Parallel O(Log n) Algorithms for Open- and Closed-Chain Rigid Multibody Systems Based on a New Mass Matrix Factorization Technique

Amir Fijany
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109

Abstract

In this paper, parallel $O(\log n)$ algorithms for computation of rigid multibody dynamics are developed. These parallel algorithms are derived by parallelization of new $O(n)$ algorithms for the problem. The underlying feature of these $O(n)$ algorithms is a drastically different strategy for decomposition of interbody force which leads to a new factorization of the mass matrix ($\mathcal{M}$). Specifically, it is shown that a factorization of the inverse of the mass matrix in the form of the *Schur Complement* is derived as $\mathcal{M}^{-1} = \mathcal{C} - \mathcal{B}^{*}\mathcal{A}^{-1}\mathcal{B}$, wherein matrices $\mathcal{C}$, $\mathcal{A}$, and $\mathcal{B}$ are block tridiagonal matrices. The new $O(n)$ algorithm is then derived as a recursive implementation of this factorization of $\mathcal{M}^{-1}$. For the closed-chain systems, similar factorizations and $O(n)$ algorithms for computation of Operational Space Mass Matrix Λ and its inverse Λ^{-1} are also derived. It is shown that these $O(n)$ algorithms are *strictly parallel*, that is, they are less efficient than other algorithms for serial computation of the problem. But, to our knowledge, they are the only known algorithms that can be parallelized and that lead to both time- and processor-optimal parallel algorithms for the problem, i.e., parallel $O(\log n)$ algorithms with $O(n)$ processors. The developed parallel algorithms, in addition to their theoretical significance, are also practical from an implementation point of view due to their simple architectural requirements.

n	Total number of Degrees-Of-Freedom (DOF) of the system
$P_{i,j}$	Position vector from point O_j to point O_i, $P_{i+1,i} = P_i$
m_i	Mass of body i
h_i, k_i	First and Second Moment of mass of body i about point O_i
$I_{i,j}$	Spatial Inertia of body i about point O_j, $I_{i,i} = I_i = \begin{bmatrix} k_i & \tilde{h}_i \\ \tilde{h}_i^* & m_i U \end{bmatrix} \in \mathbb{R}^{6x6}$ (* denotes the transpose)
$\mathcal{M} \in \mathbb{R}^{nxn}$	Symmetric Positive Definite (SPD) mass matrix
$\theta \triangleq \text{col}\{\theta_i\} \in \mathbb{R}^{nx1}$	Vector of joint positions
$Q \triangleq \text{col}\{Q_i\} \in \mathbb{R}^{nx1}$	Vector of joint velocities
$\dot{Q} \triangleq \text{col}\{\dot{Q}_i\} \in \mathbb{R}^{nx1}$	Vector of joint accelerations
$\mathcal{T} \triangleq \text{col}\{\tau_i\} \in \mathbb{R}^{nx1}$	Vector of applied (control) joint forces/torques
ω_i, $\dot{\omega}_i$	Angular and linear acceleration of body i (frame i+1)
v_i, $\dot{v}_i$	Linear velocity and acceleration of body i (point O_i)
f_i, n_i	Force and moment of interaction between body i-1 and body i
$\dot{V}_i = \begin{bmatrix} \dot{\omega}_i \\ \dot{v}_i \end{bmatrix} \in \mathbb{R}^{6x1}$	Spatial acceleration of body i
$F_i = \begin{bmatrix} n_i \\ f_i \end{bmatrix} \in \mathbb{R}^{6x1}$	Spatial force of interaction between body i-1 and body i
$H_i \in \mathbb{R}^{6xn_i}$	Spatial axis (map matrix) of joint i

Table I. Notation

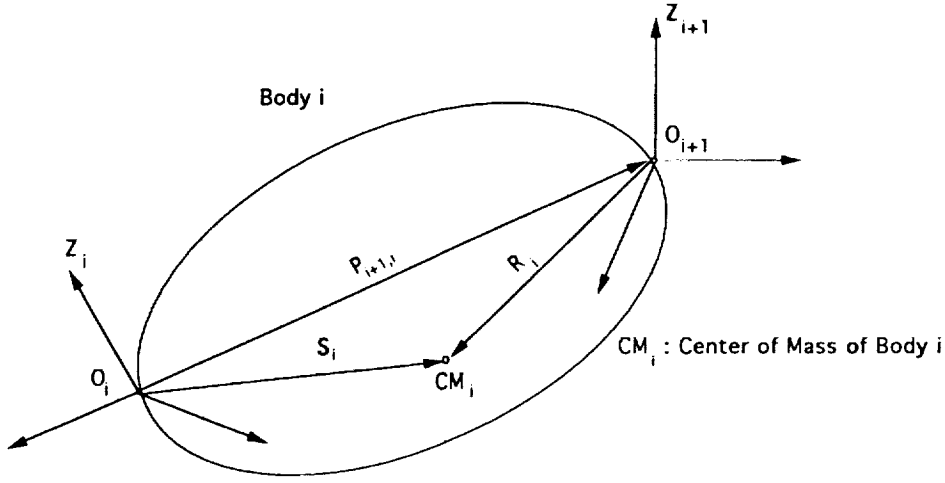

Figure 1. Body, Frames, and Position Vectors

I. Introduction

The multibody dynamics problem concerns the determination of the motion of the mechanical system, resulting from the application of a set of control forces. In the context of robotics, the dynamic simulation problem is better known as the forward dynamics problem.

From a computational point of view, the multibody dynamics problem can be stated as the solution of a linear system as

$$\mathcal{M}\dot{Q} = \mathcal{T} - b(\theta, Q) = \mathcal{F}_T, \text{ or} \tag{1}$$

$$\dot{Q} = \mathcal{M}^{-1}\mathcal{F}_T \tag{2}$$

where the vector $b(\theta, Q)$ represents the contribution of nonlinear terms and can be computed by using the recursive Newton-Euler (N-E) algorithm [3] by setting the joint accelerations to zero. Hence, in Eqs. (1)-(2), $\mathcal{F}_T \triangleq \text{col}\{F_{Ti}\} \epsilon \mathbb{R}^{nx1}$ represents the acceleration-dependent component of the control force.

The developed serial algorithms for the problem can be classified as the $O(n^3)$ algorithm [4], the $O(n^2)$ algorithm [5], and the $O(n)$ algorithms [6-13]. See also [14] for more complete references as well as an extensive analysis and comparison of these algorithms. In addition to these algorithms, which are based on rather direct methods, there is also another class of indirect (or, iterative) algorithms for solution of Eq. (1) which include the $O(n^2)$ conjugate gradient algorithms [4,15,16].

It seems that the development of serial algorithms for the problem has reached a certain level of maturity. Asymptotically, the $O(n)$ algorithms represent the fastest possible serial method for the problem, since, given the n-component input (vector of control force), the evaluation of the n-component output (joint accelerations) requires at least $O(n)$ distinct steps in the computation. Hence, any further improvement in computational efficiency of the $O(n)$ algorithms can only be achieved by reducing the coefficients (see for example [10,14] wherein this reduction has been achieved by avoiding explicit computation of the term $b(\theta, Q)$).

The relationship among the different direct algorithms is also well understood, and two fundamental results have been established [14]. The first is that, at a conceptual level, the $O(n)$ algorithms can be essentially considered as a procedure for recursive factorization and inversion of mass matrix, i.e., recursive computation of $\mathcal{M}^{-1}\mathcal{F}_T$ [9,10,11,14]. The second result is that, at a computational level, the $O(n)$ algorithms lead to the computation of the articulated-body inertia [7]. The reader is referred to [14] for an

extensive analysis of commonalities in the computation of the $O(n)$ algorithms. It should be emphasized that our analysis of the parallel computation efficiency of different algorithms relies on these two results.

Despite the significant improvement in the efficiency of serial algorithms, even the fastest algorithm is still far from providing real-time or faster-than-real-time simulation capability. With the maturity of serial algorithms, any further significant improvement in computational efficiency can be achieved only through exploitation of parallelism. This is further motivated by advances in VLSI technology that have made parallel computation a practical and low-cost alternative for achieving significant computational efficiency. However, unlike serial computation, there are few reports on the development of parallel algorithms for the problem.

The development of efficient parallel algorithms for multibody dynamics is a rather challenging problem. It represents an interesting example for which the analysis of the efficiency of a given algorithm for parallel computation is far different and more complex than that for serial computation. In fact, our previous analysis [1,2,17] and the results of this paper clearly show that those algorithms that are less efficient (in terms of either asymptotic complexity or number of operations) for serial computation provide a higher degree of parallelism and hence are more efficient for parallel computation.

A preliminary investigation of parallelism in the computation of forward dynamics, analyzing the efficiency of existing algorithms for parallel computation, is reported in [2]. The main result of this investigation was that the $O(n^3)$ algorithms provide the highest degree of parallelism and are the most efficient for parallel computation. Specifically, it was shown that

1. Theoretically, the time lower bound of $O(\log^2 n)$ can be achieved by parallelization of the $O(n^3)$ algorithms by using $O(n^3)$ processors.

2. Practically, the best parallel algorithm for the problem is of $O(n)$ which results from parallelization of the $O(n^3)$ algorithms on a two-dimensional array of $O(n^2)$ processors. This parallel algorithm, although of $O(n)$, achieves a significant speedup over the best serial $O(n)$ algorithms by reducing the coefficient of the n-dependent term on polynomial complexity by more than two orders-of-magnitude. Different approaches for parallelization of the $O(n^3)$ algorithms have also been proposed in [18,19].

The analysis in [1] also led to two additional important conclusions. The first was that, if indeed there can be a parallel algorithm achieving the time

lower bound of $O(\log n)$ with an optimal number of $O(n)$ processors, then this parallel algorithm can only be derived by parallelization of an $O(n)$ serial algorithm. However, the analysis in [1] showed that the parallelism in the existing $O(n)$ algorithms was bounded, that is, at best only a constant speedup in the computation can be achieved, leading to the parallel $O(n)$ algorithms. More specifically, it was shown that the recurrence for computation of the articulated-body inertia is strictly serial and cannot be parallelized (see Sec. II.D). Hence, the second conclusion in [1] was that if the forward dynamics problem is to have the time lower bound of $O(\log n)$ for its computation, it can only result from a totally different serial $O(n)$ algorithm. Such an algorithm can only be derived by a global reformulation of the problem and not an algebraic transformation in the computation of existing $O(n)$ algorithms.

Physically, a given algorithm for multibody dynamics can be classified based on its force decomposition strategy. Mathematically, the algorithm can be classified based on the resulting factorization of the mass matrix which corresponds to the specific force decomposition (see Sec. II.B and C). A new algorithm based on a global reformulation of the problem is, then, the one that starts with a different and new force decomposition strategy and results in a new factorization of mass matrix.

Interestingly, a recently developed $O(n)$ algorithm in [21-24] for a single serial chain represents such a global reformulation of the problem. It differs from the existing $O(n)$ algorithms in the sense that it is based on a different strategy for force decomposition (see Sec. III). We will show that this strategy leads to a new and completely different factorization of $\mathcal{M}^{-1}$. This factorization, in turn, results in a new $O(n)$ algorithm for the problem which is strictly efficient for parallel computation, that is, it is less efficient than other $O(n)$ algorithms for serial computation but, as will be shown, it can be parallelized to achieve the time lower bound of $O(\log n)$ with $O(n)$ processors. We show that this factorization of $\mathcal{M}^{-1}$ also directly leads to new factorizations and $O(n)$ algorithms for closed-chain systems. Again, these new $O(n)$ algorithms for closed-chain systems can be parallelized to derive both time- and processor-optimal parallel algorithms for the problem, i.e., $O(\log n)$ parallel algorithms with $O(n)$ processors. Furthermore, the new factorizations for both open- and closed-chain systems can be uniformly described in terms of the *Schur Complement* and provide different and deeper physical insights into the problem.

This paper is organized as follows. In Sec. II, the $O(n)$ algorithms, i.e, the Articulated-Body Inertia algorithm and recursive factorization and inversion of mass matrix, are briefly reviewed. In Sec. III, the Constraint Force algorithm and the new factorization of mass matrix are derived. In Sec. IV, new factorizations and $O(n)$ algorithms for closed-chain systems are presented. In Sec. V, parallel $O(\log n)$ algorithms for both open- and closed-chain systems are briefly presented. Finally, some concluding remarks are made in Sec. VI.

II. The $O(n)$ Algorithms: Recursive Factorization and Inversion of Mass Matrix

A. Notation and Preliminaries

In our discussion of the $O(n)$ algorithms, a set of spatial notations is used which, though slightly different from those in [8-11, 21-24], allows a clear understanding and comparison of the algorithms (see also Table I and Fig. 1). For the sake of clarity, the spatial quantities are shown with upper-case italic letters. Here, only joints with one revolute DOF are considered. However, all the results can be extended to the systems with joints having different and more DOF's.

With any vector V, a tensor $\tilde{V}$ can be associated whose representation in any frame is a skew symmetric matrix as

$$\tilde{V} = \begin{bmatrix} 0 & -V_{(z)} & V_{(y)} \\ V_{(z)} & 0 & -V_{(x)} \\ -V_{(y)} & V_{(x)} & 0 \end{bmatrix}$$

where $V_{(X)}$, $V_{(y)}$, and $V_{(z)}$ are the components of V in the considered frame. The tensor $\tilde{V}$ has the properties that $\tilde{V}^* = -\tilde{V}$ and $\tilde{V}_1 V_2 = V_1 \times V_2$. A matrix $\hat{V}$ associated to the vector V is defined as

$$\hat{V} = \begin{bmatrix} U & \tilde{V} \\ 0 & U \end{bmatrix} \text{ and } \hat{V}^* = \begin{bmatrix} U & 0 \\ -\tilde{V} & U \end{bmatrix} \in \mathbb{R}^{6\times 6}$$

where here (as well as through the rest of the paper) U and 0 stand for unit and zero matrices of appropriate size. The spatial forces acting on two rigidly connected points A and B are related as

$$F_B = \hat{P}_{A,B} F_A$$

where $P_{A,B}$ denotes the position vector from B to A. If the linear and angular velocities of point A are zero then

$$\dot{V}_A = (\hat{P}_{A,B})^* \dot{V}_B$$

The matrix $\hat{P}_{A,B}$ has the properties $\hat{P}_{A,B}\hat{P}_{B,C} = \hat{P}_{A,C}$ and $(\hat{P}_{A,B})^{-1} = \hat{P}_{B,A}$.

In derivation of equations of motion, it is assumed that the nonlinear term $b(\theta, Q)$ is explicitly computed by using the recursive N-E algorithm. For both the articulated-body algorithm (as shown in [14]) and the new algorithm, this explicit computation can be avoided. However, this does not affect the efficiency of the algorithms for parallel computation. In fact, as for the $O(n^3)$ and $O(n^2)$ algorithms [16,17], the explicit computation of $b(\theta, Q)$ provides additional parallelism which can be exploited to further increase the speedup in the computation.

By computing the term $b(\theta, Q)$ and subtracting it from $\mathcal{T}$ (Eq. 1), i.e., by explicitly computing $\mathcal{F}_T$, the multibody system can be assumed to be a system at rest which upon the application of the control force $\mathcal{F}_T$ accelerates in space. The equation of motion for body i, as a single rigid body, is given as

$$F_i = I_i \dot{V}_i$$

and as an interconnected member of the serial chain is given as (Fig. 1)

$$\dot{V}_i = \hat{P}^*_{i-1}\dot{V}_{i-1} + H_i\ddot{Q}_i \tag{3}$$

$$F_i = I_i\dot{V}_i + \hat{P}_i F_{i+1} \tag{4}$$

Eqs. (3)-(4) represent the simplified N-E algorithm (with nonlinear terms excluded) for the serial chain.

Equation (4) represents the interbody force-decomposition strategy of the N-E formulation. As shown in [9-11], this force-decomposition strategy leads to a specific factorization of M. To see this, let us rewrite Eqs. (3)-(4) as

$$\dot{V}_i - \hat{P}^*_{i-1}\dot{V}_{i-1} = H_i\ddot{Q}_i \tag{5}$$

$$F_i - \hat{P}_i F_{i+1} = I_i\dot{V}_i \tag{6}$$

and define

$$\mathcal{H} \triangleq \text{diag}\{H_i\} \in \mathbb{R}^{6n\times n}$$

$$\mathcal{I} \triangleq \text{diag}\{I_i\} \in \mathbb{R}^{6n\times 6n}$$

$$\dot{\mathcal{V}} \triangleq \text{col}\{\dot{V}_i\} \in \mathbb{R}^{6n\times 1}$$

$$\mathcal{F} \triangleq \text{col}\{F_i\} \in \mathbb{R}^{6n\times 1}$$

$$\mathcal{P} = \begin{bmatrix} U & & & & \\ -\hat{P}_{n,n-1} & U & & & \\ 0 & -\hat{P}_{n-1,n-2} & U & & \\ 0 & 0 & & & \\ 0 & 0 & & -\hat{P}_{2,1} & U \end{bmatrix} \varepsilon \mathbb{R}^{6nx6n}$$

$$\mathcal{P}^{-1} = \begin{bmatrix} U & & & & \\ \hat{P}_{n,n-1} & U & & & \\ \hat{P}_{n,n-2} & \hat{P}_{n-1,n-2} & U & & \\ & & & & \\ \hat{P}_{n,1} & \hat{P}_{n-1,1} & & \hat{P}_{n,1} & U \end{bmatrix} \varepsilon \mathbb{R}^{6nx6n}$$

Eqs. (5)-(6) can now be rewritten in a global form as

$$\mathcal{P}^{*}\dot{V} = \mathcal{H}\dot{Q} \tag{7}$$

$$\mathcal{P}\mathcal{F} = \mathcal{J}\dot{V} \tag{8}$$

A factorization of mass matrix, associated with the force decomposition in Eq. (4), can now be derived as

$$\mathcal{F}_T = \mathcal{H}^{*}\mathcal{F} = \mathcal{H}^{*}\mathcal{P}^{-1}\mathcal{J}\dot{V} = \mathcal{H}^{*}\mathcal{P}^{-1}\mathcal{J}(\mathcal{P}^{*})^{-1}\mathcal{H}\dot{Q} \tag{9}$$

which, in comparison with Eq. (1), represents a factorization of $\mathcal{M}$ as

$$\mathcal{M} = \mathcal{H}^{*}\mathcal{P}^{-1}\mathcal{J}(\mathcal{P}^{*})^{-1}\mathcal{H} \tag{10}$$

Although the matrices $\mathcal{P}^{-1}$, $\mathcal{J}$, and $(\mathcal{P}^{*})^{-1}$ are square and have trivial inverses, the matrices $\mathcal{H}^{*}$ and $\mathcal{H}$ are not square. This prevents the computation of $\mathcal{M}^{-1}$ from the above factorization.

B. The Articulated-Body Inertia (A-BI) Algorithm

The Articulated-Body Inertia (A-BI) algorithm is based on a decomposition of F_i as [8]

$$F_i = I_i^A \dot{V}_i + T_i^A \tag{11}$$

where I_i^A is the articulated-body inertia of body i. The force T_i^A is a function of I_j^A and F_{Tj} for j = n to i+1. If I_j^A (and hence T_j^A), for j = n to i, is computed, then the projection of Eq. (6) along the joint axis i leads to a new equation with $\dot{V}_i$ as the only unknown

$$F_{Ti} = H_i^{*} F_i = H_i^{*} I_i^A \dot{V}_i + H_i^{*} T_i^A \tag{12}$$

Starting from i = 1, the joint accelerations can then be recursively computed from Eq. (12). This clearly explains the motivation behind the specific force decomposition in Eq. (11), which, unlike the one in Eq. (4), leads to the solution for joint accelerations.

The computational steps of the A-BI algorithm are given as [8]

For i = n to 1

$$I_i^A = I_i + \hat{P}_i\left(I_{i+1}^A - I_{i+1}^A H_{i+1}(H_{i+1}^* I_{i+1}^A H_{i+1})^{-1} H_{i+1}^* I_{i+1}^A\right)\hat{P}_i^* \qquad I_n^A = I_n \qquad (13)$$

$$T_i^A = \hat{P}_i\left(T_{i+1}^A - I_{i+1}^A H_{i+1}(H_{i+1}^* I_{i+1}^A H_{i+1})^{-1}(F_{Ti+1} - H_{i+1}^* T_{i+1}^A)\right) \qquad T_n^A = 0 \qquad (14)$$

For i = 1 to n

$$\dot{Q}_i = (F_{Ti} - H_i^* I_i^A \hat{P}_{i-1}^* \dot{V}_{i-1} - H_i^* T_i^A)(H_{i+1}^* I_{i+1}^A H_{i+1})^{-1} \qquad \dot{V}_0 = 0 \qquad (15)$$

$$\dot{V}_i = \hat{P}_{i-1}^* \dot{V}_{i-1} + H_i \dot{Q}_i \qquad (16)$$

C. Recursive Factorization and Inversion of Mass Matrix

In [9-11], starting with the factorization in Eq. (10), an alternate factorization of the mass matrix in terms of square factors is derived as

$$M = (U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})\mathcal{D}(U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})^* \qquad (17)$$

$$\mathcal{E}_P \triangleq U - \mathcal{P} \in \mathbb{R}^{6nx6n}$$

$$\mathcal{I}^A \triangleq \text{diag}\{I_i^A\} \in \mathbb{R}^{6nx6n}$$

$$\mathcal{D} \triangleq \text{diag}\{D_i\} = \text{diag}\{H_i^* I_i^A H_i\} \in \mathbb{R}^{nxn} \qquad (18)$$

$$\mathcal{G} \triangleq \text{diag}\{G_i\} = \mathcal{I}^A \mathcal{H} \mathcal{D}^{-1} \in \mathbb{R}^{6nxn} \qquad (19)$$

$$\mathcal{K} \triangleq \mathcal{E}_P \mathcal{G} \in \mathbb{R}^{6nxn} \qquad (20)$$

The nxn matrices $(U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})$, $\mathcal{D}$, and $(U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})^*$ are, respectively, lower triangular, diagonal, and upper triangular. The factorization in Eq. (17) represents the LDL* factorization of the SPD mass matrix (which is unique) in an analytical form. Furthermore, due to the positive definiteness of $\mathcal{M}$, the matrix $\mathcal{D}$ is nonsingular, that is, $D_i \neq 0$ (this is also proved in [8]).

In [9-11] it is shown that the inverse of the factor $(U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})$ can be derived in an analytical form as

$$(U + \mathcal{H}^* \mathcal{P}^{-1} \mathcal{K})^{-1} = (U - \mathcal{H}^* \psi \mathcal{K}) \qquad (21)$$

where $\Psi = \{\psi_{i,j}\} \in \mathbb{R}^{6nx6n}$ is a lower triangular matrix with

$\psi_{i,1} = U$ and $\psi_{i,j} = \hat{P}_{i,j}(U - G_i H_i^*) \in \mathbb{R}^{6\times 6}$, i = n to 1 and j = i-1 to 1 (22)

From Eqs. (17) and (21), a factorization of $\mathcal{M}^{-1}$ is derived as

$$\mathcal{M}^{-1} = (U - \mathcal{H}^* \Psi \mathcal{K})^* \mathcal{D}^{-1} (U - \mathcal{H}^* \Psi \mathcal{K}) \tag{23}$$

The significant contribution of the work in [9-11] is to exploit further structure of the mass matrix (in addition to the symmetry and positive-definiteness) and explicitly obtain the above factorization of $\mathcal{M}^{-1}$. It also demonstrates that the force decomposition in Eq. (11) corresponds to this factorization of $\mathcal{M}^{-1}$. If the articulated-body inertia is computed from Eq. (13) and the terms $\mathcal{K}$ and Ψ are computed according to Eqs. (18)-(20) and (22), then from Eqs. (2) and (23) the solution for $\dot{Q}$ is obtained as

$$\dot{Q} = (U - \mathcal{H}^* \Psi \mathcal{K})^* \mathcal{D}^{-1} (U - \mathcal{H}^* \Psi \mathcal{K}) \mathcal{F}_T \tag{24}$$

In [9-11,14] it is shown that the recursive implementation of Eq. (24) results in an $O(n)$ algorithm whose computational steps (with some minor modifications) correspond to those in Eqs. (13)-(16).

D. Parallelism in the $O(n)$ Algorithms

The main bottleneck in parallel computation of the A-BI algorithm is the computation of I_i^A from Eq. (13), which can be represented, at an abstract level, as the solution of a set of first-order nonlinear recurrences

$$X_i = C_i + \phi_2(X_{i+1})/\phi_1(X_{i+1}) = C_i + \phi(X_{i+1})$$

where C_i is a constant, ϕ_1 and ϕ_2 are polynomials of first and second degree, and deg ϕ = Max (deg ϕ_1, deg ϕ_2) = 2. It is well known that the parallelism in computation of nonlinear recurrences of the above form and with deg ϕ>1 is bounded [25,26], that is, regardless of the number of processors used, their computation can be speeded up only by a constant factor. This is due to the fact that the data dependency in nonlinear recurrences and particularly those containing division is stronger than in linear recurrences [26]. Hence, the parallelism in the $O(n)$ articulated-body based algorithms is bounded and their parallelization leads to parallel $O(n)$ algorithms which are faster than the serial algorithms only by a constant factor. Note that a rather simple model was used to describe the nonlinear recurrences for computation of the articulated-body inertia, while they are far more complex than those usually studied in the literature, e.g., in [25,26].

However, the computations in Eqs. (14)-(16) can be fully parallelized since they can be transformed into a set of first-order linear recurrences (here,

due to the lack of space, we do not discuss these transformations). This clearly indicates that the main obstacle in parallelization of the $O(n)$ articulated-body based algorithms is the computation of the articulated-body inertia. It should also be mentioned that the $O(n)$ algorithm in [7], which was originally developed for serial chains with 3-DOF spherical joints, involves nonlinear recurrences which are even more complex than those for computation of articulated-body inertia.

III. The Constraint Force Algorithm

A. Basic Force Decomposition and Algorithm

The algorithm in [21-24] is based on a decomposition of interbody force as

$$F_i = H_i F_{Ti} + W_i F_{Si} \tag{25}$$

where F_{Si} is the constraint force and W_i is the orthogonal complement of H_i which is defined [27,28] by

$$H_i H_i^* + W_i W_i^* = U \tag{26}$$

The matrix H_i is a projection matrix and hence

$$H_i^* H_i = U \tag{27}$$

It then follows that the matrix W_i is also a projection matrix and that [27]

$$H_i^* W_i = W_i^* H_i = 0 \text{ and } W_i^* W_i = U \tag{28}$$

For a joint i with ni DOF's (ni<6), it follows that $H_i \varepsilon \mathbb{R}^{6xni}$ and $W_i \varepsilon \mathbb{R}^{6x(6-ni)}$. For a more detailed discussion on these projection matrices see [27,28].

The decomposition in Eq. (25) seems to be more natural (and perhaps more physically intuitive) than those in Eqs. (4) and (11) since it expresses the interbody force in terms of two physically more basic components: the control (or, working) force and the constraint (or, nonworking) force. In fact, as stated in [21], the basic idea of the algorithm was first presented in [29] for a system of particles, and later in [30] it was extended to rigid body systems. However, both works were concerned with the *constraint stabilization* problem and the algorithm had not been used as an alternative procedure for the dynamic simulation problem. Also, the independent derivation of the algorithm in [21-24] was mainly motivated by its suitability for parallel iterative solution of the dynamic simulation problem.

It is not surprising that the algorithm has not been considered as a viable alternative for direct serial and parallel solution of the multibody dynamics problem. The decomposition in Eq. (25) naturally leads to the explicit

computation of the constraint (and interbody) forces, which has also motivated the designation of the algorithm as the Constraint Force (CF) algorithm. In fact, researchers have always argued that since the constraint forces are nonworking forces, their computation is not needed and leads to computational inefficiency. Consequently, the elimination of the constraint forces from the equations of motion has always been considered as a necessary first step in the derivation of efficient algorithms.

Here, for the sake of clarity and self-completeness, we first redrive the algorithm as presented in [21-24]. We then show that the force decomposition in Eq. (25) leads to a new factorization of $\mathcal{M}^{-1}$. This allows a better understanding of the algorithm as well as its comparison with other algorithms, particularly the recursive factorization and inversion of mass matrix.

Equation (25) can be written in global form as

$$\mathcal{F} = \mathcal{H}\mathcal{F}_T + \mathcal{W}\mathcal{F}_S \tag{29}$$

with $\mathcal{W} \triangleq \text{diag}\{W_i\} \in \mathbb{R}^{6n \times 5n}$ and $\mathcal{F}_S \triangleq \text{col}\{F_{Si}\} \in \mathbb{R}^{5n \times 1}$. For global matrices $\mathcal{H}$ and $\mathcal{W}$, Eqs. (26)-(28) are written as

$$\mathcal{H}\mathcal{H}^* + \mathcal{W}\mathcal{W}^* = U, \quad \mathcal{H}^*\mathcal{W} = \mathcal{W}^*\mathcal{H} = 0, \text{ and } \mathcal{H}^*\mathcal{H} = \mathcal{W}^*\mathcal{W} = U \tag{30}$$

From Eqs. (7)-(8) and (30), it follows that

$$\dot{\mathcal{V}} = \mathcal{J}^{-1}\mathcal{P}\mathcal{F} \tag{31}$$

$$\mathcal{W}^*\mathcal{P}^*\dot{\mathcal{V}} = \mathcal{W}^*\mathcal{H}\ddot{Q} = 0 \tag{32}$$

$$\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{F} = 0 \tag{33}$$

and substituting Eq. (29) into Eq. (33) yields

$$\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}(\mathcal{H}\mathcal{F}_T + \mathcal{W}\mathcal{F}_S) = 0 \Rightarrow \mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W}\mathcal{F}_S = -\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_T, \text{ or} \tag{34}$$

$$\mathcal{A}\mathcal{F}_S = -\mathcal{B}\mathcal{F}_T \tag{35}$$

where $\mathcal{A} \triangleq \mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W} \in \mathbb{R}^{5n \times 5n}$ and $\mathcal{B} \triangleq \mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} \in \mathbb{R}^{5n \times n}$. The global constraint force, $\mathcal{F}_S$, is computed as the solution of the linear system in Eq. (35), where $\mathcal{A}$ is a symmetric, positive-definite, block tridiagonal matrix. The global interbody force ($\mathcal{F}$) and acceleration ($\dot{\mathcal{V}}$) are then computed from Eqs. (29) and (31). Finally, the joint accelerations are computed from Eqs. (7) and (30) as

$$\mathcal{H}^*\mathcal{H}\ddot{Q} = \mathcal{H}^*\mathcal{P}^*\dot{\mathcal{V}} \Rightarrow \ddot{Q} = \mathcal{H}^*\mathcal{P}^*\dot{\mathcal{V}} \tag{36}$$

The solution of the linear system in Eq. (35) represents the most computationally intensive part of the algorithm. In [21-24], exploiting the structure of matrix $\mathcal{A}$ (i.e., symmetry, positive-definiteness, and block

tridiagonal form), a set of iterative algorithms for solution of Eq. (35) is developed. It is shown that these iterative algorithms can be efficiently parallelized and implemented on a simple architecture with n processors while the rest of the computation is performed serially. Although the computational complexity of the developed parallel iterative algorithms is still of $O(n)$, the extensive simulation in [21-24] has shown that the algorithms achieve speedup over the serial A-BI algorithm.

B. A New Factorization of $\mathcal{M}^{-1}$

Here, we extend the work in [21-24] by first deriving an operator form of the algorithm and then showing that the force decomposition in Eq. (25) indeed leads to a new and interesting factorization of $\mathcal{M}^{-1}$. From Eqs. (29) and (34) the global interbody force can be computed as

$$\mathcal{F} = \left(\mathcal{H} - \mathcal{W}(\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\right)\mathcal{F}_T \tag{37}$$

From Eqs. (8) and (37), $\dot{\mathcal{V}}$ is computed as

$$\dot{\mathcal{V}} = \mathcal{J}^{-1}\mathcal{P}\left(\mathcal{H} - \mathcal{W}(\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\right)\mathcal{F}_T \tag{38}$$

and finally from Eqs. (36) and (38), $\dot{\mathcal{Q}}$ can be computed as

$$\dot{\mathcal{Q}} = \left(\mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} - \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W}(\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\right)\mathcal{F}_T \tag{39}$$

which represents a compact operator form of the algorithm. In comparison with Eq. (2), an operator form of $\mathcal{M}^{-1}$, in terms of its decomposition into a set of simpler operators, is given as

$$\mathcal{M}^{-1} = \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} - \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W}(\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W})^{-1}\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} \tag{40}$$

C. Alternate Approach for Factorization of $\mathcal{M}^{-1}$ based on the Schur Complement

The operator form of $\mathcal{M}^{-1}$ given by Eq. (40) represents an interesting mathematical construct. To see this, let

$$\mathcal{C} \triangleq \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} \in \mathbb{R}^{nxn}$$

$\mathcal{M}^{-1}$ is now written as

$$\mathcal{M}^{-1} = \mathcal{C} - \mathcal{B}^*\mathcal{A}^{-1}\mathcal{B} \tag{41}$$

Consider a matrix $\mathcal{L}$ defined as

$$\mathcal{L} \triangleq \begin{bmatrix} \mathcal{A} & \mathcal{B} \\ \mathcal{B}^* & \mathcal{C} \end{bmatrix} \in \mathbb{R}^{6nx6n} \tag{42}$$

then $\mathcal{C} - \mathcal{B}^*\mathcal{A}^{-1}\mathcal{B}$ is the *Schur Complement* of $\mathcal{A}$ in $\mathcal{L}$ [31] and is designated as $(\mathcal{L}/\mathcal{A})$. The structure of matrix $\mathcal{L}$ motivates a different and simpler approach for derivation of the algorithm. Assume that the spatial acceleration of body i is written in terms of two components: one generated by acceleration of DOF's ($\dot{Q}_i$), and the other generated by acceleration of Degrees-Of-Constraint (DOC's), denoted as $\dot{\sigma}_i \in \mathbb{R}^{5x1}$ (of course, by definition $\dot{\sigma}_1 = 0$). Then rewrite Eqs. (5) and (7) as

$$\dot{V}_i - \hat{P}^*_{i-1}\dot{V}_{i-1} = H_i\dot{Q}_i + W_i\dot{\sigma}_i \tag{43}$$

$$\mathcal{P}^*\dot{V} = \mathcal{H}\dot{Q} + \mathcal{W}\dot{\Sigma} \tag{44}$$

with $\dot{\Sigma} \triangleq \text{col}\{\dot{\sigma}\} \in \mathbb{R}^{5nx1}$. From Eqs. (8), (29), and (44), it then follows that

$$\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W}\mathcal{F}_S + \mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_T = \mathcal{H}\dot{Q} + \mathcal{W}\dot{\Sigma} \tag{45}$$

$\dot{\Sigma}$ and $\dot{Q}$ can be obtained by multiplying both sides of Eq. (45) first by $\mathcal{W}^*$ and then by $\mathcal{H}^*$ as

$$\mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W}\mathcal{F}_S + \mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_T = \dot{\Sigma} \tag{46}$$

$$\mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W}\mathcal{F}_S + \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_T = \dot{Q}, \text{ or} \tag{47}$$

$$\mathcal{A}\mathcal{F}_S + \mathcal{B}\mathcal{F}_T = \dot{\Sigma} \tag{48}$$

$$\mathcal{B}^*\mathcal{F}_S + \mathcal{C}\mathcal{F}_T = \dot{Q} \tag{49}$$

Eq. (39) is then obtained by setting $\dot{\Sigma} = 0$ and using the Guassian elimination for solving for $\mathcal{F}_T$. If the vectors of total acceleration ($\dot{a}_G$) and total force ($\mathcal{F}_G$) are defined as

$$a_G \triangleq \begin{bmatrix} \dot{\Sigma} \\ \dot{Q} \end{bmatrix} \in \mathbb{R}^{6nx1} \text{ and } \mathcal{F}_T \triangleq \begin{bmatrix} \mathcal{F}_S \\ \mathcal{F}_T \end{bmatrix} \in \mathbb{R}^{6nx1}$$

then Eqs. (48)-(49) can be written as

$$\mathcal{L}\mathcal{F}_G = \dot{a}_G \tag{50}$$

The matrix $\mathcal{L}$ can be interpreted as the inverse of the augmented mass matrix; it relates the total force and acceleration. $\mathcal{M}^{-1}$ is then the Schur Complement of $\mathcal{A}$ in $\mathcal{L}$, that is,

$$\mathcal{M}^{-1} = (\mathcal{L}/\mathcal{A}) \tag{51}$$

It should be mentioned that an even simpler physical interpretation of the algorithm along with an alternate direct approach for derivation of Eqs. (48)-(49) can be given by noting the physical interpretation of the operators $\mathcal{H}$, $\mathcal{P}$,

$\mathcal{I}^{-1}$, W, etc. and the matrices $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ [32].

It should also be pointed out that by using the matrix identity

$$(C - XDY)^{-1} = C^{-1} + C^{-1}X(D^{-1} - YC^{-1}X)YC^{-1} \tag{52}$$

in Eqs. (40)-(41) an operator expression of $\mathcal{M}$ can be obtained as

$$\begin{aligned}\mathcal{M} &= \mathcal{C}^{-1} + \mathcal{C}^{-1}\mathcal{B}^{*}(\mathcal{A} - \mathcal{B}\mathcal{C}^{-1}\mathcal{B}^{*})^{-1}\mathcal{B}\mathcal{C}^{-1} \\ &= (\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{I}^{-1}\mathcal{P}\mathcal{H})^{-1} + (\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{I}^{-1}\mathcal{P}\mathcal{H})^{-1}(\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{I}^{-1}\mathcal{P}W)\Big[(W^{*}\mathcal{P}^{*}\mathcal{I}^{-1}\mathcal{P}W) - (W^{*}\mathcal{P}^{*}\mathcal{I}^{-1}\mathcal{P}\mathcal{H}) \\ &\quad (\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{I}^{-1}\mathcal{P}\mathcal{H})^{-1}(\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{I}^{-1}\mathcal{P}W)\Big]^{-1}(W^{*}\mathcal{P}^{*}\mathcal{I}^{-1}\mathcal{P}\mathcal{H})(\mathcal{H}^{*}\mathcal{P}^{*}\mathcal{I}^{-1}\mathcal{P}\mathcal{H})^{-1}\end{aligned} \tag{53}$$

An $O(n^2)$ algorithm for computation of the mass matrix can be derived based on the above expression of $\mathcal{M}$ (which is asymptotically as fast as the best serial algorithms). However, this operator expression of $\mathcal{M}$ is significantly more complex and its associated algorithm is less efficient than other operator expressions and their associated algorithms in Eqs. (10) and (17).

D. Serial Computation of the Constraint Force Algorithm

An efficient serial implementation of the $O(n)$ CF algorithm is based on rewriting Eq. (39) as

$$\dot{Q} = \mathcal{H}^{*}\mathcal{P}^{*}\Big(U - \mathcal{I}^{-1}\mathcal{P}W(W^{*}\mathcal{P}^{*}\mathcal{I}^{-1}\mathcal{P}W)^{-1}W^{*}\mathcal{P}^{*}\Big)\mathcal{I}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_{T} \tag{54}$$

Here, the key to achieving greater efficiency for both serial and parallel computation is to simply perform, as much as possible, the matrix-vector multiplication instead of matrix-matrix multiplication. In this regard, the matrices $\mathcal{B}$, $\mathcal{B}^{*}$, and $\mathcal{C}$ do not need to be computed explicitly and only the explicit computation of $\mathcal{A}$ is needed. Given $\mathcal{F}_{T}$, the computational steps of the algorithm consist of a sequence of matrix-vector multiplications and vector additions where the matrices, except for $\mathcal{A}^{-1}$, are either bidiagonal or diagonal block matrices. Multiplication of a vector by $\mathcal{A}^{-1}$ is equivalent to the solution of a symmetric, positive-definite, block tridiagonal system.

The vector $\mathcal{F}_{T}$ can be computed in $O(n)$ steps by using the N-E algorithm [3]. The matrix-vector multiplications with diagonal or bidiagonal block matrices can be performed in $O(n)$ steps. The solution of the block tridiagonal system can also be obtained in $O(n)$ steps by using block LDL^{*} factorization [33] in $O(n)$ steps. Therefore, the computational complexity of the serial CF algorithm is of $O(n)$.

Note that, however, Eq. (54) is presented in a coordinate-free form. Hence, before its implementation the tensors and vectors involved in its computation

should be projected onto a suitable frame. The choice of optimal frame for projection of equations and other issues regarding efficient serial implementation of the CF algorithm are extensively discussed in [1], wherein the computation cost in terms of number of operations is also evaluated (see Table II). However, as can be seen, even with the most efficient schemes for serial implementation, the CF algorithm is significantly less efficient than the other algorithms for serial computation (see Table II). For large n, the A-BI algorithm is more efficient than the CF algorithm by a factor of about 2.5 (in terms of the total number of operations) for serial computation. Obviously, for smaller n (say n<12), the CF algorithm is also significantly less efficient than the other $O(n^2)$ or $O(n^3)$ algorithms.

It should be mentioned that the explicit computation of $\mathcal{M}^{-1}$ (though it is not usually needed) can be performed in $O(n^2)$ steps. To see this, note that in Eq. (41) the computation of the term $\mathcal{A}^{-1}\mathcal{B}$ is equivalent to the solution of a block tridiagonal system with n right-hand sides which can be computed in $O(n^2)$ steps. $\mathcal{M}^{-1}$ can then be explicitly computed by performing a matrix-matrix multiplication and a matrix-matrix addition, each in $O(n^2)$ steps. This leads to a total computational complexity of $O(n^2)$ for explicit evaluation of $\mathcal{M}^{-1}$.

IV. New Mass Matrix Factorizations for Computation of Closed-Chain Multibody Dynamic Systems

In this section we briefly discuss the application of the new factorization of $\mathcal{M}^{-1}$ to the computation of dynamics of closed-chain systems. Our discussion follows the treatment of the problem as presented in [34-37], wherein it is shown that the main computational problems are the evaluation of the Operational Space Mass Matrix Λ [38,39] and its inverse Λ^{-1}. Note that the computation of Λ is also required for the task space dynamic control of single robot arms [38,39]. The matrices Λ^{-1} and Λ are defined as

$$\Lambda = (\mathcal{J}\mathcal{M}^{-1}\mathcal{J}^*)^{-1}\varepsilon\mathbb{R}^{6x6} \text{ and } \Lambda^{-1} = \mathcal{J}\mathcal{M}^{-1}\mathcal{J}^*\varepsilon\mathbb{R}^{6x6} \tag{55}$$

where $\mathcal{J}\varepsilon\mathbb{R}^{6x6n}$ is the Jacobian matrix. In [34-37] recursive $O(n)$ algorithms are developed for computation of Λ^{-1}, and the matrix Λ is then computed by explicit inversion of Λ^{-1}. The main computational step in these algorithms is the computation of the articulated-body inertia as in Eq. (13). Therefore, as discussed in Sec. II.D, these $O(n)$ algorithms are also strictly serial.

Here, we show that the new factorization of $\mathcal{M}^{-1}$ directly leads to new factorizations of both Λ^{-1} and Λ as well as new $O(n)$ algorithms for their

computation. These new factorizations are similar to that of M^{-1} since they can be described in terms of the Schur Complement and thus provide simple physical interpretation and a different and deeper insight into the problem. More importantly, however, the resulting algorithms can be parallelized to derive both time- and processor-optimal parallel algorithms, i.e., $O(\log n)$ parallel algorithms with $O(n)$ processors, for computation of Λ^{-1} and Λ.

A. A New Factorization of the Inverse of Operational Space Mass Matrix Λ^{-1}

An operator expression of $\mathcal{J}$ is derived in [34,35]. Using the notation of this paper, this operator expression is given as

$$\mathcal{J} = \beta(\mathcal{P}^*)^{-1}\mathcal{H} \tag{56}$$

where $\beta = [\hat{P}_n^* \; 0 \; 0 \; \ldots \; 0] \in \mathbb{R}^{6\times 6n}$. From Eqs. (40) and (56), an operator expression of Λ^{-1} is then derived as

$$\begin{aligned}\Lambda^{-1} &= \beta(\mathcal{P}^*)^{-1}\mathcal{H}\{\mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H} - \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}W(W^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}W)^{-1}W^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\}\mathcal{H}^*(\mathcal{P})^{-1}\beta^* \\ &= \beta\{(\mathcal{P}^*)^{-1}(\mathcal{H}\mathcal{H}^*)\mathcal{P}^*(\mathcal{J}^{-1} - \mathcal{J}^{-1}\mathcal{P}W(W^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}W)^{-1}W^*\mathcal{P}^*\mathcal{J}^{-1})\mathcal{P}(\mathcal{H}\mathcal{H}^*)(\mathcal{P})^{-1}\}\beta^*\end{aligned} \tag{57}$$

The above expression can be simplified by noting that from Eq. (30), we have

$$\mathcal{H}\mathcal{H}^* = U - WW^* \tag{58}$$

By inserting Eq. (58) into Eq. (57) and after some involved algebraic manipulations, a simple operator expression of Λ^{-1} is derived as

$$\Lambda^{-1} = \beta\mathcal{J}^{-1}\beta^* - \beta\mathcal{J}^{-1}\mathcal{P}W(W^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}W)^{-1}W^*\mathcal{P}^*\mathcal{J}^{-1}\beta^* \tag{59}$$

This expression can be further simplified since

$$\mathcal{D} = \beta\mathcal{J}^{-1}\beta^* = \hat{P}_n^* I_n^{-1}\hat{P}_n = I_{n,n+1}^{-1} \tag{60}$$

$$\mathcal{E}^* = \beta\mathcal{J}^{-1}\mathcal{P}W = [\hat{P}_n^* I_n^{-1}W_n \; 0 \; 0 \; \ldots \; 0] \in \mathbb{R}^{6\times 5n} \tag{61}$$

This factorization of Λ^{-1} is then written in the form of the Schur Complement

$$\Lambda^{-1} = \mathcal{D} - \mathcal{E}^*\mathcal{A}^{-1}\mathcal{E} \tag{62}$$

Note that the matrix $\mathcal{A}$ is the same as in Eq. (41). Let us define a matrix $\mathcal{L}'$

$$\mathcal{L}' \triangleq \begin{bmatrix} \mathcal{A} & \mathcal{E} \\ \mathcal{E}^* & \mathcal{D} \end{bmatrix} \in \mathbb{R}^{6n\times 6n} \tag{63}$$

Λ^{-1} is then the Schur Complement of $\mathcal{A}$ in $\mathcal{L}'$, i.e., $\Lambda^{-1} = (\mathcal{L}'/\mathcal{A})$. As in the previous section, based on the Schur Complement factorization of Λ^{-1} and the structure of matrix $\mathcal{L}'$, a set of linear equations can be formed leading to both simple physical interpretation and alternative derivation of this factorization of Λ^{-1} (see [40] for a more detailed discussion).

From Eq. (62), Λ^{-1} can be explicitly computed in $O(n)$ steps as follows. The term $\mathcal{A}^{-1}\mathcal{E}$ can be computed in $O(n)$ steps since it is equivalent to the solution of a block tridiagonal system with six right-hand sides. Λ^{-1} is then computed by performing a matrix-matrix multiplication and a matrix-matrix addition, each with a cost of $O(1)$, leading to a total computation complexity of $O(n)$. However, usually the multiplication of Λ^{-1} by a vector, say F_{n+1}, rather than the explicit computation of Λ^{-1} is needed. In this case, it is significantly more efficient to directly compute $\Lambda^{-1}F_{n+1}$ rather than first explicitly compute Λ^{-1} and then perform the matrix-vector multiplication. Note that the computation of $\Lambda^{-1}F_{n+1}$ is also done in $O(n)$ steps.

B. A New Factorization of Operational Space Mass Matrix Λ

The operator expression of Λ is derived by using the matrix identity in Eq. (52) for inverting the matrix Λ^{-1} in Eq. (62) as

$$\Lambda = (\mathcal{D} - \mathcal{E}^*\mathcal{A}^{-1}\mathcal{E})^{-1} = \mathcal{D}^{-1} - \mathcal{D}^{-1}\mathcal{E}^*(\mathcal{E}\mathcal{D}^{-1}\mathcal{E}^* - \mathcal{A})^{-1}\mathcal{E}\mathcal{D}^{-1} = (\beta\mathcal{J}^{-1}\beta^*)^{-1} - \tag{64}$$

$$(\beta\mathcal{J}^{-1}\beta^*)^{-1}\beta\mathcal{J}^{-1}\mathcal{P}W\{W^*\mathcal{P}^*(\mathcal{J}^{-1}\beta^*(\beta\mathcal{J}^{-1}\beta^*)^{-1}\beta\mathcal{J}^{-1} - \mathcal{J}^{-1})\mathcal{P}W\}^{-1}W^*\mathcal{P}^*\mathcal{J}^{-1}\beta^*(\beta\mathcal{J}^{-1}\beta^*)^{-1}$$

The factorization of Λ can be further simplified by noting that

$$\mathcal{G} = \mathcal{D}^{-1} = (\beta\mathcal{J}^{-1}\beta^*)^{-1} = (I^{-1}_{n,n+1})^{-1} = I_{n,n+1} \tag{65}$$

$$\beta\mathcal{J}^{-1}\mathcal{P}W = [\hat{P}^*_n I^{-1}_n W_n \; 0 \; 0 \; \ldots \; 0] \tag{66}$$

$$\mathcal{R}^* = (\beta\mathcal{J}^{-1}\beta^*)^{-1}\beta\mathcal{J}^{-1}\mathcal{P}W = [(\hat{P}_n)^{-1} I_n (\hat{P}^*_n)^{-1}\hat{P}^*_n I^{-1}_n W_n \; 0 \; 0 \; \ldots \; 0] \tag{67}$$

$$= [\hat{P}_{n,n+1} W_n \; 0 \; 0 \; \ldots \; 0]$$

$$\mathcal{J}'^{-1} = \mathcal{J}^{-1}\beta^*(\beta\mathcal{J}^{-1}\beta^*)^{-1}\beta\mathcal{J}^{-1} - \mathcal{J}^{-1} = \mathrm{Diag}\{I'^{-1}_i\} \varepsilon \mathbb{R}^{6nx6n} \tag{68}$$

with $I'^{-1}_n = 0$ and $I'^{-1}_i = -I^{-1}_i$, i = n-1, n-2, ..., 1

$$\mathcal{S} = W^*\mathcal{P}^*\mathcal{J}'^{-1}\mathcal{P}W \tag{69}$$

Note that the matrix $\mathcal{S}$ is a rank one modification of matrix $\mathcal{A}$ in Eq. (41). The factorization of Λ is then written in terms of the Schur Complement as

$$\Lambda = \mathcal{G} - \mathcal{R}^*\mathcal{S}^{-1}\mathcal{R} \tag{70}$$

Let us define a matrix $\mathcal{L}''$ as

$$\mathcal{L}'' \triangleq \begin{bmatrix} \mathcal{S} & \mathcal{R} \\ \mathcal{R}^* & \mathcal{G} \end{bmatrix} \varepsilon \mathbb{R}^{6nx6n} \tag{71}$$

Λ is then the Schur Complement of $\mathcal{S}$ in $\mathcal{L}''$, i.e., $\Lambda = (\mathcal{L}''/\mathcal{S})$. Again, based on the Schur Complement factorization of Λ^{-1} and the structure of matrix $\mathcal{L}'$, a set of linear equations can be formed leading to both simple physical

interpretation and alternative derivation of this factorization of Λ^{-1} (see [40] for a more detailed discussion).

As for Λ^{-1}, from Eq. (70) Λ can be explicitly computed in $O(n)$ steps since the term $\mathcal{Y}^{-1}\mathcal{R}$ can be computed as the solution of a block tridiagonal system with six right-hand sides. Λ is then computed by performing a matrix-matrix multiplication and a matrix-matrix addition, each with a cost of $O(1)$, leading to a total computation complexity of $O(n)$. However, usually the multiplication of Λ by a vector, say $\dot{V}_{n+1}$, rather than the explicit computation of Λ is needed. In this case, it is significantly more efficient to directly compute $\Lambda\dot{V}_{n+1}$ rather than first explicitly compute Λ^{-1} and then perform the matrix-vector multiplication. Again, note that the computation of $\Lambda\dot{V}_{n+1}$ is done in $O(n)$ steps.

V. Parallel $O(\log n)$ Algorithms for Computation of Open- and Closed-Chain Rigid Multibody Systems

The parallel implementation of the CF algorithm for a serial open-chain system is extensively discussed in [1]. Here, we briefly present the results of [1] and their extension to the computation of closed-chain systems.

A. Parallel $O(\log n)$ Algorithms for Open-Chain Rigid Multibody Systems

The computation of the parallel CF algorithm is performed as follows.

Step I. Projection and Computation of Matrix $\mathcal{A}$

The projection of vectors and tensors and the explicit computation of matrix $\mathcal{A}$ is performed in $O(1)$ steps with n processors.

Step II. Computation of $\mathcal{F}_T$

By using the algorithm in [20], $\mathcal{F}_T$ is computed in $O(\log n)+O(1)$ steps with n processors.

Step III. Computation of $\dot{\Sigma}_T = \mathcal{B}^*\mathcal{F}_T = \mathcal{W}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_T$ and $\dot{Q}_T = \mathcal{C}\mathcal{F}_T = \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{H}\mathcal{F}_T$

The computation of $\dot{\Sigma}_T$ and $\dot{Q}_S$ involves two sequences of matrix-vector multiplication wherein the matrices are bidiagonal or diagonal block matrices and is performed in $O(1)$ steps with n processors.

Step IV. Computation of $\mathcal{F}_S = \mathcal{A}^{-1}\dot{\Sigma}_T$

The SPD block tridiagonal system is solved in $O(\log n)+O(1)$ steps with n processors and by using the Odd-Even Elimination (OEE) variant of the cyclic reduction algorithm [41].

Step V. Computation of $\dot{Q}_S = \mathcal{B}\mathcal{F}_S = \mathcal{H}^*\mathcal{P}^*\mathcal{J}^{-1}\mathcal{P}\mathcal{W}\mathcal{F}_S$ and $\dot{Q} = \dot{Q}_S + \dot{Q}_T$

The computation of $\dot{Q}_S$ is similar to that of $\dot{Q}_T$ in Step III and is performed in $O(1)$ steps with n processors.

As can be seen, the overall computational complexity of the parallel CF algorithm is of $O(\log n)+O(1)$ by using n processors. In [1] it is shown that the algorithm can be efficiently implemented on an SIMD parallel architecture with n processors and with a Shuffle-Exchange augmented with Nearest-Neighbor (SENN) interconnection. The SENN interconnection allows a perfect mapping, i.e., with no topological variation, of the parallel algorithm since it perfectly matches the inherent communication structure of different steps of the algorithm and thus leads to minimum communication cost. In [1] the computation and communication cost of the parallel CF are evaluated as $(732m+653a)\lceil\log_2 n\rceil+(542m+439a)$ and $(134\lceil\log_2 n\rceil+49)c$, where m, a, and c stand for the cost of multiplication, addition, and communicating a single datum ($\lceil x\rceil$ is the smallest integer greater than or equal to x).

Note that the parallel algorithm, while achieving the time lower bound in the computation, remains highly compute-bound. The ratio of the computation cost over communication cost is greater than 10, which indicates that the parallel algorithm has a rather large grain size and thus can be efficiently implemented on commercially available MIMD parallel architectures. In fact, the parallel CF algorithm is currently being implemented on an MIMD Hypercube parallel architecture. Furthermore, the parallel CF algorithm allows the exploitation of parallelism at several computational levels. In [1] it is shown that a greater speedup in the computation can be achieved by exploiting a multilevel parallelism and implementing the algorithm on an architecture with 3n processors.

B. Parallel $O(\log n)$ Algorithms for Closed-Chain Rigid Multibody Systems

1. Computation of Λ^{-1} and $\Lambda^{-1}F_{n+1}$

The explicit evaluation of matrix $\mathcal{A}$, similar to Step I of Sec. V.A, can be performed in $O(1)$ steps with n processors. The term $\mathcal{A}^{-1}\mathcal{E}$ in Eq. (62) represents the solution of an SPD block tridiagonal system with 6 right-hand sides and can be computed in $O(\log n)+O(1)$ steps with n processors by using the OEE variant of the cyclic reduction algorithm. Λ^{-1} is then computed by performing a matrix-matrix multiplication and a matrix-matrix addition wherein each operation can be performed in $O(1)$ steps with $O(1)$ processors. This leads to a computational complexity of $O(\log n)+O(1)$ with n processors, which

indicates a both time- and processor-optimal parallel algorithm for evaluating Λ^{-1}. Similar results with greater computation efficiency can be obtained when evaluating $\Lambda^{-1}F_{n+1}$ since the solution of an SPD block tridiagonal system with a single right-hand side is needed.

2. Computation of Λ^{-1} and $\Lambda^{-1}F_{n+1}$

The explicit evaluation of matrix $\mathcal{P}$, similar to that of $\mathcal{A}$, can be performed in $O(1)$ steps with n processors. The rest of the computation in Eq. (70) is similar to that in Eq. (62). Therefore, both Λ^{-1} and $\Lambda^{-1}F_{n+1}$ can be computed in $O(\log n)+O(1)$ steps with n processors, which indicates both time- and processor-optimal parallel algorithms for the computations. Note, however, that, unlike the matrix $\mathcal{A}$, which is always SPD, for some configurations, the matrix $\mathcal{P}$ can become singular [34-37], and thus special care should be taken in computation of Eq. (70). However, it should be mentioned that the new and simple factorization of both Λ^{-1} and Λ provides much better insight for the analysis of the singularity in the computation of Λ [40].

VI. Discussion and Conclusion

In this paper, we presented parallel $O(\log n)$ algorithms for computation of open- and closed-chain rigid multibody dynamics. These parallel algorithms were derived from new $O(n)$ algorithms for the problem. These $O(n)$ algorithms are based on a new force-decomposition strategy which results in new factorizations of $\mathcal{M}^{-1}$, Λ^{-1}, and Λ, presented in Eqs. (40), (62), and (70).

Some important conceptual features of these new algorithms and their underlying factorizations can be summarized as follows.

1. The factorizations of $\mathcal{M}^{-1}$, Λ^{-1}, and Λ are very similar and can be described in terms of the Schur Complement. Due to this similarity, both serial and parallel algorithms involve similar computational steps.

2. Compared to the previous factorization of mass matrix, which is based on a multiplicative decomposition of $\mathcal{M}^{-1}$ (see Eq. (23)), the new factorization leads to an additive decomposition of $\mathcal{M}^{-1}$ which involves simpler matrices, i.e., block tridiagonal matrices.

3. Unlike the previous approaches, wherein Λ^{-1} is first recursively computed and then Λ is obtained by explicit inversion of Λ^{-1}, independent factorizations for both Λ^{-1} and Λ are derived, which allows direct computation of either of them.

From a computational point of view, the main feature of the new algorithms is that they are *strictly* parallel algorithms. That is, as was shown through

Algorithm		Computation Cost	Number of Processors
Serial	A-BI	586n - 371	-
	CF	1500n - 755	-
Parallel	SL CF	$1385\lceil\log_2 n\rceil + 981$	n
	ML CF	$780\lceil\log_2 n\rceil + 595$	3n
	$O(n^3)$	$6n+69\lceil\log_2 n\rceil + 340$	n(n+1)/2

A-BI: Articulated-Body Inertia Algorithm. CF: Constraint Force Algorithm
SL CF: Single Level Parallel CF. ML CF: Multilevel Parallel CF
$O(n^3)$: Parallel $O(n^3)$ algorithm in [2].

Table II. Computation Costs of Serial and Parallel Algorithms

an extensive analysis in [1] for a single serial chain, the algorithm is less efficient than other $O(n)$ algorithms for serial computation (see Table II). However, based on the analysis in Sec. II.D, they are the only known algorithms that can be parallelized and lead to both time- and processor-optimal parallel algorithms for the problem.

The computation costs of different serial and parallel algorithms for the problem are presented in Table II, wherein it is assumed that $m = a$. As can be seen, for small n, the parallel algorithm resulting from parallelization of the $O(n^3)$ algorithm with $O(n^2)$ processors is the most efficient. However, as n increases, so does the efficiency of the parallel $O(\log n)$ algorithms.

As the last point, it should be emphasized that the parallel algorithms developed in this paper--in addition to being theoretically significant by proving, for the first time, the existence of both time- and processor-optimal parallel algorithms for the problem--are also highly practical from an implementation point of view. This is due to their large grain size and simple communication and processor interconnection requirements. In fact, these algorithms are currently being implemented on a Hypercube parallel architecture.

Acknowledgments
The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration (NASA). The author gratefully acknowledges many insightful discussions with Dr. G. Rodriguez of JPL regarding different aspects of this research work.

REFERENCES

1. A. Fijany, "Parallel *O*(Log N) Algorithms for Rigid Multibody Dynamics," JPL Eng. Memorandum (Internal Document) EM 343-92-1258, August 1992.
2. A. Fijany and A.K. Bejczy, "Techniques for Parallel Computation of Mechanical Manipulator Dynamics. Part II: Forward Dynamics," in *Advances in Control and Dynamic Systems, Vol. 40: Advances in Robotic Systems Dynamics and Control,* C.T. Leondes (Ed.), pp. 357-410, Academic Press, March 1991.
3. J.Y.S. Luh, M.W. Walker, and R.P.C. Paul, "On-Line Computational Scheme for Mechanical Manipulator," ASME J. Dynamic Syst., Meas., Control, Vol. 102, pp. 69-76, June 1980.
4. M.W. Walker and D.E. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanism," ASME J. Dynamic Systems, Measurement, and Control, Vol. 104, pp. 205-211, 1982.
5. D.E. Rosental, "Triangularization of Equations of Motion for Robotic Systems," J. Guidance, Control, and Dynamics, Vol. 11, pp. 278-281, 1988.
6. A.F. Vereshchagin, "Computer Simulation of the Dynamics of Complicated Mechanism of Robot Manipulators," Engineering Cybernetics, Vol. 6, pp. 65-70, 1974.
7. W.W. Armstrong, "Recursive Solution to the Equation of Motion of an N-Link Manipulator," Proc. 5th World Congress on Theory of Machines and Mechanisms, pp. 1343-1346, 1979.
8. R. Featherstone, "The Calculation of Robot Dynamics Using Articulated-Body Inertia," Int. J. Robotics Research, Vol. 2(1), pp. 13-30, 1983.
9. G. Rodriguez, "Kalman Filtering, Smoothing and Recursive Robot Arm Forward and Inverse Dynamics," IEEE J. Robotics and Automation, Vol. RA-3(6), pp. 624-639, Dec. 1987.
10. G. Rodriguez and K. Kreutz-Delgado, "Spatial Operator Factorization and Inversion of the Manipulator Mass Matrix," IEEE Trans. Robotics and Automation, Vol. RA-8(1), pp. 65-76, Feb. 1992.
11. G. Rodriguez, K. Kreutz, and A. Jain, "A Spatial Operator Algebra for Manipulator Modeling and Control," Int. J. Robotics Research, vol. 10(4), pp. 371-381, Aug. 1991.
12. D. Rosental, "Order N Formulation for Equations of Motion of Multibody Systems," Proc. SDIO/NASA Workshop on Multibody Simulation, pp. 1122-1150, Sept. 1987.
13. D.S. Bae and E.J. Haug, "A Recursive Formulation for Constraint Mechanical System Dynamics: Part I. Open Loop Systems," Mech. Struct. & Mach., Vol. 15(3), pp. 359-382, 1987.
14. A. Jain, "Unified Formulation of Dynamics for Serial Rigid Multibody Systems," J. Guidance, Control, and Dynamics, Vol. 14(3), pp. 531-542, May/June 1991.
15. A. Fijany and R.E. Scheid, "Efficient Conjugate Gradient Algorithms for Computation of the Manipulator Forward Dynamics," Proc. NASA Conf. Space Telerobotics, Vol. IV, pp. 329-340, Jan. 1989.
16. A. Fijany and R.E. Scheid, "Fast Parallel Preconditioned Conjugate Gradient Algorithms for Robot Manipulator Dynamics Simulation," To appear in J. of Intelligent & Robotic Systems: Theory & Applications, 1992. Also, in JPL Eng. Memorandum (Internal Document) EM 343-1196, Aug. 1991.
17. A. Fijany and A.K. Bejczy, "Parallel Algorithms and Architecture for Computation of Robot Manipulator Forward Dynamics," Proc. IEEE Int. Conf. Robotics and Automation, pp. 1156-1162, April 1991, Sacramento, CA.
18. H. Kasahara, H. Fujii, and M. Iwata, "Parallel Processing of Robot Motion Simulation," Proc. 10th IFAC World Congress, July 1987.
19. C.S.G. Lee and P.R. Chang, "Efficient Parallel Algorithms for Robot Forward Dynamics Computation," IEEE Trans. Syst., Man, and Cybern., Vol. 18(2), pp. 238-251, March/April 1988.

20. C.S.G. Lee and P.R. Chang,"Efficient Parallel Algorithms for Robot Inverse Dynamics Computation," IEEE Trans. Syst., Man, and Cybern., Vol. 16(4), pp. 532-542, July/August 1986.
21. I. Sharf, *Parallel Simulation Dynamics for Open Multibody Chains*, Ph.D. Diss., Univ. of Toronto, Canada, Nov. 1990.
22. I. Sharf and G.M.T. D'Eleuterio,"Parallel Simulation Dynamics for Rigid Multibody Chains," Proc. 12th Biennial ASME Conf. on Mechanical Vibration and Noise, Montreal, Canada, Sept. 1989.
23. I. Sharf and G.M.T. D'Eleuterio,"Computer Simulation of Elastic Chains Using a Recursive Formulation," Proc. IEEE Int. Conf. Robotics and Automation, pp. 1539-1545, Philadelphia, PA, 1988.
24. I. Sharf and G.M.T. D'Eleuterio,"Parallel Simulation Dynamics for Elastic Multibody Chains," Proc. IEEE Int. Conf. Robotics and Automation, pp. 740-747, Cincinnati, OH, 1990.
25. J. Miklosko and V.E. Kotov (Eds.), *Algorithms, Software, and Hardware of Parallel Computers*. Springer-Verlag, 1984.
26. L. Hyafil and H.T. Kung,"The Complexity of Parallel Evaluation of Linear Recurrences," J. ACM, Vol. 24(3), pp. 513-521, July 1977.
27. P.C. Hughes and G.B. Sincarsin,"Dynamics of an Elastic Multibody Chain: Part B - Global Dynamics," Dynamics and Stability of Systems, Vol. 4(3&4), pp. 227-244, 1989.
28. C.J. Damaren and G.M.T. D'Eleuterio,"On the Relationship between Discrete-Time Optimal Control and Recursive Dynamics for Elastic Multibody Chains," Contemporary Mathematics, Vol. 97, pp. 61-77, 1989.
29. J. Baumgarte,"Stabilization of Constraints and Integrals of Motion in Dynamical Systems," Computer Methods in Applied Mechanics and Engineering, Vol. (1), pp. 1-16, 1972.
30. R.E. Roberson,"Constraint Stabilization for Rigid Bodies: An Extension of Baumgarte's Method," Proc. IUTAM Symp. Dynamics of Multibody Systems, pp. 274-289, Munich, 1978.
31. R.W. Cottle,"Manifestation of Schur Complement," Linear Algebra and its Application, Vol. 8, pp. 189-211, 1974.
32. A. Fijany, I. Sharf, and G.M.T. D'Eleuterio,"Parallel O(Log N) Algorithms for Computation of Manipulator Forward Dynamics," Submitted to IEEE Trans. Robot. Automat.
33. G.H. Golub and C.F. Van Loan, *Matrix Computations*, 2nd Edition, The Johns Hopkins Univ. Press, 1989.
34. G.Rodriguez,"Recursive Forward Dynamics for Multiple Robot Arms Moving a Common Task Object," IEEE Trans. Robot. Automat., Vol. 5(4), Aug. 1989.
35. G.Rodriguez and K. Kreutz,"Recursive Mass Matrix Factorization and Inversion: An Operator Approach to Open- and Closed-Chain Multibody Dynamics," Jet Propulsion Lab. Publication 88-11, March 1988.
36. K.W. Lilly and D.E. Orin,"Efficient O(n) Computation of the Operational Space Inertia Matrix," Proc. IEEE Int. Conf. Robotics & Automation, pp. 1014-1019, Cincinnati, OH, May 1990.
37. S. McMillan, P. Sadayappan, D.E. Orin,"Efficient Dynamic Simulation of Multiple-Manipulator Systems with Singular Configurations," Proc. IEEE Int. Conf. Robotics & Automation, May 1992.
38. O. Khatib,"The Operational Space Formulation in the Analysis, Design, and Control of Manipulators," 3rd Int. Symp. Robotics Research, 1985.
39. O. Khatib,"A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," IEEE J. Robot. Automat., Vol. RA-3, pp. 43-53, Feb. 1987.
40. A. Fijany,"New Factorization Techniques and O(n) Algorithms for Computation of Operational Space Mass Matrix and its Inverse," In preparation.
41. R.W. Hockney and C.R. Jesshope, *Parallel Computers*, Adam Hilger Ltd., 1981.