Inverse Kinematics of Multilink Flexible Robots for High-Speed Applications

Joonh Cheong, Member, IEEE, Wan Kyun Chung, Member, IEEE, and Youngh Youn

Abstract—A straightforward inverse kinematic algorithm for multilink flexible robots is proposed to improve the control performance. The inclusion of a dynamic constraint maximizes the performance of feedback controllers in high-speed applications. To overcome the computational difficulty, the singular perturbation approach is employed, which decomposes the inverse kinematics into an averaged part (low part) and a parasitic part (fast part). The solution of the averaged part is considered the desired inverse kinematics, while the parasitic part is intentionally removed. The parameter expansion method is carried out to obtain the solution sequentially. The initial expansion method, which is a refined version of the expansion method, reduces computing time considerably. The formula in discrete time offers efficiency in computer applications. In addition, a requirement on differentiability of the desired task trajectory is derived.

Index Terms—Flexible robots, inverse kinematics, singular perturbation, tracking control.

I. INTRODUCTION

The output tracking control [11, 2] may be one of the most important tasks in robot manipulation, because the majority of tasks being carried out by the industrial manipulators are painting, spraying, welding, and other manipulation by the end effectors. In the case of multilink flexible robots, the output tracking tends to be quite difficult because of the mechanical flexibility. This challenging problem has often been tackled by the inverse dynamic control method, whose goal is to obtain the feedback torque that generates the given output. Two types of inverse dynamic control methods have been popular: the frequency-domain inversion method [1] and the time-domain inversion method [4]. These methods require long pre- and post-acceleration time because of the infinite time integration. The nonlinear inversion method [5] treats general output tracking problems for various nonlinear systems, but there are still many constraints on the forms of nonlinearities. It may also be difficult to apply this approach to the systems which have an unstable zero dynamics. Numerical optimization methods [6, 7] were also addressed; however, their resulting solutions are usually oscillatory that they can hardly be used for real applications. The input shaping technique [8] might be regarded as a type of inverse dynamic control method, because the input torque is calculated to follow the final trajectory exactly and, at the same time, nullify the residual vibration. The shaping technique is applicable to linear systems for the present. These inverse dynamic control methods are vulnerable to uncertainties. If there are modeling uncertainties, a considerable amount of complexity, or other unmodeled nonlinearities, the calculated torque will not produce the desired output trajectory. Even if the accurate torque could be numerically obtained, the implementation of the exact amount of torque would not be guaranteed. These uncertainties necessitate combining the inverse dynamic method with the feedback of joint and deflection. The cooperation with the feedback control eliminates the merit of the inverse dynamic control after all. On the other hand, if the feedback control is to be used, the desired joint trajectory and deflection trajectory should be calculated beforehand. The calculation could be achieved by solving the forward dynamics with inverse torque, but this often results in oscillatory solutions if the use of feedback control is indispensable, and if there is a sophisticated inverse kinematic algorithm, we do not have to rely on the previous methods of inverse dynamics. The inverse kinematic solution, which is by far easier to get, can be used for the feedback control, and even for the feed-forward control.

Some researchers have been trying to develop a perfect feedback controller to achieve flawless tracking. However, such an excellent control has not been reported yet, and we are skeptical about the idea, which pays no serious consideration to the deflection and joint trajectory. The inverse kinematics of multilink flexible robots has not been dealt with seriously despite its fundamental importance in control.

Our study on the inverse kinematics begins with an assumption that the desired deflection has been usually set to zero, and that the joint trajectory has been solved on top of that condition. Obviously, this solution is not effective in achieving good tracking performance, since the bending deflection by the dynamic force is not being compensated for during the motion. There are a few works noticeable in the area of the inverse kinematics of flexible robots. Recursive numerical kinematic algorithms for a lumped mass-spring model were presented by Dan et al. [9] and Lukic et al. [10]. They employed a nice inertial property for a lumped mass-spring model to cancel the inertial terms. However, these methods are not applicable to the general dynamic model. Besides, the online application is almost impossible because of the use of iteration. For an easily realizable method, the closed-loop inverse kinematic algorithm was addressed by Siciliano [11] and Siciliano and Villani [12],
who combined kinematic constraint with the static equilibrium condition when the gravitational and external forces are exerted on the robot. In theory, the increase of kinematic control gain makes the task reconstruction error arbitrarily small. Nonetheless, the method is useful only for slow motion or set point control, since the robot’s dynamic force is not considered.

There could be numerous methods and solutions in the inverse kinematics of nontrivial flexible manipulators, while, in the case of rigid manipulators, the inverse kinematic solution is unique. Among the possible methods, we present a specific inverse kinematic algorithm satisfying the kinematic and dynamic constraints. The proposed inverse kinematic method is effective for high acceleration/speed motions, and it definitely increases the tracking performance, since the natural dynamic constraint is being used. Mathematically, the complete inverse kinematics is represented by a system of the second-order differential equations. The differential equations are not suitable by the common integration method. To make them solvable, the original equations are decomposed into the slow part and fast part by the singular perturbation approach. The solution from the slow part is considered our inverse kinematic solution, since the slow part captures the stable and averaged behavior of the original system, and it is suitable through the proposed sequential way.

This paper is organized as follows. Section II is devoted to the preliminary results of the kinematics and dynamics of the flexible robots. Section III deals with the inverse kinematic algorithm in detail. Section IV presents numerical and experimental results which compare the proposed and the previous methods. Finally, a summary on the methodology of the proposed inverse kinematics and some concluding remarks are made in Section V.

II. PRELIMINARIES

A. Equations of Motion

Fig. 1 shows the coordinate frames and kinematic definitions of a typical two-link flexible robot tracking a desired trajectory. All the flexible links are assumed to be Euler–Bernoulli beams, where the shear force–shortening effect and rotary inertia are neglected. For general n-link flexible robots, the vibration $\delta_i(x_i, t)$ of each link can be represented by a series form as

$$\delta_i(x, t) = \sum_{k=1}^{n} \phi_{i,k}(x, t) n_i(t), \quad i = 1, \ldots, n$$

where $\phi_{i,k}$ is the modal shape function, $n_i$ is the time function, and $n_i$ is the significant number of modes. The mode shapes are determined according to the boundary conditions of each flexible link. The indexes $i$ and $k$ denote the link number and the mode number, respectively. The deflection at the end of the link is defined as

$$\delta_i(t) = \delta_i(t, l_i), \quad i = 1, \ldots, n$$

where $l_i$ is the length of the $i$th link. The generalized coordinate of this system is chosen as

$$\theta = [\theta_1, \ldots, \theta_n, e_1, \ldots, e_{n-1}, c_1, \ldots, c_{n-1}]^T = [\theta_0]^T$$

where $\theta \in \mathbb{R}^n$ is the joint variable and $u \in \mathbb{R}^m$ is the deflection variable, and $m = \sum_{i=1}^{n} n_i$ is the total number of the deflection. According to the Lagrangian dynamics using the assumed modes (13), the equation of motion becomes

$$M(q)q + C(q, \dot{q}) + K + g(q) = \tau$$

That is

$$\begin{bmatrix}
M(q) & M_{ij}(q) \\
M_{ji}(q) & M_{jj}(q)
\end{bmatrix} \begin{bmatrix}
\dot{q} \\
\ddot{q}
\end{bmatrix} = \begin{bmatrix}
C_{ij}(q, \dot{q}) \\
C_{jj}(q, \dot{q})
\end{bmatrix} \begin{bmatrix}
\dot{q} \\
\ddot{q}
\end{bmatrix} + \begin{bmatrix}
K_{ij} \\
K_{jj}
\end{bmatrix} + \begin{bmatrix}
g_{ij} \\
g_{jj}
\end{bmatrix} + \begin{bmatrix}
0 \\
\tau
\end{bmatrix}.$$

where $\tau$ and $\tau$ denote the rigid part and the flexible part, respectively. $M(q) \in \mathbb{R}^{(n+m) \times (n+m)}$ is the inertia matrix, $C(q, \dot{q}) \in \mathbb{R}^{(n+m) \times (n+m)}$ is the Coriolis and centrifugal matrix, $K \in \mathbb{R}^{(n+m) \times (n+m)}$ is the stiffness matrix, $g \in \mathbb{R}^{(n+m)}$ is the gravity vector, $\tau \in \mathbb{R}^{(n+m)}$ is the input torque. The explicit vibration coupling between links is made solely by the inertia coupling. The stiffness matrix is given in a diagonal form, consisting of clamped natural frequencies of all the links. The synthesis of the inertia and stiffness matrices produces the global modes at each configuration. The zero block in the $K$ matrix represents the clamped boundary conditions of the flexible links at the sides of the joints.

B. Forward and Inverse Kinematic Consideration

The goal of the inverse kinematics is to obtain joint and deflection trajectories for the desired output trajectory $x_d$. The forward kinematic relations satisfying the forward kinematic relations

$$\begin{align*}
x_d &= h(\theta, u_1) \\
x_d &= J(\theta, u_1)u_2 - J_1(\theta, u_1) \theta_1 - J_2(\theta, u_1)u_3 \\
x_d &= J_3(\theta, u_1)u_3 + J_4(\theta, u_1)u_2 \\
\end{align*}$$

where $h(\theta, u_1)$ is the forward kinematic position map, and $J(\theta, u_1) = (h(\theta, u_1))_\theta \in \mathbb{R}^{(n+m) \times n}$ is the Jacobian matrix. The subscript $\theta$ refers to the desired value of the original variable. The Jacobian matrices can be divided into two sub-Jacobian matrices as $J(\theta, u_1) = [J_1(\theta, u_1) J_4(\theta, u_1)]$, where
\begin{align*}
J_s(\theta, v_0) &= \partial h(\theta, v_0) / \partial \theta \in \mathbb{R}^{m \times n} \quad \text{and} \quad J_t(\theta, v_0) = \partial h(\theta, v_0) / \partial v_0 \in \mathbb{R}^{m \times n}, \quad \text{respectively. For convenience, assume}
\text{that the robot is a non-undamped planar rigid body. In spite of the assumption, the dimension of } v_0 \text{ is still larger than that of }\theta \text{ because of the vibrational degrees of freedom (DOFs). There should be an additional number of constraint equations for the unique inverse kinematics solution. In many papers, the desired deflection is assumed to be zero, or should there be a considerable amount of deflection caused by the gravitational force, the deflection is calculated just to meet the static balance, such that}
\end{align*}

\begin{equation}
K_{ff} v = -g_f. \tag{4}
\end{equation}

By using this condition, it is straightforward to get the unique solution. However, this solution is limited to achieving good performance in high-speed motions, where the dynamic force from the inertial and Coriolis acceleration is significant. Such a large dynamic force causes the elastic links to bend as much as the static force, such as gravitational or external force. Therefore, the dynamic force of the robot should be incorporated into inverse kinematics. Even in low-speed motions, the dynamic forces should be considered in order to obtain accurate tracking results. The effects of the robot’s dynamic force on the elastic part can be seen clearly by

\begin{equation}
M_f \ddot{\theta}_f + C_f \dot{\theta}_f + G_f \theta_f + K_{ff} \theta_f = 0 \tag{5}
\end{equation}

which is the lower part of (2). This relation is always satisfied, no matter what the controller may be. In fact, this is a type of holonomic constraint. If all the dynamic terms are removed, (5) is reduced to a static condition in (4).

**III. INVERSE KINEMATIC ALGORITHM USING DYNAMIC CONSTRAINT**

Several differential inverse kinematics [14] is often implemented in combination with the feedback of the kinematic error to prevent the accumulation of the error. Recently, the closed-loop inverse kinematic (CLK) algorithms were modified and applied to flexible robots by Siciliano [11] and Siciliano and Vicinioni [12]. In their papers, the robot was considered to be statically in contact with a surface and suffering from the gravitational force around a fixed desired point. High-speed motions are not likely to work well in this approach.

**A. Expansion Method**

To make up for the weak feature of the modified CLK, a new inverse approach is proposed, where both of the kinematic and dynamic constraints are merged. For the kinematic constraint, the acceleration-level forward kinematics in (3) is used in combination with the feedback of velocity and the position kinematic errors, such that

\begin{align*}
\dot{x}_f &= J_f(\theta, v_0) \dot{\theta}_f + J_t(\theta, v_0) \dot{v}_0 \\
&+ K_1 (J_f(\theta, v_0) \dot{\theta}_f - x_f) + K_2 (J_t(\theta, v_0) \dot{v}_0 - x_f) \tag{6}
\end{align*}

where \( K_1 > 0 \) and \( K_2 > 0 \) are the velocity and the position control gains, respectively. This equation can be simplified to

\begin{equation}
\dot{e}_f - K_1 \dot{e}_f + K_2 \dot{v}_0 = 0 \tag{7}
\end{equation}

where \( e_f = x_f - h(\theta, v_0) \) is the kinematic error. The dynamic constraint in (5) makes up the rest of the system’s dimension. For the time being, free motion is considered as the default task. The complete formulation of the proposed inverse kinematics is then, obtained by (5) and (6) such that

\begin{equation}
\begin{pmatrix}
J_s(\theta, v_0) & J_t(\theta, v_0) \\
M_f(\theta, v_0) & M_{ft}(\theta, v_0) \\
N_s(\theta, \theta, v_0, v_0, r_0) & N_{st}(\theta, \theta, v_0, v_0, r_0) \\
N_{ts}(\theta, \theta, v_0, v_0, r_0) & N_{tt}(\theta, \theta, v_0, v_0, r_0)
\end{pmatrix}
\begin{pmatrix}
\dot{\theta}_f \\
\dot{v}_0
\end{pmatrix}
= \begin{pmatrix}
\ddot{x}_f \\
K_{ff} \theta_f
\end{pmatrix} \tag{8}
\end{equation}

where

\begin{align*}
N_s(\theta, \theta, v_0, v_0, r_0) &= K_f J_f(\theta, v_0) \dot{\theta}_f + J(\theta, v_0) \dot{v}_0 + K_f h(\theta, v_0), \\
N_t(\theta, \theta, v_0, v_0, r_0) &= C_f(\theta, \theta, v_0, v_0) \dot{\theta}_f + C_{ft}(\theta, \theta, v_0, v_0) \dot{v}_0 + K_f h(\theta, v_0), \\
\end{align*}

\begin{align*}
u_0 &= -x_f + K_1 \dot{\theta}_f + K_2 \dot{v}_0.
\end{align*}

The first row of (8) is the kinematic constraint, and the second row is the dynamic constraint. If the equation is rewritten in terms of \( \dot{e}_f \) and \( \dot{v}_0 \), a well-arranged normal form of nonlinear differential equation [15] will be obtained; the first part describes the linearized output dynamics as (7), and the second part becomes the zero dynamics. If one integrates (8) directly to get the inverse kinematic solution, the result will blow up or be permanently oscillating, because the unstable zero dynamics makes the system eventually unstable. The stability of the zero dynamics can be checked by keeping track of the system’s eigenvalues after linearization. Normally, the zero dynamics is ill-handled after external input is removed. Thus, if all \( e_f \) terms and other external input are set to zero, the reduced zero-dynamic equation becomes

\begin{equation}
\begin{pmatrix}
M_f & M_{ft} J^{-1}_f \ J_t \\
M_{ft} & M_{ft} J^{-1}_f \ J_t
\end{pmatrix}
\begin{pmatrix}
\dot{\theta}_f \\
\dot{v}_0
\end{pmatrix}
= \begin{pmatrix}
K_{ff} \theta_f
\end{pmatrix} \tag{9}
\end{equation}

In the above, \( M_f = M_f J^{-1}_f \ J_t \), is not always positive definite. Some of the eigenvalues of the matrix may be zero at certain configurations, or they may be negative at other configurations. Naturally, the zero-dynamic system is not always stable even with damping.

To obtain a stable and feasible solution of the inverse kinematics, the singular perturbation approach is employed, rather than the direct integration. Physical observability has found that flexible robots can be treated effectively using the singular perturbation method, since the stiffness of the flexible link has a large numerical value. The singular perturbation method separates the entire motion into the slow and the fast parts according to the time scales. The slow part (outer solution) covers the whole domain of interest outside of the boundary layer, while the fast part (inner solution) covers the parasitic motion. The role of composite controller [16], [17] is to make each submotion asymptotically stable. The fundamental idea of the singular perturbation method is equally applicable to the inverse kinematic system in (8), except that there is no external controller available. If we know the exact values of parameters, and if we are able to set the initial conditions exactly in the equilibrium manifold of the slow motion, the fast motion may not arise at all. And the slow motion captures the overall behavior of the original motion. However, such an ideal situation is not happening, nor is it realizable even in a computer simulation, and
the resulting solution will be oscillating or unstable. The undesirable fast motion is to be eliminated from the overall motion intentionally.

Let us define a singular parameter \( \mu = \frac{1}{\sigma_0(K_{ff})_{min}} \), where \( \sigma_0 \) is the minimum singular value of the concerned variable, and \( \mu \) serves as a measure of the separation of time scale. Substituting (15) into (18) yields:

\[
\begin{bmatrix}
J_{p}(\hat{\Theta}, \hat{\Theta}) & J_{p}(\hat{\Theta}, \hat{\Theta}) \\
M_{p}(\hat{\Theta}, \hat{\Theta}) & M_{p}(\hat{\Theta}, \hat{\Theta}) \\
N_{p}(\hat{\Theta}, \hat{\Theta}) & N_{p}(\hat{\Theta}, \hat{\Theta}) \\
N_{p}(\hat{\Theta}, \hat{\Theta}) & N_{p}(\hat{\Theta}, \hat{\Theta})
\end{bmatrix}
\begin{bmatrix}
\hat{\Theta} \\
\hat{\Theta}
\end{bmatrix}
+ \begin{bmatrix}
\mu \dot{\Theta} \\
\mu \dot{\Theta}
\end{bmatrix}
= \begin{bmatrix}
\dot{\Theta} \\
\dot{\Theta}
\end{bmatrix}
\]

We seek the solution of the slow motion written in the power series of \( \mu \) as

\[
\hat{\Theta} = \sum_{i=0}^{\infty} \mu^{i} \hat{\Theta}_{i}, \quad \hat{\Theta} = \sum_{i=0}^{\infty} \mu^{i} \hat{\Theta}_{i} (12)
\]

where \( k \geq 0 \) is the order of solution. Ideally, as \( k \) increases, it approaches the exact solution. However, we should avoid solutions of too high an order, since they contain the terms that are changing abruptly and may cause instability \( (18) \). A criterion to choose the desired order will be discussed later. Introducing (12) into (11) and expanding all the matrices up to the 4th order yield:

\[
\begin{bmatrix}
\sum_{i=0}^{\infty} \mu^{i} J_{p}^{(1)} \\
\sum_{i=0}^{\infty} \mu^{i} J_{p}^{(2)} \\
\sum_{i=0}^{\infty} \mu^{i} M_{p}^{(1)} \\
\sum_{i=0}^{\infty} \mu^{i} M_{p}^{(2)} \\
\sum_{i=0}^{\infty} \mu^{i} N_{p}^{(1)} \\
\sum_{i=0}^{\infty} \mu^{i} N_{p}^{(2)}
\end{bmatrix}
\begin{bmatrix}
\hat{\Theta}_{0} \\
\hat{\Theta}_{1} \\
\hat{\Theta}_{2} \\
\hat{\Theta}_{3} \\
\hat{\Theta}_{4} \\
\hat{\Theta}_{5}
\end{bmatrix}
= \begin{bmatrix}
\sum_{i=0}^{\infty} \mu^{i} J_{p}^{(1)} \\
\sum_{i=0}^{\infty} \mu^{i} J_{p}^{(2)} \\
\sum_{i=0}^{\infty} \mu^{i} M_{p}^{(1)} \\
\sum_{i=0}^{\infty} \mu^{i} M_{p}^{(2)} \\
\sum_{i=0}^{\infty} \mu^{i} N_{p}^{(1)} \\
\sum_{i=0}^{\infty} \mu^{i} N_{p}^{(2)}
\end{bmatrix}
\begin{bmatrix}
\dot{\Theta}_{0} \\
\dot{\Theta}_{1} \\
\dot{\Theta}_{2} \\
\dot{\Theta}_{3} \\
\dot{\Theta}_{4} \\
\dot{\Theta}_{5}
\end{bmatrix}
\]

or

\[
J_{p}^{(0)} \dot{\Theta}_{0} + N_{p}^{(0)} = \dot{\Theta}_{0}
\]

The solution in (15) is called the generating solution, since it is used for obtaining the other higher order solutions. Rewriting (15) in detail yields:

\[
J_{p}^{(0)} \dot{\Theta}_{0} + J_{p}^{(1)} \dot{\Theta}_{1} = \dot{z}_{0} + K_{1} \left( \dot{z}_{2} - J_{p}^{(0)} \dot{\Theta}_{0} \right) + K_{2} \left( \dot{z}_{3} - \dot{z}_{0} \right)
\]

where \( K_{1} = x_{2} - h(\Theta, 0) \). In fact, (15) is the same as in the inverse kinematics of the conventional rigid link robot. Equation (16) is the dynamics constraint to be solved after (13). The next order equation requires \( \dot{s}_{0} \) and \( \dot{\mu} \). The explicit time derivative of \( \dot{s}_{0} \) is

\[
\dot{s}_{0} = -K_{f}^{1} \left( M_{f}^{(0)} \dot{\Theta}_{0} + M_{f}^{(1)} \dot{\Theta}_{1} + N_{f}^{(0)} \right)
\]

Similarly, by taking time derivative to the above equation, \( \dot{\mu} \) is obtained

\[
\dot{\mu} = -K_{f}^{1} \left( M_{f}^{(0)} \dot{\Theta}_{0} + M_{f}^{(1)} \dot{\Theta}_{1} + N_{f}^{(0)} \right)
\]

In the above two equations, the third- and fourth-order derivatives of \( \dot{\Theta}_{0} \) are required, but they are not available by the algorithm. They should be replaced with lower order derivatives by employing the following relations:

\[
\dot{\Theta}_{0} = \left( J_{p}^{(0)} \right)^{\frac{1}{2}} \left( J_{p}^{(0)} \dot{\Theta}_{0} + N_{p}^{(0)} \dot{s}_{0} \right)
\]

which is obtained by differentiating (15) with respect to time. In the same way, the fourth-order derivative of \( \dot{\Theta}_{0} \) can be rewritten by lower-order derivatives, up to the second order. It should be remembered that \( \dot{s}_{0} \) must be smooth enough so that its higher order time derivatives are continuous. Since \( \dot{s}_{0} \) contains \( \dot{\mu} \), and since \( \dot{\mu} \) needs \( \dot{s}_{0} \), \( \dot{\mu} \) should be at least four times continuously differentiable.

Like the zeroth-order equation, the first-order equation is obtained by collecting terms multiplied by the first power of \( \mu \) in (13) and (14) as

\[
J_{p}^{(1)} \dot{\Theta}_{1} + N_{p}^{(1)} = -J_{p}^{(0)} \dot{\Theta}_{0} - J_{p}^{(0)} \dot{\Theta}_{0}
\]

where the terms on the right-hand side (RHS) of (18) are obtained from the previous step (the zeroth-order solution). Also, \( N_{p}^{(1)} \) contains terms known previously, although they are not given explicitly. It can be verified easily that (18) is exponentially stable with bounded perturbation by the generating solution. Thus, the first-order solution in (18) and (19) is also bounded and stable. Following the same procedures brings the general 4th-order equation to be

\[
J_{p}^{(4)} \dot{\Theta}_{4} + N_{p}^{(4)} = -\sum_{i=1}^{4} J_{p}^{(i-1)} \dot{\Theta}_{i} + \sum_{i=1}^{4} J_{p}^{(i-1)} \dot{\Theta}_{i}
\]

Now that all \( \dot{\Theta}_{4} \) and \( \dot{\Theta}_{5} \) are computed, the inverse kinematical solution \( \dot{\Theta}_{0} \) and \( \dot{\Theta}_{1} \) is obtained by adding up every order as in (12).

**Initial Conditions**: The initial condition should be in the equilibrium state of the system. Otherwise, the future solution
will be away from the desired one. If the robot is initially in the stationary condition, it satisfies:

\begin{align}
\dot{x}_i &= h_i(\theta_i, \nu_i) \\
\ddot{x}_i &= 0 \\
\nu_i &= 0
\end{align}

at \( i = 0 \). The initial conditions can be rewritten separately according to the order of \( \mu \) as follows:

\begin{align}
x_0 &= K_f^{ij} g_j^n \\
\dot{x}_0 &= 0 \\
\nu_0 &= K_f^{ij} g_j^n
\end{align}

(22)

where \( K_f^{ij} \) and \( g_j^n \) are the \( i \)-th coefficient of power series expansion, such that:

\begin{align}
h_i(\theta_j, \nu_i) &= \sum_{j=0}^{\infty} \mu^j h_j^{ij} + g_j^{ij} \theta_j + \sum_{j=0}^{\infty} \nu_j g_j^{ij}, \\
g_j^{ij} &= \sum_{j=0}^{\infty} \mu^j g_j^ij
\end{align}

In more detail, \( h_i^{ij} \) and \( g_j^{ij} \) for \( i \geq 1 \) can be given by:

\begin{align}
h_i^{ij} &= A_i(\theta_j, \nu_i) + D_i(\theta_j, \nu_i) \alpha_i = 0 \\
g_j^{ij} &= C_j(\theta_j, \nu_i) + D_j(\theta_j, \nu_i) \alpha_i - K_f^{ij} \alpha_i
\end{align}

(24)

where \( A_i, D_i, C_i, \) and \( D_i \) are matrices of appropriate dimensions, and \( \alpha_i \) is invertible, especially. Using (23) and (24), every initial value of \( \theta_i \) and \( \alpha_i \) for \( i \geq 2 \) is determined sequentially.

To establish the procedure for determining approximation order, we need to select a threshold value \( \alpha_i \) and whether the fraction of the solution is significant or not. If the actual scale of the \( i \)-th order solution is smaller than a tolerance, \( \alpha_i \), such that:

\begin{align}
\|x_i^{ij} \| < \alpha_i
\end{align}

(25)

\( \alpha_i \) or other higher solution fragments can be disregarded in the solution. Otherwise, \( \alpha_i \) should be included in the solution, and the same test should be given to the \((i-1)\)th order. Since the speed, acceleration, and higher time derivatives of solutions are changing according to the shape and generation method of the desired trajectory, the approximate solution by a few lower orders might not cover the region of abrupt change to meet the condition in (25). To achieve strict performance, the additional higher order solutions are required. Or, we have to slow down the speed of the trajectory. As for the tolerance, it cannot be set to one specific value, just as there is no actual boundary value of stiffness to separate between the flexible links and the rigid links. Depending on the need of accuracy, calculation time, or other factors, it may be set to a certain value.

The inverse kinematic method introduced above is conceptually very simple; however, it has never been tried before. Since the fast subsystem is neglected, the final solution corresponds to the averaged solution of the original singularly perturbed system. The algorithm can be implemented online by disregarding higher order.

### B. Implicit Expansion Method

The previous expansion method requires the series expansion of all the participating matrices, as well as the variables themselves. This expansion is computationally expensive because it requires large memory and calculating time. Sometimes the calculation is so huge and complicated that it does not always look useful. To avoid the shortcomings, an implicit expansion method is proposed, where the matrix expansion is not required at all.

Still, the implicit expansion method preserves the same accuracy as the previous expansion method. In the sequel, the expansion method in Section II-A will be called the explicit expansion method to differentiate from the implicit expansion method. Suppose that the \( i \)-th order solution of \( \theta_j \) and \( \nu_i \) based on the implicit expansion method is:

\begin{align}
\theta_{E,i} &= \sum_{k=0}^{i} \mu^k \theta_{E,i,k} + R(\theta_{E,i}) \\
\nu_{E,i} &= \sum_{k=0}^{i} \mu^k \nu_{E,i,k} + R(\nu_{E,i})
\end{align}

(26)

respectively. Here, \( R(\theta_{E,i}) \) and \( R(\nu_{E,i}) \) terms denote the negligible terms higher than the \( i \)-th order. The implicit expansion method solves \( \theta_{E,i} \) and \( \nu_{E,i} \) directly from (11) without explicit expansion. Substituting (26) into (11) yields:

\begin{align}
J_{E,i}(\theta_{E,i}, \nu_{E,i}, \Psi_{E,i}) &= J_{E,i}(\theta_{E,i}, \nu_{E,i}, \Psi_{E,i}) + N_i(\theta_{E,i}, \nu_{E,i}, \Psi_{E,i}) - \Psi_{E,i}
\end{align}

(27)

\begin{align}
M_{E,i}(\theta_{E,i}, \nu_{E,i}, \Psi_{E,i}) &= M_{E,i}(\theta_{E,i}, \nu_{E,i}, \Psi_{E,i}) + M_{E,i}(\theta_{E,i}, \nu_{E,i}, \Psi_{E,i}) - \Psi_{E,i} - \Psi_{E,i}
\end{align}

(28)

All terms retaining \( \mu^{i+1} \) or higher order explicitly are to be removed. Since \( \Psi_{E,i} = \sum_{j=0}^{i} \mu^j \Psi_{E,i,j} \), the term \( \mu^{i+1} \Psi_{E,i} \) is in the order of \( \mu^{i+1} \) explicitly, and it should be eliminated. A convenient way to do the elimination is to use \( \nu_{E,i} \) instead of \( \nu_{E,i} \). Following the suggestion and performing elementary algebra makes (27) and (28) written as:

\begin{align}
J_{E,i}(\theta_{E,i}, \nu_{E,i}, \Psi_{E,i}) &= J_{E,i}(\theta_{E,i}, \nu_{E,i}, \Psi_{E,i}) + N_i(\theta_{E,i}, \nu_{E,i}, \Psi_{E,i}) - \Psi_{E,i}
\end{align}

(29)

and:

\begin{align}
\Psi_{E,i} &= -K_f^{ij} \left( M_{E,i}(\theta_{E,i}, \nu_{E,i}, \Psi_{E,i}) + N_i(\theta_{E,i}, \nu_{E,i}, \Psi_{E,i}) \right)
\end{align}

(30)

where the index variable \( k \) is added to all the matrices only to show that they are related to the \( k \)-th order model. In the above, the matrices are never expanded as power series. Lumped terms (29) and (30) are the basic equations for the implicit expansion method in the \( i \)-th order. The terms in the RHSs of those equations are known from the previous stage, playing as the input to the foregoing order. If \( i = 0 \), those equations become:

\begin{align}
J_{E,0}(\theta_{E,0}) &= J_{E,0}(\theta_{E,0}, \Psi_{E,0}) = \Psi_{E,0}
\end{align}

(31)

\begin{align}
\Psi_{E,0} &= -K_f^{ij} \left( M_{E,0}(\theta_{E,0}, \nu_{E,0}, \Psi_{E,1}) + N_i(\theta_{E,0}, \nu_{E,0}, \Psi_{E,1}) \right)
\end{align}

(32)

since \( \nu_{E,0} = 0 \). From (31), \( \theta_{E,0}, \nu_{E,0}, \) and \( \theta_{E,0} \) are obtained. Then, \( \Psi_{E,0} \) is obtained by putting them into (32). Since (31) and (32) are exactly the same as the zeroth-order equation of the explicit expansion method, it is true that \( \theta_{E,0} = \theta_0 \) and \( \Psi_{E,0} = \Psi_0 \). The differentiation of (33) results in \( \Psi_{E,1} \) as:

\begin{align}
\Psi_{E,1} &= -K_f^{ij} \left( M_{E,1}(\theta_{E,1}, \nu_{E,1}, \Psi_{E,1}) + N_i(\theta_{E,1}, \nu_{E,1}, \Psi_{E,1}) \right)
\end{align}

(33)

and another differentiation yields \( \Psi_{E,2} \). To obtain \( \Psi_{E,0} \) and \( \Psi_{E,1} \), \( \theta_{E,0} \) and \( \theta_{E,1} \) are required, but they can be replaced with lower order derivatives as was done in the explicit expansion method.
The first-order solution is obtained simply in the same way by setting \( k = 1 \) to (29) and (30). The zeroth solution corrects the first-order equation in this case, making the solution more accurate. It is straightforward to obtain the higher order solutions recursively. The calculation is repeated until the final-order solution is obtained.

The amount of calculation for higher orders will be increasing. Generally speaking, as the link becomes more flexible, the final solution requires higher orders. However, if the link becomes stiff, just a lower order solution will be enough for sufficient accuracy. The parameter \( \mu \) becomes smaller as the link goes stiffer. Consequently, the order of negligible terms of \( \mathcal{O}(\mu^{k+1}) \) becomes reduced, and the final solution remains in low order. It is interesting to rewrite (30) as follows:

\[
\Psi_{k+1} = -\tilde{K}_{ff}^{-1} \left( M_{ff,k} \phi_{k+1} + R(J_{f,k}) \right) + \tilde{K}_{ff}^{-1} \left( \sum_{i=0}^{k} \mu^i J_{fi,k} + R(J_{f,i,k}) \right) \Psi_{k+1} + \tilde{K}_{ff}^{-1} R(N_{f,k})
\]

By considering the infinite series forms of \( \Theta_{k+1} \) and \( \Psi_{k+1} \), (40) and (41) can be separated into subequations, according to the order of \( \mu \) like the explicit expansion method. And the resulting subequations are the same as (20) and (21) at any order up to \( k \). Therefore, the actual order of solution in (29) and (30) will be:

\[
\tilde{\Theta}_{k+1} = \tilde{\Psi}_{k+1} + \sum_{i=0}^{k} \mu^i \Theta_{k+1},
\]

\[
\tilde{\Theta}_{k+1} = \tilde{\Psi}_{k+1} + \sum_{i=0}^{k} \mu^i \Theta_{k+1},
\]

where \( \mathcal{O}(\mu^{k+1}) \) terms result from \( \mathcal{R}(\cdot) \) terms of each matrix. Note that the solution form in the above is the same as the superposition in (26), and the 4th-order solutions of the explicit and implicit methods present the same accuracy.

To investigate the effect of the residual terms in the accuracy of the next-order solution, the (4 + 4th-order) system is considered as:

\[
\left( \sum_{i=0}^{4} \mu^i J_{fi,k} + R(J_{f,i,k}) \right) \Psi_{k+1} + \left( \sum_{i=0}^{4} \mu^i N_{fi,k} + R(N_{f,i,k}) \right) = \mu_{k+1} \left( \sum_{i=0}^{4} \mu^i J_{fi,k} + R(J_{f,i,k}) \right) \Psi_{k+1} + \mu_{k+1} \left( \sum_{i=0}^{4} \mu^i N_{fi,k} + R(N_{f,i,k}) \right)
\]

\[
\tilde{\Theta}_{k+1} = -\tilde{K}_{ff}^{-1} \left( \sum_{i=0}^{4} \mu^i J_{fi,k} + R(J_{f,i,k}) \right) \Psi_{k+1} + \tilde{K}_{ff}^{-1} \left( \sum_{i=0}^{4} \mu^i N_{fi,k} + R(N_{f,i,k}) \right) \Psi_{k+1}
\]
Again, subequations can be collected according to the order of $\mu$ up to the $(k+1)$th-order after we plug the forms of $\Psi_k$, $\Theta_k$, and $\Theta_k$ in (38) and (39). The $C^{(k+1)}$ term contained in $\Psi_k$ becomes $C^{(k+2)}$, since $\Theta_k$ is always multiplied by $\mu$ in the $(k+1)$th-order equation. The $C^{(k+1)}$ term in the $k$-th-order solution does not affect the significant part of the $(k+1)$th-order solution. Finally, the $(k+1)$th-order solution becomes

$$
\begin{align*}
\dot{\Theta}_{k+1} &= \sum_{i=0}^{k+1} \frac{\partial^{i} J^{(k+1)}}{\partial x^i} \dot{x}_{k+1} + C^{(k+2)}, \\
\dot{\Psi}_{k+1} &= \sum_{i=0}^{k+1} \frac{\partial^{i} J^{(k+2)}}{\partial x^i} \dot{x}_{k+1}, \\
\dot{\Theta}_{k+2} &= \sum_{i=0}^{k+1} \frac{\partial^{i} J^{(k+2)}}{\partial x^i} \dot{x}_{k+1}, \\
\dot{\Psi}_{k+2} &= \sum_{i=0}^{k+1} \frac{\partial^{i} J^{(k+3)}}{\partial x^i} \dot{x}_{k+1}, \\
\dot{\Theta}_{k+3} &= \sum_{i=0}^{k+1} \frac{\partial^{i} J^{(k+4)}}{\partial x^i} \dot{x}_{k+1}, \\
\dot{\Psi}_{k+3} &= \sum_{i=0}^{k+1} \frac{\partial^{i} J^{(k+5)}}{\partial x^i} \dot{x}_{k+1}, \\
\dot{\Theta}_{k+4} &= \sum_{i=0}^{k+1} \frac{\partial^{i} J^{(k+6)}}{\partial x^i} \dot{x}_{k+1}, \\
\dot{\Psi}_{k+4} &= \sum_{i=0}^{k+1} \frac{\partial^{i} J^{(k+7)}}{\partial x^i} \dot{x}_{k+1}.
\end{align*}
$$

(42)

In this way, the solutions of the explicit method and the implicit method have the same accuracy for any given order.

Note that the stability of the $(k+1)$th-order implicit kinematic equation needs to be examined for the whole time domain. Assume that $x_k$ and $\Psi_{k-1}$ are given, and they are uniformly continuous up to the second-order derivatives. By modifying the output error in (17), the $(k+1)$th-order output error is defined as

$$
e_{k+1} = \dot{x}_k - (\dot{\Theta}_{k+1} + \dot{\Psi}_{k+1}).$$

(43)

See that $e_{k+1}$ contains just one independent variable, $\Theta_{k+1}$. On the other hand, $e_k$ consists of two independent variables, $\Theta_k$ and $e_k$. Equation (29) can be rewritten in terms of $e_k$ such that

$$
e_{k+1} + K_{1}e_k + K_{2}e_k = 0.$$

(44)

Thus, only if $K_{1}$ and $K_{2}$ are positive definite, $e_{k+1}$ is exponentially stable for given $x_k$ and $\Psi_{k-1}$. If $e_{k+1}$ is linearized as

$$
e_{k+1} = J_k(\Theta_k, \mu \Psi_{k-1}) (\Theta_{k-1} - \Theta_{k}),$$

where $J_k$ is the unique solution such that $x_k = h(\Theta_k, \mu \Psi_{k-1})$, there is a one-to-one relation between $\Theta_k$ and $\Theta_{k-1}$, whenever $J_k$ is invertible. As $e_{k+1}$ goes to zero, $\Theta_{k-1}$ approaches $\Theta_{k}$. Hence, $\Theta_{k}$ is uniformly continuous and bounded for the whole trajectory. However, $\Theta_{k}$ has its first two derivatives are uniformly continuous and bounded automatically according to (30). In addition, this means that the $C^{(k+2)}$ terms in the $k$th-order solution are uniformly bounded if we recall that $\Theta_k$ and $\Theta_{k-1}$ are bounded. Therefore, the proof that the explicit expansion method is equivalent to the explicit expansion method is completed, and the solutions from the two methods are bounded at any given order.

If the robot is in contact with environment, the contact force will be nonzero. Since the contact force brings about the deformation of flexible links, it is natural to include the effect of the force in the inverse kinematics to deal with constrained motions. Simply by putting the contact force into the previous formulation in (11), a modified inverse kinematics is obtained as

$$
\begin{align*}
\begin{bmatrix}
J_{E}(\Theta_k, \mu \Psi_{k-1}) & J_{E}(\Theta_{k-1}, \mu \Psi_{k-2}) & \ldots & J_{E}(\Theta_{k-(m-1)}, \mu \Psi_{k-(m-2)})
\end{bmatrix}
\begin{bmatrix}
\dot{\Theta}_k \\
\dot{\Theta}_{k-1} \\
\vdots \\
\dot{\Theta}_{k-(m-1)}
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
\vdots \\
0
\end{bmatrix}
\end{align*}
$$

(45)

where $f_c$ is the desired contact force. The remaining procedure is the same as the noncontact situation. However, one should be careful in designating the output trajectory and the contact force. The contact force should be given only when the robot’s trajectory is really constrained in physical surfaces. Otherwise, the robot motion will not converge to the desired equilibrium, because it is not physically reachable. Also, make sure that $f_c$ is given to be smooth enough to guarantee the continuity of $\dot{\Theta}_k$ and $\dot{\Psi}_k$.

**Initial Conditions:** For the $(k+1)$th-order implicit equation, the initial conditions are given by

$$
\begin{align*}
\Theta_k(0) &= \alpha_k, \\
\Theta_{k-1}(0) &= \hat{\Theta}_{k-1}(0), \\
\Psi_k(0) &= \hat{\Psi}_k(0), \\
\Psi_{k-1}(0) &= \hat{\Psi}_{k-1}(0), \\
\beta_k &= \beta_{k-1} = \beta_0.
\end{align*}
$$

(46)

(47)

where $\alpha_k$ and $\beta_k$ are chosen to satisfy the kinematic constraint and static equilibrium condition, such that

$$
x_k(0) = h(\alpha_k, \beta_{k-1}),$$

(48)

with $\beta_0 = 0$. Every $\alpha_k$ and $\beta_k$ can be found easily after a few numerical iterations.

### C. Discrete-Time Integration Formula

Since almost every control system is realized by a digital computer, it is desirable to rewrite the implicit algorithm based on discrete time. Assume that the $(k+1)$th-order inverse kinematics solution is our desired solution and that the robot is not in contact with environment. The discrete version of the differential equation becomes a simple algebraic equation that uses the state of the previous time, $x(t_j) = k(t_j), \dot{\theta}(t_j)$, and $\dot{\psi}(t_j)$, $i = 0, 1, \ldots, k$, are obtained by putting the solutions of the previous time. That is

$$
\begin{align*}
\Psi_{k+1} &= -K_{1}(t_j) \begin{bmatrix}
M \dot{\Theta}_{k+1}(t_j) & M \dot{\Theta}_{k+2}(t_j) & \ldots & M \dot{\Theta}_{k+1}(t_j)
\end{bmatrix}
\begin{bmatrix}
\dot{\Theta}_{k+1}(t_j) \\
\dot{\Theta}_{k+2} \\
\vdots \\
\dot{\Theta}_{k+1}(t_j)
\end{bmatrix}
\end{align*}
$$

(49)

with sampling time $\Delta t = t_{k+1} - t_k$. The first and the second time derivatives of $\dot{\Psi}_{k+1}$ are obtained by the numerical deriva-
Fig. 2. Discrete sequence of the implicit inverse kinematics.

\[
\begin{align*}
\dot{\Psi}^e_1(t_i) &= \dot{\Psi}^e_1(t_{i-1}) + \frac{(1 - \nu)}{\Delta t} \left( \Psi^e_1(t_{i-1}) - \Psi^e_1(t_{i-1}) \right) \\
\ddot{\Psi}^e_1(t_i) &= \ddot{\Psi}^e_1(t_{i-1}) + \frac{(1 - \nu)}{\Delta t} \left( \dot{\Psi}^e_1(t_{i-1}) - \dot{\Psi}^e_1(t_{i-1}) \right)
\end{align*}
\]  

(49)

where \( \dot{\Psi}^e \) and \( \ddot{\Psi}^e \) are estimated velocity and acceleration of \( \Psi^e \), respectively, and \( \nu \) is a parameter on the filter's time constant. In most of the cases, there is not much difference between the estimated and actual values of \( \dot{\Psi}^e \) and \( \ddot{\Psi}^e \), as long as the desired task trajectory is sufficiently smooth. The next step is to update \( \dot{\theta}_e(t_j) \) using the previous time-joint variables \( \dot{\theta}_e(t_{j-1}) \) and current time-deflection variables \( \dot{\Psi}^e(t_j) \). Inserting the Jacobian matrices of the differential equation in (29) gives the joint acceleration as

\[
\begin{align*}
\dot{\theta}_e(t_j) &= J_e^T(\dot{\theta}_e(t_{j-1}), \mu \dot{\Psi}^e(t_j)) \\
&= \begin{cases} 
\dot{\theta}_e(t_j) - N_1(\dot{\theta}_e(t_{j-1}), \mu \dot{\Psi}^e(t_j)), \\
\dot{\theta}_e(t_j) - N_2(\dot{\theta}_e(t_{j-1}), \mu \dot{\Psi}^e(t_j)) 
\end{cases} \\
&= J_e(\dot{\theta}_e(t_{j-1}), \mu \dot{\Psi}^e(t_j)).
\end{align*}
\]

(50)

Table 1: Physical parameters

<table>
<thead>
<tr>
<th>Component</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total length of link 1</td>
<td>l1</td>
<td>0.561 [m]</td>
</tr>
<tr>
<td>Total length of link 2</td>
<td>l2</td>
<td>0.571 [m]</td>
</tr>
<tr>
<td>Length of flexible link 1</td>
<td>l1</td>
<td>0.521 [m]</td>
</tr>
<tr>
<td>Length of flexible link 2</td>
<td>l2</td>
<td>0.521 [m]</td>
</tr>
<tr>
<td>Mass of link 1</td>
<td>M1</td>
<td>0.04 [kg]</td>
</tr>
<tr>
<td>Mass of link 2</td>
<td>M2</td>
<td>0.10 [kg]</td>
</tr>
<tr>
<td>Mass of elbow mechanism</td>
<td>M0</td>
<td>0.20 [kg]</td>
</tr>
<tr>
<td>Stiffness of link 1</td>
<td>K1</td>
<td>21.46 [Nm/rad]</td>
</tr>
<tr>
<td>Stiffness of link 2</td>
<td>K2</td>
<td>0.054 [Nm/rad]</td>
</tr>
</tbody>
</table>

After these procedures are carried out from \( t = 0 \) to \( t = l_i \), time is forwarded to \( t = l_{i+1} \). By succeeding it to the final
time $t_f$, the proposed inverse kinematic algorithm will be completed. Fig. 2 summarizes the steps of the proposed algorithm. As shown above, all the procedures are performed sequentially and deterministically. Thus, the calculation time is predictable, and the algorithm can be implemented online.

Remarks: As mentioned in the explicit expansion method, that $x_k$ is highly differentiable is a prerequisite for obtaining a higher order solution of the inverse kinematics. For example, to obtain $\dot{x}_k$, $\ddot{x}_k$, and their first two time derivatives, $x_k$ should be continuously differentiable up to $2k+1$. In the implicit method, the same condition on the differentiability is required to obtain $\dot{\theta}_{k,1}$ and $\dot{\Psi}_{k-1}$. It is possible to select $\dot{\theta}_{k,1}, \dot{\Psi}_{k-1}$, and their first two derivatives as the set of the desired solutions, as long as the constant error $\dot{\Psi}_{k-1}$ in (58) is small. In that case, $x_k$ is to be $2k+2$ times continuously differentiable.

IV. NUMERICAL AND EXPERIMENTAL RESULTS

A three-dimensional (3-D) two-link flexible robot, shown in Fig. 3, is used for numerical and experimental study. For clear demonstration, the base joint is locked, and only the two-link plane motion is allowed. The joint angles are measured by optical encoders, and deflections are measured by strain gauges attached along the sides of the links. The physical parameters of the experimental robot are summarized in Table I. See [39] for more detail. A Pentium III IBM PC is used as a controller unit with realtime OS. The software algorithm is processed at every 1 ms.

A. Numerical Results

To see the performance and characteristics of the proposed inverse kinematics, the numerical solution of the inverse kinematics is calculated for a triangular trajectory composed of three successive line segments, i.e., $\Delta x_k = (-0.3, 0.0, 0.0) \text{ m}$, and $\Delta x_{k+1} = (0.3, 0.0, 0.0) \text{ m}$, for each equal $t_f$ seconds. Two different speeds are considered: one is slow such that $t_f = 3.5 \text{ s}$, and the other is rather fast, such that $t_f = 1.5 \text{ s}$. Each line trajectory is described as the 5-th order polynomial function parameterized by time as

$$f(t) = \sum_{i=0}^{11} a_i t^i$$

where $a_i$ is a real constant, which is determined by the boundary conditions as

$$f(0) = \theta_1 f(t_f) = \theta_2 f(t_f) = 0$$
$$f'(0) = 0 \quad f'(t_f) = 0$$
$$\cdots \quad f''(0) = 0 \quad f''(t_f) = 0$$

where $f''(i)$ means the $i$-th order time derivative of $f$. For the first segment of line tracking, the desired trajectory is given as

$$x_k(t) = \begin{cases} x_0 + \Delta x_1 f(t), & 0 \leq t < t_f \\ x_0 + \Delta x_2, & t_f \leq t < t_f + t_c \end{cases}$$

In the second segment, the desired trajectory is given as
Fig. 3: Tracking performance using the proposed algorithm: $\tau_2 = 3.5$ s.

where $x_0$ is the initial position of the robot. We set $x_0 = (1.022, -0.208)$ m as the initial position of the robot. A short interval time $\Delta t$ is added before tracking the next line segment, in order to reduce excessive vibration. The second and third line segments can be parameterized by time in the same way. The equations of motion are determined by procedures similar to [20]. The mode shapes of each link are normalized such that the stiffness matrix becomes a diagonal matrix, and every diagonal element is the square of a natural frequency of each link. Following the definition in (10), we have $\mu \approx 1/290$, which is the inverse of the stiffness for the first vibration mode (2.73 Hz) of the first link. The given trajectory allows sixth-order continuous time derivatives, and we determine the desired solution to be $\dot{\theta}_{2,D}$ and $\ddot{\theta}_{2,D}$ for the proposed implicit inverse kinematics algorithm.

The inverse kinematic solution is obtained from the discrete formula in (48) and (50), succeeding the procedure from $t = 0$ to the end of tracking time. The required gains are set to $K_1 = 2I$ and $K_2 = 20I$. The filter parameter $\nu$ is set to 0.95.

The proposed inverse kinematics is compared with the previous inverse kinematics in [11], which considers the kinematic constraint and static deflection compensation. For a brief introduction, the formulation in [11] is summarized as

$$\dot{\theta}_i = -K_i \dot{J}_i (x_i - \dot{h}(\theta_i, x_i))$$

$$\ddot{\theta}_i = -\left( K_{ii} + \frac{\partial h(\theta_i)}{\partial \theta_i} \right) \ddot{h}$$

where $J_i = J_i (K_{ii} + \frac{\partial h(\theta_i)}{\partial \theta_i})^{-1} \frac{\partial h(\theta_i)}{\partial \dot{\theta}_i} \ddot{\theta}_i$ and $\ddot{\theta}_i = \ddot{y}_i$.

We use $K_{ii} = 1000I$, which is high enough to keep the inverse kinematic solution following the desired output path.

Figs. 4 and 5 show the solutions of the trajectory when $t_1 = 3.5$ s and $t_2 = 1.5$ s, respectively. The rest interval is set $\tau_2 = 0.3$ s. In each figure, the solution of the previous method labeled as “DS” is compared with the first three solutions of the proposed method. The joint solutions of the zeroth order are very far off from the others, since the deflection caused by the gravitational force is not considered. In both cases of different speeds, the first-and-second order solutions of the proposed inverse kinematics are very close to each other. This implies that only the first-order solution is sufficiently accurate. When $t_1 = 3.5$ s, the solution of the previous method is very close to the first-and-second order solutions of the proposed method. But, when $t_1 = 1.5$ s, there is a big difference between the results. For the faster trajectory, the solutions of the proposed algorithm contain slight oscillations, and the solution is smooth as that of the previous algorithm, because of the reflection of the dynamic forces to the inverse kinematics. Although not presented in this paper, all the output errors are bounded and remain in a very negligible region. As the order of solution increases, the output error becomes much smaller.

8 Experimental Results

Using the inverse kinematic solutions, we conducted experimental tests on the Pohang University of Science and
Technology (POSTECH) (Pohang, Korea) flexible robot. Before starting the triangular trajectory, the robot was sent to the initial configuration from the home position (straight down configuration), and then it was commanded to track the triangular trajectory. After finishing the job, it was sent back to the home position. The control law adopted was the proportional-integral-derivative (PID) composite controller of the form \( \tau = \tau_s + \tau_f \), where

\[
\tau_s = M_s \left( \theta_s + \lambda \dot{\theta}_s \right) + C_s \left( \theta_s + \lambda \dot{\theta}_s \right) + K_s \alpha - K_s \alpha \\
\tau_f = M_f \left( K_f \eta + K_f \eta + K_f \int \omega dt \right)
\]

where \( \tau_s \) is the controller for the slow subsystem, and \( \tau_f \) is the fast subsystem. In the above, \( \alpha = \beta_s + \lambda \int \omega dt \) identifies the combined joint error, \( \omega = \theta_s - \theta \), and \( \eta = \dot{\omega} \) is the coordinate of the system mode. In the slow control part, \( \lambda \) is working as the bandwidth parameter of the closed-loop joint system. Usually, \( \lambda \) is set to be smaller than the fundamental frequency [21]. The first two vibration modes were considered as the significant vibration modes; the modal analysis and test showed that the rest of the modes were comparatively high, and that they could be disregarded with proper signal filtering. The control gains of \( \tau_f \) were chosen to satisfy the Routh criteria in the symmetrical coupled flexible system [22], [23]. The gains of \( \tau_s \) were given in the diagonal form, whereas the gains of \( \tau_f \) were given in a nondiagonal symmetric form as follows:

\[
K_{f1} = K_{f1}' M_f / T_f' \\
K_{f2} = K_{f2}' M_f / T_f' \\
K_{f3} = K_{f3}' M_f / T_f'
\]

where \( K_{f1}', K_{f2}', \) and \( K_{f3}' \) are diagonal matrices. These gains should be selected carefully. Each gain of \( \tau_s \) and \( \tau_f \) not only has to stabilize its corresponding subsystem, but also has to minimize the counter excitation between \( \tau_s \) and \( \tau_f \). The best tuning is extremely complicated, and it is beyond the scope of this paper. Table II shows the control gains used in the experiments. The gains in \( \tau_f \) are much larger than those in \( \tau_s \) because \( \tau_f \) is designed on the expanded time scale. When the first time scale is given to the normal time scale, all the gains are multiplied by the inverse of \( \alpha \), the time-scale parameter, resulting in a very high gain controller. The high gain controller makes the fast subsystem approach the equilibrium motion very quickly.
and the motion of the whole system looks like a slow subsystem. A gain-tuning strategy for this controller can be found in [23]. Since the previous inverse kinematic algorithm cannot provide acceleration quantities by itself, the feedforward torque terms in $\tau_i$ were omitted to compare the two algorithms on the same conditions.

Figs. 6-9 show the experimental results of the tracking control for the triangular trajectories obtained before in the numerical simulation. The same controller and the same control gains were used for each case. When $T_f = 1.5$ s, the output tracking results are nearly the same for the two inverse kinematic algorithms, as shown in Figs. 6 and 7, but the proposed algorithm shows better results. The superiority of the proposed algorithm is clearly seen when $T_f = 1.5$ s. Fig. 8 shows the control performance when the solution of the previous inverse kinematics was used. There are undamped oscillations during the trajectory tracking. The settling of the vibration also takes much time. Sharp peaks can be noticed in Fig. 8(a) and (b) when there are changes of acceleration. The output tracking performance is very poor. On the contrary, the capability of the proposed kinematics algorithm is very good for suppression of oscillation and output tracking, as shown in Fig. 9. The proposed inverse kinematics algorithm brings out much difference in performance between the slow and the fast cases. This elucidates that the solution of the proposed inverse kinematics can accommodate a substantial amount of dynamic force arising in the fast motion. Eventually, the tracking performance is highly enhanced in broad range of dynamic motion.

V. CONCLUSION

There have been many research results on the inverse kinematics of multibody flexible robots, even though it is the fundamental work required in motion control. Most of the studies on the tracking control of flexible robots have tried to solve it in terms of inverse dynamics, which looks very complex and difficult to realize in practice. From this motivation, we formulated an inverse kinematics algorithm, which is much easier for obtaining solutions than the inverse dynamics approach. By imposing the dynamic constraint to the inverse kinematics algorithm, we made the resulting solutions effective in high-speed tasks. The singular perturbation technique is applied to reduce the inverse kinematic equation to a lower dimension and to make the recursive form of differential-algebraic equation. The recursive equation could be solved by the explicit expansion method. An implicit method was also devised, which is equivalent to the explicit one, but does not require explicit expansion of the involved matrices and variables. The numerical simulation and experiment tests proved the effectiveness of the proposed algorithm. We found that the consideration of the dynamic force is indispensable in achieving high speed and high accuracy control of flexible systems. This would help expand the application domains of the flexible robot.

REFERENCES

Van Kyun Cheng (S83-M91) received the B.S. degree in mechanical design from Seoul National University, Seoul, Korea, in 1981. He received the M.S. degree in mechanical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejon, Korea, in 1983, and the Ph.D. degree in production engineering from RPI in 1987.

He is currently a Professor in the School of Mechatanical Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Korea, where he has been on the faculty since 1987. In 1988, he was a Visiting Professor at the Robotics Institute of Carnegie-Mellon University, Pittsburgh, PA. In 1993, he was a Visiting Scholar at the University of California, Berkeley. His research interests include mobile robots, underwater robotics, and development of control algorithms for precision motion control. He is a Director of the National Research Laboratory of Intelligent Space Robot Navigation.

Youngil Youn received the B.S. degree from Illinois State University, Normal, in 1984, the M.S. degree in mechanical engineering and engineering mechanics in 1987 and 1983, respectively, and the Ph.D. degree in mechanical engineering in 1978, all from the University of Wisconsin-Madison.

He was the Engineering Director for the Human Upper Extremity Laboratory at the hospital of the medical school, University of Iowa, Iowa City, in 1974. In 1978, he transferred to the Catholic University of America, Washington, DC, and became a teaching professor in the Department of Mechanical Engineering. He was a consultant to the National Institute of Standards and Technology in 1981 and served as the first biomechanics lab at the institute. He served as an Adjunct Professor at the Department of Orthopedics at the Medical College of Virginia, in 1964, he became the first Chair of the Mechatronics Engineering Department, Pohang University of Science and Technology (POSTECH), Pohang, Korea. After that, he became the Dean of Academic Affairs, and then the University's Vice President for six years. Currently, he is the Director of the POSTECH Center for Bio-Mechatronics, Human machine research, in art and robotics, and biomimetic mechanisms.