), solve (64) to obtain $\boldsymbol{\nu}$, and compute $\dot{\boldsymbol{q}}^{(k)}$ from (39).
4) Update $\boldsymbol{q}^{(k)}$ according to (6).

V. Numerical Evaluations

A. Computational Evaluation Using Manipulators

The proposed method and other existing algorithms were tested on the IK of the manipulator consisting of spherical joints to analyze their performance with respect to the number of DoF. In the numerical evaluation, the two manipulators with different DoF are used: $N = 60$, 600. Similar to the evaluation in [27], the target position and orientation of the end-effector are given as

TABLE I
Comparison of the Computation Time ([ms]) (And the Number of Iterations) When Solving the IK of the Two Manipulators With Different DoF

	N = 60	N = 600
LM-PFD(ABA)	0.7 (7)	8.3 (9)
LM-PFD(AVD)	0.7 (7)	6.0 (8)
LM	6.6 (7)	1545.3 (9)
QN	3.6 (122)	1259.1 (465)
CG	16.0 (934)	8503.7 (18002)

$$\boldsymbol{p}_e^d \triangleq \begin{bmatrix} 0.5 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{R}_e^d \triangleq \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \tag{67}$$

In addition, the target position of each link is set at

$$\boldsymbol{p}_i^d \triangleq \begin{bmatrix} \frac{1.5\times i}{N-2} & 0 & 0 \end{bmatrix}^T \left(1 \le i \le N_J = \frac{N}{3}\right) \tag{68}$$

The following methods were evaluated:

- (LM): the LM method with the damping design proposed by Sugihara [21].
- (LM-PFD(ABA)): the LM method [21] solving the pseudo FD problem in (22) by the ABA method [28]
- (LM-PFD(AVD)): the LM method [21] solving the pseudo FD problem according to the approximated virtual dynamics (AVD) denoted in (64)
- (QN): the quasi-Newton method (BFGS formula) with the gradient computation method [27]
- (CG): the conjugate gradient method (DY formula [35]) with the gradient computation method [27].

In the line search process, step size α was computed according to the Armijo condition [36]. The gradient of the cost function was computed by the decomposed gradient computation described in [27], which particularly demonstrates its effectiveness when applied to the IK for the system with large DoFs. In this letter, the Sugihara's method [21] is utilized for all IK methods based on the LM methods (LM, LM-PFDs). According to the method, the damping factor is designed such that $\boldsymbol{\Sigma}_Q = (\widehat{f}_k + N\sqrt{\epsilon_d})\boldsymbol{I}$, where $\widehat{f}_k$ represents the value of the cost function in (48) at k-th iteration, and ϵ_d is the double-precision machine epsilon. In the case of LM-PFD(AVD), damping factor μ is designed such that: $\mu = N\sqrt{\epsilon_d}$. The iterative computation is repeated until the error cost function, f, is less than 1.0×10^{-6} The method was implemented on a mobile workstation with an Intel(R) Core(TM) i9-12900H.

The comparison of each optimization method is summarized in Table I with the results of the computational time and the number of iterations that are required until convergence. The proposed methods (i.e., LM-PFDs(ABA&AVD)) show the best performance from the viewpoint of both convergent speed and computational cost of each iteration. In general, the complexity of one iterative computation of each method can usually be represented as follows: LM: O(N^3), QN: O(N^2), LM-PFDs, CG: O(N^2). The order of the convergent speed of each method is generally expected as follows: LM, QN, and CG methods. The above-mentioned characteristics can be seen in Table I.

LM-PFD(ABA) solves the exact same IK problem as the standard LM method. In contrast, LM-PFD(AVD) approximates

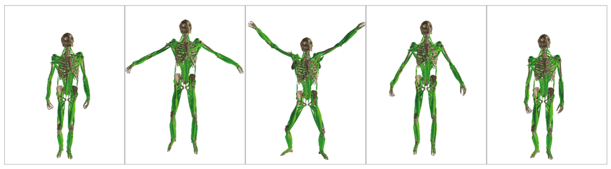

Fig. 2. Snapshot of the jumping motion used for validation.

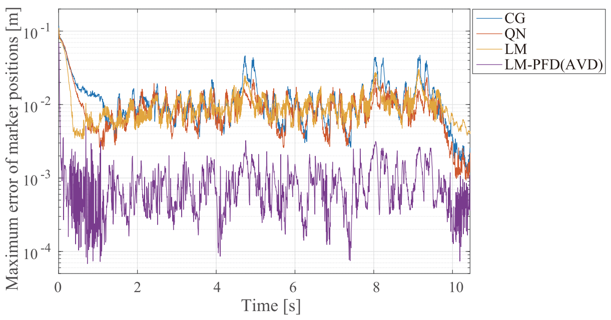

Fig. 3. The comparison of the tracking errors when solving the musculoskeletal IK among the several methods: LM-PFD(AVD), LM, QN, and CG.

the IK problem by introducing μ. As shown in Table I, the approximation results in a better computational performance compared to the other methods.

B. Musculoskeletal IK

Next, the computational performance of the proposed method and that of the other methods are compared based on the IK computation of the human musculoskeletal model used in [10]. In this musculoskeletal model, the skeletal system is represented as a multibody kinematic chain, and the muscles and tendons are approximated as massless wires. Note that the total number of wires is larger than the number of anatomically classified muscles/tendons because, for example, muscles with a large volume are represented by multiple wires. In addition, virtual links (6 DoF) are located so as to represent the branches of muscles or tendons. Therefore, the total DoF of the model exceeds that of the whole joints. In this study, the model with 155 DOFs and 338 wires was evaluated.

To confirm the IK tracking performance of IK in the case of quick movements, the jumping motion of a human subject was measured by a commercial optical motion capture system (Motion Analysis Corp.). The system measures the position of 34 reflective markers attached to the anatomical points of the human body at 200 Hz. Our study protocol was approved by the local institutional review board and conformed to the guidelines outlined in the Declaration of Helsinki (1983). From the measured trajectories of the markers, the musculoskeletal IK was computed by minimizing the sum of the error between the measured and estimated marker position. The following IK algorithms were evaluated: LM, LM-PFD(AVD), QN, and CG. As the original ABA algorithm work only for open kinematic chains, LM-PFD(ABA) cannot be utilized for the IK of the musculoskeletal model and has been excluded from the evaluation. The snapshots of the reconstructed jumping motion of the musculoskeletal model are shown in Fig. 2.

The iterative computation of each IK algorithm stops within 5 ms according to the frame rate of the motion capture system. As the tracking errors cannot be zero owing to measurement noise and modeling errors, the sufficiently convergent trajectory from IK with multiple repetitions of the iterative computations was used as the baseline to be compared. Let $\boldsymbol{p}_{M,i}$ and $\hat{\boldsymbol{p}}_{M,i}$ be $i(1 \leq i \leq 34)$-th marker's position reconstructed from each IK result and the baseline trajectory, respectively. The maximum marker tracking error, $\max(\{||\hat{\boldsymbol{p}}_{M,i} - \boldsymbol{p}_{M,i}||\})$, of each IK algorithm from the baseline is shown in Fig. 3. The LM-PFD(AVD) method showed the best tracking performance, and there is little difference from the convergent trajectory. The tracking error increases when the subject is in fast motion. The maximum error of the proposed method is at most 3 mm except at the beginning when the IK tracking starts. This error is in the same order as the spatial resolution of the motion capture system. The maximum error of any other method is over 2 cm, which is not in the acceptable range.

VI. Conclusion

This letter presented LM-PFD, a fast IK method for large-DoF multibody systems. In the method, the IK problem is replaced with the pseudo FD problem by assuming the virtual system, the inertial properties of which are derived from the weighting or damping factors in the LM method. The method can significantly accelerate the computational speed of the LM method by utilizing efficient FD algorithms. In this letter, the basic implementation of LM-PFD utilizing ABA method [29] was initially introduced for open kinematic chains. Furthermore, the enhanced method for closed kinematic chains, particularly for wire-driven systems, was also proposed.

The proposed method efficiently addresses the issue of the LM method about the computational speed in the case of large-DoF systems, while maximizing the strengths of the LM method in terms of convergence speed throughout iterations and computational robustness. These characteristics have been proved by comparing the proposed method and other existing algorithms in the numerical evaluation. The method was also tested on solving IK of the human musculoskeletal model [10]. The computational time when using the model with approximately 150 DoF and 300 wires was within 5 ms.

In our future work, the computational acceleration by introducing the pseudo FD problem will be also applied to other kinematics or dynamics problems such as muscle tension estimation [10] or motion optimization [32], [33].

Appendix A
Articulated-Body Algorithm [28]

This appendix section introduces the actual formulations of solving (22) by the articulated-body algorithm [28].

Given $\overline{\boldsymbol{\Sigma}}_{L,i}$, $\boldsymbol{\Sigma}_{Q,j}$ and $\boldsymbol{g}_i$, the articulated-body inertia and momentum, $\boldsymbol{I}_i^A$ and $\boldsymbol{p}_i^A$, can be computed recursively from the child-side of the kinematic chain as follows:

$$\boldsymbol{I}_i^A = \overline{\boldsymbol{\Sigma}}_{L,i} + \sum_{j \in \mathcal{C}(i)} {}^j\boldsymbol{X}_i{}^T(\boldsymbol{I}_j^A - \boldsymbol{U}_j\boldsymbol{D}_j^{-1}\boldsymbol{U}_j{}^T){}^j\boldsymbol{X}_i \tag{69}$$

$$\boldsymbol{p}_i^A = \sum_{j \in \mathcal{C}(i)} {}^j\boldsymbol{X}_i{}^T(\boldsymbol{p}_j^A + \boldsymbol{U}_j\boldsymbol{D}_j{}^{-1}\boldsymbol{u}_j) \tag{70}$$

where $\mathcal{C}(i)$ means the children of link i, and

$$ {}^{j}\boldsymbol{X}_i \triangleq \begin{bmatrix} \boldsymbol{R}_j{}^T\boldsymbol{R}_i & \boldsymbol{O} \\ \boldsymbol{R}_j{}^T[(\boldsymbol{p}_i - \boldsymbol{p}_j)\times]\boldsymbol{R}_i & \boldsymbol{R}_j{}^T\boldsymbol{R}_i \end{bmatrix} \quad (71)$$

$$\boldsymbol{U}_i \triangleq \boldsymbol{I}_i^A\boldsymbol{K}_i \quad (72)$$

$$\boldsymbol{D}_i \triangleq \boldsymbol{K}_i{}^T\boldsymbol{U}_i + \boldsymbol{\Sigma}_{Q,j} \quad (73)$$

$$\boldsymbol{u}_i \triangleq \boldsymbol{g}_i - \boldsymbol{K}_i{}^T\boldsymbol{p}_i^A \quad (74)$$

After computing (69) and (70) recursively, the joint and link velocity, $\dot{\boldsymbol{q}}_j$ and $\boldsymbol{\nu}_j$, can be computed recursively from the parent-side of the kinematic chain as follows:

$$\dot{\boldsymbol{q}}_j = \boldsymbol{D}_i^{-1}(\boldsymbol{u}_i - \boldsymbol{U}_i{}^T{}^i\boldsymbol{X}_{\mathcal{P}(i)}\boldsymbol{\nu}_{\mathcal{P}(i)}) \quad (75)$$

$$\boldsymbol{\nu}_i = {}^i\boldsymbol{X}_{\mathcal{P}(i)}\boldsymbol{\nu}_{\mathcal{P}(i)} + \boldsymbol{K}_i\dot{\boldsymbol{q}}_j \quad (76)$$

where $\mathcal{P}(i)$ indicates the parent of link i.

It should be noted that matrix $\boldsymbol{D}_i$ needs to be positive definite in the ABA method. In this letter, the virtual joint inertia, $\boldsymbol{\Sigma}_{Q,i}$, is positive definite, and the virtual link inertia, $\overline{\boldsymbol{\Sigma}}_{L,i}$, is positive semi-definite. According to (73), even if $\overline{\boldsymbol{\Sigma}}_{L,i} = \boldsymbol{O}$, matrix $\boldsymbol{D}_i$ is always positive definite.

References

[1] D. E. Whiteny, "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. Man- Mach. Syst.*, vol. 10, no. 2, pp. 47–53, Jun. 1969.

[2] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *Proc. IEEE 23rd Conf. Decis. Control*, 1984, vol. 23, pp. 1359–1363.

[3] J. Lenarcic, "An efficient numerical approach for calculating the inverse kinematics for robot manipulators," *Robotica*, vol. 3, pp. 21–26, 1985.

[4] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *ASME J. Dyn. Syst. Meas. 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 Trans. Syst., Man, Cybern.*, vol. 16, no. 1, pp. 93–101, Jan. 1986.

[6] T. Sugihara and Y. Nakamura, "Whole-body cooperative balancing of humanoid robot using COG jacobian," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2002, pp. 2575–2580.

[7] S. Kajita et al., "Resolved momentum control: Humanoid motion planning based on the linear and angular momentum," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2003, pp. 1644–1650.

[8] K. Yamane and Y. Nakamura, "Natural motion animation through constraining and deconstraining at will," *IEEE Trans. Vis. Comput. Graph.*, vol. 9, no. 3, pp. 352–360, Jul./Sep. 2003.

[9] S. Oota et al., "Four-dimensional quantitative analysis of the gait of mutant mice using coarse-grained motion capture," in *Proc. IEEE Annu. Int. Conf. Eng. Med. Biol. Soc.*, 2009, pp. 5227–5230.

[10] Y. Nakamura, K. Yamane, Y. Fujita, and I. Suzuki, "Somatosensory computation for man-machine interface from motion-capture data and musculoskeletal human model," *IEEE Trans. Robot.*, vol. 21, no. 1, pp. 58–66, Feb. 2005.

[11] S. L. Delp et al., "OpenSim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Trans. Biomed. Eng.*, vol. 54, no. 11, pp. 1940–1950, Nov. 2007.

[12] Z. Cao, G. Hidalgo, T. Simon, S. E. Wei, and Y. Sheikh, "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 172–186, Jan. 2021.

[13] D. Mehta et al., "VNect: Real-time 3D human pose estimation with a single RGB camera," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–14, 2017.

[14] T. Ohashi, Y. Ikegami, and Y. Nakamura, "Synergetic reconstruction from 2D pose and 3D motion for wide-space multi-person video motion capture in the wild," *Image Vis. Comput.*, vol. 104, 2020, Art. no. 104028.

[15] Y. Yoshiyasu, R. Sagawa, K. Ayusawa, and A. Murai, "Skeleton transformer networks: 3D human pose and skinned mesh from single RGB image," in *Proc. 14th Asian Conf. Comput. Vis.*, 2019, pp. 485–500.

[16] Y. Sankai, "HAL: Hybrid assistive limb based on cybernics," in *Robotics Research Springer Tracts in Advanced Robotics*. Berlin, Germany: Springer, 2011, vol. 66, pp. 25–34.

[17] H. Kobayashi, T. Aida, and T. Hashimoto, "Muscle suit development and factory application," *Int. J. Automat. Technol.*, vol. 3, no. 6, pp. 709–715, 2009.

[18] A. Murai, K. Kurosaki, K. Yamane, and Y. Nakamura, "Musculoskeletal-see-through mirror: Computational modeling and algorithm for whole-body muscle activity visualization in real time," *Prog. Biophys. Mol. Biol.*, vol. 103, no. 2/3, pp. 310–317, 2010.

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

[20] 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. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 2939–2944.

[21] T. Sugihara, "Solvability-unconcerned inverse kinematics by the Levenberg–Marquardt method," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 984–991, Oct. 2011.

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

[23] W. Suleiman, "On inverse kinematics with inequality constraints: New insights into minimum jerk trajectory generation," *Adv. Robot.*, vol. 30, no. 17/18, pp. 1164–1172, 2016.

[24] M. Sekiguchi and N. Takesue, "Fast and robust numerical method for inverse kinematics with prioritized multiple targets for redundant robots," *Adv. Robot.*, vol. 34, no. 16, pp. 1068–1078, 2020.

[25] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *J. Soc. Ind. Appl. Math.*, vol. 11, no. 2, pp. 431–441, 1963.

[26] D. Di 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.

[27] K. Ayusawa and Y. Nakamura, "Fast inverse kinematics algorithm for large DOF system with decomposed gradient computation based on recursive formulation of equilibrium," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 3447–3452.

[28] R. Featherstone, "The calculation of robot dynamics using articulated -body inertias," *Int. J. Robot. Res.*, vol. 2, no. 1, pp. 13–30, 1983.

[29] R. Featherstone, "A divide-and-conquer articulated-body algorithm for parallel o(log(n)) calculation of rigid-body dynamics. Part 1: Basic algorithm," *Int. J. Robot. Res.*, vol. 18, no. 9, pp. 867–875, 1999.

[30] K. Yamane and Y. Nakamura, "Efficient parallel dynamics computation of human figures," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2002, pp. 530–537.

[31] J. Luh, M. Walker, and R. Paul, "Resolved-acceleration control of mechanical manipulators," *IEEE Trans. Autom. Control*, vol. 25, no. 3, pp. 468–474, Jun. 1980.

[32] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," 2018, *arXiv:1801.02854*.

[33] K. Ayusawa and E. Yoshida, "Comprehensive theory of differential kinematics and dynamics towards extensive motion optimization framework," *Int. J. Robot. Res.*, vol. 37, no. 13/14, pp. 1554–1572, 2018.

[34] P. R. Amestoy, T. A. Davis, and I. S. Duff, "An approximate minimum degree ordering algorithm," *SIAM J. Matrix Anal. Appl.*, vol. 17, no. 4, pp. 886–905, 1996.

[35] Y.-H. Dai and Y. Yuan, "A nonlinear conjugate gradient method with a strong global convergence property," *SIAM J. Optim.*, vol. 10, no. 1, pp. 177–182, 1999.

[36] R. Fletcher, *Practical Methods of Optimization*. 2nd ed. Hoboken, NJ, USA: Wiley, 1987.