Engineering Notes

ENGINEERING NOTES are short manuscripts describing new developments or important results of a preliminary nature. These Notes cannot exceed 6 manuscript pages and 3 figures; a page of text may be substituted for a figure and vice versa. After informal review by the editors, they may be published within a few months of the date of receipt. Style requirements are the same as for regular contributions (see inside back cover).

Triangularization of Equations of Motion for Robotic Systems

Dan E. Rosenthal*
Symbolic Dynamics, Mountain View, California

Introduction

IN this Note, two main points are made: 1) A new algorithm, which constructs the equations of motion for robotic systems in $O(n^2)$ operations, is derived. 2) The use of computer symbol manipulation[1,2] to implement the algorithm is shown to provide vast reduction in the final operation counts.

The new algorithm develops the equations in a highly uncoupled triangular form, so that no matrix decomposition is required to solve for the joint accelerations. The methods used here are based on Kane's equations, which we have found to be highly useful for investigating multibody dynamics.

A generic chain system is indicated in Fig. 1. The bodies move in an inertial frame N. We assume that each body is attached to its lower neighbor via a revolute joint. The bodies are assigned the labels 1 to n, as are the joints. The case of more complicated joints can be accommodated by the method derived here, but the details are shown only for revolute joints. Each joint connects two bodies, labeled i and k. The vector from the mass center of i to the mass center of k is $\boldsymbol{r}^{ik}$. It is made up of two vectors, one fixed on i, the other fixed on k. These vectors are called $\boldsymbol{ri}_k$ and $\boldsymbol{rk}_k$. The first is the vector from the mass center of i to the joint, and the second from the mass center of k to the joint.

We introduce generalized u_k, $k = 1, n$ as follows:

$$u_k = {}^i\boldsymbol{\omega}^k \cdot \boldsymbol{\lambda}_k \tag{1}$$

where $\boldsymbol{\lambda}_k$ is a unit vector in the direction of the lower joint of body k. Here $u_k\boldsymbol{\lambda}_k$ is the angular velocity vector of body k in its lower body, body i.

With these preliminaries, we are ready to proceed. In what follows, three methods are presented. The first is the "basic method," which requires $O(n^3)$ operations to compute the equations of motion, followed by Choleski decomposition to solve the equations for the joint accelerations. The next method reduces the cost of computing the equations to $O(n^2)$ but still requires Choleski decomposition to solve for the accelerations. The final method provides a means to eliminate the matrix decomposition by forming the equations so that a triangular mass matrix rather than a square mass matrix is produced.

A certain amount of kinematic analysis must precede formulation of the dynamical equations. This analysis follows.

Kinematical Analysis

1) Form partial angular velocity vectors $\boldsymbol{\omega}_r^k$ and partial velocity vectors $\boldsymbol{v}_r^k$:

$$\boldsymbol{\omega}_r^k = \begin{cases} \boldsymbol{\lambda}_r & \text{for } r \le k \\ 0 & \text{for } k < r \end{cases} \tag{2}$$

The partial velocity vectors are computed recursively. First, $\boldsymbol{v}_r^r$ is computed, followed by $\boldsymbol{v}_r^k$ for $k > r$. The index of the inboard body of body k is i.

$$\boldsymbol{v}_r^k = \begin{cases} -\boldsymbol{\lambda}_k \times \boldsymbol{rk}_k & \text{for } r = k \\ \boldsymbol{v}_r^i + \boldsymbol{\omega}_r^i \times \boldsymbol{ri}_k - \boldsymbol{\omega}_r^k \times \boldsymbol{rk}_k & \text{for } k > r \\ 0 & \text{for } k < r \end{cases} \tag{3}$$

Again, $\boldsymbol{ri}_k$ is the vector from the mass center of body i to the inboard hinge of body k. $\boldsymbol{rk}_k$ is the vector from the mass center of body k to its inboard hinge point.

2) Form angular velocity vectors $\boldsymbol{\omega}^k$:

$$\boldsymbol{\omega}^k = \boldsymbol{\omega}^i + \boldsymbol{\omega}_k^k u_k \quad k = 1,n \tag{4}$$

3) Form acceleration remainder terms $\boldsymbol{\alpha}_t^k$ and $\boldsymbol{\alpha}_t^k$. The angular acceleration vector $\boldsymbol{\alpha}^k$ can be expressed

$$\boldsymbol{\alpha}^k = \sum_{r=1}^{k} \boldsymbol{\omega}_r^k \dot{u}_r + \boldsymbol{\alpha}_t^k \tag{5}$$

where the vector $\boldsymbol{\alpha}_t^k$ is called the angular acceleration remainder for body k. Equation (5) follows from the fact that the angular velocity vector $\boldsymbol{\omega}^k$ can be written

$$\boldsymbol{\omega}^k = \sum_{r=1}^{k} \boldsymbol{\omega}_r^k u_r \tag{6}$$

Differentiating Eq. (6), we find that

$$\boldsymbol{\alpha}^k = \sum_{r=1}^{k} \boldsymbol{\omega}_r^k \dot{u}_r + \sum_{r=1}^{k} u_r^N \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\omega}_r^k \tag{7}$$

so that we may identify

$$\boldsymbol{\alpha}_t^k = \sum_{r=1}^{k} u_r^N \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\omega}_r^k \tag{8}$$

The vector $\boldsymbol{\alpha}_t^k$ may be computed recursively by differentiating Eq. (4) to find

$$\boldsymbol{\alpha}^k = \boldsymbol{\alpha}^i + \boldsymbol{\omega}_k^k \dot{u}_k + u_k \boldsymbol{\omega}^k \times \boldsymbol{\omega}_k^k \tag{9}$$

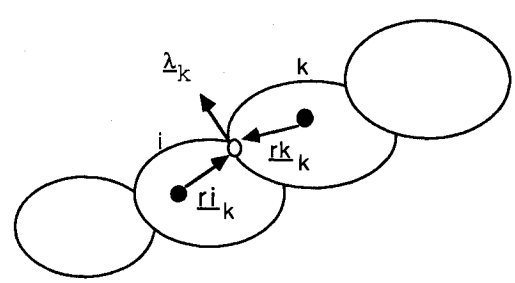

Fig. 1 Illustration of adjacent body connections.

Received Dec. 2, 1986; revision received March 3, 1987.

*Senior Scientist.

Substituting Eq. (5) into Eq. (9), we find that

$$\alpha_t^k = \alpha_t^i + u_k \omega_k \times \omega_k^k \tag{10}$$

The linear acceleration vector a^k may be expressed

$$a^k = \sum_{r=1}^{k} v_r^k \dot{u}_r + a_t^k \tag{11}$$

where the remainder term a_t^k is given by the formula

$$a_t^k = a_t^i + \alpha_t^i \times r i_k + \omega^i \times \omega^i \times r i_k - \alpha_t^k \times r k_k - \omega^k \times \omega^k \times r k_k \tag{12}$$

Equation (12) follows from the expression for the linear velocity of the mass center of body k,

$$v^k = v^i + \omega^i \times r i_k - \omega^k \times r k_k \tag{13}$$

Differentiating Eq. (13) yields

$$a^k = a^i + \alpha^i \times r i_k + \omega^i \times \omega^i \times r i_k - \alpha^k \times r k_k - \omega^k \times \omega^k \times r k_k \tag{14}$$

The motivation for Eq. (11) comes from the fact that the vector v^k may be written

$$v^k = \sum_{r=1}^{k} v_r^k u_r \tag{15}$$

and differentiating Eq. (15), making use of Eqs. (10) and (11), comparing to Eq. (14) leads to Eq. (12).

The remainder vectors α_t^k and a_t^k are not functions of the $\dot{u}$. In fact, that is how the separation into the remainder vector and the part of the acceleration that does contain $\dot{u}$ terms was made. The recursion formulas allow the remainder vectors to be propagated for each body in the system. We never calculate the body linear velocity, angular acceleration, or linear acceleration vectors since these contain the $\dot{u}$ we are trying to solve for. We make use of the remainder vectors in the following sections.

4) Form partial linear momentum vectors and partial angular momentum vectors F_r^k and T_r^k, respectively. These vectors are given by

$$F_r^k = m_k v_r^k \tag{16a}$$

and

$$T_r^k = I_k \cdot \omega_r^k \tag{16b}$$

Here m_k is the mass of body k, while I is the central inertia dyadic of body k. We also form momentum remainder vectors F_t^k and T_t^k as

$$F_t^k = m_k a_t^k - F^k \tag{17}$$

$$T_t^k = I_k \cdot \alpha_t^k + \omega^k \times (I_k \cdot \omega^k) - T^k \tag{18}$$

Here F^k and T^k are the external force and torque acting on body k, exclusive of any nonworking constraint forces or constraint torques.

We now proceed to derive the dynamical equations.

Dynamical Analysis

Basic Method

We form the equations

$$M\dot{u} = \tau \tag{19}$$

where the mass matrix M is an $n \times n$ symmetric matrix, and τ is an $n \times 1$ column vector. A generic entry m_{rj} of the mass matrix is given by

$$m_{rj} = \sum_{k=j}^{n} v_r^k \cdot F_j^k + \omega_r^k \cdot T_j^k \tag{20}$$

and an entry τ_r is given by

$$\tau_r = \sum_{k=j}^{n} v_r^k \cdot F_t^k + \omega_r^k \cdot T_t^k \tag{21}$$

Each element of the mass matrix is computed by summing over a (growing) subset of the bodies in the system. Since there are $O(n^2)$ elements in the matrix, this algorithm leads to a computational cost that grows as $O(n^3)$. Once formed, Eq. (19) may be solved for $\dot{u}$ vector, in preparation for numerical integration. A Choleski decomposition, which takes advantage of the symmetry of the mass matrix, is normally used to accomplish this. One is also free to regard Eq. (19) as a set of algebraic differential equations and use an integration scheme that can operate directly on Eq. (19), though this requires Jacobians of the mass matrix to be formed. An improvement in the algorithm is available.

Improved Method

Equation (3) provides a recursion relationship that has not been fully exploited in forming Eq. (19). We rewrite Eq. (3) as

$$v_r^k = v_r^i + \omega_r^i \times r^{ik} \tag{22}$$

where r^{ik} is the vector from the mass center of body i to the mass center of body k. Equation (22) allows us to first "shift" the partial momentum vectors for body k to the mass center of body i, add these vectors to those of body i, and *then* form the dot products indicated in Eq. (20). By starting from the highest numbered body, body n, we can form each element of the mass matrix at fixed cost. We form the last row of the mass matrix

$$m_{nj} = v_n^n \cdot F_j^n + \omega_n^n \cdot T_j^n \tag{23}$$

and τ_n

$$\tau_n = -v_n^n \cdot F_t^n - \omega_n^n \cdot T_t^n \tag{24}$$

We next shift each vector F_r^n, F_t^n, T_r^n, T_t^n to the mass center of body $n-1$ via the shifting formulas

$$F_r^{n-1} = F_r^{n-1} + F_r^n \tag{25}$$

$$F_t^{n-1} = F_t^{n-1} + F_t^n \tag{26}$$

$$T_r^{n-1} = T_r^{n-1} + T_r^n + r^{ik} \times F_r^n \tag{27}$$

$$T_t^{n-1} = T_t^{n-1} + T_t^n + r^{ik} \times F_t^n \tag{28}$$

The next-to-last row of the mass matrix is found as in Eq. (23):

$$m_{n-1,j} = v_{n-1}^{n-1} \cdot F_j^{n-1} + \omega_{n-1}^{n-1} \cdot T_j^{n-1} \tag{29}$$

and τ_{n-1} is found as in Eq. (24):

$$\tau_{n-1} = v_{n-1}^{n-1} \cdot F_t^{n-1} + \omega_{n-1}^{n-1} \cdot T_t^{n-1} \tag{30}$$

This process is continued until the complete mass matrix and the τ vector has been computed. This results in the same equations of motion as in the basic method but, since each element of the mass matrix is computed at fixed cost rather than by a summation, the total cost of computing the matrices is $O(n^2)$. The decomposition step is still $O(n^3)$, so that the simulation cost is still $O(n^3)$.

Further improvement is still available, as we show in the following section.

Triangularized Equations

Further improvement is obtained by making use of each equation of motion as soon as it is formed rather than assembling the complete system of equations and then solving the resulting set of linear equations.

The last equation of Eq. (19) is

$$\sum_{j=1}^{n} m_{nj}\dot{u}_j = \tau_n \tag{31}$$

This equation may be solved for $\dot{u}_n$:

$$\dot{u}_n = 1/m_{nn}(\tau_n - \sum_{j=1}^{n-1} m_{nj}\dot{u}_j) \tag{32}$$

This result is used to modify the partial momentum vectors. A brief discussion is required to motivate the following derivation. First, we note that, by our choice of generalized speeds, the linear and angular velocity of body k does not depend on any u_r for $r>k$. Hence, the same statement may be made for the linear and angular acceleration of body k. Thus, the appearance of $\dot{u}_n$ in each equation is due solely to the contribution of body n to each equation, for $\dot{u}_n$ does not appear in kinematical expression for any body except body n. Similar statements may be made for each lower numbered $\dot{u}_r$; $\dot{u}_{n-1}$ appears in the equations because it appears in the acceleration vector for bodies $n-1$ and n.

From Eq. (5), we can define the inertia torque for body k

$$T^{*k} = \sum_{r=1}^{k} T_r^k \dot{u}_r + T_t^k \tag{33}$$

The inertia force for body k is

$$F^{*k} = \sum_{r=1}^{k} F_r^k \dot{u}_r + F_t^k \tag{34}$$

where we have defined these vectors in terms of the partial momentum vectors introduced earlier. Equation (32) is now substituted into Eqs. (33) and (34) for body n. This yields

$$F^{*n} = \sum_{r=1}^{n-1} F_r^n \dot{u}_r + F_t^n \tag{35}$$

where

$$F_r^n = F_r^n - m_{nr}/m_{nn}F_n^n \tag{36}$$

$$F_t^n = F_t^n + \tau_n/m_{nn}F_n^n \tag{37}$$

$$T_r^n = T_r^n - m_{nr}/m_{nn}T_n^n \tag{38}$$

$$T_t^n = T_t^n + \tau_n/m_{nn}T_n^n \tag{39}$$

We note that the subscript r in Eq. (35) now ranges from 1 to $n-1$, rather than from 1 to n, as before. Therefore, when we apply the shifting algorithm, $\dot{u}_n$ will not be introduced in any subsequent equation. We continue the process of forming an equation of motion, solving for the highest $\dot{u}$ present in the equation, iterating the partial momentum vectors for the body presently under consideration, and then shifting these modified momentum vectors to the mass center of the next body.

Carrying out this procedure until all the equations have been computed yields the final set of equations:

$$L\dot{u} = \tau \tag{40}$$

where L is lower triangular with 1's along the diagonal. The solution of Eq. (38) may be obtained by simple back substitution, an $O(n^2)$ operation. We note that applying the Choleski decomposition to Eq. (19) also yields a lower matrix with 1's along the diagonal but at greater cost than the present method. The equations presented to this point have been in vector form. Of course, in actual computation, direction cosine matrices must be formed so as to be able to produce the vector sums required in the analysis.

This leads to a final cost of the present method:

$$\text{multiplies} = 18n^2 + 87n - 72 \tag{41}$$

and

$$\text{adds} = (26\ 1/2)n^2 + (73\ 1/2)n - 66 \tag{42}$$

For $n = 6$, this method yields a slight reduction compared to the most efficient method of Ref. 3: 1458 multiplies and 1329 adds compared to 1627 multiplies and 1261 adds for method 3 of Ref. 3. For manipulators possessing >6 degrees of freedom, the present algorithm can present ever-increasing savings. In a microprocessor-based implementation, the present method may be easier to implement since it does not require an equation solver to be programmed.

As an aside, an interesting interpretation of the last method can be given. We consider what would be required if, at some point in the equation derivation procedure, we wished to impose a constraint on the system. Normally, we would modify the equations already derived by adding constraint forces, and we could use the constraint equations to eliminate some of the $\dot{u}$ in favor of a subset that we regard as independent. From this point of view, we can interpret each *equation of motion* as a constraint equation since we use it to eliminate one $\dot{u}$ each time. The interesting point here is that no modification to the already derived equations is necessary since the equations are self-consistent; the autonomous motion of the system requires no constraint forces to be present.

We conclude in the next section by showing that, for some robots of practical interest, the computational bounds we have discussed here are extremely pessimistic. If one uses a symbol manipulation program to construct *literal* equations of motion rather than utilizing a numerical multibody program, simplification in the physics can be exploited. Such simplifications arise when any of the body coordinate frames are parallel to body principal axes, the joint axis is parallel to a coordinate axis, or the body mass center has no offset from the joint.

Computer Derivation of Equations of Motion

Figure 2 shows the geometry of a particular robot called the Stanford Arm. This robot possesses 6 degrees of freedom: 5 revolute joints and 1 sliding joint. Explicit equations of motion for this arm were derived by Kane and Levinson.[4]

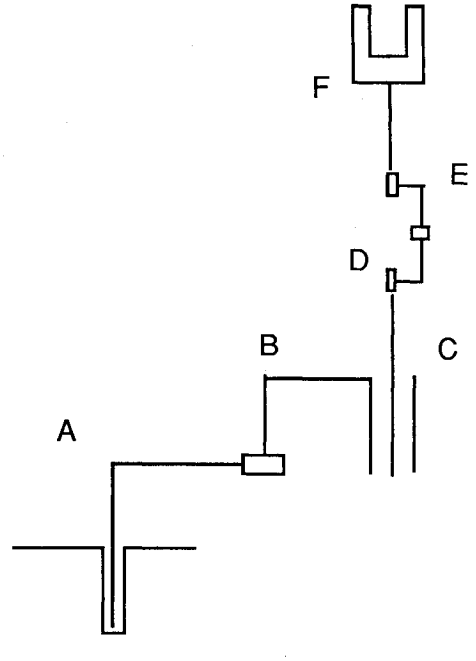

Fig. 2 Schematic representation of the Stanford Arm.

```
 TEMP(1)= ((((.5D-03*(WPK(4,5,3)*WPK(4,5,3)))+((.5D-03*(WPK(4,5,1)
& *WPK(4,5,1)))+(.001*(C5*C5))))+((VPK(4,5,3)*VPK(4,5,3))+((.01*
& (S5*S5))+(VPK(4,5,1)*VPK(4,5,1)))))+(((.01*(C5*C5))+(.05*
& (S5*S5)))+((4.*((.01*(C5*C5))+((.01*(S5*S5))+(VPK(4,4,1)*
& VPK(4,4,1)))))+((.01*(C5*C5))+(.4*(S5*S5))))))
 TEMP(2)= (((.6*((VPK(4,6,3)*VPK(4,6,3))+((VPK(4,6,1)*VPK(4,6,1))+
& (VPK(4,6,2)*VPK(4,6,2)))))+((.5D-03*(WPK(4,6,3)*WPK(4,6,3)))+
& ((.2D-03*(WPK(4,6,2)*WPK(4,6,2)))+(.5D-03*(WPK(4,5,1)*WPK(4,5,1)
& )))))+TEMP(1))
 M(1,1)= (.08+(((.5*((VPK(4,7,3)*VPK(4,7,3))+((VPK(4,6,2)*VPK(4,6,
& 2))+(VPK(4,7,1)*VPK(4,7,1)))))+((.003*(WPK(4,7,3)*WPK(4,7,3)))+
& ((.001*(WPK(4,7,1)*WPK(4,7,1)))+(.002*(WPK(4,6,2)*WPK(4,6,2)))))
& )+TEMP(2)))
```

Fig. 3 Output from symbolic equation generator.

They reported an operation count of 394 adds and 646 multiplies. Slightly improved operation counts can be realized by using computer programs that utilize symbol manipulation. One such program, SD/EXACT,[1] produces the equations of motion for this arm, which require only 390 adds and 576 multiplies. Thus, equations for this robot can be obtained at less than one-third the operations count predicted by the bounds derived earlier. These results are true for almost every industrial robot in existence. For this problem, the CPU time required to construct the equations of motion is 67 s on a VAX 780 computer. Figure 3 shows the SD/EXACT output for the $m(1,1)$ term.

Conclusions

This Note has presented a new multibody algorithm that produces highly uncoupled equations of motion. The new algorithm has a computational complexity that grows as $O(n^2)$ in the number of degrees of freedom and requires no matrix decompositions to solve for the joint accelerations. When implemented with a symbol manipulator, the new algorithm produces a substantial reduction in operation counts for robotic equations of motion.

References

[1]Rosenthal, D.E. and Sherman, M.A., "Symbolic Multibody Equations via Kane's Method," *Journal of Astronautical Sciences,* Vol. 34, No. 3, July-Sept. 1986, pp. 223-239.

[2]Kreuzer, E.J., "Dynamical Analysis of Mechanisms Using Symbolical Equation Manipulation," *Proceedings of the Fifth World Congress on the Theory of Machines and Mechanisms,* 1979.

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

[4]Kane, T.R. and Levinson, D.A., "The Use of Kane's Dynamical Equations in Robotics," *International Journal of Robotics Research,* Vol. 2, Fall 1983.

Algorithm to Generate Geodetic Coordinates from Earth-Centered Earth-Fixed Coordinates

Atul Nautiyal*
Defence Research and Development Laboratory
Hyderabad, India

Introduction

THE transformation of Earth-centered Earth-fixed (ECEF) coordinates to geodetic coordinates is required in many space position measurement systems. In a recent article,[1] Lupash developed an iterative algorithm (described as algorithm A in Ref. 1) to obtain geodetic coordinates from ECEF coordinates. Further, he compared it with the most efficient available iterative algorithm (described as algorithm B in Ref. 1). It has been observed by Lupash that algorithm A has a convergence problem near the equator, whereas algorithm B has a convergence problem near the pole. In this Note, an iterative algorithm is developed that is free from convergence problems and has better convergence property than both algorithms A and B.

Received Dec. 18, 1986; revision received March 16, 1987.

*Scientist, Systems Group B.

Mathematical Formulation

The following relations expressing ECEF coordinates X_e, Y_e, Z_e of a point P in terms of geodetic coordinates are well known.

$$X_e = (r + h)\cos\phi\cos\lambda \tag{1}$$

$$Y_e = (r + h)\cos\phi\sin\lambda \tag{2}$$

$$Z_e = [(1 - e^2)r + h]\sin\phi \tag{3}$$

and

$$r = a/(1 - e^2\sin^2\phi)^{1/2}$$

where, for the point P (see Fig. 1),

ϕ = geodetic latitude
λ = geodetic longitude
h = altitude normal to reference ellipsoid
a = ellipsoidal equatorial radius (a = 6378, 135 m based on Model WGS-72)
e^2 = square of eccentricity of reference ellipsoid (e^2 = 0.006694317778 for Model WGS-72)
b = ellipsoidal polar radius ($b = a\sqrt{1 - e^2}$)
θ = geocentric latitude

Futher, in Fig. 1, P_1 is the point of intersection of the reference ellipsoid and the normal through P and

ψ = geocentric latitude of P_1

Let the ECEF coordinates of P_1 be (X_1, Y_1, Z_1). Using Eqs. (1-3) with $h = 0$, for the point P_1, we get

$$\sqrt{X_1^2 + Y_1^2} = r\cos\phi \tag{4}$$

$$Z_1 = (1 - e^2)r\sin\phi \tag{5}$$

From Eqs. (4) and (5), we obtain

$$\tan^2\psi = Z_1^2/(X_1^2 + Y_1^2) = (1 - e^2)^2\tan^2\phi \tag{6}$$

We now consider the intersection of the reference ellipsoid and the plane passing through the point P, the center of the Earth, and the pole N (see Fig. 2). The ellipse obtained by this