Multibody System Dynamics **9:** 63–85, 2003.

Simulation of Industrial Manipulators Based on the UDU^T Decomposition of Inertia Matrix ⋆

SUBIR KUMAR SAHA
Department of Mechanical Engineering, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110 016, India; E-mail: saha@mech.iitd.ernet.in

(Received: 28 February 2001; accepted in revised form: 14 September 2001)

Abstract. The **UDU**^{T} – **U** and **D** are respectively the upper triangular and diagonal matrices – decomposition of the generalized inertia matrix of an n-link serial manipulator, introduced elsewhere, is used here for the simulation of industrial manipulators which are mainly of serial type. The decomposition is based on the application of the Gaussian elimination rules to the recursive expressions of the elements of the inertia matrix that are obtained using the Decoupled Natural Orthogonal Complement matrices. The decomposition resulted in an efficient order n, i.e., $\mathcal{O}(n)$, recursive forward dynamics algorithm that calculates the joint accelerations. These accelerations are then integrated numerically to perform simulation. Using this methodology, a computer algorithm for the simulation of any n degrees of freedom (DOF) industrial manipulator comprising of revolute and/or prismatic joints is developed. As illustrations, simulation results of three manipulators, namely, a three-DOF planar manipulator, and the six-DOF Stanford arm and PUMA robot, are reported in this paper.

Key words: manipulators, forward dynamics, recursive algorithm, simulation.

1. Intrduction

Simulation of industrial manipulators is defined here as, *given the joint forces/ torques, find the joint positions*. It is performed in two different steps, namely, (i) solve for the joint accelerations from the dynamic equations of motion, i.e., forward dynamics, and (ii) integrate the joint accelerations. Thus, one requires the dynamic equations of motion, i.e., the dynamic model, of the system at hand. The conventional approach to obtain the dynamic model of a mechanical system, consisting of rigid bodies coupled by kinematic pairs or joints, is to use either Newton–Euler (NE) or Euler–Lagrange (EL) equations. While the NE equations are obtained from the free-body diagrams, the EL equations result from the kinetic and potential energy of the system. The former is not suitable for motion simulation, as it finds the internal forces and torques that do not affect the motion of the system. Alternatively, EL equations give independent set of equations that are good for motion simulation, however, require complex calculations for the partial derivatives. With the advent of digital computation, a series of new methods in the

⋆ This paper is based on the presentation made in the EUROMECH 404 Colloquium held in Lisbon, Portugal, September 20–23, 1999.

study of dynamic modeling of mechanical systems were reported in the literature [1–4]. If these formalisms are used in simulation they normally result in order n^3, i.e., $\mathcal{O}(n^3)$, forward dynamics algorithms [5–7]. These methods use the following strategy to solve for the joint accelerations:

(i) first, the elements of the inertia matrix are evaluated;
(ii) then, the decomposition of the inertia matrix, namely, the Cholesky decomposition [8] is performed numerically; and
(iii) finally, the joint accelerations are solved by backward and forward substitutions [8], respectively.

Since the complexity of the Cholesky decomposition is of order n^3, $\mathcal{O}(n^3)$, the resulting forward dynamics algorithm also requires $\mathcal{O}(n^3)$ computations. There are, however, alternative approaches which find the forward dynamics algorithm recursively and whose computational complexity is in the order n, $\mathcal{O}(n)$ [9–15]. There are also parallel order n algorithms [16, 17] that require special computer hardware, namely, a parallel architecture. Because of the special computer hardware requirement parallel algorithms are not discussed further.

The algorithm by Armstrong [9] was the first to recursively solve the equations of motion of an n-link manipulator mounted on a spacecraft. Later, Featherstone [10] introduced the concept of *articulated body inertia* (ABI), which is obtained by writing the equations of motion written for hinge points. The approach by Stejskal and Valasek [14] is also similar to ABI approach, however, the book provides a good comparison of different formalisms. Bae and Haug [11] used the variational principle to obtain an $\mathcal{O}(n)$ algorithm. Rodriguez [12], on the other hand, presented a novel robot inverse and forward dynamics algorithms based on Kalman filtering and smoothing technique arising in the state estimation theory. The methodology has been used to solve several other dynamics problems as well [18]. Schiehlen [13] and Saha [15], however, have independently proposed a Linear Algebra approach to arrive at recursive forward dynamics algorithms. It is interesting to note that, compared to the $\mathcal{O}(n^3)$ schemes [5], advantage of an $\mathcal{O}(n)$ scheme in terms of the computational complexity starts when n is large, e.g., $n \geq 10$ for the algorithm presented in this paper, as evident in Table I. The $\mathcal{O}(n)$ algorithms, however, calculate the joint accelerations that are smooth functions of time [19]. As a result, numerical integration is faster and, hence, the total computer time for simulation may be less while comparing with the $\mathcal{O}(n^3)$ schemes. Besides, $\mathcal{O}(n)$ algorithms provide physical interpretations of the terms they calculate and the intermediate steps they follow. Thus, the recursive robot forward dynamics is becoming more and more popular.

In this paper, an order n algorithm [15] is implemented efficiently for the dynamic simulation of industrial manipulators which are mainly serial types. The characteristics of the approach are:

- extension of the concept of the *Natural Orthogonal Complement* [3] to define the Decoupled Natural Orthogonal Complement [15] matrices in the derivation of the dynamic equations of motion;
- analytical decomposition of the generalized inertia matrix;
- uniform development of the recursive inverse and forward dynamics algorithms for the serial multi-body systems [20];
- explicit analytical inversion of the inertia matrix [21], which may facilitate the deeper understanding of the dynamics involved in a multi-body system;
- it is extendable to the parallel platform type closed-loop systems, as shown in [22].

However, the contributions of this paper are as follows:

1. efficient implementation of the forward dynamics algorithm [15], as presented in Section 3;
2. computational complexity analysis of the above-mentioned implementation, also done in Section 3;
3. simulation of several industrial robots, namely, a three-DOF planar arm, the six-DOF Stanford arm, and a six-DOF PUMA robot; and
4. comparison and verification of the simulation results with those available in the literature [7, 11].

The paper is organized as follows. Section 2 presents the simulation scheme [15], whereas Section 3 provides the computer algorithm for the purpose of simulation. The simulation results are then given in Section 4. Finally, conclusions are made in Section 5.

2. Simulation

Simulation is defined as the computation of the joint accelerations from the dynamic equations of motion, i.e., forward dynamics, followed by numerical integration. It is assumed that the dynamic model, i.e., the equations of motion, of an n-DOF serial manipulator, as shown in Figure 1, be represented as

$$\mathbf{I}\ddot{\boldsymbol{\theta}} = \boldsymbol{\phi} \quad \text{where} \quad \boldsymbol{\phi} \equiv \boldsymbol{\tau} - \mathbf{C}\dot{\boldsymbol{\theta}}, \tag{1}$$

in which $\mathbf{I}$ and $\mathbf{C}$ are the $n \times n$ generalized inertia matrix (GIM), and the matrix of convective inertia terms, respectively. Moreover, the n-dimensional vectors, $\dot{\boldsymbol{\theta}}$ and $\ddot{\boldsymbol{\theta}}$, are the joint velocity and accelerations, respectively, i.e., the first and second time derivatives of the joint position vector, $\boldsymbol{\theta}(\equiv [\theta_1, \ldots, \theta_n]^T)$. Moreover, $\boldsymbol{\tau}(\equiv [\tau_1, \ldots, \tau_n]^T)$, is the n-dimensional vector of generalized forces due to external driving moments and forces and those resulting from gravity and dissipation, etc. In forward dynamics, the main interest is how to solve for vector $\ddot{\boldsymbol{\theta}}$. Thus, it is assumed that the vector $\boldsymbol{\phi}$ is known, which can be computed efficiently from an

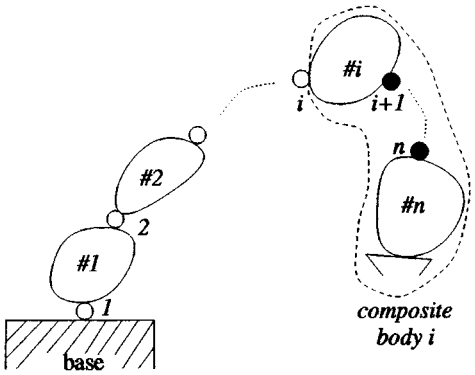

Figure 1. An n-body serial manipulator.

$\mathcal{O}(n)$ inverse dynamics algorithm while $\ddot{\boldsymbol{\theta}} = \mathbf{0}$ [15]. The GIM, $\mathbf{I}$ of Equation (1), is now expressed using the the Decoupled Natural Orthogonal Complement Matrices (DeNOC) matrices [15] as

$$\mathbf{I} \equiv \mathbf{N}_d^T \tilde{\mathbf{M}} \mathbf{N}_d \quad \text{and} \quad \tilde{\mathbf{M}} \equiv \mathbf{N}_l^T \mathbf{M} \mathbf{N}_l, \tag{2}$$

where the $6n \times 6n$ generalized mass matrix, $\mathbf{M}$, and the $6n \times 6n$ and $6n \times n$ DeNOC matrices, $\mathbf{N}_l$ and $\mathbf{N}_d$, respectively, are defined as

$$\mathbf{M} \equiv \begin{bmatrix} \mathbf{M}_1 & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{O} & \mathbf{M}_2 & \cdots & \mathbf{O} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{O} & \mathbf{O} & \cdots & \mathbf{M}_n \end{bmatrix}, \quad \mathbf{N}_l \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} & \cdots & \mathbf{O} \\ \mathbf{B}_{21} & \mathbf{1} & \cdots & \mathbf{O} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{n1} & \mathbf{B}_{n2} & \cdots & \mathbf{1} \end{bmatrix},$$

$$\mathbf{N}_d \equiv \begin{bmatrix} \mathbf{p}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{p}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{p}_n \end{bmatrix}, \tag{3}$$

in which the 6×6 matrix, $\mathbf{B}_{ij}$, and the six-dimensional vector, $\mathbf{p}_i$, are associated with the relation between the *twist* of the ith link, $\mathbf{t}_i (\equiv [\boldsymbol{\omega}_i^T, \ \mathbf{v}_i^T]^T) - \boldsymbol{\omega}_i$ and $\mathbf{v}_i$ being the three-dimensional vectors of angular velocity and linear velocity of the mass center of the ith link, C_i, as shown in Figure 2, respectively – and that of the jth link, $\mathbf{t}_j$, i.e.,

$$\mathbf{t}_i = \mathbf{B}_{ij} \mathbf{t}_j + \mathbf{p}_i \dot{\theta}_i, \tag{4}$$

where the joint rate, $\dot{\theta}_i$, is the time derivative of ith joint angle, θ_i, as shown in Figure 2, for a revolute joint. Matrices, $\mathbf{M}_i$, $\mathbf{B}_{ij}$, and vector $\mathbf{p}_i$ are given by

$$\mathbf{M}_i \equiv \begin{bmatrix} \mathbf{I}_i & \mathbf{O} \\ \mathbf{O} & m_i \mathbf{1} \end{bmatrix}, \quad \mathbf{B}_{ij} \equiv \begin{bmatrix} \mathbf{1} & \mathbf{O} \\ \mathbf{C}_{ij} & \mathbf{1} \end{bmatrix} \quad \text{and} \quad \mathbf{p}_i \equiv \begin{bmatrix} \mathbf{e}_i \\ \mathbf{e}_i \times \mathbf{d}_i \end{bmatrix}, \tag{5}$$

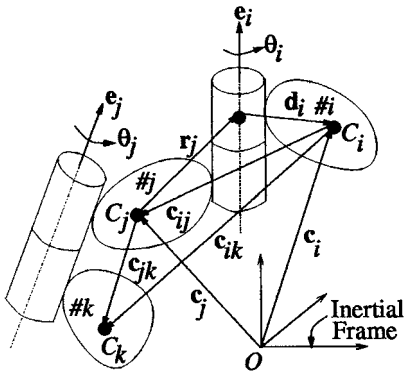

Figure 2. A coupled system.

in which $\mathbf{I}_i$ and m_i are the 3×3 inertia tensor about C_i and the mass of the ith link, respectively. Moreover, $\mathbf{1}$ and $\mathbf{O}$ being the 3×3 identity and zero matrices, respectively, which, henceforth, should be understood as of dimensions compatible to the size of the matrix in which they appear. Furthermore, $\mathbf{C}_{ij}$ is the 3×3 cross-product tensor, associated to the vector, $\mathbf{c}_{ij} (\equiv \mathbf{c}_j - \mathbf{c}_i)$, as indicated in Figure 2, which when operates on any three-dimensional Cartesian vector, $\mathbf{x}$, results in a cross-product vector, i.e., $\mathbf{C}_{ij}\mathbf{x} \equiv \mathbf{c}_{ij} \times \mathbf{x}$. Substituting the expressions of $\mathbf{M}$, $\mathbf{N}_l$, and $\mathbf{N}_d$, Equation (3), into Equation (2), each element of the GIM, can be expressed as

$$i_{ij} = \mathbf{p}_i^T \tilde{\mathbf{M}}_i \mathbf{B}_{ij} \mathbf{p}_j \quad \text{where} \quad \tilde{\mathbf{M}}_i = \mathbf{M}_i + \mathbf{B}_{i+1,i}^T \tilde{\mathbf{M}}_{i+1} \mathbf{B}_{i+1,i} \tag{6}$$

for $i = 1, \ldots, n$; $j = 1, \ldots, i$. Note that the 6×6 symmetric matrix, $\tilde{\mathbf{M}}_i$, for $i = 1, \ldots, n$, is computed recursively in which $\tilde{\mathbf{M}}_{n+1} = \mathbf{O}$, because there is no $(n+1)$st body in the system. Matrix $\tilde{\mathbf{M}}_i$ is interpretted as the mass matrix of the *composite body*, i, i.e., $\tilde{\mathbf{M}}_i$ represents the mass and inertia properties of the system comprising of $(n - i + 1)$ rigidly connected bodies, namely, bodies $\#i, \ldots, \#n$, as indicated in Figure 1 by the dotted line.

2.1. UDU^T DECOMPOSITION

The decomposition is based on the *reverse* Gaussian Elimination (RGE) of the GIM, $\mathbf{I}$ of Equations (1) and (6), as outlined in Appendix B. It is explained in the following steps:

1. The RGE of the GIM, $\mathbf{I}$ of Equation (1), as given in Appendix B, can be written as

$$\mathbf{EI} = \mathbf{L}_2 \quad \text{where} \quad \mathbf{E} \equiv \mathbf{E}_2 \ldots \mathbf{E}_n, \tag{7}$$

in which the $n \times n$ matrix, $\mathbf{E}_k$, for $k = n, \ldots, 2$, is the *elementary upper triangular matrix* (EUTM) defined in Appendix B, and $\mathbf{E}$ and $\mathbf{L}_2$, respectively, are the $n \times n$ upper and lower triangular matrices.

2. An essential property of the EUTM, $\mathbf{E}_K$, similar to the *elementary lower triangular matrix* [8], is

$$\mathbf{E}_k^{-1} \equiv (\mathbf{1} - \boldsymbol{\alpha}_k \boldsymbol{\lambda}_k^T)^{-1} = \mathbf{1} + \boldsymbol{\alpha}_k \boldsymbol{\lambda}_k^T, \tag{8}$$

where $\boldsymbol{\alpha}_k$ and $\boldsymbol{\lambda}_k$ are defined in Equations (21) and (22), respectively. Using Equation (8), the GIM, $\mathbf{I}$, is written from Equation (7) as

$$\mathbf{I} = \mathbf{U}\mathbf{L}_2 \quad \text{where} \quad \mathbf{U} \equiv \mathbf{E}^{-1}. \tag{9}$$

In Equation (9), $\mathbf{U}$ and $\mathbf{L}_2$ are the $n \times n$ upper and lower triangular matrices, respectively. Moreover, from the inverse of the EUTM, Equation (8), it is clear that the diagonal elements of $\mathbf{U}$ are unity and the above-diagonal elements are the components of the vector, $\boldsymbol{\alpha}_k$, for $k = 2, \ldots, n$, that are evaluated in Equation (25).

3. Since the factorization given by Equation (9) is not unique [8], a unique decomposition is obtained by normalizing the elements of $\mathbf{L}_2$ as

$$\mathbf{L}_2 = \mathbf{D}\mathbf{L} \quad \text{where} \quad \mathbf{D} \equiv \text{diag}[\hat{m}_1, \ \ldots, \ \hat{m}_n], \tag{10}$$

$\mathbf{D}$ being the $n \times n$ diagonal matrix, whose non-zero elements are those of the matrix, $\mathbf{L}_2$, as calculated in Equation (26). Hence, the diagonal elements of matrix $\mathbf{L}$ are unity.

4. Since, the GIM, $\mathbf{I}$, is a symmetric positive definite matrix, $\mathbf{L} \equiv \mathbf{U}^T$ [8], and, hence, the desired decomposition of the manipulator GIM, $\mathbf{I}$ of Equation (1), is given as

$$\mathbf{I} = \mathbf{U}\mathbf{D}\mathbf{U}^T, \tag{11}$$

where the elements of the matrices, $\mathbf{U}$ and $\mathbf{D}$, are evaluated using Equations (25–28).

2.2. FORWARD DYNAMICS ALGORITHM

Having found the decomposition of the GIM, given by Equation (11), an order n, i.e., $\mathcal{O}(n)$, forward dynamics algorithm [15] is presented in three steps, namely A, B, and C, below:

A. Solution for $\hat{\boldsymbol{\tau}}$: The solution, $\hat{\boldsymbol{\tau}} = \mathbf{U}^{-1}\boldsymbol{\phi}$, is evaluated as

$$\hat{\tau}_i = \tau_i - \mathbf{p}_i^T \boldsymbol{\eta}_{i,i+1}, \tag{12}$$

where $\hat{\tau}_n \equiv \tau_n$, and the six-dimensional vector, $\boldsymbol{\eta}_{i,i+1}$, is obtained as

$$\boldsymbol{\eta}_{i,i+1} \equiv \mathbf{B}_{i+1,i}^T \boldsymbol{\eta}_{i+1} \quad \text{and} \quad \boldsymbol{\eta}_{i+1} \equiv \boldsymbol{\psi}_{i+1} \hat{\tau}_{i+1} + \boldsymbol{\eta}_{i+1,i+2}, \tag{13}$$

in which $\boldsymbol{\eta}_{n,n+1} = \mathbf{0}$. The new vector, $\boldsymbol{\psi}_{i+1}$, is the six-dimensional vector which is evaluated in Appendix B.

B. Solution for $\tilde{\boldsymbol{\tau}}$: The solution of the equation, $\mathbf{D}\tilde{\boldsymbol{\tau}} = \hat{\boldsymbol{\tau}}$, involves the inverse of the diagonal matrix, $\mathbf{D}$ of Equation (10), which is simple, namely, $\mathbf{D}^{-1}$ has only nonzero diagonal elements that are the reciprocal of the corresponding diagonal elements of $\mathbf{D}$. Vector $\tilde{\boldsymbol{\tau}}$ is obtained as follows: For $i = 1, \ldots, n$,

$$\tilde{\tau}_i = \hat{\tau}_i / \hat{m}_i.$$

The scalar, $\hat{m}_i$, is defined in Equation (26).

C. Solution for $\ddot{\boldsymbol{\theta}}$: In this step, $\ddot{\boldsymbol{\theta}} \equiv \mathbf{U}^{-T} \tilde{\boldsymbol{\tau}}$, is calculated, for $i = 2, \ldots, n$, as

$$\ddot{\theta}_i = \tilde{\tau}_i - \boldsymbol{\psi}_i^T \boldsymbol{\mu}_{i,i-1}, \tag{14}$$

where $\ddot{\theta}_1 \equiv \tilde{\tau}_1$, and the six-dimensional vector, $\boldsymbol{\mu}_{i,i-1}$, is obtained from

$$\boldsymbol{\mu}_{i,i-1} \equiv \mathbf{B}_{i,i-1} \boldsymbol{\mu}_{i-1} \quad \text{and} \quad \boldsymbol{\mu}_{i-1} \equiv \mathbf{p}_{i-1} \ddot{\theta}_{i-1} + \boldsymbol{\mu}_{i-1,i-2}, \tag{15}$$

in which $\tilde{\boldsymbol{\mu}}_{10} = \mathbf{0}$.

3. Computer Algorithm

In this section, the computer algorithm of the forward dynamics scheme given in Section 2.2 is presented. The implementation details are provided with the associated computation count in terms of the number of multiplications/divisions (M) and additions/subtraction (A). Here, components of a vector or matrix in a frame, say, $\mathcal{F}$, is denoted with $[\cdot]_{\mathcal{F}}$, where '$\cdot$' is the vector or the matrix. The required input to run the algorithm are now given below: For $i = 1, \ldots, n$,

- *Constant* Denavit–Hartenberg (DH) parameters [23] – the DH parameters, as defined in Appendix A with reference to Figure 8 – of the system under study, i.e., a_i, b_i, and α_i, for a revolute pair, and a_i, b_i, and θ_i, for a prismatic pair.
- Mass of each body, m_i, and the vector denoting the distance of the $(i+1)$st joint from the ith mass center, C_i, in the $(i+1)$st frame, i.e., $[\mathbf{r}_i]_{i+1}$.
- Inertia tensor of the ith link about its mass center, C_i, in the $(i+1)$st frame, $[\mathbf{I}_i]_{i+1}$.
- Initial value for the *variable* DH parameter, i.e., θ_i, for a revolute pair, and b_i, for a prismatic pair, and their first time derivatives, i.e., $\dot{\theta}_i$ and $\dot{b}_i$, respectively.
- Time history of the input joint forces/torques, i.e., τ_i.
- Each component of the six-dimensional vector, $\boldsymbol{\phi}$ of Equation (1), i.e., ϕ_i.

The implementation based on the three steps of Section 2.2, i.e., A, B, and C, is presented next, along with their computational complexity count written inside the symbols, { and }, at the right of the first line of each step or sub-step.

A. Solution for $\hat{\boldsymbol{\tau}}$: Calculate

1. the three-dimensional vector, $[\mathbf{d}_i]_i$: For $i = 1, \ldots, n$, $\{4M\ 2A(n)\}$

$$[\mathbf{d}_i]_{i+1} = [\mathbf{a}_i]_{i+1} - \mathbf{Q}_i[\mathbf{r}_i]_{i+1} : 4M\ 2A(10M\ 7A),$$

where the three-dimensional vector, $[\mathbf{a}_i]_i$, and the 3×3 orthogonal matrix, $\mathbf{Q}_i$, that defines the orientation of the $(i+1)$st link with respect to the ith one, are given as

$$[\mathbf{a}_i]_i \equiv \begin{bmatrix} a_i \cos\theta_i \\ a_i \sin\theta_i \\ b_i \end{bmatrix} \quad \text{and} \quad \mathbf{Q}_i \equiv \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i \end{bmatrix},$$

in which the DH parameters, a_i, b_i, α_i, and θ_i, are input. Note here that a straightforward calculation of the expression, $([\mathbf{a}_i]_i - \mathbf{Q}_i[\mathbf{r}_i]_{i+1})$, would require $10M\ 7A - 8M\ 4A$ for $\mathbf{Q}_i[\mathbf{r}_i]_{i+1}$, $2M$ to obtain $[\mathbf{a}_i]_i$, and $3A$ for the subtraction – that is indicated within the parentheses. However, if one looks at the resulting expressions, it becomes obvious that many terms assoiciated with a_i, α_i, and the components of $[\mathbf{r}_i]_{i+1}$ are constants and can be computed off-line. Thus, the number of computations required in each time step reduces to $4M\ 2A$, which is counted. It is also pointed out here that the structure of the 3×3 orthogonal matrix, $\mathbf{Q}_i$, is exploited to reduce the complexity of its multiplication with an arbitrary three-dimensional vector from $9M\ 6A$ to $8M\ 4A$.

2. the three-dimensional vector, $[\mathbf{c}_{i,i-1}]_i$: For $i = n, \ldots, 2$, $\{3A(n-1)\}$

$$[\mathbf{c}_{i,i-1}]_i = -[\mathbf{r}_{i-1}]_i - [\mathbf{d}_i]_i : 3A$$

The above computation requires only $3A$ because $[\mathbf{r}_{i-1}]_i$ is input.

3. the 6×6 matrix, $[\hat{\mathbf{M}}_i]_i$: For the purpose of showing the computation details, the following definition of $\hat{\mathbf{M}}_i$, as defined in Equation (26), is introduced:

$$\hat{\mathbf{M}}_i \equiv \begin{bmatrix} \hat{\mathbf{I}}_i & \mathbf{F}_i^T \\ \mathbf{F}_i & \mathbf{G}_i \end{bmatrix}, \tag{16}$$

where $\hat{\mathbf{I}}_i$, $\mathbf{F}_i$, and $\mathbf{G}_i$ are 3×3 block matrices, in which $\hat{\mathbf{I}}_i$ and $\mathbf{G}_i$ are symmetric.

(a) $[\hat{\mathbf{M}}_n]_n$: For $i = n$, $\{16M\ 17A\}$

$$[\hat{\mathbf{I}}_n]_n = \mathbf{Q}_n[\mathbf{I}_n]_{n+1}\mathbf{Q}_n^T : 16M\ 17A;$$

$$[\mathbf{F}_n]_n = \mathbf{O} : \text{nil}; \quad [\mathbf{G}_n]_n = m_n\mathbf{1} : \text{nil},$$

where $[\mathbf{I}_n]_{n+1}$ and m_n are input, and the two matrix multiplications require only $16M\ 17A$, instead of $2(27M\ 18A) = 54M\ 36A$. This is due to the structure of $\mathbf{Q}_i$ and the symmetric elements of $\mathbf{I}_i$. Moreover, matrices $[\mathbf{F}_n]_n$ and $[\mathbf{G}_n]_n$ do not require any computation, which are indicated with the word 'nil',

(b) $[\hat{\mathbf{M}}_i]_i$: For $i = n-1, \ldots, 2$, $\{(86n-179)M\ (101n-215)A\}$

$$[\mathbf{G}_i]_{i+1} = m_i\mathbf{1} + [\mathbf{G}_{i+1,i+1}]_{i+1} : 3A; \quad [\mathbf{G}_i]_i = \mathbf{Q}_i[\mathbf{G}_i]_{i+1}\mathbf{Q}_i^T : 16M\ 17A;$$

$$[\mathbf{G}_{i+1,i+1}]_{i+1}[\mathbf{C}_{i+1,i}]_{i+1} : \begin{cases} 9M\ 2A, & \text{if } i = n-1, \\ 15M\ 9A, & \text{otherwise;} \end{cases}$$

$$\rightarrow [\mathbf{F}_i]_{i+1} = [\mathbf{F}_{i+1,i+1}]_{i+1} + (\,) : \begin{cases} 6A, & \text{if } i = n-1, \\ 9A, & \text{otherwise;} \end{cases}$$

$$[\mathbf{F}_i]_i = \mathbf{Q}_i[\mathbf{F}_i]_{i+1}\mathbf{Q}_i^T : 24M\ 22A;$$

$$[\mathbf{F}_{i+1,i+1}]_{i+1}^T[\mathbf{C}_{i+1,i}]_{i+1} - [\mathbf{C}_{i+1,i}]_{i+1}[\mathbf{F}_i]_{i+1} : \begin{cases} 14M\ 15A, & \text{if } i = n-1, \\ 15M\ 18A, & \text{otherwise;} \end{cases}$$

$$\rightarrow [\hat{\mathbf{I}}_i]_{i+1} = [\hat{\mathbf{I}}_{i+1,i+1}]_{i+1} + (\,) : 6A; [\hat{\mathbf{I}}_i]_i = \mathbf{Q}_i[\hat{\mathbf{I}}_i]_{i+1}\mathbf{Q}_i^T : 16M\ 17A$$

where to reduce the clumsiness of the expressions symbol ( ) is used to substitute a compound expression computed just left to it before $\rightarrow$ sign. For example, to find $[\mathbf{F}_i]_{i+1}$, $(\,) \equiv [\mathbf{G}_{i+1,i+1}]_{i+1}[\mathbf{C}_{i+1,i}]_{i+1}$. This convention will be followed throughout this section.

4. the six-dimensional vector, $[\hat{\boldsymbol{\psi}}_i]_i$: To show the computation steps, vector $\hat{\boldsymbol{\psi}}_i$, as in Equation (26), is defined as

$$\hat{\boldsymbol{\psi}}_i \equiv [\hat{\boldsymbol{\psi}}_{i(t)}^T,\ \hat{\boldsymbol{\psi}}_{i(b)}^T]^T, \tag{17}$$

where (t) and (b) in the subscripts stand for the three-dimensional top and bottom vectors containing respectively the top and bottom three components of $\hat{\boldsymbol{\psi}}_i$. Similar definition associated to other six-dimensional vectors will be used henceforth.

(a) $[\hat{\boldsymbol{\psi}}_n]_n$: For $i = n$, $\{2M\}$

$$[\hat{\boldsymbol{\psi}}_{n(t)}]_n = [\hat{\mathbf{I}}_n]_n[\mathbf{e}_n]_n : \text{nil}; \quad [\hat{\boldsymbol{\psi}}_{n(b)}]_n = m_n[\mathbf{e}_n]_n \times [\mathbf{d}_n]_n : 2M,$$

where $[\mathbf{e}_n]_n \equiv [\mathbf{e}_i]_i \equiv [0, 0, 1]^T$. Thus, computations associated with $[\mathbf{e}_i]_i$ either not required or simplified.

(b) $[\hat{\boldsymbol{\psi}}_i]_i$: For $i = n-1, \ldots, 2$ $\{12M\ 12A(n-3)\}$

$$[\hat{\mathbf{I}}_i]_i[\mathbf{e}_i]_i : \text{nil}; \rightarrow [\hat{\boldsymbol{\psi}}_{i(t)}]_i = (\,) + [\mathbf{F}_i]_i^T([\mathbf{e}_i]_i \times [\mathbf{d}_i]_i) : 6M\ 6A$$

$$[\mathbf{F}_i]_i[\mathbf{e}_i]_i : \text{nil}; \rightarrow [\hat{\boldsymbol{\psi}}_{i(b)}]_i \;=\; () + [\mathbf{G}_i]_i([\mathbf{e}_i]_i \times [\mathbf{d}_i]_i) : 6M\ 6A$$

5. the scalar $\hat{m}_i$: For $i = n, \ldots, 1$, $\{2M\ 2A(n)\}$

$$[\mathbf{e}_i]_i^T[\hat{\boldsymbol{\psi}}_{i(t)}]_i : \text{nil}; \rightarrow \hat{m}_i = () + ([\mathbf{e}_i]_i \times [\mathbf{d}_i]_i)^T[\hat{\boldsymbol{\psi}}_{i(b)}]_i : 2M\ 2A$$

6. the six-dimensional vector, $[\boldsymbol{\psi}_i]_i$: For $i = n, \ldots, 2$, $\{(6n-7)M\}$

$$[\boldsymbol{\psi}_{i(t)}]_i = [\hat{\boldsymbol{\psi}}_{i(t)}]_i/\hat{m}_i : 3M; \quad [\boldsymbol{\psi}_{i(b)}]_i = [\hat{\boldsymbol{\psi}}_{i(b)}]_i/\hat{m}_i : \begin{cases} 2M, & \text{if } i = n, \\ 3M, & \text{otherwise;} \end{cases}$$

7. the 6×6 matrix, $[\hat{\mathbf{M}}_{ii}]_i$: A definition similar to Equation (16) is used here.

   (a) $[\hat{\mathbf{M}}_{nn}]_n$: For $i = n$, $\{15M\ 8A\}$

$$[\hat{\boldsymbol{\psi}}_{n(t)}]_n[\boldsymbol{\psi}_{n(t)}]_n^T : 6M; \rightarrow [\hat{\mathbf{I}}_{nn}]_n = [\hat{\mathbf{I}}_n]_n - () : 6A;$$
$$[\mathbf{H}_{nn}]_n = -[\hat{\boldsymbol{\psi}}_{n(b)}]_n[\boldsymbol{\psi}_{n(t)}]_n^T : 6M$$
$$[\hat{\boldsymbol{\psi}}_{n(b)}]_n[\boldsymbol{\psi}_{n(b)}]_n^T : 3M; \rightarrow [\mathbf{G}_{nn}]_n = m_n\mathbf{1} - () : 2A$$

   (b) $[\hat{\mathbf{M}}_{ii}]_i$: For $i = n-1, \ldots, 2$, $\{21M\ 21A(n-2)\}$

$$[\hat{\boldsymbol{\psi}}_{i(t)}]_i[\boldsymbol{\psi}_{i(t)}]_i^T : 6M; \rightarrow [\hat{\mathbf{I}}_{ii}]_i = [\hat{\mathbf{I}}_i]_i - () : 6A; [\hat{\boldsymbol{\psi}}_{i(b)}]_i[\boldsymbol{\psi}_{i(t)}]_i^T : 9M;$$
$$\rightarrow [\mathbf{H}_{ii}]_i = [\mathbf{H}_i]_i - () : 9A;\ [\hat{\boldsymbol{\psi}}_{i(b)}]_i[\boldsymbol{\psi}_{i(b)}]_i^T : 6M;$$
$$\rightarrow [\mathbf{G}_{ii}]_i = [\mathbf{G}_i]_i - () : 6A$$

8. the six-dimensional vector $[\boldsymbol{\eta}_i]_i$:

   (a) $[\boldsymbol{\eta}_n]_n$: For $i = n$, $[\boldsymbol{\eta}_n]_n = \hat{\tau}_n[\boldsymbol{\psi}_n]_n$ $\{5M\}$
   (b) $[\boldsymbol{\eta}_i]_i$: For $i = n-1, \ldots, 2$ $\{6M\ 6A(n-2)\}$

$$\hat{\tau}_i[\boldsymbol{\psi}_i]_i : 6M; \rightarrow [\boldsymbol{\eta}_i]_i = () + [\boldsymbol{\eta}_{i,i+1}]_i : 6A$$

9. the six-dimensional vector, $[\boldsymbol{\eta}_{i-1,i}]_{i-1}$:
   For $i = n, \ldots, 2$, $\{(22n-26)M\ (14n-18)A\}$

$$[\boldsymbol{\eta}_{i(b)}]_i \times [\mathbf{c}_{i,i-1}]_i : \begin{cases} 4M\ 1A, & \text{if } i = n, \\ 6M\ 3A, & \text{otherwise;} \end{cases} \rightarrow [\boldsymbol{\eta}_{i-1,i(t)}]_i = [\boldsymbol{\eta}_{i(t)}]_i + () : 3A;$$
$$[\boldsymbol{\eta}_{i-1,i(t)}]_{i-1} = \mathbf{Q}_{i-1}[\boldsymbol{\eta}_{i-1,i(t)}]_i : 8M\ 4A;$$
$$[\boldsymbol{\eta}_{i-1,i(b)}]_i = [\boldsymbol{\eta}_{i(b)}]_i : \text{nil};$$

$$[\boldsymbol{\eta}_{i-1,i(b)}]_{i-1} = \mathbf{Q}_{i-1}[\boldsymbol{\eta}_{i-1,i(b)}]_i : \begin{cases} 6M\ 2A, & \text{if } i = n, \\ 8M\ 4A, & \text{otherwise;} \end{cases}$$

10. the scalar, τ_i:

(a) For $i = n$, $\hat{\tau}_n = \phi_n$ {nil}
(b) For $i = n-1, \ldots, 1$ $\{2M\ 3A(n-1)\}$

$$[\mathbf{e}_i]_i^T[\boldsymbol{\eta}_{i,i+1(t)}]_i : \text{nil};$$
$$\rightarrow ( ) + ([\mathbf{e}_i]_i \times [\mathbf{d}_i]_i)^T[\boldsymbol{\eta}_{i,i+1(b)}]_i : 2M\ 2A; \rightarrow \hat{\tau}_i = \phi_i - ( ) : 1A$$

B. Solution for $\tilde{\boldsymbol{\tau}}$: Calculate for $i = 1, \ldots, n$, $\{nM\}$

$$\tilde{\tau}_i = \hat{\tau}_i / \hat{m}_i : 1M.$$

C. Solution for $\ddot{\boldsymbol{\theta}}$: Calculate

1. the scalar, $\ddot{\theta}_i$:

(a) $\ddot{\theta}_1$: For $i = 1$, $\ddot{\theta}_1 = \tilde{\tau}_1$ {nil}
(b) the scalar, $\ddot{\theta}_i$: For $i = 2, \ldots, n$, $\{6M\ 6A(n-1)\}$

$$[\boldsymbol{\psi}_{i(t)}]_i^T[\boldsymbol{\mu}_{i,i-1(t)}]_i : 3M\ 2A \rightarrow ( ) + [\boldsymbol{\psi}_{i(b)}]_i^T[\boldsymbol{\mu}_{i,i-1(b)}]_i : 3M\ 3A;$$
$$\rightarrow \ddot{\theta}_i = \tilde{\tau}_i - ( ) : 1A$$

2. the six-dimensional vector, $[\boldsymbol{\mu}_i]_{i+1}$:

(a) $[\boldsymbol{\mu}_1]_2$: For $i = 1$, $\{12M\ 2A\}$

$$[\boldsymbol{\mu}_{1(t)}]_1 = \ddot{\theta}_1[\mathbf{e}_1]_1 : \text{nil};\ [\boldsymbol{\mu}_{1(t)}]_2 = \mathbf{Q}_1^T[\boldsymbol{\mu}_{1(t)}]_1 : 4M$$
$$[\boldsymbol{\mu}_{1(b)}]_i = \ddot{\theta}_1([\mathbf{e}_1]_1 \times [\mathbf{d}_1]_1) : 2M;\ [\boldsymbol{\mu}_{1(b)}]_2 = \mathbf{Q}_1^T[\boldsymbol{\mu}_{1(b)}]_1 : 6M\ 2A$$

(b) $[\boldsymbol{\mu}_i]_{i+1}$: For $i = 2, \ldots, n$, $\{18M\ 11A(n-1)\}$

$$\ddot{\theta}_i[\mathbf{e}_i]_i : \text{nil}; \rightarrow [\boldsymbol{\mu}_{i(t)}]_i = ( ) + [\boldsymbol{\mu}_{i,i-1(t)}]_i : 1A;$$
$$[\boldsymbol{\mu}_{i(t)}]_{i+1} = \mathbf{Q}_i^T[\boldsymbol{\mu}_{i(t)}]_i : 8M\ 4A$$
$$\ddot{\theta}_i([\mathbf{e}_i]_i \times [\mathbf{d}_i]_i) : 2M; \rightarrow [\boldsymbol{\mu}_{i(b)}]_i = ( ) + [\boldsymbol{\mu}_{i,i-1(b)}]_i : 2A;$$
$$[\boldsymbol{\mu}_{i(b)}]_{i+1} = \mathbf{Q}_i^T[\boldsymbol{\mu}_{i(b)}]_i : 8M\ 4A$$

Table I. Computational complexities in forward dynamics.

Algorithm	M	A	$n = 6$	$n = 10$
Proposed	$\underline{191n - 284}$	$\underline{187n - 325}$	$\underline{862M\ 797A}$	$\underline{1626M\ 1545A}$
Featherstone [10]	$199n - 198$	$174n - 173$	$996M\ 871A$	$1792M\ 1567A$
Valasek [14]	$226n - 343$	$206n - 345$	$1013M\ 891A$	$1917M\ 1715A$
Brandl et al. [14]	$250n - 222$	$220n - 198$	$1278M\ 1122A$	$2278M\ 2002A$
Walker and Orin [5] (as implemented by Featherstone [10])	$\frac{1}{6}n^3 + \frac{23}{2}n^2 + \frac{115}{3}n - 47$	$\frac{1}{6}n^3 + 7n^2 + \frac{233}{6}n - 46$	$633M\ 480A$	$1653M\ 1209A$

M: Multiplication/Division; A: Addition/Subtraction

3. $[\boldsymbol{\mu}_{i,i-1}]_i$: For $2, \ldots, n$, $\{6M\ 6A(n-1)\}$

$$[\boldsymbol{\mu}_{i,i-1(t)}]_i = [\boldsymbol{\mu}_{i-1(t)}]_i : \text{nil};\ [\mathbf{c}_{i,i-1}]_i \times [\boldsymbol{\mu}_{i-1(t)}]_i : 6M\ 3A;$$
$$\rightarrow [\boldsymbol{\mu}_{i,i-1(b)}]_i = [\boldsymbol{\mu}_{i-1(t)}]_i + (\,) : 3A$$

The total complexity of the proposed implementation is then obtained by adding the above number of multiplications and additions inside { and }. The result is shown in Table I, where comparison with other algorithms is also made. The complexity of the proposed algorithm, compared to other $\mathcal{O}(n)$ algorithms, is found to be the best in terms of the number of multiplications required. Otherwise, it is very similar to the one reported by Featherstone [10].

A computer program in C++ is written for the implementation of the forward dynamics algorithm presented in this section. Moreover, an integration program based on the Runge–Kutta–Fehlberg [24] (RKF) formula is also developed in C++ to solve the dynamic equations of motion written in the state-space form. The simulation results are presented and discussed in Section 4.

4. Results

In order to test the simulation scheme presented in Sections 2 and 3, simulation of three manipulators, namely, (i) a three-DOF all revolute planar arm; (ii) the six-DOF Stanford arm with five revolute joints and one prismatic joint, as shown in Figure 3(i); and (iii) a six-DOF PUMA robot with all revolute joints, are performed. Two types of simulations are done: (a) free-fall and (b) forced. In the case of free-fall, it is assumed that no external joint forces/torques are acting. The motion of the manipulator is due to pure gravity only. In forced simulation, joint forces/torques, i.e., vector $\boldsymbol{\tau}$ of Equation (1), required to follow a given trajectory, e.g., Equation (18), are given as input, whereas vector $\boldsymbol{\phi}$ is calculated from an

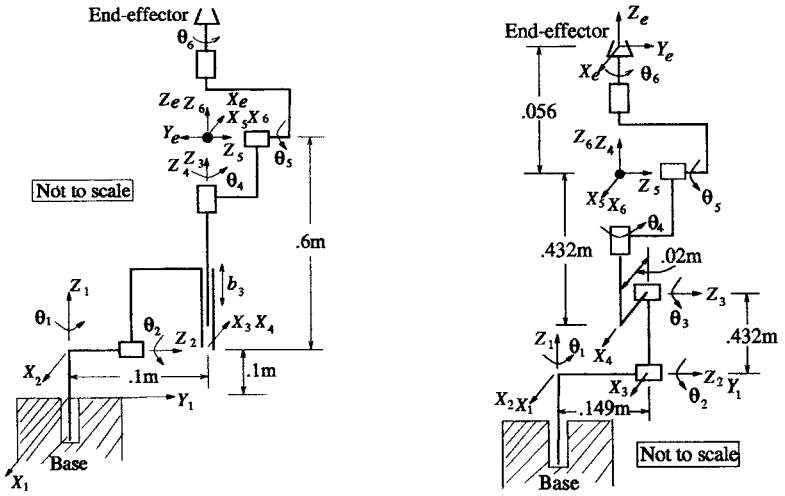

Figure 3. Six-DOF serial manipulators.

inverse dynamics program, say, as in [20], while $\ddot{\boldsymbol{\theta}} = \mathbf{0}$.

$$\theta_i(\text{or } b_i) = \frac{1}{2}\left[\frac{2\pi}{T}t - \sin\left(\frac{2\pi}{T}t\right)\right];$$

$$\theta_i\text{: for a revolute joint;} \quad b_i\text{: for a prismatic joint.} \tag{18}$$

Three-DOF arm: The simulation results obtained for the three-DOF planar arm are compared with those obtained from the explicit dynamic equations of motion, as in Angeles [7] or Saha [21]. The results in both the cases, i.e., free-fall and forced, exactly match. Note that, for the forced simulation, the joint torques are calculated to maintain the joint trajectory given by Equation (18) with the following values: $T = 10.0$ sec; and initial conditions at time, $t = 0$, i.e., $\dot{\theta}_i(0) = \theta_i(0) = 0$, for $i = 1, 2, 3$. The plots are not shown here, as the plots for the more complex manipulators, namely, the Stanford arm and a PUMA robot, obtained from the same algorithm are given next and discussed.

The Stanford arm: The Denavit–Hartenberg (DH) parameters [23] of the Stanford arm, Figure 3(i), along with its mass and inertia properties, are shown in Table II(i). The definitions of the DH parameters, as indicated in Figure 8, used to find the values in Table II(i) are given in Appendix A.

1. *Free-fall simulation*. First, the free-fall simulation of the Stanford arm is carried out with the following values: $T = 10.0$ sec; $\theta_i(0) = 0$, for $i \neq 2, 3$, $\theta_2(0) = 90°$, $b_3(0) = 0$; $\dot{\theta}_i(0) = 0$, for $i \neq 3$, and $\dot{b}_3(0) = 0$. Time step, ΔT, for numerical integration is taken as, $\Delta T = 0.01$ sec. Variations of the joint positions vs. time are shown in Figures 4(i)(a–f).

Table II. DH parameters, and mass and inertia properties.

i	a_i	b_i	α_i	θ_i	m_i	r_x	r_y	r_z	I_{xx}	I_{yy}	I_{zz}
	(m)	(m)	(deg)	(deg)	(kg)		(m)			(kg-m^2)	
(i) For the Stanford arm [Figure3(i)]											
1	0	0.1	–90	θ_1[0]	9	0	–0.1	0	0.01	0.02	0.01
2	0	0.1	–90	θ_2[90]	6	0	0	0	0.05	0.06	0.01
3	0	b_3	0	0[0]	4	0	0	0	0.4	0.4	0.01
4	0	0.6	90	θ_4[0]	1	0	–0.1	0	0.001	0.001	0.0005
5	0	0	–90	θ_5[0]	0.6	0	0	0	0.0005	0.0005	0.0002
6	0	0	0	θ_6[0]	0.5	0	0	0	0.003	0.001	0.002
(ii) For the PUMA Robot [Figure 3(ii)]											
1	0	0	–90	θ_1[0]	10.521	0	0	0.054	1.612	–1.612	0.5091
2	0.432	0.149	0	θ_2[0]	15.761	0.292	0	0	0.4898	8.0783	8.2672
3	0.02	0	90	θ_3[0]	8.767	–0.02	0	–0.197	3.3768	3.3768	–0.3009
4	0	0.432	–90	θ_4[0]	1.052	0	–0.057	0	0.181	–0.1273	0.181
5	0	0	90	θ_5[0]	1.052	0	0	–0.007	0.0735	0.1273	–0.0735
6	0	0.056	0	θ_6[0]	0.351	0	0	0.019	0.0071	0.0071	0.0141

Values inside [ ] under column θ_i and b_i denote initial value.

As no results for such simulation are readily available they are interpreted intuitively. For example, referring to Figure 3(i) and the initial configuration of the Stanford arm given in Table II(i), the motion of joint 3 should increase sharply once the joint 2 turns more than 90°. This is true, as evident from Figures 4(i)(c) and (b), respectively. In Figure 4(i)(b), the joint 2 turns 90° from its initial position of 90°, i.e., 1.5708 rad, when the value becomes zero. This happens a little after 0.5 sec, when the displacement of joint 3 starts picking up (Figure 4(i)(c)).

2. *Forced simulation*. The forced simulation is performed next with the joint force/torques calculated to maintain the trajectory given by Equation (18) with the following values: $T = 10.0$ sec; $\theta_i(0) = 0$, for $i \neq 2, 3$, $\theta_2(0) = 90°$, $b_3(0) = 0$; $\theta_i(T) = 60°$, for $i \neq 3$, $b_3(T) = 0.1$ m. The calculated joint force/torques are shown in Figure 5, which will be the input, vector $\boldsymbol{\tau}$ of Equation (1), for the forced simulation. The initial conditions are same as in the free-fall simulation. The simulated joint positions are shown in Figures 6(i)(a–f). The results show that the system deviates from the desired trajectory, i.e., Equation (18), after about 2.5 sec. In order to verify the integration results, two standard routines available in NEWMOS* software are used. Initially, *SHAMGOR* algorithm, which is based on the Adams–Bashforth–Moulton [24] (ABM) formula, is used. The comparison

* A simulation software in C at the Institute of B Mechanics, University of Stuttgart, Germany.

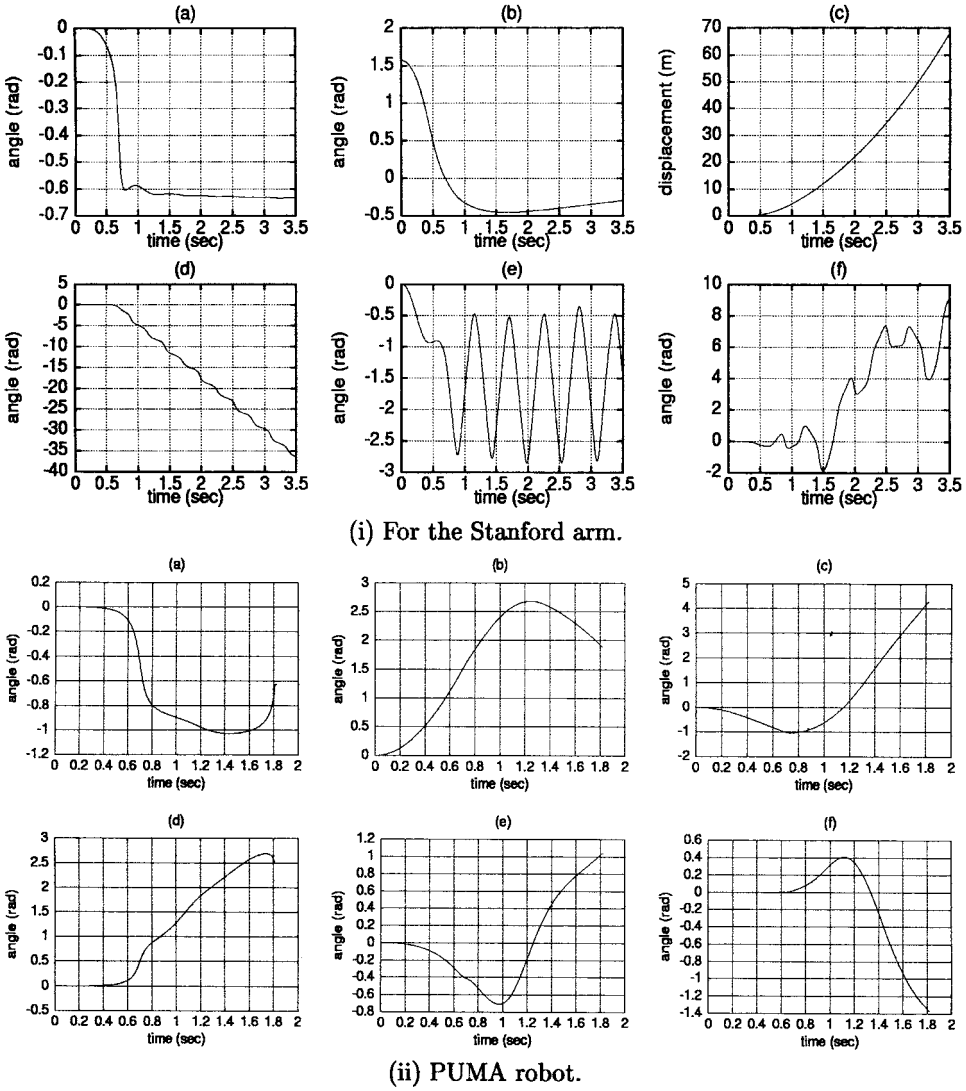

Figure 4. Free fall simulation: Positions at joint (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6.

of the results are shown in Figures 6(i)(a–f). Later, *DVERK* routine of NEWMOS based on the RKF formula is also used, which produced similar plots. Note that in both the cases tolerance value is taken as 10^{-6}.

The overall simulation result is then verified by comparing the simulated positions of the end-effector with those reported in Bae and Haug [11], where comparison was done with a commercial software, DADS. Since the present dynamic formulation is in the joint space, a forward kinematics program is written to obtain the end-effector positions. The results are shown in Figures 7(a–c). Note here that the displacement of the end-effector along Y_1, as in Figure 7b, is the end-effector

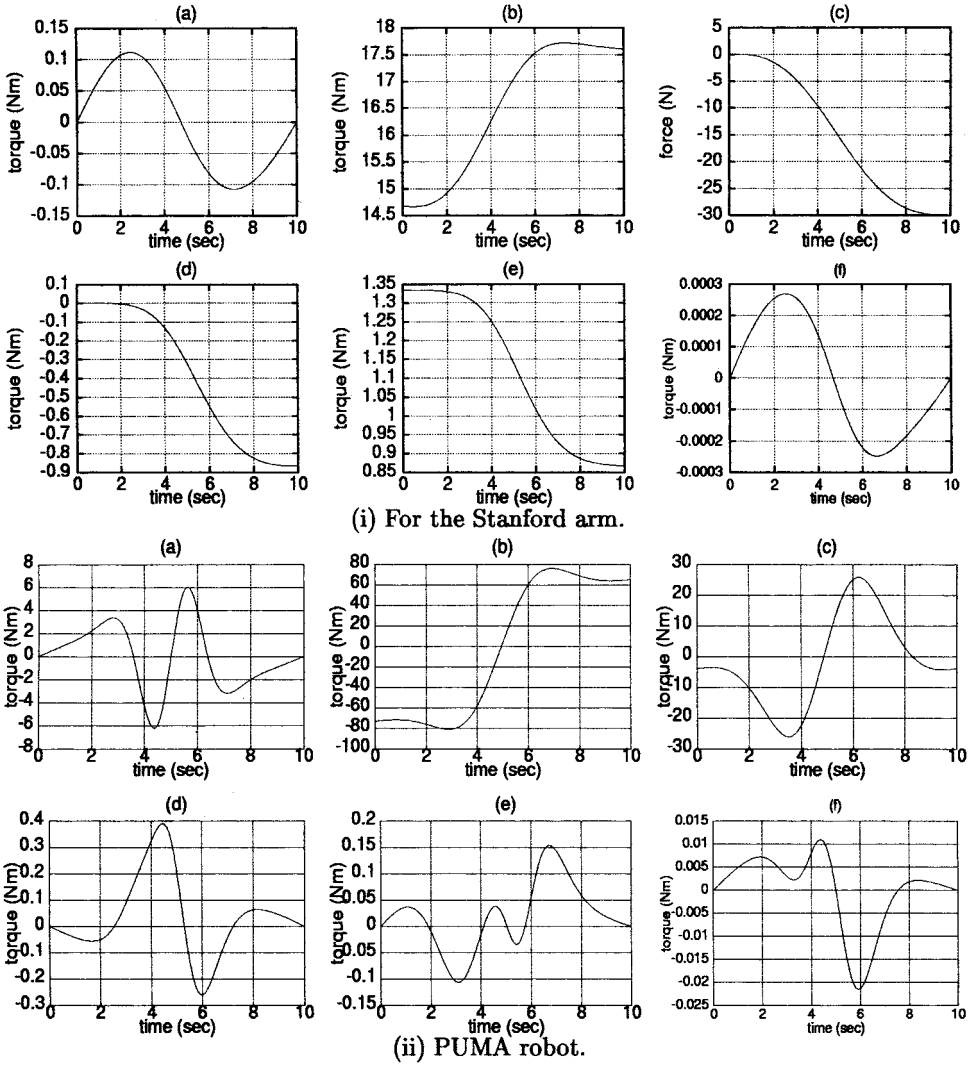

Figure 5. Force/torques at joint (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6.

displacement plot given in Bae and Haug [11]. The results show a similar trend.

A PUMA robot: The DH parameters of the PUMA robot, as shown in Figure 3(ii), along with its mass and inertia properties are given in Table II(ii). The simulations are performed with the in-house developed RKF based integration scheme, as the results for the Stanford arm with this scheme is very close to the other established algorithms.

1. *Free-fall simulation*. Free-fall simulation of the PUMA robot is carried out similar to the Stanford arm, where $T = 10.0$ sec; $\dot{\theta}_i = \theta_i(0) = 0$, for $i = 1, \ldots, 6$,

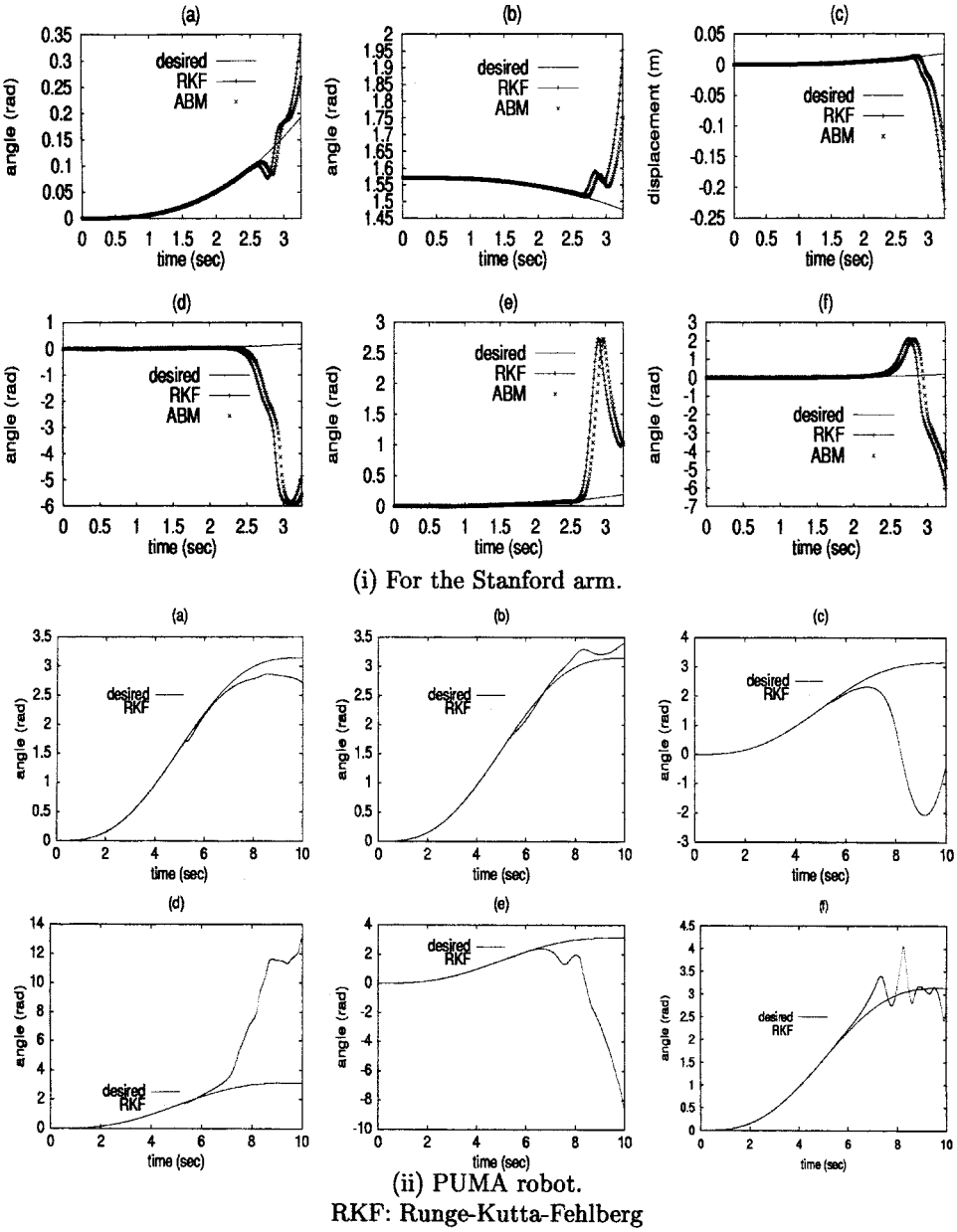

Figure 6. Forced simulation: Positions at joint (a) 1; (b) 2; (c) 3; (d) 4; (e) 5; (f) 6.

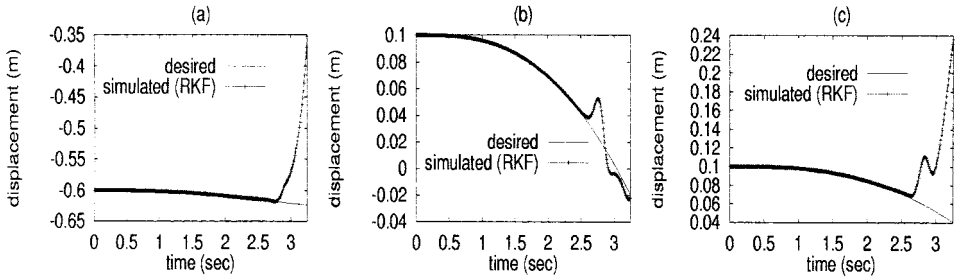

Figure 7. Forced simulation: End-effector positions along (a) X_1; (b) Y_1; (c) Z_1, axes.

and $\Delta T = 0.01$ sec. Variations of the joint positions vs. time are shown in Figures 4(ii)(a–f). It is clear from Figure 3(ii) that due to the length, $a_3 = 0.02$, the joint 2 will rotate in the positive direction, which is evident from Figure 4(ii)(b).

2. *Forced simulation*. The forced simulation is performed next with the joint force/torques calculated to maintain the trajectory given by Equation (18) with the following values: $T = 10.0$ sec; $\theta_i(0) = 0$, and $\theta_i(T) = \pi$, for $i = 1, \ldots, 6$. The calculated joint torques are shown in Figure 5(ii). Using the same initial conditions as in the free-fall simulation, the simulated joint positions are shown in Figures 6(ii)(a–f). The results show that the system deviates from the desired trajectory, Equation (18), after about 6 sec.

5. Conclusions

Based on the $\mathbf{UDU}^T$ decomposition of the generalized inertia matrix resulting from the dynamic equations of motion, i.e., $\mathbf{I}$ of Equation (1), an efficient implementation of the order n, $\mathcal{O}(n)$, forward dynamics algorithm [15] is presented. The complexity analysis of the proposed implementation is also carried out, which is compared in Table I with other algorithms. Note that the proposed implementation, compared to other $\mathcal{O}(n)$ algorithms, is the best in terms of multiplication count, even better than the one reported by the author earlier [15], $(201n-335)M$ $(193n-361)A$.

Using the proposed algorithm, simulations of three manipulators, namely, a three-DOF arm, the six-DOF Stanford arm, and a six-DOF PUMA robot, are performed, whose results are given in Section 4 and discussed. The comparison of the simulation results, mainly, the one with Bae and Haug [11] for the Stanford arm validates the developed simulation algorithm and program.

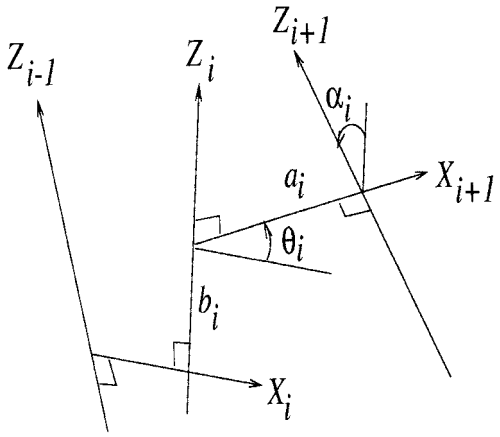

Figure 8. DH nomenclature.

Appendix A. Denavit–Hartenberg Nomenclature

In order to describe the the Denavit–Hartenberg (DH) nomenclatures [23] that is followed here, note that the manipulator under study, as shown in Figure 1, consists of $n + 1$ bodies, namely, a fixed "base" and n links denoted by #1, ..., #n, coupled by n kinematic pairs, say, a revolute or a prismatic, numbered as $1, \ldots, n$, respectively. Referring to Figure 1, the ith pair couples the $(i - 1)$st and the ith links. Moreover, a coordinate system X_i, Y_i, Z_i is attached to the $(i - 1)$st link. Then, for the first n frames, DH parameters are defined according to the following rules (referring to Figure 8):

1. Z_i is the axis of the ith pair. Its positive direction can be chosen arbitrarily.
2. X_i is defined as the common perpendicular to Z_{i-1} and Z_i, directed from the former to the latter. The origin of the ith frame, O_i, is the point where X_i intersects Z_i. If these two axes intersect, the positive direction of X_i is chosen arbitrary. The origin, O_i, is coincides with the origin of the $(i - 1)$st frame, O_{i-1}.
3. the distance between Z_i and Z_{i+1} is defined as a_i, which is *nonnegative*.
4. the Z_i coordinate of the intersection of the X_{i+1} axis with Z_i is defined as b_i, which thus can be either positive or negative. For a prismatic joint, b_i is variable.
5. the angle between Z_i and Z_{i+1} is defined as α_i, and is measured about the positive direction of X_{i+1}, and
6. the angle between X_i and X_{i+1} is defined as θ_i, and is measured about the positive direction of Z_i. For a revolute joint, θ_i is variable.

Since no $(n + 1)$st link exists the above definitions do not apply to the $(n + 1)$st frame and it can be chosen at will.

Appendix B: RGE of the GIM, I

Conventionally, the Gaussian elimination (GE) [8] begins from the first column of the matrix under interest. In the proposed elimination, it is assumed that the GE of matrix **I**, Equation (1), whose elements are given by Equation (6), starts from the nth column. This is to obtain recursive relations starting from the nth body. Thus, the name *reverse* Gaussian elimination (RGE) is used. In RGE, after the annihilation of the first $(n-1)$ elements of the nth column, the modified inertia matrix, denoted by $\mathbf{L}_n$, is looks like

$$\mathbf{L}_n \equiv \begin{bmatrix} i_{11}^{(n)} & & \text{sym} & 0 \\ \vdots & \ddots & & 0 \\ i_{n-1,1}^{(n)} & \cdots & i_{n-1,n-1}^{(n)} & 0 \\ i_{n1} & \cdots & i_{n,n-1} & i_{nn} \end{bmatrix}, \tag{19}$$

where i_{nn} is the *pivot* [8] and $i_{ij}^{(n)}$ are the modified elements of **I**, whereas 'sym' denotes the symmetric elements of the $(n-1)\times(n-1)$ matrix, resulting from the deletion of the nth row and column of matrix $\mathbf{L}_n$. Equation (19) is realized by premultiplying matrix **I** with the *elementary upper triangular matrix* (EUTM) of order n and index n, as done in GE with *elementary lower triangular matrix* (ELTM) [8]. An EUTM of order n and index k, denoted by $\mathbf{E}_k$, is defined as

$$\mathbf{E}_k \equiv \mathbf{1} - \boldsymbol{\alpha}_k \boldsymbol{\lambda}_k^T, \tag{20}$$

where $\mathbf{1}$ is the $n\times n$ identity matrix and the n-dimensional vectors, $\boldsymbol{\alpha}_k$ and $\boldsymbol{\lambda}_k$, are

$$\boldsymbol{\alpha}_k \equiv [\alpha_{1k}, \ \ldots, \ \alpha_{k-1,k}, \ 0, \ \ldots, \ 0]^T, \tag{21}$$

$$\boldsymbol{\lambda}_k \equiv [\,0, \quad \ldots, \quad 0, \quad 1, \ \ldots, \ 0]^T. \tag{22}$$

From Equations (21) and (22), the EUTM, $\mathbf{E}_k$, Equation (20), has the following structure:

$$\mathbf{E}_k \equiv \begin{bmatrix} 1 & 0 & \cdots & -\alpha_{1k} & \cdots & 0 \\ & \ddots & \vdots & \vdots & \vdots & \vdots \\ & & 1 & -\alpha_{k-1,k} & \cdots & 0 \\ & & & 1 & \cdots & 0 \\ & & 0's & & \ddots & \vdots \\ & & & & & 1 \end{bmatrix}, \tag{23}$$

where 0's imply zeros. Moreover, in the RGE, the modified matrix after the annihilation of the first $k-1$ elements of the kth column is expressed as

$$\mathbf{L}_k = \mathbf{E}_k \mathbf{L}_{k+1}. \tag{24}$$

Note that, if $k = n$ and $\mathbf{L}_{n+1} \equiv \mathbf{I}$, the matrix, $\mathbf{L}_n$, as in Equation (19), follows from Equation (24). Furthermore, the elements of $\mathbf{E}_k$ and $\mathbf{L}_k$, namely, α_{ik} and $i_{ij}^{(k)}$, respectively, are computed from the following scheme:

- For $k = n, \ldots, 2$; Do $i = k-1, \ldots, 1$; Do $j = i, \ldots, 1$

$$\alpha_{ik} = \mathbf{p}_i^T \boldsymbol{\psi}_{ik} \quad \text{and} \quad i_{ij}^{(k)} = \mathbf{p}_i^T \hat{\mathbf{M}}_{ik} \mathbf{B}_{ij} \mathbf{p}_j \tag{25}$$

  end do j; end do i; end for k.

In Equation (25), matrix $\mathbf{B}_{ij}$, and vectors $\mathbf{p}_i$ and $\mathbf{p}_j$ are defined in Equation (5), whereas the six-dimensional vector $\hat{\boldsymbol{\psi}}_{ik}$ and the terms associated to it are given as

$$\hat{\boldsymbol{\psi}}_k \equiv \hat{\mathbf{M}}_k \mathbf{p}_k, \quad \hat{\boldsymbol{\psi}}_{ik} \equiv \mathbf{B}_{ki}^T \hat{\boldsymbol{\psi}}_k, \quad \hat{m}_k \equiv \mathbf{p}_k^T \hat{\boldsymbol{\psi}}_k, \tag{26}$$

$$\boldsymbol{\psi}_k \equiv \frac{\hat{\boldsymbol{\psi}}_k}{\hat{m}_k} \quad \text{and} \quad \boldsymbol{\psi}_{ik} \equiv \frac{\hat{\boldsymbol{\psi}}_{ik}}{\hat{m}_k}. \tag{27}$$

In Equation (26), the definition, $\hat{\mathbf{M}}_k \equiv \hat{\mathbf{M}}_{k,k+1}$, is used to simplify the notations. The 6×6 matrix, $\hat{\mathbf{M}}_{ik}$ of Equation (25), and $\hat{\mathbf{M}}_k$ of Equation (26), are evaluated from the following relations:

$$\hat{\mathbf{M}}_{ik} = \mathbf{M}_i + \mathbf{B}_{i+1,i}^T \hat{\mathbf{M}}_{i+1,k} \mathbf{B}_{i+1,i} \quad \text{and} \quad \hat{\mathbf{M}}_{kk} = \hat{\mathbf{M}}_k - \hat{\boldsymbol{\psi}}_k \boldsymbol{\psi}_k^T, \tag{28}$$

where $k = n, \ldots, 2$; $i = k-1, \ldots, 1$. Moreover, for $i = k-1$, $\hat{\mathbf{M}}_{i+1,k} \equiv \hat{\mathbf{M}}_{kk}$, and $\hat{\mathbf{M}}_n \equiv \mathbf{M}_n$.

Note that, in contrast to the definition of $\tilde{\mathbf{M}}_i$ associated to the *composite body*, i, as in Equation (6), the matrix, $\hat{\mathbf{M}}_i (\equiv \hat{\mathbf{M}}_{i,i+1})$, as in Equation (28), for $k = i+1$, has the following interpretation: It represents the mass and inertia properties of the *articulated body* i, which is defined as the system consisting of $(n-i+1)$ bodies, $\#i, \ldots, \#n$, that are coupled by joints $i+1, \ldots, n$, i.e, replace the fixed joints of the ith composite body, as shown in Figure 1 by the dotted line, by kinematic pairs, say, a revolute joint or a prismatic. Matrix, $\hat{\mathbf{M}}_i$ is the *articulated-body inertia of link* i [10], and the *state estimation error covariance* [12] that satisfies the discrete Riccati equations. Thus, it is commented that a correlation exists between the Gaussian elimination technique and the Kalman filtering approach, which can be exploited for the deeper understanding of the dynamics characteristics of a complex mechanical system. Finally, the scalar, $\hat{m}_i$, is interpreted as the moment of inertia of the *articulated body*, i, about the axis of rotation of the ith joint.

Acknowledgments

The research work reported in this paper was possible under the Department of Science and Technology (DST), Government of India, Grant No. HR/OY/E-02/96. The last part of the work is, however, completed during the author's research stay in the Institute of B Mechanics, University of Stuttgart, Germany, as a Humboldt Fellow in 1999. The author sincerely acknowledges the financial supports from

both the DST and the AvH, and thanks Prof. Schiehlen for hosting the author in his institute.

References

1. Huston, H. and Passerello, C.E., 'On constraint equations–A new approach', *ASME J. Appl. Mech.* **41**, 1974, 1130–1131.
2. Wehage, R.A. and Haug, E.J., 'Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems', *ASME J. Mech. Design* **104**, 1982, 247–255.
3. Angeles, J. and Lee, S., 'The formulation of dynamical equations of holonomic mechanical systems using a natural orthogonal complement', *ASME J. Appl. Mech.* **55**, 1988, 243–244.
4. Sciavicco, L. and Siciliano, B., *Modeling and Control of Robot Manipulators*, McGraw-Hill, New York, 1996.
5. Walker, M.W. and Orin, D.E., 'Efficient dynamic computer simulation of robotic mechanisms', *ASME J. Dynam. Systems, Measurements, Control* **104**, 1982, 205–211.
6. Kane, T.R. and Levinson, D.A., 'The use of Kane's dynamical equations for robotics', *Internat. J. Robotic Res.* **2**(3), 1983, 3–21.
7. Angeles, J., *Rational Kinematics*, Springer-Verlag, New York, 1997.
8. Golub, G.H. and Van Loan, C.F. *Matrix Computations*, John Hopkins University Press, Baltimore, MD, 1983.
9. Armstrong, W.W., 'Recursive solution to the equations of motion of an n-link manipulator', in *Proceedings of the 5th World Congress on Theory of Machines and Mechanics*, Montreal, Canada, ASME, New York, Vol. 2, 1979, 1343–1346.
10. Featherstone, R., 'The calculation of robot dynamics using articulated-body inertias', *Internat. J. Robotic Res.* **2**(1), 1983, 13–30.
11. Bae, D. and Haug, E.J., 'A recursive formulation for constrained mechanical system dynamics: Part I. Open loop systems', *Mech. Structures Mach.* **15**(3), 1987, 359–382.
12. Rodriguez, G., 'Kalman filtering, smoothing, and recursive robot arm forward and inverse dynamics', *IEEE Trans. Robotics & Automation* **RA-3**(6), 1987, 624–639.
13. Schiehlen, W., 'Computational aspects in multibody system dynamics', *Comput. Methods Appl. Mech. Engrg.* **90**(1–3), 1991, 569–582.
14. Stejskal, V. and Valasek, M., *Kinematics and Dynamics of Machinery*, Marcel Dekker, New York, 1996.
15. Saha, S.K., 'A decomposition of the manipulator inertia matrix', *IEEE Trans. Robotics & Automation* **13**(2), 1997, 301–304.
16. Bae, D. and Haug, E.J., 'A recursive formulation for constrained mechanical systems dynamics: Part III, Parallel processing implementation', *Mech. Structures Mach.* **16**, 1987, 249–269.
17. Fijany, A., Sharf, I. and D'Eleuterio, M.T.D., 'Parallel $\mathcal{O}(\log N)$ algorithms for computation of manipulator forward dynamics, *IEEE Trans. Robotics & Automation* **11**(3), 1995, 389–400.
18. Rodriguez, G., Jain, A. and Kreutz-Delgado, K., 'A spatial operator algebra for manipulation modeling and control', *Internat. J. Robotic Res.* **10**(4), 1991, 371–381.
19. Ascher, U.M., Pai, D.K. and Cloutier, B.P., 'Forward dynamics, elimination methods, and formulation stiffness in robot simulation', *Internat. J. Robotic Res.* **16**(6), 1997, 749–775.
20. Saha, S.K., 'Dynamic modeling of serial multi-body systems using the decoupled natural orthogonal complement matrices', *ASME J. Appl. Mech.* **66**(4), 1999, 986–996.
21. Saha, S.K., 'Analytical expression for the inverted inertia matrix of serial robots', *Internat. J. Robotic Res.* **18**(1), 1999, 116–124.
22. Saha, S.K. and Schiehlen, W.O., 'Recursive kinematics and dynamics for closed loop multibody systems', *Internat. J. Mech. Structures Machines* **29**(2), 2001, 143–175.

23. Denavit, J. and Hartenberg, R.S., 'A kinematic notation for lower-pair mechanisms based on matrices', *ASME J. Appl. Mech.* **77**, 1955, 215–221.
24. Shampine, L.F., *Numerical Solution of Ordinary Differential Equations*, Chapman and Hall, New York, 1994.