To appear in *IEEE Transactions on Robotics and Automation.*

Efficient Computation of Articulated-Body Inertias Using Successive Axial Screws [†]

Scott McMillan David E. Orin

Abstract

The articulated-body (AB) algorithm for dynamic simulation of chains of rigid bodies was developed by Featherstone [1]. The most costly step in this algorithm is the computation of the AB inertias at each link which involves a spatial (6×6) congruence transformation. The amount of computation required is closely coupled to the kinematic modeling technique used. This paper examines this computation in detail and presents an efficient step-by-step procedure for its evaluation in a serial chain with revolute and prismatic joints using modified Denavit-Hartenberg parameters for modeling the kinematics. The result is a very efficient procedure using successive axial screws that reduces the computational requirements of the AB algorithm by about 15% from results obtained by Brandl, Johanni, and Otter [2]. The procedure developed defines a general approach and can be used to improve the efficiency of spatial congruence transformations of other types of matrices, such as spatial rigid-body inertias (used in the Composite Rigid-Body simulation algorithm [3, 4]).

1. Introduction

The desire for real-time dynamic simulation of robotic systems has been shown in applications such as human-in-the-loop teleoperation where communication delays are significant (as in both underwater and space operations), and for hardware-in-the-loop testing and development of controllers for these systems. A number of efficient algorithms have been

[†] This work was supported in part by an AT&T Ph.D. Scholarship at The Ohio State University, by Grant No. BCS-9109989 from the National Science Foundation to the Naval Postgraduate School, and by Grant No. BCS-9311269 from the National Science Foundation to The Ohio State University.

S. McMillan is with the Department of Computer Science, Naval Postgraduate School, Monterey, CA 93943. D. E. Orin is with the Department of Electrical Engineering, The Ohio State University, Columbus, OH 43210.

developed in an effort to achieve the needed computational rates. One of these is the Articulated-Body (AB) algorithm which was first developed by Featherstone [1] and later improved by Brandl, Johanni, and Otter [2]. The advantage of this algorithm is its O(N) complexity, where N is the number of links. As such, the amount of computation only grows linearly with the number of links in the system [1, 2, 5].

The AB algorithm has three steps. In the first, quantities that depend on the state (joint positions and velocities) are computed recursively for each link from the base of the chain to the end-effector. The second is an inward recursion from the end-effector to the base that computes the forces exerted and AB inertias "felt" at each joint. Then, the joint accelerations are computed in the final, outward recursion. The computation of the AB inertias in the second step of the AB algorithm involves a spatial congruence transformation. This step accounts for a significant portion of the computation in the algorithm; therefore, efficient computation of these is critical for achieving an efficient implementation of the algorithm.

This paper presents a very efficient step-by-step procedure for the computation of the AB inertias. In the next section, the notation used in this paper is defined, including the equations for computing the AB inertias and the modified Denavit-Hartenberg (MDH) notation used to model the kinematics of the system and define the transformation matrices. In Section 3, the procedure for efficient computation is given for revolute joints. Then, a brief explanation of the procedure for prismatic joints is given in Section 4.

2. Background and Notation

This section begins with the pertinent equations for the computation of the AB inertias followed by an explanation of the spatial notation used. Since the amount of computation is strongly coupled to kinematic modeling of the links in the serial chain, the last part of this section describes the MDH notation and resulting definition of the transformation matrices used in this paper.

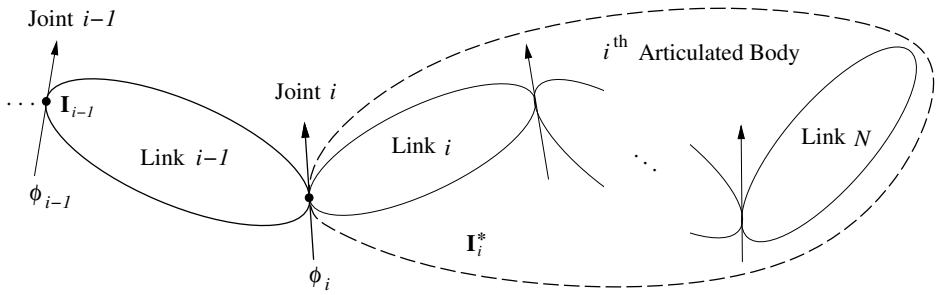

Figure 1: Quantities for computing the AB inertia for links $i-1$ through N.

A. Computation of an AB Inertia

Figure 1 illustrates the physical quantities used to compute $\mathbf{I}^*_{i-1}$, the $(i-1)$st AB inertia at joint $i-1$ for the articulated set of links $i-1$ through N. Physically, the $(i-1)$st AB inertia is the inertia "felt" at joint $i-1$ from all of the links outboard of this joint when the joints are free to move (that is, no joint torques). This is computed with the following equation:

$$\mathbf{I}^*_{i-1} \;=\; \mathbf{I}_{i-1} + {}^i\mathbf{X}^T_{i-1}\mathbf{N}_i\,{}^i\mathbf{X}_{i-1}, \tag{1}$$

where $\mathbf{I}_{i-1}$ is the spatial inertia of link $i-1$ and ${}^i\mathbf{X}_{i-1}$ is the spatial transformation matrix between link $i-1$'s and i's coordinate system.

In this discussion, the spatial notation developed by Lilly and Orin [4] is used which combines rotational and translational quantities into single six-dimensional vectors and 6×6 matrices. The spatial inertia relates the spatial acceleration of the link to the resultant spatial force exerted on it. In this 6×6 matrix, the link's mass and inertia quantities are combined as follows:

$$\mathbf{I}_{i-1} \;=\; \begin{bmatrix} \bar{\boldsymbol{I}}_{i-1} & m_{i-1}\tilde{\boldsymbol{s}}_{i-1} \\ m_{i-1}\tilde{\boldsymbol{s}}^T_{i-1} & m_{i-1}\mathbf{1}_3 \end{bmatrix}, \tag{2}$$

where $\bar{\boldsymbol{I}}_{i-1}$ is the 3×3 moment of inertia tensor for the link about the body-fixed coordinate system, m_{i-1} is its mass, and $\boldsymbol{s}_{i-1}$ is the vector from the origin of the link's coordinate system to its center of mass. In addition, $\mathbf{1}_3$ denotes the 3×3 identity matrix, and the tilde above a Cartesian vector signifies that its components should be combined in a skew symmetric

matrix such that $\tilde{\boldsymbol{b}}\boldsymbol{a} = \boldsymbol{b} \times \boldsymbol{a}$.

The spatial transformation matrix is used to transform spatial quantities to different coordinate systems and is defined as follows:

$$ {}^{i}\mathbf{X}_{i-1} = \begin{bmatrix} {}^{i}\boldsymbol{R}_{i-1} & \mathbf{0} \\ {}^{i}\boldsymbol{R}_{i-1}\, {}^{i-1}\tilde{\boldsymbol{p}}_{i}{}^{T} & {}^{i}\boldsymbol{R}_{i-1} \end{bmatrix}, \tag{3} $$

where ${}^{i}\boldsymbol{R}_{i-1}$ is the 3×3 rotation matrix from link $(i-1)$'s coordinate system to link i's coordinate system, and ${}^{i-1}\boldsymbol{p}_i$ is the Cartesian vector, expressed in link $(i-1)$'s coordinate system, from its origin to link i's.

In the last term of Eq. (1), $\mathbf{N}_i$ is called the reflected AB inertia. This reflected inertia is that portion of the ith AB inertia that is felt on the inboard side of joint i which is free to move. This is computed as follows [1]:

$$ \mathbf{N}_i = \mathbf{I}_i^* - \mathbf{I}_i^* \boldsymbol{\phi}_i \left(\boldsymbol{\phi}_i^T \mathbf{I}_i^* \boldsymbol{\phi}_i \right)^{-1} \boldsymbol{\phi}_i^T \mathbf{I}_i^*, \tag{4} $$

where $\boldsymbol{\phi}_i$ is the spatial joint axis vector which defines the motion in the joint. For revolute joints which allow *rotation* about the link's z-axis, $\boldsymbol{\phi}_i$ is equal to $[0\ 0\ 1\ \ 0\ 0\ 0]^T$. For prismatic joints with *translation* along the link's z-axis, $\boldsymbol{\phi}_i$ is equal to $[0\ 0\ 0\ \ 0\ 0\ 1]^T$.

Through a recursion that begins with $\mathbf{I}_N^* = \mathbf{I}_N$, $\mathbf{I}_{i-1}^*$ is the combination of link $i-1$'s inertia, $\mathbf{I}_{i-1}$, and $\mathbf{I}_i^*$. The latter is reflected across joint i (indicated by the vector, $\boldsymbol{\phi}_i$) in Eq. (4), and then transformed to link $(i-1)$'s coordinate system in Eq. (1) with a spatial (6×6) congruence transformation. While the significant portion of the computation comes from the transformation of the reflected inertia, understanding of the reflection itself is also important to the development of the efficient computation in this paper.

For revolute links, $\boldsymbol{\phi}_i = [0\ 0\ 1\ 0\ 0\ 0]^T$ is substituted into Eq. (4), leading to the following form for $\mathbf{N}_i$:

$$ \mathbf{N} = \begin{bmatrix} \boldsymbol{N}_{11} & \boldsymbol{N}_{12} \\ \boldsymbol{N}_{21} & \boldsymbol{N}_{22} \end{bmatrix} \tag{5} $$

$$= \left[\begin{array}{ccc|ccc} n_{11} & n_{12} & 0 & n_{14} & n_{15} & n_{16} \\ n_{12} & n_{22} & 0 & n_{24} & n_{25} & n_{26} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline n_{14} & n_{24} & 0 & n_{44} & n_{45} & n_{46} \\ n_{15} & n_{25} & 0 & n_{45} & n_{55} & n_{56} \\ n_{16} & n_{26} & 0 & n_{46} & n_{56} & n_{66} \end{array}\right], \tag{6}$$

where the subscript i, indicating the coordinate system, has been dropped from the notation for clarity, and the (jk)th element is defined as follows:

$$n_{jk} = [\mathbf{I}^*]_{jk} - \frac{[\mathbf{I}^*]_{j3}\,[\mathbf{I}^*]_{3k}}{[\mathbf{I}^*]_{33}}. \tag{7}$$

The resulting matrix is symmetric (notice the indices); therefore, only the elements in the upper (or lower) triangular portion of the matrix need to be computed. Furthermore, the third row and column are zero when the joint is revolute. The physical explanation for this is that inertia is not felt on the inboard side of the joint along the spatial dimension for which the joint is free to move. As a result, only 15 non-zero elements must be computed. The most efficient implementation of this step requires only [20M,15A]*. For prismatic joints, the sixth row and column of $\mathbf{N}_i$ are zero resulting in the same amount of computation.

B. MDH Parameters and the Transformation Matrix

The definition of the spatial transformation matrix, ${}^{i}\mathbf{X}_{i-1}$, is determined by the method used to describe the kinematics. Since the amount of computation in the congruence transformation operation in Eq. (1) is closely tied to this, finding the best method is the key to achieving a very efficient procedure. For the vast majority of cases, links in robotic systems are attached to one another with single degree-of-freedom revolute or prismatic joints. For these, it is especially efficient to describe the transformations between links using a modified form of the Denavit-Hartenberg notation [6]. Details of the resulting modified Denavit-Hartenberg (MDH) parameters can be found in [7] (except that α_i and a_i in this paper corresponds to α_{i-1} and a_{i-1} in the book).

*This is shorthand for 20 multiplications/divisions and 15 additions/subtractions.

Using these MDH parameters to model the kinematics, transformations can be represented as two successive axial (z- and x-axis) screw operations.[†] With this approach, the rotation matrices are maintained as two successive planar rotations about the z- and x-axes as follows:

$$
\begin{aligned}
{}^{i}\boldsymbol{R}_{i-1} &= \boldsymbol{R}_{z,\theta_i}\boldsymbol{R}_{x,\alpha_i} && (8)\\
&= \begin{bmatrix} \mathrm{c}\theta_i & \mathrm{s}\theta_i & 0 \\ -\mathrm{s}\theta_i & \mathrm{c}\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}\alpha_i & \mathrm{s}\alpha_i \\ 0 & -\mathrm{s}\alpha_i & \mathrm{c}\alpha_i \end{bmatrix}, && (9)
\end{aligned}
$$

where α_i is commonly referred to as the link twist angle and θ_i is the joint angle, and where each is an MDH parameter. Also note that $s\alpha_i$, $c\alpha_i$, $s\theta_i$, and $c\theta_i$ correspond to $\sin\alpha_i$, $\cos\alpha_i$, $\sin\theta_i$, and $\cos\theta_i$, respectively.

As a result, the spatial transformation matrix can be expressed as successive axial screw transformations about the z- and x-axes as follows:

$$
\begin{aligned}
{}^{i}\mathbf{X}_{i-1} &= \mathbf{X}_{z,d_i,\theta_i}\,\mathbf{X}_{x,a_i,\alpha_i}, && (10)\\
&= \begin{bmatrix} \boldsymbol{R}_{z,\theta_i} & \mathbf{0} \\ \boldsymbol{R}_{z,\theta_i}\,\tilde{\boldsymbol{p}}_z^T & \boldsymbol{R}_{z,\theta_i} \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_{x,\alpha_i} & \mathbf{0} \\ \boldsymbol{R}_{x,\alpha_i}\,\tilde{\boldsymbol{p}}_x^T & \boldsymbol{R}_{x,\alpha_i} \end{bmatrix}, && (11)
\end{aligned}
$$

where $\boldsymbol{p}_z = [0\ 0\ d_i]^T$ and $\boldsymbol{p}_x = [a_i\ 0\ 0]^T$. Note that a_i and d_i are the other two MDH parameters, called the link length and joint offset, respectively.

3. Efficient Congruence Transformation: Revolute Joints

By far, the most computationally expensive step in the procedure outlined in the previous section is the congruence transformation of the reflected inertia. This section develops the efficient procedure for this computation specifically. Also because of its prevalence and use in the literature to compare relative performance between algorithms, the case of the transformation across a revolute joint is examined in detail.

[†] An *axial screw* represents a rotation and translation about and along a single coordinate axis.

The idea for the procedure comes from the method by which vectors are efficiently transformed. Featherstone showed that a very efficient method is achieved for the spatial transformation of a vector when the axial screw transformations of Eq. (10) are applied to the vector in succession [8]. That is, the vector is multiplied by $\mathbf{X}_{x,a_i,\alpha_i}$ and the result is multiplied by $\mathbf{X}_{z,d_i,\theta_i}$ to achieve the final result. This paper presents an analogous approach for congruence transformations of matrices to achieve an efficient procedure. That is, *two* axial screw congruence transformations are applied to the reflected inertia matrix in succession.

Continuing the use of the 3×3 block notation of Eq. (5), both axial screw transformations can be written the same way as follows:

$$\mathbf{X}^T\mathbf{N}\mathbf{X} = \begin{bmatrix} \boldsymbol{R}^T & \tilde{\boldsymbol{p}}\boldsymbol{R}^T \\ \mathbf{0} & \boldsymbol{R}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{N}_{11} & \boldsymbol{N}_{12} \\ \boldsymbol{N}_{21} & \boldsymbol{N}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \mathbf{0} \\ \boldsymbol{R}\tilde{\boldsymbol{p}}^T & \boldsymbol{R} \end{bmatrix} \tag{12}$$

$$= \begin{bmatrix} \boldsymbol{R}^T\boldsymbol{N}_{11}\boldsymbol{R} + \tilde{\boldsymbol{p}}\boldsymbol{R}^T\boldsymbol{N}_{22}\boldsymbol{R}\tilde{\boldsymbol{p}}^T + \boldsymbol{R}^T\boldsymbol{N}_{12}\boldsymbol{R}\tilde{\boldsymbol{p}}^T + \left(\boldsymbol{R}^T\boldsymbol{N}_{12}\boldsymbol{R}\tilde{\boldsymbol{p}}^T\right)^T & \boldsymbol{R}^T\boldsymbol{N}_{12}\boldsymbol{R} + \tilde{\boldsymbol{p}}\boldsymbol{R}^T\boldsymbol{N}_{22}\boldsymbol{R} \\ \left(\boldsymbol{R}^T\boldsymbol{N}_{12}\boldsymbol{R} + \tilde{\boldsymbol{p}}\boldsymbol{R}^T\mathbf{N}_{22}\boldsymbol{R}\right)^T & \boldsymbol{R}^T\boldsymbol{N}_{22}\boldsymbol{R} \end{bmatrix}, \tag{13}$$

where the symmetry in $\mathbf{N}$ (i.e., $\boldsymbol{N}_{21} = \boldsymbol{N}_{12}^T$) has been used to simplify its form, and the subscripts denoting z- or x-axial congruence transformations have been omitted for generality. The result of the z-axial congruence transformation is used in the x-axial transformation which has exactly the same form. Table 1 lists the nine steps that are identified in each axial screw congruence transformation which leads to a particularly efficient procedure.

This approach is different from previous approaches for transformation of spatial (and Composite Rigid-Body) inertia matrices by Lilly and Orin [4] and reflected AB inertia matrices in both the first draft of this paper and the work of Brandl, Johanni, and Otter [2]. In those, the full spatial transformation matrix from Eq. (3) is used in Eq. (13) (it has the same form). Then only successive planar *rotations* were applied to the terms of the form, $\boldsymbol{R}^T\boldsymbol{N}\boldsymbol{R}$, to achieve a certain level of efficiency. At first this seems more efficient since it results in only one set of nine steps from Table 1. However, it leads to more computation because ${}^i\boldsymbol{R}_{i-1}$ and ${}^{i-1}\boldsymbol{p}_i$ are much more complex than that of either $\boldsymbol{R}_{z,\theta_i}$ and $\boldsymbol{p}_z$ or $\boldsymbol{R}_{x,\alpha_i}$ and $\boldsymbol{p}_x$ in the successive axial screw approach developed in this paper.

Table 1: On-line computational cost of the congruence transformation across a revolute joint.

		z-axis		x-axis	
Step	Computation	$\times$	$+$	$\times$	$+$
A.	Trigonometric quantities	2	3	0	0
B.	$\boldsymbol{R}^T \boldsymbol{N}_{11} \boldsymbol{R}$	4	5	4	1
C.	$[\mathbf{X}^T \mathbf{N} \mathbf{X}]_{22} = \boldsymbol{R}^T \boldsymbol{N}_{22} \boldsymbol{R}$	8	7	8	7
D.	$\boldsymbol{R}^T \boldsymbol{N}_{12} \boldsymbol{R}$	8	10	10	6
E.	$\tilde{\boldsymbol{p}}(\boldsymbol{R}^T \boldsymbol{N}_{22} \boldsymbol{R})$	5	0	5	0
F.	$(\tilde{\boldsymbol{p}} \boldsymbol{R}^T \boldsymbol{N}_{22} \boldsymbol{R}) \tilde{\boldsymbol{p}}^T$	3	0	3	0
G.	$(\boldsymbol{R}^T \boldsymbol{N}_{12} \boldsymbol{R}) \tilde{\boldsymbol{p}}^T$	4	0	6	0
H.	$\left[\mathbf{X}^T \mathbf{N} \mathbf{X}\right]_{11} = (B) + (F) + (G) + (G)^T$	0	9	0	11
I.	$\left[\mathbf{X}^T \mathbf{N} \mathbf{X}\right]_{12} = \left[\mathbf{X}^T \mathbf{N} \mathbf{X}\right]_{21}^T = (D) + (E)$	0	6	0	6
	Total:	34	40	36	31

A. z-Axial Screw Congruence Transformation

The following discussion presents the first half of the computation, the congruence transformation of the reflected inertia matrix using the z-axial screw. Each of the nine steps is presented separately below and the results are tabulated in the middle columns of Table 1. Note that the final result of this half, $\mathbf{N}^z$, requires a total of [34M, 40A] to compute and has the same zero row and column of the original reflected inertia. This will further reduce the computational requirements of the second half of the congruence transformation, using the x-axial screw, to follow.

Step A_z: Trigonometric quantities. This step includes the computation of trigonometric quantities that will be required in more than one of the subsequent steps of this procedure. By computing all such quantities once, at the beginning, some redundant multiplications and additions are eliminated. For the z-axial screw congruence transformation, these involve functions of the revolute joint angle variable, θ: $\mathrm{s}^2\theta = \sin\theta \sin\theta$, $\mathrm{c}^2\theta - \mathrm{s}^2\theta = 1 - \mathrm{s}^2\theta - \mathrm{s}^2\theta$, $\mathrm{s}\theta\mathrm{c}\theta = \sin\theta\cos\theta$, and $2\mathrm{s}\theta\mathrm{c}\theta = \mathrm{s}\theta\mathrm{c}\theta + \mathrm{s}\theta\mathrm{c}\theta$, where addition is opted for over multiplication. A total of [2M, 3A] are required to compute the desired quantities.

Also, note that only one sine/cosine pair must be computed on-line for each revolute joint.

Step B_z: $\boldsymbol{R}_{z,\theta}^T \boldsymbol{N}_{11} \boldsymbol{R}_{z,\theta}$. This step requires a 3×3 congruence transformation of a symmetric matrix with a zero row and column using the z-planar rotation matrix. After multiplying these matrices together, certain terms are identified in different elements of the result which leads to the following matrix:

$$\boldsymbol{N}^B = \begin{bmatrix} n_{11}+k_1 & k_2 & 0 \\ k_2 & n_{22}-k_1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{14}$$

where

$$k_1 = (n_{22}-n_{11})\mathrm{s}^2\theta - n_{12}(2\mathrm{s}\theta\mathrm{c}\theta), \tag{15}$$

$$k_2 = n_{12}(\mathrm{c}^2\theta - \mathrm{s}^2\theta) - (n_{22}-n_{11})(\mathrm{s}\theta\mathrm{c}\theta). \tag{16}$$

Using the results of Step A_z, [4M, 3A] are required to compute k_1 and k_2. With an additional [0M, 2A] to compute the (1,1) and (2,2) elements of this matrix, this step requires a total of [4M, 5A]. Note that the resulting matrix is symmetric, and the z-planar congruence transformation has preserved the zero row and column.

Step C_z: $\boldsymbol{R}_{z,\theta}^T \boldsymbol{N}_{22} \boldsymbol{R}_{z,\theta}$. As in the previous, this step requires a 3×3 congruence transformation of a symmetric matrix using the z-planar rotation matrix, but the difference in this step is that the matrix is full, in general. This yields the following result:

$$\boldsymbol{N}^C = \boldsymbol{N}_{22}^z = \begin{bmatrix} n_{44}+k_1 & k_2 & n_{46}\mathrm{c}\theta - n_{56}\mathrm{s}\theta \\ k_2 & n_{55}-k_1 & n_{46}\mathrm{s}\theta + n_{56}\mathrm{c}\theta \\ n_{46}\mathrm{c}\theta - n_{56}\mathrm{s}\theta & n_{46}\mathrm{s}\theta + n_{56}\mathrm{c}\theta & n_{66} \end{bmatrix}, \tag{17}$$

where

$$k_1 = (n_{55}-n_{44})\mathrm{s}^2\theta - n_{45}(2\mathrm{s}\theta\mathrm{c}\theta) \tag{18}$$

$$k_2 = n_{45}(\mathrm{c}^2\theta - \mathrm{s}^2\theta) - (n_{55}-n_{44})(\mathrm{s}\theta\mathrm{c}\theta). \tag{19}$$

These two terms have the same form as in Step B_z and also require [4M, 3A] to compute. An additional [0M, 2A] are needed to compute the (1,1) and (2,2) elements, and [4M, 2A]

are needed to compute the (1,3) and (2,3) elements for a total of [8M, 7A]. Computation of the other elements are unnecessary due to the symmetry of the result.

Step D_z: $\boldsymbol{R}_{z,\theta}^T \boldsymbol{N}_{12} \boldsymbol{R}_{z,\theta}$. This step requires a 3×3 congruence transformation of a non-symmetric matrix with a zero row using the z-planar rotation matrix. This results in the following matrix:

$$\boldsymbol{N}^D = \begin{bmatrix} n_{14}+k_1 & n_{15}-k_2 & n_{16}\mathrm{c}\theta - n_{26}\mathrm{s}\theta \\ n_{24}-k_2 & n_{25}-k_1 & n_{16}\mathrm{s}\theta + n_{26}\mathrm{c}\theta \\ 0 & 0 & 0 \end{bmatrix}, \tag{20}$$

where

$$k_1 = (n_{25}-n_{14})\mathrm{s}^2\theta - (n_{15}+n_{24})(\mathrm{s}\theta\mathrm{c}\theta) \tag{21}$$

$$k_2 = (n_{25}-n_{14})(\mathrm{s}\theta\mathrm{c}\theta) + (n_{15}+n_{24})\mathrm{s}^2\theta. \tag{22}$$

A total of [8M, 10A] is required to compute this matrix: [4M, 4A] for k_1 and k_2; [0M, 4A] for elements (1,1), (1,2), (2,1) and (2,2); and [4M, 2A] for elements (1,3) and (2,3). The result still has a zero row.

Step E_z: $\tilde{\boldsymbol{p}}_z(\boldsymbol{R}_{z,\theta}^T \boldsymbol{N}_{22} \boldsymbol{R}_{z,\theta})$. This step uses the results of Step C_z [n_{jk}^C represents the (jk)th element of $\boldsymbol{N}^C$ in Eq. (17)] and the definition for $\boldsymbol{p}_z$ from Eq. (11). This leads to the following matrix multiplication:

$$\begin{bmatrix} 0 & -d & 0 \\ d & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} n_{11}^C & n_{12}^C & n_{13}^C \\ n_{12}^C & n_{22}^C & n_{23}^C \\ n_{13}^C & n_{23}^C & n_{33}^C \end{bmatrix} = \begin{bmatrix} -n_{12}^C d & -n_{22}^C d & -n_{23}^C d \\ n_{11}^C d & n_{12}^C d & n_{13}^C d \\ 0 & 0 & 0 \end{bmatrix}. \tag{23}$$

Since elements (1,1) and (2,2) are the same (except for sign), this step only requires [5M, 0A] to compute. Also note that the sparse structure of $\tilde{\boldsymbol{p}}_z$ is a direct result of using axial screws and leads to a significant reduction in computation over previous approaches.

Step F_z: $(\tilde{\boldsymbol{p}}_z \boldsymbol{R}_{z,\theta}^T \boldsymbol{N}_{22} \boldsymbol{R}_{z,\theta}) \tilde{\boldsymbol{p}}_z^T$. Using the result of Eq. (23) in Step E_z, the following matrix

results:

$$\begin{bmatrix} n_{11}^E & n_{12}^E & n_{13}^E \\ n_{21}^E & n_{22}^E & n_{23}^E \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & d & 0 \\ -d & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -n_{12}^E d & n_{11}^E d & 0 \\ -n_{22}^E d & n_{21}^E d & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{24}$$

Since $-n_{22}^E d = n_{11}^E d$, the result is symmetric and only [3M, 0A] are required.

Step G_z: $(\boldsymbol{R}_{z,\theta}^T \boldsymbol{N}_{12} \boldsymbol{R}_{z,\theta})\tilde{\boldsymbol{p}}_z^T$**.** Using the result of Eq. (20) in Step D_z, the following non-symmetric matrix results:

$$\begin{bmatrix} n_{11}^D & n_{12}^D & n_{13}^D \\ n_{21}^D & n_{22}^D & n_{23}^D \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & d & 0 \\ -d & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -n_{12}^D d & n_{11}^D d & 0 \\ -n_{22}^D d & n_{21}^D d & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{25}$$

requiring [4M, 0A].

Step H_z: $\left[\mathbf{X}_{z,d,\theta}^T \, \mathbf{N} \, \mathbf{X}_{z,d,\theta}\right]_{11}$**.** This step completes the computation required for the upper left block of Eq. (13). To accomplish this, the results of Steps B_z, F_z, and G_z must be added together as shown in Table 1: $B_z + F_z + G_z + G_z^T$. The resulting matrix, $\boldsymbol{N}_{11}^z$, is symmetric with a zero row and column (the third); therefore, the three upper triangular elements of the result are needed. With 3 additions for each of the elements, the amount of computation required in this step is [0M, 9A].

Step I_z: $\left[\mathbf{X}_{z,d,\theta}^T \, \mathbf{N} \, \mathbf{X}_{z,d,\theta}\right]_{12}$**.** To complete the computation for the upper right block of Eq. (13) for the congruence transformation using the z-axial screw, the results of Steps D_z and E_z [Eqs. (20) and (23), respectively] are added together. Neither of these matrices nor the result, $\boldsymbol{N}_{12}^z$, is symmetric, but the third row is still zero; therefore, [0M, 6A] are required in this step.

B. x-Axial Screw Congruence Transformation

To complete the computation, a congruence transformation using the x-axial screw is applied to the 6×6 symmetric matrix, $\mathbf{N}^z$, resulting from the previous computation. The

result of this step is a full 6×6 symmetric matrix, ${}^{i-1}\mathbf{N}_i$, that requires [36M, 31A] as detailed in the following discussion.

Step A_x: Trigonometric quantities. The same quantities that were computed in the first step using θ must also be computed using α: $\mathrm{s}^2\alpha$, $\mathrm{c}^2\alpha - \mathrm{s}^2\alpha$, $\mathrm{s}\alpha\mathrm{c}\alpha$, and $2\mathrm{s}\alpha\mathrm{c}\alpha$. Since α is always constant these terms can be computed off-line.

Step B_x: $\boldsymbol{R}_{x,\alpha}^T \boldsymbol{N}_{11}^z \boldsymbol{R}_{x,\alpha}$. This step performs a 3×3 congruence transformation on $\boldsymbol{N}_{11}^z$, the result of Step H_z, using the x-planar rotation matrix. This leads to the following matrix:

$$\boldsymbol{R}_{x,\alpha}^T \begin{bmatrix} n_{11}^z & n_{12}^z & 0 \\ n_{12}^z & n_{22}^z & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{R}_{x,\alpha} = \begin{bmatrix} n_{11}^z & n_{12}^z\mathrm{c}\alpha & n_{12}^z\mathrm{s}\alpha \\ n_{12}^z\mathrm{c}\alpha & n_{22}^z - (n_{22}^z\mathrm{s}^2\alpha) & n_{22}^z(\mathrm{c}\alpha\mathrm{s}\alpha) \\ n_{12}^z\mathrm{s}\alpha & n_{22}^z(\mathrm{c}\alpha\mathrm{s}\alpha) & (n_{22}^z\mathrm{s}^2\alpha) \end{bmatrix}. \tag{26}$$

Using results from Step A_x, this step requires only [4M, 1A].

Step C_x: $\boldsymbol{R}_{x,\alpha}^T \boldsymbol{N}_{22}^z \boldsymbol{R}_{x,\alpha}$. This step performs a 3×3 congruence transformation on $\boldsymbol{N}_{22}^z$, the result of Step C_z, using the x-planar rotation matrix. The result is the lower right 3×3 block of the completed congruence transformation, $\left[{}^{i-1}\mathbf{N}_i\right]_{22}$, as follows:

$$\boldsymbol{N}^C = \left[{}^{i-1}\mathbf{N}_i\right]_{22} = \begin{bmatrix} n_{44}^z & n_{45}^z\mathrm{c}\alpha - n_{46}^z\mathrm{s}\alpha & n_{45}^z\mathrm{s}\alpha + n_{46}^z\mathrm{c}\alpha \\ n_{45}^z\mathrm{c}\alpha - n_{46}^z\mathrm{s}\alpha & n_{55}^z + k_1 & k_2 \\ n_{45}^z\mathrm{s}\alpha + n_{46}^z\mathrm{c}\alpha & k_2 & n_{66}^z - k_1 \end{bmatrix}, \tag{27}$$

where n_{jk}^z is the jkth element of $\mathbf{N}^z$, the final result of the z-axial screw congruence transformation, and

$$k_1 = (n_{66}^z - n_{55}^z)\mathrm{s}^2\alpha - n_{56}^z(2\mathrm{s}\alpha\mathrm{c}\alpha) \tag{28}$$

$$k_2 = n_{56}^z(\mathrm{c}^2\alpha - \mathrm{s}^2\alpha) - (n_{66}^z - n_{55}^z)(\mathrm{s}\alpha\mathrm{c}\alpha). \tag{29}$$

Computing k_1 and k_2 requires [4M, 3A], [0M, 2A] are required to compute the (2,2) and (3,3) elements, and [4M, 2A] are required to compute the remaining elements, for a total of [8M, 7A].

Step D_x: $\boldsymbol{R}_{x,\alpha}^T \boldsymbol{N}_{12}^z \boldsymbol{R}_{x,\alpha}$. This step performs a 3×3 congruence transformation on $\boldsymbol{N}_{12}^z$, the result of Step I_z, using the x-planar rotation matrix. The third row of $\boldsymbol{N}_{12}^z$ is zero

which results in the following matrix:

$$\boldsymbol{N}^D \;=\; \boldsymbol{R}_{x,\alpha}^T\, \boldsymbol{N}_{12}^z \boldsymbol{R}_{x,\alpha} \;=\; \begin{bmatrix} n_{14}^z & n_{15}^z \mathrm{c}\alpha - n_{16}^z \mathrm{s}\alpha & n_{15}^z \mathrm{s}\alpha + n_{16}^z \mathrm{c}\alpha \\ n_{24}^z \mathrm{c}\alpha & n_{25}^z - k_1 & n_{26}^z + k_2 \\ n_{24}^z \mathrm{s}\alpha & k_2 & k_1 \end{bmatrix}, \tag{30}$$

where

$$k_1 \;=\; n_{25}^z \mathrm{s}^2\alpha + n_{26}^z (\mathrm{s}\alpha \mathrm{c}\alpha) \tag{31}$$

$$k_2 \;=\; n_{25}^z (\mathrm{s}\alpha \mathrm{c}\alpha) - n_{26}^z \mathrm{s}^2\alpha. \tag{32}$$

The computation of k_1 and k_2 requires [4M, 2A], and the remaining elements of Eq. (30) require [6M, 4A], for a total of [10M, 6A].

Step E_x: $\tilde{\boldsymbol{p}}_x(\boldsymbol{R}_{x,\alpha}^T \boldsymbol{N}_{22}^z \boldsymbol{R}_{x,\alpha})$. This uses the result of Eq. (27) in Step C_x and the definition for $\boldsymbol{p}_x$ from Eq. (11):

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -a \\ 0 & a & 0 \end{bmatrix} \begin{bmatrix} n_{11}^C & n_{12}^C & n_{13}^C \\ n_{12}^C & n_{22}^C & n_{23}^C \\ n_{13}^C & n_{23}^C & n_{33}^C \end{bmatrix} \;=\; \begin{bmatrix} 0 & 0 & 0 \\ -n_{13}^C a & -n_{23}^C a & -n_{33}^C a \\ n_{12}^C a & n_{22}^C a & n_{23}^C a \end{bmatrix}. \tag{33}$$

Since elements (2,2) and (3,3) of this result are the same (within a sign), this step only requires [5M, 0A] to compute.

Step F_x: $(\tilde{\boldsymbol{p}}_x \boldsymbol{R}_{x,\alpha}^T \boldsymbol{N}_{22}^z \boldsymbol{R}_{x,\alpha}) \tilde{\boldsymbol{p}}_x^T$. Using the result of Eq. (33) in Step E_x, the following matrix results:

$$\begin{bmatrix} 0 & 0 & 0 \\ n_{21}^E & n_{22}^E & n_{23}^E \\ n_{31}^E & n_{32}^E & n_{33}^E \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & a \\ 0 & -a & 0 \end{bmatrix} \;=\; \begin{bmatrix} 0 & 0 & 0 \\ 0 & -n_{23}^E a & n_{22}^E a \\ 0 & -n_{33}^E a & n_{32}^E a \end{bmatrix}. \tag{34}$$

This result is symmetric since $-n_{33}^E a = n_{22}^E a$, so this step requires [3M, 0A].

Step G_x: $(\boldsymbol{R}_{x,\alpha}^T \boldsymbol{N}_{12}^z \boldsymbol{R}_{x,\alpha}) \tilde{\boldsymbol{p}}_x^T$. Using the result of Eq. (30) in Step D_x, the following

nonsymmetric matrix results from this step:

$$\begin{bmatrix} n_{11}^D & n_{12}^D & n_{13}^D \\ n_{21}^D & n_{22}^D & n_{23}^D \\ n_{31}^D & n_{32}^D & n_{33}^D \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & a \\ 0 & -a & 0 \end{bmatrix} = \begin{bmatrix} 0 & -n_{13}^D a & n_{12}^D a \\ 0 & -n_{23}^D a & n_{22}^D a \\ 0 & -n_{33}^D a & n_{32}^D a \end{bmatrix}. \tag{35}$$

This step requires [6M, 0A].

Step H_x: $\left[\mathbf{X}_{x,a,\alpha}^T \, \mathbf{N}^z \, \mathbf{X}_{x,a,\alpha}\right]_{11}$**.** This step finishes the computation required for the upper left 3×3 block of the complete congruence transformation. To accomplish this, the results of Steps B_x, F_x, and G_x must be added together as shown in Table 1. The resulting matrix is symmetric; therefore, only the six upper triangular elements of the result are needed. Because some elements to be added are zero, the amount of computation required in this step is [0M, 11A].

Step I_x: $\left[\mathbf{X}_{x,a,\alpha}^T \, \mathbf{N}^z \, \mathbf{X}_{x,a,\alpha}\right]_{12}$**.** To finish the computation for the upper right block of the complete congruence transformation, the results of Eqs. (30) and (33) [Steps D_x and E_x, respectively] are added together. Neither of these matrices nor the result is symmetric, but the result of Eq. (33) has three zero elements. Therefore [0M, 6A] are required in this step.

4. Efficient Congruence Transformation: Prismatic Joints

For transformations across prismatic joints, the sixth row and column (instead of the third) of the reflected inertia matrix are zero. Therefore, the computation for the congruence transformation of this matrix differs only slightly from the results of the procedure above for a revolute joint. Since the joint and link twist angles, θ and α, are constant, all of the computation for Step A_z can be performed off-line. The requirements for Steps B and C are switched; Step D requires the same amount of computation, but is redefined for the transposed structure of the upper right-hand block; Step E_z requires two fewer multiplications while Step G_z requires two more; and Step H_z requires two more additions while Step I_z requires two fewer. The total computational requirement is [68M, 68A] which is [2M, 3A] less than the [70M, 71A] required for a revolute joint.

5. Summary and Conclusions

This paper has developed a new approach to computing AB inertias in articulated mechanisms with revolute and prismatic joints. This involves three steps: (1) computation of the reflected inertia $\mathbf{N}_i$ [in Eq. (4)] which requires [20M, 15A], (2) transformation of this reflected inertia (the focus of this paper), and (3) addition of a link's spatial inertia [in Eq. (1)] requiring [0M, 15A]. Using modified Denavit-Hartenberg parameters, and applying two successive axial screw transformations (analogous to the efficient procedure for transforming vectors in [8]) has resulted in a very efficient approach to the second step which requires [70M, 71A] for revolute joints and [68M, 68A] for prismatic joints. This represents more than a 20% improvement over the results of the first draft of this paper, [89M, 90A], which only used successive planar rotations instead.

Using these results, the most efficient AB algorithm for simulating a serial chain with a fixed base and N revolute joints has been developed in [9]. The resulting algorithm requires [$(224N$-259)M, $(205N$-248)A] to compute the desired joint accelerations. This represents about 15% fewer operations than in the algorithm developed by Brandl, Johanni, and Otter [2]. Note that despite the very efficient procedure developed in this paper, the computation of the AB inertias still accounts for nearly half of the total computation in the AB simulation algorithm.

This approach has also been used to achieve efficient transformations of other types of matrices. A summary of the results from various sources [8, 10] are listed in Table 2. In particular, note that the result for the transformation of a spatial inertia matrix improves on one developed in [4] by about 30%. This leads to a significant improvement in the competing approach to dynamic simulation using the Composite Rigid-Body (CRB) method [3].

Another advantage of using this approach is its modularity. If either a and α are both zero or d and θ are both zero, half of the computation can be easily omitted by simply skipping the corresponding axial screw congruence transformation step. This also occurs when using Khalil and Kleinfinger's guidelines for efficient kinematic modeling of tree-structured mechanisms [11]. When a branch occurs in the tree, a constant z-axial transformation is in-

cluded in the model (in addition to MDH parameters). In the AB simulation algorithm, this adds a z-axial congruence transformation to the routine computing the AB inertias. Using the approach outlined here, this will only incur an additional [32M, 37A] (θ is constant).

These approaches to modeling tree-structured mechanisms and efficient transformations have both been implemented in a simulator developed for an underwater walking vehicle [12]. The software simulation library, called DynaMechs, is available via anonymous FTP at the following URL: `ftp://cs.nps.navy.mil/pub/dynamechs`. The efficiency gained by using this approach has resulted in dynamic simulations that execute in real time on SGI workstations with the MIPS R4x00 processor.

6. References

[1] R. Featherstone, "The Calculation of Robot Dynamics Using Articulated-Body Inertias," *The Int. Journal of Robotics Research,* The MIT Press, vol. 2, pp. 13–30, Spring 1983.

[2] H. Brandl, R. Johanni, and M. Otter, "A Very Efficient Algorithm for the Simulation of Robots and Similar Multibody Systems Without Inversion of the Mass Matrix," in *Proc. of IFAC/IFIP/IMACS Int. Symp. on Theory of Robots*, (Vienna, Austria), December 1986.

[3] M. W. Walker and D. E. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, pp. 205–211, September 1982.

Table 2: Summary of efficient transformation operations across revolute joints using successive axial screws.

	Trig.	z-axis	x-axis	Total
Cartesian Vectors [8]		[4M, 2A]	[4M, 2A]	[8M, 4A]
Spatial Vectors [8]		[10M, 6A]	[10M, 6A]	[20M, 12A]
General 3×3 [10]	[2M, 0A]	[12M, 12A]	[12M, 12A]	[26M, 24A]
Symmetric 3×3 [10]	[2M, 3A]	[8M, 7A]	[8M, 7A]	[18M, 17A]
Symmetric 6×6 [10]	[2M, 3A]	[42M, 43A]	[42M, 43A]	[86M, 89A]
Spatial/CRB Inertia	[2M, 3A]	[15M, 16A]	[15M, 16A]	[32M, 35A]
Reflected AB Inertia	[2M, 3A]	[32M, 37A]	[36M, 31A]	[70M, 71A]

[4] K. W. Lilly and D. E. Orin, "Alternate Formulations for the Manipulator Inertia Matrix," *International Journal of Robotics Research,* The MIT Press, vol. 10, pp. 64–74, February 1991.

[5] G. Rodriguez and K. Kreutz-Delgado, "Spatial Operator Factorization and Inversion of the Manipulator Mass Matrix," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 65–76, February 1992.

[6] J. Denavit and R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *ASME Journal of Applied Mathematics*, pp. 215–221, June 1955.

[7] J. J. Craig, *Introduction to Robotics: Mechanics and Control.* Reading, MA: Addison-Wesley, 1986.

[8] R. Featherstone, *Robot Dynamics Algorithms.* Boston: Kluwer Academic Publishers, 1987.

[9] S. McMillan, D. E. Orin, and R. B. McGhee, "Efficient Dynamic Simulation of an Underwater Vehicle with a Robotic Manipulator," To appear in *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 25, August 1995.

[10] S. McMillan, *Computational Dynamics for Robotic Systems on Land and Under Water.* PhD dissertation, The Ohio State University, Columbus, Ohio, Summer 1994.

[11] W. Khalil and J. F. Kleinfinger, "A New Geometric Notation for Open and Closed-Loop Robots," in *IEEE Int. Conf. on Robotics and Automation*, (San Francisco, CA), pp. 1174–1179, 1986.

[12] S. McMillan, D. E. Orin, and R. B. McGhee, "DynaMechs: An Object Oriented Software Package for Efficient Dynamic Simulation of Underwater Robotic Vehicles," in *Underwater Robotic Vehicles: Design and Control* (J. Yuh, ed.), pp. 73–98, TSI Press, 1995.