Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea • May 21-26, 2001

# $O(N)$ Forward Dynamics Computation of Open Kinematic Chains Based on the Principle of Virtual Work

Katsu YAMANE[*1*2] Yoshihiko NAKAMURA[*1*3]
(E-mail: katz@ynl.t.u-tokyo.ac.jp) (E-mail: nakamura@ynl.t.u-tokyo.ac.jp)
[*1] Department of Mechano-Informatics, University of Tokyo
7-3-1 Hongo Bunkyo-ku Tokyo, 113-8656 JAPAN
[*2]Research Fellow of the Japan Society for the Promotion of Science
[*3]CREST, Japan Science and Technology Corporation

## Abstract

This paper describes an efficient algorithm for the forward dynamics of open kinematic chains with $O(N)$ complexity, where $N$ is the number of links in the chain. The method is based on the Principle of Virtual Work and does not use any theory in linear algebra or the concept of Articulated Body Inertia. The idea of this method is to add a link one by one from the leaf links to the root evaluating the constraint force at each new joint. The algorithm consists of two iterative procedures: from the leaf links to the root to compute the constraint forces, and from the root to the leaf to compute the joint accelerations. Some numerical examples show the efficiency of the proposed algorithm. Similarity and differences with other $O(N)$ algorithms are also discussed.

**Key Words**: Forward Dynamics, Open Kinematic Chains, $O(N)$ Complexity, Principle of Virtual Work.

## 1 Introduction

Forward dynamics computation of kinematic chains has a wide range of applications and therefore many research have been made in this area. The most basic and conventional methods have $O(N^3)$ or $O(N^2)$ complexities where $N$ is the number of link in the chain, due to the explicit computation of inverse of the inertial matrix[1, 2]. These methods may be enough for simple kinematic chains with relatively small $N$, but obviously not for complex systems such as humanoid robots because the computation time grows too fast along with the increase of $N$.

For such applications, a forward dynamics algorithm should have $O(N)$ complexity, i.e. the amount of computation increases linearly to $N$. The first practical algorithm to achieve this complexity is the Articulated Body Algorithm (ABA) proposed by Featherstone[3]. ABA uses the concept of Articulated Body Inertia (ABI), which relates the acceleration of a body in the chain (called handle) with the force applied to it (test force). The key point of ABA is that ABI is computed by a recursive computation starting from the leaf links and ending at the root link, and that the size of ABI is constant for any $N$.

Other methods with $O(N)$ complexity but without ABI include Constraint Force Algorithm (CFA) by Fijany et al.[4] and Baraff's method using Lagrange multipliers[5]. Orginal version of CFA is applicable only to unbranched serial chains, but Featherstone and Fijany[6] extended it to allow short branches by partially using ABI. These methods make use of advanced theories in linear algebra to solve the equation of motion efficiently.

In this paper, we propose a new method for the forward dynamics computation of open kinematic chains based on the Principle of Virtual Work which we have employed in [7] to compute the dynamics of structure-varying kinematic chains. The $O(N)$ complexity of the method presented in this paper is realized by the efficient evaluation of the equation of motion derived on the basis of the Principle of Virtual Work, instead of linear algebra theories or the concept of ABI.

The rest of this paper is organized as follows. In section 2, we present the basic equations of motion based on the Priciple of Virtual Work. We then derive the $O(N)$ forward dynamics algorithm for unbranched serial chains in section 3, which is extended to branched open kinematic chains in section 4. In section 5 we present several numerical examples that demonstrate the efficiency of the proposed algorithm. Discussions on the relationship between our algorithm and other several ones are given in section 6, followed by the concluding remarks.

## 2 Dynamics of Kinematic Chain Using the Principle of Virtual Work

The basis of the method proposed in this paper is the dynamics computation of closed kinematic chains originally proposed by Nakamura and Ghodoussi[8]. Practically the original method was only applicable

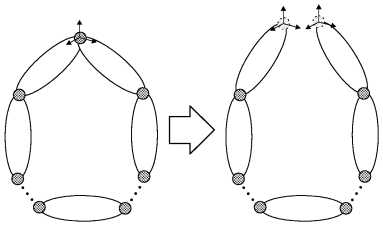

Figure 1: A closed kinematic chain (left) and its corresponding virtual open kinematic chain (right)

to planar or simple closed kinematic chains, however the restriction was then removed to handle general closed kinematic chains by Nakamura and Yamane[7]. These methods first virtually cut some joints to get open kinematic chains, and then use the Principle of Virtual Work to compute the constraint forces at the cut joints. The outline is given in this section.

Consider a closed kinematic chain shown in the left-hand side of Fig.1. The closed loop is cut at a joint and a virtual open kinematic chain is generated as shown in the right-hand side. The frame formerly attached to the cut joint is now split into a pair of frames attached to two links at the both sides of the cut joint, subject to some constraint conditions.

The dynamics of the closed kinematic chain is described as

$$\boldsymbol{\tau}_G = \boldsymbol{A}\ddot{\boldsymbol{q}}_G + \boldsymbol{b} \tag{1}$$

where $\boldsymbol{q}_G \in \boldsymbol{R}^{N_G}$ is the generalized coordinates, $\boldsymbol{A} \in \boldsymbol{R}^{N_G \times N_G}$ the inertial matrix, $\boldsymbol{b} \in \boldsymbol{R}^{N_G}$ the velocity dependent forces, $\boldsymbol{\tau}_G \in \boldsymbol{R}^{N_G}$ the generalized forces applied to on the chain, and $N_G$ the degrees of freedom (DOF) of the chain. There are two typical ways to select $\boldsymbol{q}_G$: use position and orientation of all links[2, 5], or joint angles[1, 7]. The latter is efficient in the sense that the number of unknowns to solve becomes much smaller than the former. The advantage of the former approach is, on the other hand, that the mass matrix $\boldsymbol{A}$ and the constraint matrix (described later) tend to be sparse. Some methods make use of this fact to derive efficient solutions[5].

In the actual closed kinematic chain, constraint forces/torques are applied to maintain the constraint condition. Suppose $N_{loop}$ closed loops are cut at $N_{loop}$ joints and let $\dot{\boldsymbol{r}}_C \in \boldsymbol{R}^{6N_{loop}}$ denote the spatial relative velocities of all pairs of frames at the cut joints. The constraint conditions are expressed as

$$\boldsymbol{K}\dot{\boldsymbol{r}}_C = \boldsymbol{O} \tag{2}$$

where $\boldsymbol{K} \in \boldsymbol{R}^{N_C \times 6N_{loop}}$ is the constraint matrix and $N_C \leq 6N_{loop}$ is the total number of constraints. In the following discussion we treat $\boldsymbol{K}$ as a constant matrix, which is often the for with normal types of joints, although it is not difficult to handle time-varying constraint matrices such as those for nonholonomic constraints.

Next we define $\boldsymbol{H}_C$, the Jacobian matrix of $\boldsymbol{r}_C$ with respect to the generalized coordinates, as

$$\boldsymbol{H}_C \triangleq \frac{\partial \boldsymbol{r}_C}{\partial \boldsymbol{q}_G}. \tag{3}$$

Using $\boldsymbol{H}_C$, the constraint equation Eq.(2) is rewritten as

$$\boldsymbol{H}\dot{\boldsymbol{q}}_G = \boldsymbol{O} \tag{4}$$

where $\boldsymbol{H} \triangleq \boldsymbol{K}\boldsymbol{H}_C$. Differentiating Eq.(4) yields the relationship of acceleration:

$$\boldsymbol{H}\ddot{\boldsymbol{q}}_G + \dot{\boldsymbol{H}}\dot{\boldsymbol{q}}_G = \boldsymbol{O}. \tag{5}$$

On the other hand, the generalized force applied to the system can be compute by applying the Principle of Virtual Work to Eq.(4) as

$$\boldsymbol{\tau}_G = \boldsymbol{\tau}_W + \boldsymbol{H}^T\boldsymbol{\tau}_C \tag{6}$$

where $\boldsymbol{\tau}_W \in \boldsymbol{R}^{N_G}$ includes the effects of joint forces/torques and external forces to the generalized force and $\boldsymbol{\tau}_C$ is the constraint force.

Combining Eqs.(1)(5)(6) yields the following equation:

$$\begin{pmatrix} \boldsymbol{A} & -\boldsymbol{H}^T \\ \boldsymbol{H} & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{q}}_G \\ \boldsymbol{\tau}_C \end{pmatrix} = \begin{pmatrix} \boldsymbol{\tau}_W - \boldsymbol{b} \\ -\dot{\boldsymbol{H}}\dot{\boldsymbol{q}}_G \end{pmatrix} \tag{7}$$

by which we can compute the two unknowns, the generalized acclerations $\ddot{\boldsymbol{q}}_G$ and the constraint forces $\boldsymbol{\tau}_C$. However, explicitly solving Eq.(7) requires $O(N_G^3 + N_C^3)$ computations which is not acceptable when $N_G$ is very large.

## 3 $O(N)$ Algorithm for Serial Chains

In this section we describe the new $O(N)$ forward dynamics algorithm for unbranched serial chains. General open kinematic chains with branches are handled in the next section.

The basic approach here is to add links one by one, starting from a single independent link, and see the amount of computations we need to evaluate the effect of the new joint using the Principle of Virtual Work. If it is independent of the number of links already we have, then the total asymptotic complexity becomes $O(N)$.

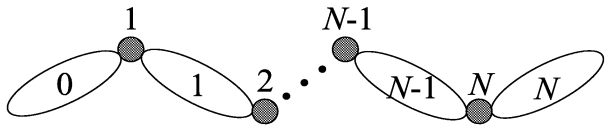


Figure 2: A serial chain with $N+1$ links and $N$ joints

### 3.1 Notations

Consider a serial kinematic chain shown in Fig.2 which consists of $N+1$ links numbered 0 to $N$ connected by $N$ joints numbered 1 to $N$.

We define the following variables for link $k$:

$$\begin{aligned}
\boldsymbol{A}_k \in \boldsymbol{R}^{6\times 6} &: \text{spatial inertial matrix}\\
\boldsymbol{b}_k \in \boldsymbol{R}^{6} &: \text{velocity dependent forces}\\
\boldsymbol{f}_k \in \boldsymbol{R}^{6} &: \text{external spacial forces}\\
\dot{\boldsymbol{\theta}}_k \in \boldsymbol{R}^{6} &: \text{spacial velocity.}
\end{aligned}$$

The numbering rule lets joint $m$ connect links $m-1$ and $m$. We define the following variables for joint $m$:

$$\begin{aligned}
\dot{\boldsymbol{r}}_{Cm} \in \boldsymbol{R}^{6} &: \text{spatial relative velocity of links } m \text{ and } m-1 \text{ at joint } m\\
\boldsymbol{H}_{Cm,k} \in \boldsymbol{R}^{N_{Cm}\times 6} &\triangleq \frac{\partial \boldsymbol{r}_{Cm}}{\partial \boldsymbol{\theta}_k} \in \boldsymbol{R}^{6\times 6}(k=m-1,m)\\
\boldsymbol{n}_{Jm} \in \boldsymbol{R}^{6-N_{Cm}} &: \text{joint torque}\\
\boldsymbol{n}_{m} \in \boldsymbol{R}^{N_{Cm}} &: \text{constraint force}\\
N_{Cm} &: \text{number of constraint condition.}
\end{aligned}$$

$\boldsymbol{H}_{Cm,k}$ is the Jacobian matrix of the relative velocity of links $m$ and $m-1$ with respect to the velocity of link $k$, which is non-zero only for $k=m, m-1$. Using $\boldsymbol{H}_{Cm,m}$ and $\boldsymbol{H}_{Cm,m-1}$, $\dot{\boldsymbol{r}}_{Cm}$ is computed by

$$\dot{\boldsymbol{r}}_{Cm} = \boldsymbol{H}_{Cm,m}\dot{\boldsymbol{\theta}}_m - \boldsymbol{H}_{Cm,m-1}\dot{\boldsymbol{\theta}}_{m-1}. \tag{8}$$

Let the constraint conditions at joint $m$ be expressed in the following form:

$$\boldsymbol{K}_m\dot{\boldsymbol{r}}_{Cm} = \boldsymbol{O} \tag{9}$$

where $\boldsymbol{K}_m$ is an $N_{Cm}\times 6$ matrix. Similarly, let the joint velocity $\dot{\boldsymbol{q}}_m \in \boldsymbol{R}^{6-N_{Cm}}$ be expressed in the following form:

$$\dot{\boldsymbol{q}}_m = \boldsymbol{K}_{Jm}\dot{\boldsymbol{r}}_{cm} \tag{10}$$

where $\boldsymbol{K}_{Jm}$ is a $(6-N_{Cm})\times 6$ matrix with the following property:

$$\boldsymbol{K}_m\boldsymbol{K}_{Jm}^T = \boldsymbol{O}. \tag{11}$$

Differentiating Eq.(8) yields the relationship of relative acceleration:

$$\begin{aligned}
\ddot{\boldsymbol{r}}_{Cm} &= \boldsymbol{H}_{Cm,m}\ddot{\boldsymbol{\theta}}_m - \boldsymbol{H}_{Cm,m-1}\ddot{\boldsymbol{\theta}}_{m-1}\\
&\quad + \dot{\boldsymbol{H}}_{Cm,m}\dot{\boldsymbol{\theta}}_m - \dot{\boldsymbol{H}}_{Cm,m-1}\dot{\boldsymbol{\theta}}_{m-1}.
\end{aligned}\tag{12}$$

Using Eq.(9) and (12) we get the acceleration constraint as follows:

$$\begin{aligned}
&\boldsymbol{H}_{m,m}\ddot{\boldsymbol{\theta}}_m - \boldsymbol{H}_{m,m-1}\ddot{\boldsymbol{\theta}}_{m-1}\\
&\quad +\dot{\boldsymbol{H}}_{m,m}\dot{\boldsymbol{\theta}}_m - \dot{\boldsymbol{H}}_{m,m-1}\dot{\boldsymbol{\theta}}_{m-1} = \boldsymbol{O}
\end{aligned}\tag{13}$$

where $\boldsymbol{H}_{m,k} \triangleq \boldsymbol{K}_m\boldsymbol{H}_{Cm,k}$ $(k=m,m-1)$. Similarly, relationship of joint accelerations is expressed as:

$$\begin{aligned}
\ddot{\boldsymbol{q}}_m &= \boldsymbol{H}_{Jm,m}\ddot{\boldsymbol{\theta}}_m - \boldsymbol{H}_{Jm,m-1}\ddot{\boldsymbol{\theta}}_{m-1}\\
&\quad + \dot{\boldsymbol{H}}_{Jm,m}\dot{\boldsymbol{\theta}}_m - \dot{\boldsymbol{H}}_{Jm,m-1}\dot{\boldsymbol{\theta}}_{m-1}
\end{aligned}\tag{14}$$

where $\boldsymbol{H}_{Jm,k} \triangleq \boldsymbol{K}_{Jm}\boldsymbol{H}_{Cm,k}$ $(k=m,m-1)$.

Since link accelerations, constraint forces and joint accelerations change as the number of links increases, we must specify at which stage the value was computed. In the following discussions, we let the left-upper index indicate the number of links involved when the value was computed. For example, ${}^i\boldsymbol{n}_j (i \geq j)$ denotes the constraint force at joint $j$ when we had joints 1 to $i$.

### 3.2 Basic Equations

First consider the case we only have links 0 to $i$ connected by constraints 1 to $i$. Link $k$ is subject to joint and constraint forces at joints $k$ and $k-1$, therefore its equation of motion is derived from the Priciple of Virtual Work as:

$$\begin{aligned}
\boldsymbol{A}_k{}^i\ddot{\boldsymbol{\theta}}_k + \boldsymbol{b}_k &= \boldsymbol{f}_k + \boldsymbol{H}_{Jk,k}^T\boldsymbol{n}_{Jk} - \boldsymbol{H}_{Jk+1,k}^T\boldsymbol{n}_{Jk+1}\\
&\quad + \boldsymbol{H}_{k,k}^T{}^i\boldsymbol{n}_k - \boldsymbol{H}_{k+1,k}^T{}^i\boldsymbol{n}_{k+1}.
\end{aligned}\tag{15}$$

with the exceptions of links 1 to $i-1$, whose equations of motion are written as:

$$\boldsymbol{A}_0\ddot{\boldsymbol{\theta}}_0 + \boldsymbol{b}_0 = \boldsymbol{f}_0 - \boldsymbol{H}_{J1,1}^T\boldsymbol{n}_{J1} - \boldsymbol{H}_{1,1}^T{}^i\boldsymbol{n}_1 \tag{16}$$

$$\boldsymbol{A}_i\ddot{\boldsymbol{\theta}}_i + \boldsymbol{b}_i = \boldsymbol{f}_i + \boldsymbol{H}_{Ji,i}^T\boldsymbol{n}_{Ji} + \boldsymbol{H}_{i,i}^T{}^i\boldsymbol{n}_i. \tag{17}$$

Combining Eqs.(15)–(17) of links 0 to $i$ and Eqs.(13)(14) of joints 1 to $i$ yields the equations in global form:

$$\boldsymbol{\Lambda}_i{}^i\ddot{\boldsymbol{\Theta}}_i = \boldsymbol{T}_i + \boldsymbol{\Gamma}_i^{Ti}\boldsymbol{N}_i \tag{18}$$

$$\boldsymbol{\Gamma}_i{}^i\ddot{\boldsymbol{\Theta}}_i = \boldsymbol{\gamma}_i \tag{19}$$

$${}^i\ddot{\boldsymbol{Q}}_i = \boldsymbol{\Gamma}_{Ji}{}^i\ddot{\boldsymbol{\Theta}}_i - \boldsymbol{\gamma}_{Ji} \tag{20}$$

where

$$
\begin{aligned}
\boldsymbol{\Lambda}_i &\triangleq diag(\boldsymbol{A}_k)\ (k=0,1,\ldots,i) \in \boldsymbol{R}^{6(i+1)\times 6(i+1)} \\
{}^i\ddot{\boldsymbol{\Theta}}_i &\triangleq \left( \begin{array}{cccc} {}^i\ddot{\boldsymbol{\theta}}_0^T & {}^i\ddot{\boldsymbol{\theta}}_1^T & \ldots & {}^i\ddot{\boldsymbol{\theta}}_i^T \end{array} \right)^T \in \boldsymbol{R}^{6(i+1)} \\
\boldsymbol{B}_i &\triangleq \left( \begin{array}{cccc} \boldsymbol{b}_0^T & \boldsymbol{b}_1^T & \ldots & \boldsymbol{b}_i^T \end{array} \right)^T \in \boldsymbol{R}^{6(i+1)} \\
\boldsymbol{F}_i &\triangleq \left( \begin{array}{cccc} \boldsymbol{f}_0^T & \boldsymbol{f}_1^T & \ldots & \boldsymbol{f}_i^T \end{array} \right)^T \in \boldsymbol{R}^{6(i+1)} \\
{}^i\boldsymbol{N}_i &\triangleq \left( \begin{array}{cccc} {}^i\boldsymbol{n}_1^T & {}^i\boldsymbol{n}_2^T & \ldots & {}^i\boldsymbol{n}_i^T \end{array} \right)^T \in \boldsymbol{R}^{{}^iN_C} \\
\boldsymbol{N}_{Ji} &\triangleq \left( \begin{array}{cccc} \boldsymbol{n}_{J1}^T & \boldsymbol{n}_{J2}^T & \ldots & \boldsymbol{n}_{Ji}^T \end{array} \right)^T \in \boldsymbol{R}^{6i-{}^iN_C} \\
{}^i\ddot{\boldsymbol{Q}}_i &\triangleq \left( \begin{array}{cccc} {}^i\ddot{\boldsymbol{q}}_1^T & {}^i\ddot{\boldsymbol{q}}_2^T & \ldots & {}^i\ddot{\boldsymbol{q}}_i^T \end{array} \right)^T \in \boldsymbol{R}^{6i-{}^iN_C} \\
\boldsymbol{T}_i &\triangleq \boldsymbol{F}_i + \boldsymbol{\Gamma}_{Ji}^T \boldsymbol{N}_{Ji} - \boldsymbol{B}_i \\
\boldsymbol{\gamma}_i &\triangleq -\dot{\boldsymbol{\Gamma}}_i \dot{\boldsymbol{\Theta}}_i \\
\boldsymbol{\gamma}_{Ji} &\triangleq -\dot{\boldsymbol{\Gamma}}_{Ji} \dot{\boldsymbol{\Theta}}_i \\
{}^iN_C &\triangleq \sum_{m=1}^{i} N_{Cm}
\end{aligned}
$$

and $diag(\boldsymbol{A}_i)$ denotes a block diagonal matrix whose $i$-th entry is $\boldsymbol{A}_i$, and $\boldsymbol{\Gamma}_i \in \boldsymbol{R}^{{}^iN_C \times 6(i+1)}$ is a matrix written as

$$
\boldsymbol{\Gamma}_i = \left( \begin{array}{ccccc} -\boldsymbol{H}_{1,0} & \boldsymbol{H}_{1,1} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & -\boldsymbol{H}_{2,1} & \boldsymbol{H}_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \boldsymbol{O} \\ \boldsymbol{O} & \cdots & \boldsymbol{O} & -\boldsymbol{H}_{i,i-1} & \boldsymbol{H}_{i,i} \end{array} \right).
$$

$\boldsymbol{\Gamma}_{Ji}$ has the same structure as $\boldsymbol{\Gamma}_i$ except that $\boldsymbol{H}_{m,k} (m = 1,2,\ldots,i;\ k = m, m-1)$ are replaced by $\boldsymbol{H}_{Jm,k}$.

Solving Eq.(18) in terms of ${}^i\ddot{\boldsymbol{\Theta}}_i$ and substituting it into Eq.(19) yields

$$
\boldsymbol{\Phi}_i {}^i\boldsymbol{N}_i = \boldsymbol{\alpha}_i \tag{21}
$$

where

$$
\begin{aligned}
\boldsymbol{\Phi}_i &\triangleq \boldsymbol{\Gamma}_i \boldsymbol{\Lambda}_i^{-1} \boldsymbol{\Gamma}_i^T \\
\boldsymbol{\alpha}_i &\triangleq \boldsymbol{\gamma}_i - \boldsymbol{\Gamma}_i \boldsymbol{\Lambda}_i^{-1} \boldsymbol{T}_i.
\end{aligned}
$$

If $\boldsymbol{\Phi}_i$ is non-singular, then the constraint force ${}^i\boldsymbol{N}_i$ can be computed by

$$
{}^i\boldsymbol{N}_i = \boldsymbol{\Phi}_i^{-1} \boldsymbol{\alpha}_i \tag{22}
$$

which is not actually executed here because it requires as much as $O(N_C^3)$ computations.

The physical meaning of Eqs.(21)(22) is explained as follows. $\boldsymbol{\alpha}_i$ denotes the accelerations that should be generated at joints in order to maintain the constraint. ${}^i\boldsymbol{N}_i$, on the other hand, is the constraint forces that should be exerted at joints to maintain the constraint condition. The matrix $\boldsymbol{\Phi}_i$ can therefore be considered as a kind of intertial matrix.

Now we are ready to add link $i+1$ via joint $i+1$. The new equations of motion, constraint and joint accelerations are:

$$
\begin{aligned}
&\left( \begin{array}{cc} \boldsymbol{\Lambda}_i & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{A}_{i+1} \end{array} \right) \left( \begin{array}{c} {}^{i+1}\ddot{\boldsymbol{\Theta}}_i \\ {}^{i+1}\ddot{\boldsymbol{\theta}}_{i+1} \end{array} \right) \\
&\quad = \left( \begin{array}{cc} \boldsymbol{\Gamma}_i^T & \boldsymbol{\Gamma}_{i+1,i}^T \\ \boldsymbol{O} & \boldsymbol{H}_{i+1,i+1}^T \end{array} \right) \left( \begin{array}{c} {}^{i+1}\boldsymbol{N}_i \\ {}^{i+1}\boldsymbol{n}_{i+1} \end{array} \right) \\
&\qquad + \left( \begin{array}{c} \boldsymbol{T}_i \\ \boldsymbol{t}_{i+1} \end{array} \right)
\end{aligned} \tag{23}
$$

$$
\left( \begin{array}{cc} \boldsymbol{\Gamma}_i & \boldsymbol{O} \\ \boldsymbol{\Gamma}_{i+1,i} & \boldsymbol{H}_{i+1,i+1} \end{array} \right) \left( \begin{array}{c} {}^{i+1}\ddot{\boldsymbol{\Theta}}_i \\ {}^{i+1}\ddot{\boldsymbol{\theta}}_{i+1} \end{array} \right) = \left( \begin{array}{c} \boldsymbol{\gamma}_i \\ \boldsymbol{h}_{i+1} \end{array} \right) \tag{24}
$$

$$
\begin{aligned}
&\left( \begin{array}{c} {}^{i+1}\ddot{\boldsymbol{Q}}_i \\ {}^{i+1}\ddot{\boldsymbol{q}}_{i+1} \end{array} \right) \\
&\quad = \left( \begin{array}{cc} \boldsymbol{\Gamma}_{Ji} & \boldsymbol{O} \\ \boldsymbol{\Gamma}_{Ji+1,i} & \boldsymbol{H}_{Ji+1,i+1} \end{array} \right) \left( \begin{array}{c} {}^{i+1}\ddot{\boldsymbol{\Theta}}_i \\ {}^{i+1}\ddot{\boldsymbol{\theta}}_{i+1} \end{array} \right) \\
&\qquad - \left( \begin{array}{c} \boldsymbol{\gamma}_{Ji} \\ \boldsymbol{h}_{Ji+1} \end{array} \right)
\end{aligned} \tag{25}
$$

where

$$
\begin{aligned}
\boldsymbol{\Gamma}_{i+1,i} &\triangleq \left( \begin{array}{cccc} \boldsymbol{O} & \ldots & \boldsymbol{O} & -\boldsymbol{H}_{i+1,i} \end{array} \right) && (26) \\
\boldsymbol{\Gamma}_{Ji+1,i} &\triangleq \left( \begin{array}{cccc} \boldsymbol{O} & \ldots & \boldsymbol{O} & -\boldsymbol{H}_{Ji+1,i} \end{array} \right) && (27) \\
\boldsymbol{t}_{i+1} &\triangleq \boldsymbol{f}_{i+1} + \boldsymbol{H}_{Ji+1,i+1}^T \boldsymbol{n}_{Ji+1} - \boldsymbol{b}_{i+1} && (28) \\
\boldsymbol{h}_{i+1} &\triangleq \dot{\boldsymbol{H}}_{i+1,i} \dot{\boldsymbol{\theta}}_i - \dot{\boldsymbol{H}}_{i+1,i+1} \dot{\boldsymbol{\theta}}_{i+1} && (29) \\
\boldsymbol{h}_{Ji+1} &\triangleq \dot{\boldsymbol{H}}_{Ji+1,i} \dot{\boldsymbol{\theta}}_i - \dot{\boldsymbol{H}}_{Ji+1,i+1} \dot{\boldsymbol{\theta}}_{i+1}. && (30)
\end{aligned}
$$

Solving Eq.(23) in terms of $({}^{i+1}\ddot{\boldsymbol{\Theta}}_i^T\ {}^{i+1}\ddot{\boldsymbol{\theta}}_{i+1}^T)^T$ and substituting into Eq.(24) yields

$$
\left( \begin{array}{cc} \boldsymbol{\Phi}_i & \boldsymbol{\Phi}_{i+1,i}^T \\ \boldsymbol{\Phi}_{i+1,i} & \boldsymbol{S}_{i+1} \end{array} \right) \left( \begin{array}{c} {}^{i+1}\boldsymbol{N}_i \\ {}^{i+1}\boldsymbol{n}_{i+1} \end{array} \right) = \left( \begin{array}{c} \boldsymbol{\alpha}_i \\ \boldsymbol{a}_{i+1} \end{array} \right) \tag{31}
$$

where

$$
\begin{aligned}
\boldsymbol{\Phi}_{i+1,i} &\triangleq \boldsymbol{\Gamma}_{i+1,i} \boldsymbol{\Lambda}_i^{-1} \boldsymbol{\Gamma}_i^T && (32) \\
\boldsymbol{S}_{i+1} &\triangleq \boldsymbol{H}_{i+1,i} \boldsymbol{A}_i^{-1} \boldsymbol{H}_{i+1,i}^T + \boldsymbol{H}_{i+1,i+1} \boldsymbol{A}_{i+1}^{-1} \boldsymbol{H}_{i+1,i+1}^T \\
\boldsymbol{a}_{i+1} &\triangleq \boldsymbol{h}_{i+1} - \boldsymbol{H}_{i+1,i+1} \boldsymbol{A}_{i+1}^{-1} \boldsymbol{t}_{i+1}
\end{aligned}
$$

or in global form,

$$
\boldsymbol{\Phi}_{i+1} {}^{i+1}\boldsymbol{N}_{i+1} = \boldsymbol{\alpha}_{i+1} \tag{33}
$$

where

$$\boldsymbol{\Phi}_{i+1} \triangleq \begin{pmatrix} \boldsymbol{\Phi}_i & \boldsymbol{\Phi}_{i+1,i}^T \\ \boldsymbol{\Phi}_{i+1,i} & \boldsymbol{S}_{i+1} \end{pmatrix}. \tag{34}$$

Using the upper part of Eq.(31) and the relationship of Eq.(22), we get

$$\begin{aligned} {}^{i+1}\boldsymbol{N}_i &= \boldsymbol{\Phi}_i^{-1}(\boldsymbol{\alpha}_i - \boldsymbol{\Phi}_{i+1,i}^T {}^{i+1}\boldsymbol{n}_{i+1}) \\ &= {}^{i}\boldsymbol{N}_i - \boldsymbol{\Phi}_i^{-1}\boldsymbol{\Phi}_{i+1,i}^T {}^{i+1}\boldsymbol{n}_{i+1}. \end{aligned} \tag{35}$$

Substituting this equation into the lower part of Eq.(31) yeilds

$$\begin{aligned} {}^{i+1}\boldsymbol{n}_{i+1} &= \boldsymbol{M}_{i+1}(\boldsymbol{a}_{i+1} - \boldsymbol{\Phi}_{i+1,i} {}^{i}\boldsymbol{N}_i) && (36) \\ \boldsymbol{M}_{i+1} &\triangleq (\boldsymbol{S}_{i+1} - \boldsymbol{\Phi}_{i+1,i}\boldsymbol{\Phi}_i^{-1}\boldsymbol{\Phi}_{i+1,i}^T)^{-1} && (37) \end{aligned}$$

by which we can recursively compute ${}^{i+1}\boldsymbol{n}_{i+1}$, the constraint force at the new joint $i+1$. However, a careful look at Eqs.(36)(37) shows that the amount of computation required for evaluating both equations increases as the number of links involved increases. We need more investigation into Eqs.(36)(37) in order to derive $O(N)$ algorithm.

### 3.3 Derivation of $O(N)$ Algorithm

Reduction of computation is realized by making use of the structure of $\boldsymbol{\Gamma}_{i+1,i}$ shown in Eq.(26). Using Eq.(26), Eq.(32) can be rewritten as

$$\begin{aligned} \boldsymbol{\Phi}_{i+1,i} &= \boldsymbol{\Gamma}_{i+1,i}\boldsymbol{\Lambda}_i^{-1}\boldsymbol{\Gamma}_i^T \\ &= \begin{pmatrix} \boldsymbol{O} & \dots & \boldsymbol{O} & -\boldsymbol{H}_{i+1,i} \end{pmatrix} \\ &\quad \cdot \begin{pmatrix} \boldsymbol{A}_1^{-1} & & & \\ & \boldsymbol{A}_2^{-1} & & \\ & & \ddots & \\ & & & \boldsymbol{A}_i^{-1} \end{pmatrix} \boldsymbol{\Gamma}_i^T \\ &= \begin{pmatrix} \boldsymbol{O} & \dots & \boldsymbol{O} & -\boldsymbol{H}_{i+1,i}\boldsymbol{A}_i^{-1} \end{pmatrix} \boldsymbol{\Gamma}_i^T \\ &= \begin{pmatrix} \boldsymbol{O} & \dots & \boldsymbol{O} & \boldsymbol{S}_{i+1,i} \end{pmatrix} \end{aligned} \tag{38}$$

where

$$\boldsymbol{S}_{i+1,i} \triangleq -\boldsymbol{H}_{i+1,i}\boldsymbol{A}_i^{-1}\boldsymbol{H}_{i,i}^T. \tag{39}$$

Substituting Eq.(38) into Eq.(36) and Eq.(37) yields

$$\begin{aligned} {}^{i+1}\boldsymbol{n}_{i+1} &= \boldsymbol{M}_{i+1}(\boldsymbol{a}_{i+1} - \boldsymbol{S}_{i+1,i} {}^{i}\boldsymbol{n}_i) && (40) \\ \boldsymbol{M}_{i+1} &= (\boldsymbol{S}_{i+1} - \boldsymbol{S}_{i+1,i} {}^{i}\boldsymbol{M}_{i,i}\boldsymbol{S}_{i+1,i}^T)^{-1} && (41) \end{aligned}$$

respectively, where ${}^{i}\boldsymbol{M}_{i,i}$ denotes the $(i,i)$-th block of $\boldsymbol{\Phi}_i^{-1}$.

Eq.(40) shows that we can compute ${}^{i+1}\boldsymbol{n}_{i+1}$ recursively from ${}^{i}\boldsymbol{n}_i$ with a bounded amount of computation for any $i$ provided that we have $\boldsymbol{M}_{i+1}$. If we write $\boldsymbol{\Phi}_{i+1}^{-1}$ in a block form as

$$\boldsymbol{\Phi}_{i+1}^{-1} = \begin{pmatrix} \boldsymbol{\Omega}_i & \boldsymbol{\Omega}_{i,i+1} \\ \boldsymbol{\Omega}_{i+1,i} & \boldsymbol{\Omega}_{i+1} \end{pmatrix} \tag{42}$$

it is readily shown from Eq.(34) that each block can be written as follows:

$$\begin{aligned} \boldsymbol{\Omega}_i &= \boldsymbol{\Phi}_i^{-1} + \boldsymbol{\Phi}_i^{-1}\boldsymbol{\Phi}_{i+1,i}^T\boldsymbol{M}_{i+1}\boldsymbol{\Phi}_{i+1,i}\boldsymbol{\Phi}_i^{-1} \\ \boldsymbol{\Omega}_{i,i+1} &= -\boldsymbol{\Phi}_i^{-1}\boldsymbol{\Phi}_{i+1,i}^T\boldsymbol{M}_{i+1} \\ \boldsymbol{\Omega}_{i+1,i} &= -\boldsymbol{M}_{i+1}\boldsymbol{\Phi}_{i+1,i}\boldsymbol{\Phi}_i^{-1} \\ \boldsymbol{\Omega}_{i+1} &= \boldsymbol{M}_{i+1}. \end{aligned} \tag{43}$$

Replacing $i+1$ with $i$ shows that the $(i,i)$-th block of $\boldsymbol{\Phi}_i^{-1}$ is $\boldsymbol{M}_i$. Therefore, the computational cost for computing $\boldsymbol{M}_{i+1}$ from $\boldsymbol{M}_i$ is also bounded for any $i$.

Executing the above procedure from link 0 to $N$, we finally get ${}^{i}\boldsymbol{n}_i$ and $\boldsymbol{M}_i$ for $i = 1 \dots N$. Note that the constraint forces already computed are ${}^{i}\boldsymbol{n}_i$, not the desired ${}^{N}\boldsymbol{n}_i$, which means that we need another step to compute the constraint forces for the completed chain. This is done by a recursive procedure from link $N$ back to link 0 as follows.

After adding the last link $N$, we have the following equation corresponding to Eq.(33):

$$\boldsymbol{\Phi}_N {}^{N}\boldsymbol{N}_N = \boldsymbol{\alpha}_N. \tag{44}$$

If we divide this equation into three parts, joints 1 to $i-1$, joint $i$, and joints $i+1$ to $N$, we get

$$\begin{pmatrix} \boldsymbol{\Phi}_{i-1} & \boldsymbol{\Phi}_{i,i-1}^T & \boldsymbol{O} \\ \boldsymbol{\Phi}_{i,i-1} & \boldsymbol{S}_i & \boldsymbol{\Phi}_{N,i}^T \\ \boldsymbol{O} & \boldsymbol{\Phi}_{N,i} & \boldsymbol{\Phi}_N^{i+1} \end{pmatrix} \begin{pmatrix} {}^{N}\boldsymbol{N}_{i-1} \\ {}^{N}\boldsymbol{n}_i \\ {}^{N}\boldsymbol{N}_N^{i+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\alpha}_{i-1} \\ \boldsymbol{a}_i \\ \boldsymbol{\alpha}_N^{i+1} \end{pmatrix}. \tag{45}$$

Provided that we have ${}^{N}\boldsymbol{n}_i$, the final constraint force at joint $i$, we can compute ${}^{N}\boldsymbol{N}_{i-1}$ by

$$\begin{aligned} {}^{N}\boldsymbol{N}_{i-1} &= \boldsymbol{\Phi}_{i-1}^{-1}(\boldsymbol{\alpha}_{i-1} - \boldsymbol{\Phi}_{i,i-1}^T {}^{N}\boldsymbol{n}_i) \\ &= {}^{i-1}\boldsymbol{N}_{i-1} - \boldsymbol{\Phi}_{i-1}^{-1}\boldsymbol{\Phi}_{i,i-1}^T {}^{N}\boldsymbol{n}_i. \end{aligned} \tag{46}$$

Extracting the $(i-1)$-th block of Eq.(46) using Eq.(38) we get

$${}^{N}\boldsymbol{n}_{i-1} = {}^{i-1}\boldsymbol{n}_{i-1} - \boldsymbol{M}_{i-1}\boldsymbol{S}_{i,i-1}^T {}^{N}\boldsymbol{n}_i \tag{47}$$

which shows that we can compute ${}^{N}\boldsymbol{n}_{i-1}$ from ${}^{N}\boldsymbol{n}_i$ recursively.

Finally we compute the joint accelerations ${}^{N}\ddot{\boldsymbol{Q}}_N$. Replacing $i$ with $N$ in Eqs.(18)(20) gives

$$\begin{aligned} \boldsymbol{\Lambda}_N {}^{N}\ddot{\boldsymbol{\Theta}}_N &= \boldsymbol{T}_N + \boldsymbol{\Gamma}_N^T {}^{N}\boldsymbol{N}_N && (48) \\ {}^{N}\ddot{\boldsymbol{Q}}_N &= \boldsymbol{\Gamma}_{JN} {}^{N}\ddot{\boldsymbol{\Theta}}_N - \boldsymbol{\gamma}_{JN} && (49) \end{aligned}$$

respectively. Solving Eq.(48) in terms of ${}^N\ddot{\boldsymbol{\Theta}}_N$ and substituting into Eq.(49) yields

$$ {}^N\ddot{\boldsymbol{Q}}_N = \boldsymbol{\Phi}_{JN}{}^N\boldsymbol{N}_N - \boldsymbol{\alpha}_{JN} \tag{50} $$

where

$$ \begin{aligned} \boldsymbol{\Phi}_{JN} &= \boldsymbol{\Gamma}_{JN}\boldsymbol{\Lambda}_N^{-1}\boldsymbol{\Gamma}_N^T \\ \boldsymbol{\alpha}_{JN} &= \boldsymbol{\gamma}_N - \boldsymbol{\Gamma}_{JN}\boldsymbol{\Lambda}_N^{-1}\boldsymbol{T}_N. \end{aligned} \tag{51} $$

which can be divided into three parts, joints 1 to $i-1$, joint $i$, and joints $i+1$ to $N$ as

$$ \begin{pmatrix} {}^N\ddot{\boldsymbol{Q}}_{i-1} \\ {}^N\ddot{\boldsymbol{q}}_i \\ {}^N\ddot{\boldsymbol{Q}}_N^{i+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Phi}_{Ji-1} & \boldsymbol{\Phi}_{Ji,i-1}^T & \boldsymbol{O} \\ \boldsymbol{\Phi}_{Ji,i-1} & \boldsymbol{S}_{Ji} & \boldsymbol{\Phi}_{JN,i}^T \\ \boldsymbol{O} & \boldsymbol{\Phi}_{JN,i} & \boldsymbol{\Phi}_{JN}^{i+1} \end{pmatrix} \cdot \begin{pmatrix} {}^N\boldsymbol{N}_{i-1} \\ {}^N\boldsymbol{n}_i \\ {}^N\boldsymbol{N}_N^{i+1} \end{pmatrix} - \begin{pmatrix} \boldsymbol{\alpha}_{Ji-1} \\ \boldsymbol{a}_{Ji} \\ \boldsymbol{\alpha}_{JN}^{i+1} \end{pmatrix} \tag{52} $$

Picking up the row corresponding to joint $i$, we get

$$ \begin{aligned} {}^N\ddot{\boldsymbol{q}}_i &= \boldsymbol{S}_{Ji,i-1}{}^N\boldsymbol{n}_{i-1} + \boldsymbol{S}_{Ji}{}^N\boldsymbol{n}_i + \\ &\quad \boldsymbol{S}_{Ji+1,i}^T{}^N\boldsymbol{n}_{i+1} - \boldsymbol{a}_{Ji} \end{aligned} \tag{53} $$

by which we can compute the joint accelerations.

To summarize, the proposed algorithm consists of two recursive paths to compute the constraint force and joint accelerations:

1. from link 0 to $N$ to compute ${}^i\boldsymbol{n}_i$ and $\boldsymbol{M}_i$, and
2. from link $N$ to 0 to compute ${}^N\boldsymbol{n}_i$ and ${}^N\ddot{\boldsymbol{q}}_i$.

## 4 $O(N)$ Algorithm for Branched Chains

The method described in the previous section is extended to branched open kinematic chains by simply modifying the two paths as follows:

1. Path 1: from end links to the root link
2. Path 2: from the root link to end links

The only difference from the serial chain version is that we have more non-zero blocks in $\boldsymbol{\Phi}_{i+1,i}$ than we had in Eq.(38) on which the $O(N)$ complexity of the algorithm relies. In fact, the asymptotic complexity becomes $O(N^2)$ in the worst case if we treat a branched kinematic chain in its original form. The topics of this section is how to achieve the $O(N)$ complexity for general open kinematic chains.

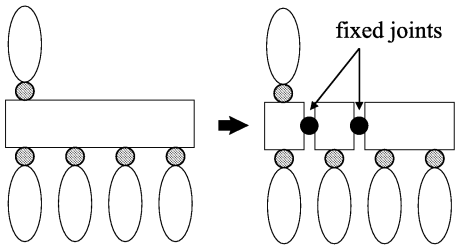


Figure 3: Split a link so that each link has no more than three connections between other links

Generally speaking, $\boldsymbol{S}_{m,n}$, the $(m,n)$-th block matrix of $\boldsymbol{\Phi}_i$, becomes nonzero if joints $m$ and $n$ a connect common link. In serial chains $\boldsymbol{\Phi}_{i+1,i}$ has only one non-zero block since joint $i+1$ is attached to link to which only two joints $i$ and $i+1$ are attached. In general open kinematic chains, on the other hand, $\boldsymbol{\Phi}_{i+1,i}$ may contain any number of non-zero blocks requiring more blocks of $\boldsymbol{\Phi}_i$ to evaluate $\boldsymbol{M}_{i+1}$. If joint $i+1$ is attached to a link with other $n_b$ joints attached, then we need $O(n_b^2)$ computation to get $\boldsymbol{M}_{i+1}$. In the worst case of $N \approx n_b$, the overall asymptotic complexity falls down to $O(N^2)$

In order to maintain the $O(N)$ complexity for branched chains, we have to limit the number of non-zero block matrices in $\boldsymbol{\Phi}_{i+1,i}$. This is achieved by splitting links with more than 3 joints as shown in Fig.3. In general, the number of joints attached to each link can be limited to 3 by splitting a link with $n$ joints into $n-2$ links connected by fixed joints. After this process, the number of non-zero blocks in each row of $\boldsymbol{\Phi}_{i+1,i}$ is limited to 2. Although the splitting increases the number of joints by $n-3$, the overall number of joints never exceeds twice of that in the original chain, so the asymptotic complexity is still $O(N)$.

## 5 Examples

The algorithm was implemented using Visual C++ and executed on a PC with a PentiumIII 850MHz processor. Figures 4 and 5 show the snapshots of dynamics simulation of serial and branched open kinematic chains.

Fig.6 shows the time to compute the joint accelerations of serial chains with 2 to 16 spherical joints. For comparison, the dashed line shows the computation time for unit vector method[1]. The new algorithm shows better performance than unit vector method for chains with more than 15 degrees of freedom (DOF), and the computation time increases linearly against

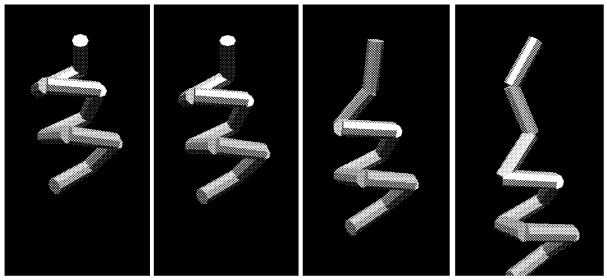

Figure 4: Dynamics simulation of a 30DOF serial chain

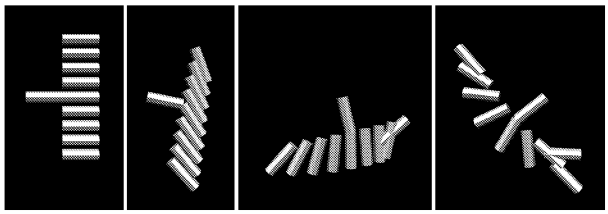

Figure 5: Dynamics simulation of a 30DOF branched chain

DOF. Note that the amount of computation for the new algorithm depends solely on the number of links, while that of unit vector method depends both on the number of links and DOF. Therefore it is recommended to minimize the number of links by utilizing multi-degrees-of-freedom joints as much as possible.

Fig.7 shows the computation time for branched chains with 2 to 12 end links hanging down from a link connected to the ground through a spherical joint, which is the most undesirable case of $N \approx n_b$ mentioned in section 4. The total degrees of freedom varies from 9 to 39. The computation time without link splitting is almost as same as unit vector method for any case, while that with link splitting increases linearly and becomes shorter for chains with more than 30DOF.

## 6 Comparison with Other Methods

This section gives intuitive comparisons between the proposed algorithm and two $O(N)$ forward dynamics algorithms ABA and CFA. Comparing the number of operations will be included in the future works.

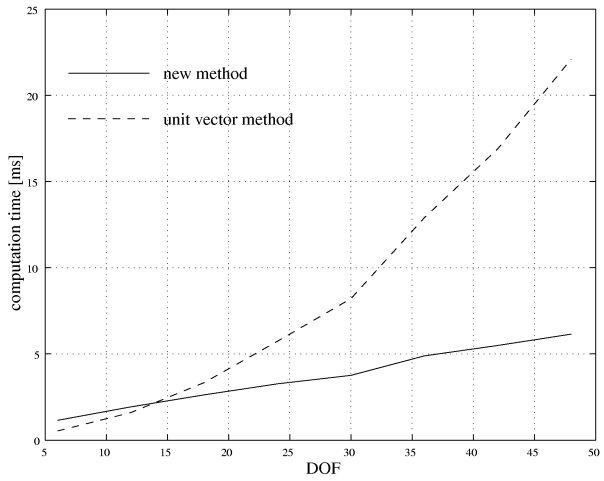


Figure 6: Computation time for serial chains; solid: proposed method, dashed: unit vector method

### 6.1 Articulated Body Algorithm

Same as our algorithm, ABA also has two main paths: one for computing ABI, another for computing joint accelerations by computing the forward dynamics of an one-joint chain. However, the physical role of each path does not coincide with that of ours.

Physical meanings of ${}^i\boldsymbol{n}_i$ and $\boldsymbol{M}_i$ in Eq.(40) can be explained as follows. Let us divide the right hand side of Eq.(40) into three parts: $\boldsymbol{a}_{i+1}$, $-\boldsymbol{S}_{i+1,i}{}^i\boldsymbol{n}_i$, and $\boldsymbol{M}_{i+1}$ multiplied to the sum of first and second parts. The first part $\boldsymbol{a}_{i+1}$ denotes the acceleration to be exerted at joint $i+1$ to maintain the constraint condition at the initial state when all links are independent. Actually, $-\boldsymbol{a}_{i+1}$ includes all bias accelerations due to velocity, gravity, external forces, joint forces, and so on. The second part $-\boldsymbol{S}_{i+1,i}{}^i\boldsymbol{n}_i$ can be interpreted as the extra acceleration at joint $i+1$ due to the constraint force at joint $i$. The sum of first and second parts, therefore, is the acceleration that should be exerted at joint $i+1$ to maintain the constraint, under the existence of joints 1 to $i$. $\boldsymbol{M}_{i+1}$ can be regarded as the inertial matrix of the chain composed of links 0 to $i$ at joint $i+1$, since it maps the acceleration and force at joint $i+1$, which is very similar to ABI.

The difference from ABI is that the inertia of link $i+1$ is included in $\boldsymbol{M}_{i+1}$ as is simply traced from Eq.(41). This is because $\boldsymbol{M}_{i+1}$ is the inertia in terms of the relative motion of links $i$ and $i+1$. The ABI when the handle is link $i$, on the other hand, will not include the inertia of link $i+1$ because the ABI is computed without assuming the existence of link $i+1$;

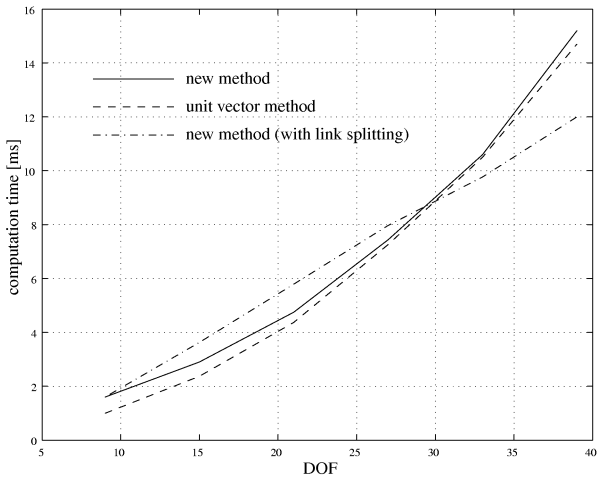


Figure 7: Computation time of branched chains; solid: proposed method without link splitting, dashed: unit vector method, dash-dotted: proposed method with link splitting

in other words, without any assumption on the source of the test force, although test force is nothing but the constraint force in a physical sense. We can also say that Path 1 of our algorithm partially executes the forward dynamics computation, and therefore includes a part of Path 2 in ABA. In addition, in our method all constraint forces are computed through the forward dynamics computation, while in ABA we need another inverse dynamics step to compute them.

### 6.2 Constraint Force Algorithm

CFA relies on the orthognality of joint space and constraint space, which was refined in the recent paper by Featherstone and Fijany[6]. In our method, this condition appears in Eq.(11), but is not use explicitly. We derive the projection between kinematic constraints and constraint forces by way of the Principle of Virtual Work.

## 7 Conclusion

The results of this paper are summarized as follows:

1. A forward dynamics algorithm with $O(N)$ complexity for unbranched and branched open kinematic chains was proposed and described in detail. The method is based on the Principle of Virtual Work, which is a quite different approach from exisitng $O(N)$ algorithms.
2. Numerical examples demonstrated the efficiency of the proposed algorithm.
3. Qualitative comparisons with other $O(N)$ algorithms were made.

Future works include a detailed evaluation of complexity and accuracy, and extension to closed kinematic chains and parallel computation.

### Acknowledgments

This research was supported by the Humanoid Robotics Project, NEDO, Japan, and the CREST Program of the Japan Science and Technology Corporation.

## References

[1] M.W. Walker and D.E. Orin: " Efficient Dynamic Computer Simulation of Robot Manipulators," *ASME Journal on Dynamic Systems, Measurement and Control*, vol.104, pp.205–211, 1982.

[2] E.J. Haug: *Computer Aided Kinematics and Dynamics of Mechanical Systems*, Allyn and Bacon Series in Engineering, 1989.

[3] R. Featherstone: *Robot Dynamics Algorithm*, Kluwer Academic Publishers, Boston, MA, 1987.

[4] A. Fijany, I. Sharf, and G.M.T. D'Eleuterio: "Parallel $O(\log N)$ Algorithms for Computation of Manipulator Forward Dynamics," *IEEE Transactions on Robotics and Automation*, vol.11, no.3, pp.389–400, 1995.

[5] David Baraff: "Linear-Time Dynamics Using Lagrange Multipliers," In *Proceedings of SIGGRAPH*, pp.137–146, 1996.

[6] R. Featherstone and A. Fijany: "A Technique for Analyzing Constrained Rigid-Body Systems, and its Application to the Constraint Force Algorithm," *IEEE Transactions on Robotics and Automation*, vol.15, no.6, pp.1140–1144, December 1999.

[7] Y. Nakamura and K. Yamane: " Dynamics Computation of Structure-Varying Kinematic Chains and Its Application to Human Figures," *IEEE Transactions on Robotics and Automation*, vol.16, no.2, pp.124–134, 2000.

[8] Y. Nakamura and M. Ghodoussi: "Dynamics Computation of Closed-Link Robot Mechanisms with Nonredundant and Redundant Actuators," *IEEE Transactions on Robotics and Automation*, vol.5, no.3, pp.294–302, 1989.