N90-22996

A Nearly-Linear Computational-Cost Scheme for the Forward Dynamics of an N-Body Pendulum

Jack C. K. Chou
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas, P. O. Box 830688, MP 32
Richardson, Texas 75083-0688, USA
Tel: (214)690-2132

Abstract

The dynamic equations of motion of an n-body pendulum with spherical joints are derived to be a mixed system of differential and algebraic equations (DAE's). The DAE's are kept in implicit form to save arithmetic and preserve the sparsity of the system and are solved by the robust implicit integration method. At each solution point, the predicted solution is corrected to its exact solution within given tolerance using Newton's iterative method. For each iteration, a linear system of the form $J\Delta\mathbf{X} = \mathbf{E}$ has to be solved. The computational cost for solving this linear system directly by LU factorization is $O(n^3)$, and it can be reduced significantly by exploring the structure of J. This paper shows that by recognizing the recursive patterns and exploiting the sparsity of the system the multiplicative and additive computational costs for solving $J\Delta\mathbf{X} = \mathbf{E}$ are $O(n)$ and $O(n^2)$, respectively. The formulation and solution method for an n-body pendulum is presented. The computational cost is shown to be nearly linearly proportional to the number of bodies.

1 INTRODUCTION

The general modeling and formulation of an open-chain multi-body system with spherical joints was presented by Chou, Singhal, and Kesavan in 1986 [1]. In this paper, we are interested in a single open kinematic chain without branching which is a special configuration of a general open-chain system. Much attention was paid to this open-chain system by researchers, such as Armstrong [2], because the system is simple and its configuration is similar to robot arms. In Armstrong's work, he presented an $O(n)$ algorithm for the computation of robot forward dynamics. However, in the conclusion of his paper he stated that his program worked only for the joints with three degrees of freedom (i.e., spherical joints). He also claimed that his algorithm can be enhanced to include prismatic and revolute joints. In the author's opinion, once we have joints which possess less than three rotational degrees of freedom connecting two bodies, the bodies are rotationally dependent. In this case, the equations of motion can not be decoupled easily, and we may not be able to find an $O(n)$ algorithm for the complete calculation of robot forward dynamics unless we use very simple and primitive integration methods or other techniques such as parallel computing. In other words, if we use an unreliable integration routine we may get invalid solutions.

A series of n rigid bodies joined sequentially by spherical joints form an n-body pendulum. The bodies are allowed to rotate freely in space, but the motion of translation between adjacent bodies is constrained by the joint. This sequence of bodies can swing in any direction with one end fixed at the ceiling through a spherical joint. Here, we derive the equations of motion of this pendulum in the form of a mixed set of differential and algebraic equations (DAE's) and solve the equations using the robust implicit integration method developed by Petzold [3,4]. The DAE's are kept in implicit form to save arithmetic and preserve the sparsity of the system. At each solution point, the predicted solution is corrected to its exact solution within the given tolerance using Newton's iterative method. For each iteration, a linear system of the form $J\Delta\mathbf{X} = \mathbf{E}$ has to be solved. The computational cost for solving this linear system directly by LU factorization is $O(n^3)$. In order to reduce computations, we explore the structure of J and take advantage of the system sparsity. This paper shows that by recognizing the recursive patterns and exploiting the sparsity of the system the multiplicative and additive computational costs for solving $J\Delta\mathbf{X} = \mathbf{E}$ are $O(n)$ and $O(n^2)$, respectively.

References

[1] S. Y. Oh and D. E. Orin, "Dynamic Computer Simulation of Multiple Closed-Chain Robotic Mechanisms," in *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, pp. 15–20, San Francisco, CA, April 1986.

[2] R. H. Lathrop, "Constrained (Closed-Loop) Robot Simulation by Local Constraint Propagation," in *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, pp. 689–694, San Francisco, CA, April 1986.

[3] H. Brandl, R. Johanni, and M. Otter, "An Algorithm for the Simulation of Multibody Systems with Kinematic Loops," in *Proceedings of the IFToMM Seventh World Congress on the Theory of Machines and Mechanisms*, Sevilla, Spain, September 1987.

[4] G. Rodriguez and K. Kreutz, "Recursive Mass Matrix Factorization and Inversion: An Operator Approach to Open- and Closed-Chain Multibody Dynamics," Technical Report 88-11, Jet Propulsion Laboratory, Pasadena, CA, March 1988.

[5] 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 *Proceedings of IFAC/IFIP/IMACS International Symposium on the Theory of Robots*, Vienna, Austria, December 1986.

[6] R. E. Roberson and R. F. Schwertassek, *Introduction to the Dynamics of Multibody Systems.* Springer-Verlag, Berlin, 1987.

[7] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 1, pp. 43–53, 1987.

[8] K. W. Lilly and D. E. Orin, "*O(N)* Recursive Algorithm for the Operational Space Inertia Matrix of a Robot Manipulator," submitted to *11th IFAC World Congress*, August 1990.

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

[10] R. Featherstone, "The Calculation of Robot Dynamics Using Articulated-Body Inertias," *The International Journal of Robotics Research*, vol. 2, no. 1, pp. 13–30, Spring 1983.

[11] K. W. Lilly, "Efficient Dynamic Simulation of Multiple Chain Robotic Mechanisms," Ph.D. Thesis, The Ohio State University, 1989.

[12] M. Amin-Javaheri and D. E. Orin, "Parallel Algorithms for Computation of the Manipulator Inertia Matrix," in *Proceedings of NASA Conference on Space Telerobotics*, Pasadena, CA, Jan./Feb. 1989.

2 MATHEMATICAL MODEL

An n-body pendulum was modeled with graphs by Chou and was presented in [5]. Based on this graph-theoretic model, the mathematical model was derived as a set of DAE's and was written symbolically as

$$\boxed{\mathbf{E}(\mathbf{X}, \dot{\mathbf{X}}, t) = \mathbf{0}} \tag{1}$$

where the vector of unknown variables is

$$\mathbf{X} = \begin{bmatrix} \mathbf{R}_b^T & \mathbf{V}_b^T & \mathbf{P}_b^T & \mathbf{S}_b^T \end{bmatrix}^T \tag{2}$$

The vectors $\mathbf{R}_b$ and $\mathbf{V}_b$ are the collection of displacements and velocities of all the bodies, and the vectors $\mathbf{P}_b$ and $\mathbf{S}_b$ are the orientation parameters (we use Euler parameters here) and their derivatives. The superscripts "u" and "v" indicate the dependent set and the independent set, respectively. The equations include the following six sets of equations:

$$\mathbf{E} \equiv \begin{bmatrix} {}^r\mathbf{F}(\dot{\mathbf{V}}_b,\ \mathbf{P}_b,\ \mathbf{S}_b,\ \dot{\mathbf{S}}_b,\ t) \\ \hline {}^t\mathbf{G}(\mathbf{R}_b,\ \mathbf{P}_b,\ t) \\ {}^t\dot{\mathbf{G}}(\dot{\mathbf{V}}_b,\ \mathbf{P}_b,\ \mathbf{S}_b,\ t) \\ \hline {}^r\mathbf{I}^1(\mathbf{P}_b,\ t) \\ {}^r\mathbf{I}^2(\mathbf{P}_b,\ \mathbf{S}_b,\ t) \\ {}^r\mathbf{I}^3(\dot{\mathbf{P}}_b^v,\ \mathbf{S}_b^v,\ t) \end{bmatrix} = \mathbf{0} \tag{3}$$

and they are a total of $14n$ scalar equations in $14n$ unknown variables.

3 SOLVING THE FULL SYSTEM

At each solution point, we can solve the system equations $\mathbf{E}$ directly, using the implicit integration method [3,4] where the predicted solution is corrected to its exact solution within a given tolerance using Newton's iterative method. For each iteration, a liner system

$$\boxed{J\,\Delta\mathbf{X} = \mathbf{E}}$$

has to be solved by LU factorization. The matrix J is the system Jacobian matrix which is specified by the formula given by Petzold [4]. The vectors $\Delta\mathbf{X}$ and $\mathbf{E}$ are the vectors of corrections and residuals, respectively. Letting the Jacobian of an implicit function $\mathbf{F}$ with respect to the variable $\mathbf{V}$ be $\boldsymbol{F}_V$, we can write the Jacobian matrix J and the vectors $\Delta\mathbf{X}$ and $\mathbf{E}$ of the full system symbolically as follows:

$$\left[\begin{array}{ccccc|c} {}^t\boldsymbol{G}_R & 0 & 0 & 0 & {}^t\boldsymbol{G}_{P^u} & {}^t\boldsymbol{G}_{P^v} \\ 0 & {}^t\dot{\boldsymbol{G}}_V & {}^t\dot{\boldsymbol{G}}_{S^u} & {}^t\dot{\boldsymbol{G}}_{S^v} & {}^t\dot{\boldsymbol{G}}_{P^u} & {}^t\dot{\boldsymbol{G}}_{P^v} \\ 0 & 0 & {}^r\boldsymbol{I}^2_{S^u} & 0 & 0 & {}^r\boldsymbol{I}^2_{P^v} \\ 0 & 0 & 0 & {}^r\boldsymbol{I}^3_{S^v} & 0 & {}^r\boldsymbol{I}^3_{P^v} \\ 0 & 0 & 0 & 0 & {}^r\boldsymbol{I}^1_{P^u} & {}^r\boldsymbol{I}^1_{P^v} \\ \hline 0 & {}^r\boldsymbol{F}_V & {}^r\boldsymbol{F}_{S^u} & {}^r\boldsymbol{F}_{S^v} & {}^r\boldsymbol{F}_{P^u} & {}^r\boldsymbol{F}_{P^v} \end{array}\right] \left[\begin{array}{c} \Delta\mathbf{R}_b \\ \Delta\mathbf{V}_b \\ \Delta\mathbf{S}_b^u \\ \Delta\mathbf{S}_b^v \\ \Delta\mathbf{P}_b^u \\ \hline \Delta\mathbf{P}_b^v \end{array}\right] = \left[\begin{array}{c} {}^t\mathbf{G} \\ {}^t\dot{\mathbf{G}} \\ {}^r\mathbf{I}^2 \\ {}^r\mathbf{I}^3 \\ {}^r\mathbf{I}^1 \\ \hline {}^r\mathbf{F} \end{array}\right] \tag{4}$$

Solving the full system means that the system is solved by implicit integration in which the full linear system, $J\Delta\mathbf{X} = \mathbf{E}$, is solved by LU factorization without reducing its size. However, some structural information in J and $\mathbf{E}$ can be utilized to reduce computations when they are evaluated.

3.1 Recursiveness and Special Structures

Some equations in $\mathbf{E}$ possess recursive properties which can be further utilized to reduce computational cost. This recursive nature also causes the sub-matrices in J to have special structures which can be exploited to minimize the computational cost.

3.1.1 Translational Kinematic Constraints ${}^t\mathbf{G}$

In [5], the translational kinematic equations were derived as

$${}^t\mathbf{G}_i \equiv \mathbf{R}_{b_i} - \mathbf{R}_{sp_1} - \boldsymbol{A}_i\mathbf{r}_i + \sum_{k=1}^{i} \boldsymbol{A}_k\mathbf{a}_k = \mathbf{0} \tag{5}$$

Let

$$\begin{cases} \boldsymbol{\lambda}_i &= \sum_{k=1}^{i} \boldsymbol{A}_k \mathbf{a}_k \\ \mathbf{x}_i &= \boldsymbol{A}_i \mathbf{a}_i \\ \boldsymbol{\lambda}_0 &= \mathbf{0} \end{cases}$$

then, (5) becomes ${}^t\mathbf{G}_i \equiv \mathbf{R}_{b_i} - \mathbf{R}_{sp_1} - \boldsymbol{A}_i \mathbf{r}_i + \boldsymbol{\lambda}_i = \mathbf{0}$, or recursively

$${}^t\mathbf{G}_i \equiv \mathbf{R}_{b_i} - \mathbf{R}_{sp_1} - \boldsymbol{A}_i \mathbf{r}_i + (\boldsymbol{\lambda}_{i-1} + \mathbf{x}_i) = \mathbf{0} \tag{6}$$

When we evaluate the residual vector ${}^t\mathbf{G}$, we only have to compute $\mathbf{R}_{b_i}$, $\mathbf{R}_{sp_1}$, $\boldsymbol{A}_i \mathbf{r}_i$, and $\mathbf{x}_i$ for $i = 1, ..., n$ once. Instead of computing $\boldsymbol{\lambda}_i$ for each equation, we only compute $\mathbf{x}_i$ and add the previous $\boldsymbol{\lambda}_{i-1}$; that is, $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_{i-1} + \mathbf{x}_i$.

The Jacobian entries corresponding to the equation ${}^t\mathbf{G}$ are ${}^t\boldsymbol{G}_R$ and ${}^t\boldsymbol{G}_P$. The Jacobian of ${}^t\mathbf{G}$ corresponding to $\mathbf{R}_b$ can be shown to be a unit matrix easily. Here we show that ${}^t\boldsymbol{G}_P$ has a special structure. It is written as

$${}^t\boldsymbol{G}_P \equiv \frac{\partial\, {}^t\mathbf{G}}{\partial\, \mathbf{P}_b} = \begin{bmatrix} \frac{\partial\, {}^t\mathbf{G}_1}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\mathbf{G}_1}{\partial\, \mathbf{p}_{b_2}} & \cdots & \frac{\partial\, {}^t\mathbf{G}_1}{\partial\, \mathbf{p}_{b_n}} \\ \frac{\partial\, {}^t\mathbf{G}_2}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\mathbf{G}_2}{\partial\, \mathbf{p}_{b_2}} & \cdots & \frac{\partial\, {}^t\mathbf{G}_2}{\partial\, \mathbf{p}_{b_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_2}} & \cdots & \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_n}} \end{bmatrix} \tag{7}$$

Since ${}^t\mathbf{G}_i$ is a function of $\mathbf{p}_{b_1}$, $\mathbf{p}_{b_2}$, ..., and $\mathbf{p}_{b_i}$, for each row i $(i = 1, ..., n)$ we have

$$\frac{\partial\, {}^t\mathbf{G}_i}{\partial\, \mathbf{p}_{b_k}} = \mathbf{0};\ k = i+1, ..., n \tag{8}$$

The matrix in (7) becomes

$${}^t\boldsymbol{G}_P = \begin{bmatrix} \frac{\partial\, {}^t\mathbf{G}_1}{\partial\, \mathbf{p}_{b_1}} & 0 & 0 & \cdots & 0 \\ \frac{\partial\, {}^t\mathbf{G}_2}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\mathbf{G}_2}{\partial\, \mathbf{p}_{b_2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_1}} & \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_2}} & \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_3}} & \cdots & \frac{\partial\, {}^t\mathbf{G}_n}{\partial\, \mathbf{p}_{b_n}} \end{bmatrix} \tag{9}$$

and it is a blocked lower-triangular matrix. Each blocked entry is a 3×4 matrix. The entries in each column i $(i = 1, ..., n)$ in (9) are derived, in Appendix D.1 in [5], as

$$\begin{cases} \frac{\partial\, {}^t\mathbf{G}_i}{\partial\, \mathbf{p}_{b_i}} &= 2\boldsymbol{E}_i\, \bar{\mathbf{a}}_i - 2\boldsymbol{E}_i\, \bar{\mathbf{r}}_i \\ \frac{\partial\, {}^t\mathbf{G}_k}{\partial\, \mathbf{p}_{b_i}} &= 2\boldsymbol{E}_i\, \bar{\mathbf{a}}_i;\ k = i+1, \ldots, n \end{cases} \tag{10}$$

Hence, (9) becomes

$${}^t\boldsymbol{G}_P = \begin{bmatrix} 2\boldsymbol{E}_1(\bar{\mathbf{a}}_1 - \bar{\mathbf{r}}_1) & 0 & 0 & \cdots & 0 \\ 2\boldsymbol{E}_1\, \bar{\mathbf{a}}_1 & 2\boldsymbol{E}_2(\bar{\mathbf{a}}_2 - \bar{\mathbf{r}}_2) & 0 & \cdots & 0 \\ 2\boldsymbol{E}_1\, \bar{\mathbf{a}}_1 & 2\boldsymbol{E}_2\, \bar{\mathbf{a}}_2 & 2\boldsymbol{E}_3(\bar{\mathbf{a}}_3 - \bar{\mathbf{r}}_3) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 2\boldsymbol{E}_1\, \bar{\mathbf{a}}_1 & 2\boldsymbol{E}_2\, \bar{\mathbf{a}}_2 & 2\boldsymbol{E}_3\, \bar{\mathbf{a}}_3 & \cdots & 2\boldsymbol{E}_n(\bar{\mathbf{a}}_n - \bar{\mathbf{r}}_n) \end{bmatrix} \tag{11}$$

To compute ${}^t\boldsymbol{G}_P$, we only have to evaluate $2\boldsymbol{E}_i\, \bar{\mathbf{a}}_i$, $2\boldsymbol{E}_i\, \bar{\mathbf{r}}_i$, and $2\boldsymbol{E}_i(\bar{\mathbf{a}}_i - \bar{\mathbf{r}}_i)$ once for $i = 1, \ldots, n$.

The matrix ${}^t\boldsymbol{G}_P$ can be further partitioned into ${}^t\boldsymbol{G}_{P^u}$ and ${}^t\boldsymbol{G}_{P^v}$ as in (4). This involves permuting selected columns; however, the special structure is retained for these matrices after they are partitioned.

3.1.2 Translational Velocity Kinematic Constraints ${}^t\dot{\mathbf{G}}$

Translational velocity constraints are written as

$$ {}^t\dot{\mathbf{G}}_i \equiv \mathbf{V}_{b_i} - \dot{\boldsymbol{A}}_i \mathbf{r}_i + \sum_{k=1}^{n} \dot{\boldsymbol{A}}_k \mathbf{a}_k = \mathbf{0} \tag{12} $$

Let

$$ \begin{cases} \dot{\boldsymbol{\lambda}}_i = \sum_{k=1}^{i} \dot{\boldsymbol{A}}_k \mathbf{a}_k \\ \dot{\mathbf{x}}_i = \dot{\boldsymbol{A}}_i \mathbf{a}_i \\ \dot{\boldsymbol{\lambda}}_0 = \mathbf{0} \end{cases} $$

then, (12) becomes ${}^t\dot{\mathbf{G}}_i \equiv \mathbf{V}_{b_i} - \dot{\boldsymbol{A}}_i \mathbf{r}_i + \dot{\boldsymbol{\lambda}}_i = \mathbf{0}$, or recursively

$$ {}^t\dot{\mathbf{G}}_i \equiv \mathbf{V}_{b_i} - \dot{\boldsymbol{A}}_i \mathbf{r}_i + (\dot{\boldsymbol{\lambda}}_{i-1} + \dot{\mathbf{x}}_i) = \mathbf{0} \tag{13} $$

Similar to (6), we only have to compute $\mathbf{V}_{b_i}$, $\dot{\boldsymbol{A}}_i \mathbf{r}_i$, and $\dot{\mathbf{x}}_i$ once. Instead of computing $\dot{\boldsymbol{\lambda}}_i$, we compute $\dot{\boldsymbol{\lambda}}_{i-1} + \dot{\mathbf{x}}_i$ recursively to save computations.

The Jacobian matrices correspond to ${}^t\dot{\mathbf{G}}$ are ${}^t\dot{\boldsymbol{G}}_V$, ${}^t\dot{\boldsymbol{G}}_S$, and ${}^t\dot{\boldsymbol{G}}_P$ in which ${}^t\dot{\boldsymbol{G}}_V$ can be shown to be a unit matrix easily, and ${}^t\dot{\boldsymbol{G}}_S$ and ${}^t\dot{\boldsymbol{G}}_P$ will be shown to possess special structures. The Jacobian matrix of ${}^t\dot{\mathbf{G}}$ with respect to $\mathbf{S}_b$ is defined as

$$ {}^t\dot{\boldsymbol{G}}_S \equiv \frac{\partial\, {}^t\dot{\mathbf{G}}}{\partial\, \mathbf{S}_b} = \begin{bmatrix} \frac{\partial\, {}^t\dot{\mathbf{G}}_1}{\partial\, \mathbf{s}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_1}{\partial\, \mathbf{s}_{b_2}} & \cdots & \frac{\partial\, {}^t\dot{\mathbf{G}}_1}{\partial\, \mathbf{s}_{b_n}} \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{s}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{s}_{b_2}} & \cdots & \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{s}_{b_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_2}} & \cdots & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_n}} \end{bmatrix} \tag{14} $$

Since ${}^t\dot{\mathbf{G}}_i$ is a function of $\mathbf{s}_{b_1}$, $\mathbf{s}_{b_2}$, ..., and $\mathbf{s}_{b_i}$, for each row i $(i = 1, ..., n)$ we have

$$ \frac{\partial\, {}^t\dot{\mathbf{G}}_i}{\partial\, \mathbf{s}_{b_k}} = 0;\ k = i+1, ..., n \tag{15} $$

Hence, (14) becomes

$$ {}^t\dot{\boldsymbol{G}}_S = \begin{bmatrix} \frac{\partial\, {}^t\dot{\mathbf{G}}_1}{\partial\, \mathbf{s}_{b_1}} & 0 & 0 & \cdots & 0 \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{s}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_2}{\partial\, \mathbf{s}_{b_2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_1}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_2}} & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_3}} & \cdots & \frac{\partial\, {}^t\dot{\mathbf{G}}_n}{\partial\, \mathbf{s}_{b_n}} \end{bmatrix} \tag{16} $$

The entries in each column i $(i = 1, ..., n)$ in (16) are derived, in Appendix D.1 in [5], as

$$ \begin{cases} \frac{\partial\, {}^t\dot{\mathbf{G}}_i}{\partial\, \mathbf{s}_{b_i}} = 2\boldsymbol{E}_i\, \bar{\mathbf{a}}_i - 2\boldsymbol{E}_i\, \bar{\mathbf{r}}_i \\ \frac{\partial\, {}^t\dot{\mathbf{G}}_k}{\partial\, \mathbf{s}_{b_i}} = 2\boldsymbol{E}_i\, \bar{\mathbf{a}}_i;\ k = i+1, \ldots, n \end{cases} \tag{17} $$

Comparing (17) with (10), we find that

$$ {}^t\dot{\boldsymbol{G}}_S \equiv \frac{\partial\, {}^t\dot{\mathbf{G}}}{\partial\, \mathbf{S}_b} = \frac{\partial\, {}^t\mathbf{G}}{\partial\, \mathbf{P}_b} \equiv {}^t\boldsymbol{G}_P \tag{18} $$

Computing ${}^t\dot{\boldsymbol{G}}_S$ is not required. Once we compute ${}^t\boldsymbol{G}_P$, we get ${}^t\dot{\boldsymbol{G}}_S$.

The Jacobian, ${}^t\dot{G}_P$, is different. Since ${}^t\dot{G}_i$ is a function of p_{b_1}, p_{b_2}, ..., and p_{b_i}, ${}^t\dot{G}_P$ is a blocked lower-triangular matrix like ${}^t\dot{G}_S$ and is written as

$$ {}^t\dot{G}_P \equiv \frac{\partial\, {}^t\dot{G}}{\partial\, P_b} = \begin{bmatrix} \frac{\partial\, {}^t\dot{G}_1}{\partial\, p_{b_1}} & 0 & 0 & \cdots & 0 \\ \frac{\partial\, {}^t\dot{G}_2}{\partial\, p_{b_1}} & \frac{\partial\, {}^t\dot{G}_2}{\partial\, p_{b_2}} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^t\dot{G}_n}{\partial\, p_{b_1}} & \frac{\partial\, {}^t\dot{G}_n}{\partial\, p_{b_2}} & \frac{\partial\, {}^t\dot{G}_n}{\partial\, p_{b_3}} & \cdots & \frac{\partial\, {}^t\dot{G}_n}{\partial\, p_{b_n}} \end{bmatrix} \tag{19} $$

The entries in each column i ($i = 1, ..., n$) in (19) are derived, in Appendix D.1 in [5], as

$$ \begin{cases} \frac{\partial\, {}^t\dot{G}_i}{\partial\, p_{b_i}} = 2\dot{E}_i\, \bar{a}_i - 2\dot{E}_i\, \bar{r}_i \\ \frac{\partial\, {}^t\dot{G}_k}{\partial\, p_{b_i}} = 2\dot{E}_i\, \bar{a}_i;\ k = i+1, \ldots, n \end{cases} \tag{20} $$

Hence, (19) becomes

$$ {}^t\dot{G}_P = \begin{bmatrix} 2\dot{E}_1(\bar{a}_1 - \bar{r}_1) & 0 & 0 & \cdots & 0 \\ 2\dot{E}_1\, \bar{a}_1 & 2\dot{E}_2(\bar{a}_2 - \bar{r}_2) & 0 & \cdots & 0 \\ 2\dot{E}_1\, \bar{a}_1 & 2\dot{E}_2\, \bar{a}_2 & 2\dot{E}_3(\bar{a}_3 - \bar{r}_3) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 2\dot{E}_1\, \bar{a}_1 & 2\dot{E}_2\, \bar{a}_2 & 2\dot{E}_3\, \bar{a}_3 & \cdots & 2\dot{E}_n(\bar{a}_n - \bar{r}_n) \end{bmatrix} \tag{21} $$

To compute ${}^t\dot{G}_P$, we only have to evaluate $2\dot{E}_i\, \bar{a}_i$, $2\dot{E}_i\, \bar{r}_i$, and $2\dot{E}_i(\bar{a}_i - \bar{r}_i)$ once for $i = 1, \ldots, n$.

The matrices ${}^t\dot{G}_S$ and ${}^t\dot{G}_P$ can be further partitioned into ${}^t\dot{G}_{S^u}$, ${}^t\dot{G}_{S^v}$, ${}^t\dot{G}_{P^u}$, and ${}^t\dot{G}_{P^v}$ as in (4). This involves permuting selected columns. However, after permuting columns the special structures are retained for these partitioned matrices.

3.1.3 Torque-Balance Equations ${}^r\mathbf{F}$

The torque-balance equations for an n-body pendulum were derived as

$$ {}^r\mathbf{F}_i \equiv \mathbf{T}^i_{b_i} - \tilde{r}_i A_i^T M_i(\dot{V}_{b_i} - g) + \tilde{a}_i A_i^T \sum_{k=i}^{n} M_k(\dot{V}_{b_k} - g) = 0 \tag{22} $$

Let

$$ \begin{cases} \tau_i = \sum_{k=i}^{n} M_k(\dot{V}_{b_k} - g) \\ z_i = M_i(\dot{V}_{b_i} - g) \\ \tau_{n+1} = 0 \end{cases} $$

then, we can write (22) as ${}^r\mathbf{F}_i \equiv \mathbf{T}^i_{b_i} - \tilde{r}_i A_i^T z_i + \tilde{a}_i A_i^T \tau_i = 0$, or recursively

$$ {}^r\mathbf{F}_i \equiv \mathbf{T}^i_{b_i} - \tilde{r}_i A_i^T z_i + \tilde{a}_i A_i^T (z_i + \tau_{i+1}) = 0 \tag{23} $$

When we compute the residual of ${}^r\mathbf{F}$, we only have to evaluate $\mathbf{T}^i_{b_i}$, z_i, $\tilde{r}_i A_i^T z_i$, and $\tilde{a}_i A_i^T \tau_i$ once for $i = 1, ..., n$. Instead of computing τ_i for each equation, we compute z_i and accumulate it recursively to save computations. The recursive term, $z_i + \tau_{i+1}$, runs in backward sequence; that is, $i = n, ..., 1$.

The Jacobian of ${}^r\mathbf{F}$ has three parts: ${}^rF_V$, ${}^rF_S$, and ${}^rF_P$. They also possess special structures. First, the Jacobian of ${}^r\mathbf{F}$ with respect to V_b is written as

$$ {}^rF_V \equiv \frac{\partial\, {}^r\mathbf{F}}{\partial\, V_b} + \sigma\frac{\partial\, {}^r\mathbf{F}}{\partial\, \dot{V}_b} = \sigma\frac{\partial\, {}^r\mathbf{F}}{\partial\, \dot{V}_b} = \sigma \begin{bmatrix} \frac{\partial\, {}^r\mathbf{F}_1}{\partial\, \dot{V}_{b_1}} & \frac{\partial\, {}^r\mathbf{F}_1}{\partial\, \dot{V}_{b_2}} & \cdots & \frac{\partial\, {}^r\mathbf{F}_1}{\partial\, \dot{V}_{b_n}} \\ \frac{\partial\, {}^r\mathbf{F}_2}{\partial\, \dot{V}_{b_1}} & \frac{\partial\, {}^r\mathbf{F}_2}{\partial\, \dot{V}_{b_2}} & \cdots & \frac{\partial\, {}^r\mathbf{F}_2}{\partial\, \dot{V}_{b_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\, {}^r\mathbf{F}_n}{\partial\, \dot{V}_{b_1}} & \frac{\partial\, {}^r\mathbf{F}_n}{\partial\, \dot{V}_{b_2}} & \cdots & \frac{\partial\, {}^r\mathbf{F}_n}{\partial\, \dot{V}_{b_n}} \end{bmatrix} \tag{24} $$

Since ${}^r\mathbf{F}_i$ is a function of $\dot{\mathbf{V}}_{b_i}$, $\dot{\mathbf{V}}_{b_{i+1}}$, ..., and $\dot{\mathbf{V}}_{b_n}$, for each column i $(i = 1, ..., n)$ we have

$$\sigma \frac{\partial {}^r\mathbf{F}_k}{\partial \dot{\mathbf{V}}_{b_i}} = \mathbf{0};\ k = i+1, ..., n \tag{25}$$

The matrix (24) becomes a blocked upper-triangular matrix and is written as

$${}^r\boldsymbol{F}_V = \sigma \begin{bmatrix} \frac{\partial {}^r\mathbf{F}_1}{\partial \dot{\mathbf{V}}_{b_1}} & \frac{\partial {}^r\mathbf{F}_1}{\partial \dot{\mathbf{V}}_{b_2}} & \cdots & \frac{\partial {}^r\mathbf{F}_1}{\partial \dot{\mathbf{V}}_{b_n}} \\ 0 & \frac{\partial {}^r\mathbf{F}_2}{\partial \dot{\mathbf{V}}_{b_2}} & \cdots & \frac{\partial {}^r\mathbf{F}_2}{\partial \dot{\mathbf{V}}_{b_n}} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial {}^r\mathbf{F}_n}{\partial \dot{\mathbf{V}}_{b_n}} \end{bmatrix} \tag{26}$$

The entries of each row i $(i = 1, ..., n)$ in ${}^r\boldsymbol{F}_V$ are derived as

$$\begin{cases} \sigma \frac{\partial {}^r\mathbf{F}_i}{\partial \dot{\mathbf{V}}_{b_i}} = -\sigma M_i \tilde{\mathbf{r}}_i A_i^T + \sigma M_i \tilde{\mathbf{a}}_i A_i^T \\ \sigma \frac{\partial {}^r\mathbf{F}_i}{\partial \dot{\mathbf{V}}_{b_k}} = \sigma M_k \tilde{\mathbf{a}}_i A_i^T;\ k = i+1,\ ...,\ n \end{cases} \tag{27}$$

Hence, (26) becomes

$${}^r\boldsymbol{F}_V = \sigma \begin{bmatrix} M_1(\tilde{\mathbf{a}}_1 - \tilde{\mathbf{r}}_1)A_1^T & M_2\tilde{\mathbf{a}}_1 A_1^T & M_3\tilde{\mathbf{a}}_1 A_1^T & \cdots & M_n\tilde{\mathbf{a}}_1 A_1^T \\ 0 & M_2(\tilde{\mathbf{a}}_2 - \tilde{\mathbf{r}}_2)A_2^T & M_3\tilde{\mathbf{a}}_2 A_2^T & \cdots & M_n\tilde{\mathbf{a}}_1 A_1^T \\ 0 & 0 & M_3(\tilde{\mathbf{a}}_3 - \tilde{\mathbf{r}}_3)A_3^T & \cdots & M_n\tilde{\mathbf{a}}_3 A_3^T \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & M_n(\tilde{\mathbf{a}}_n - \tilde{\mathbf{r}}_n)A_n^T \end{bmatrix} \tag{28}$$

To compute ${}^r\boldsymbol{F}_V$, we only have to evaluate $\sigma M_i(\tilde{\mathbf{a}}_i - \tilde{\mathbf{r}}_i)A_i^T$ and $\sigma M_k \tilde{\mathbf{a}}_i A_i^T$ once for $i = 1, ..., n$.

The Jacobian of ${}^r\mathbf{F}$ with respect to $\mathbf{P}_b$ is similar to (24). However, since ${}^r\mathbf{F}_i$ is a function of $\mathbf{p}_{b_i}$, the off-diagonal entries become zero:

$$\frac{\partial {}^r\mathbf{F}_k}{\partial \mathbf{p}_{b_i}} = \frac{\partial {}^r\mathbf{F}_i}{\partial \mathbf{p}_{b_k}} = \mathbf{0};\ k = i+1, ..., n \tag{29}$$

for $i = 1, ..., n$. Hence, ${}^r\boldsymbol{F}_P$ becomes a diagonally blocked matrix and is written as

$${}^r\boldsymbol{F}_P \equiv \frac{\partial {}^r\mathbf{F}}{\partial \mathbf{P}_b} = \begin{bmatrix} \frac{\partial {}^r\mathbf{F}_1}{\partial \mathbf{p}_{b_1}} & 0 & \cdots & 0 \\ 0 & \frac{\partial {}^r\mathbf{F}_2}{\partial \mathbf{p}_{b_2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial {}^r\mathbf{F}_n}{\partial \mathbf{p}_{b_n}} \end{bmatrix} \tag{30}$$

where

$$\begin{aligned} \frac{\partial {}^r\mathbf{F}_i}{\partial \mathbf{p}_{b_i}} &= \frac{\partial \mathbf{T}^i_{b_i}}{\partial \mathbf{p}_{b_i}} - \tilde{\mathbf{r}}_i \frac{\partial (A_i^T \mathbf{z}_i)}{\partial \mathbf{p}_{b_i}} + \tilde{\mathbf{a}}_i \frac{\partial (A_i^T \boldsymbol{\tau}_i)}{\partial \mathbf{p}_{b_i}} \\ &= \frac{\partial \mathbf{T}^i_{b_i}}{\partial \mathbf{p}_{b_i}} - 2\tilde{\mathbf{r}}_i G_i \overset{+}{\mathbf{z}}_i + 2\tilde{\mathbf{a}}_i G_i \overset{+}{\boldsymbol{\tau}}_i \\ &= \left\{ 2I_i \ddot{G}_i + 4G_i \dot{G}_i^T I_i \dot{G}_i - 4(\mathbf{s}_{b_i}^T \mathbf{p}_{b_i}) I_i \dot{G}_i - 2\tilde{\boldsymbol{\psi}}_i \dot{G}_i \right\} \\ &\quad - 2\tilde{\mathbf{r}}_i G_i \overset{+}{\mathbf{z}}_i + 2\tilde{\mathbf{a}}_i G_i \overset{+}{\boldsymbol{\tau}}_i \end{aligned} \tag{31}$$

where $\boldsymbol{\psi}_i \equiv I_i \omega_i = 2I_i G_i \dot{\mathbf{p}}_{b_i}$ and $\tilde{\boldsymbol{\psi}}_i \equiv \boldsymbol{\psi} \times$ are defined in Appendix D.2 in [5].

${}^r\boldsymbol{F}_S$ has a structure identical to that in ${}^r\boldsymbol{F}_P$. The off-diagonal entries are zero:

$$\frac{\partial {}^r\mathbf{F}_k}{\partial \mathbf{s}_{b_i}} + \sigma \frac{\partial {}^r\mathbf{F}_k}{\partial \dot{\mathbf{s}}_{b_i}} = \frac{\partial {}^r\mathbf{F}_i}{\partial \mathbf{s}_{b_k}} + \sigma \frac{\partial {}^r\mathbf{F}_i}{\partial \dot{\mathbf{s}}_{b_k}} = \mathbf{0};\ k = i+1, ..., n \tag{32}$$

for $i = 1, ..., n$. The Jacobian ${}^r\boldsymbol{F}_S$ is diagonally blocked and is written as

$${}^r\boldsymbol{F}_S \equiv \frac{\partial {}^r\mathbf{F}}{\partial \mathbf{S}_b} + \sigma \frac{\partial {}^r\mathbf{F}}{\partial \dot{\mathbf{S}}_b} = \begin{bmatrix} \frac{\partial {}^r\mathbf{F}_1}{\partial \mathbf{s}_{b_1}} + \sigma \frac{\partial {}^r\mathbf{F}_1}{\partial \dot{\mathbf{s}}_{b_1}} & 0 & \cdots & 0 \\ 0 & \frac{\partial {}^r\mathbf{F}_2}{\partial \mathbf{s}_{b_2}} + \sigma \frac{\partial {}^r\mathbf{F}_2}{\partial \dot{\mathbf{s}}_{b_2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial {}^r\mathbf{F}_n}{\partial \mathbf{s}_{b_n}} + \sigma \frac{\partial {}^r\mathbf{F}_n}{\partial \dot{\mathbf{s}}_{b_n}} \end{bmatrix} \tag{33}$$

where

$$\begin{aligned} \frac{\partial {}^r\mathbf{F}_i}{\partial \mathbf{s}_{b_i}} + \sigma \frac{\partial {}^r\mathbf{F}_i}{\partial \dot{\mathbf{s}}_{b_i}} &= \frac{\partial {}^r\mathbf{T}_{b_i}}{\partial \mathbf{s}_{b_i}} + \sigma \frac{\partial {}^r\mathbf{T}_{b_i}}{\partial \dot{\mathbf{s}}_{b_i}} \\ &= \left\{ 4\dot{G}_i G_i^T I_i G_i - 4(\mathbf{s}_{b_i}^T \mathbf{p}_{b_i}) I_i G_i + 2\tilde{\dot{\boldsymbol{\psi}}}_i G_i \right\} - \sigma(2 I_i G_i) \end{aligned} \tag{34}$$

The derivation of (31) and (34) is given in Appendix D.2 in [5].

3.1.4 Equations Concerning Euler Parameters: ${}^r\mathbf{I}^1$, ${}^r\mathbf{I}^2$, and ${}^r\mathbf{I}^3$

When we compute the residuals of ${}^r\mathbf{I}^1$, ${}^r\mathbf{I}^2$, and ${}^r\mathbf{I}^3$, there is no applicable recursive structure that can be utilized. However, their Jacobians do have some special structures because these equations are derived for each body. Matrices with diagonally blocked structures are to be expected.

First, the normality constraint for each set of Euler parameters is written as

$${}^r I_i^1 \equiv \mathbf{p}_{b_i}^T \mathbf{p}_{b_i} - 1 = 0 \tag{35}$$

Since ${}^r I_i^1$ is a function of $\mathbf{p}_{b_i}$ only, we have

$$\frac{\partial {}^r I_i^1}{\partial \mathbf{p}_{b_k}} = \frac{\partial {}^r I_k^1}{\partial \mathbf{p}_{b_i}} = \mathbf{0};\ k = i+1, ..., n \tag{36}$$

From (36), the Jacobian matrix of ${}^r\mathbf{I}^1$ with respect to $\mathbf{p}_{b_1}$, $\mathbf{p}_{b_2}$, ..., and $\mathbf{p}_{b_n}$ must be a diagonally blocked matrix. The same argument can be applied to ${}^r\mathbf{I}^2$ and ${}^r\mathbf{I}^3$ such that their Jacobians with respect to $\mathbf{P}_b$ and $\mathbf{S}_b$ are diagonally blocked.

Since the blocks concerning these three sets of equations are locally processed, as shown in [5], the Jacobian matrices ${}^r I_{P^*}^2$, ${}^r I_{P^*}^3$, and ${}^r I_{P^*}^1$ possess a similar structure which is illustrated as follows:

$$\begin{bmatrix} {}^r I_{P^*}^2 \\ \hline {}^r I_{P^*}^3 \\ \hline {}^r I_{P^*}^1 \end{bmatrix} = \left[\begin{array}{cccc} \epsilon_1 & 0 & \cdots & 0 \\ 0 & \epsilon_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \epsilon_n \\ \hline \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n \\ \hline \nu_1 & 0 & \cdots & 0 \\ 0 & \nu_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \nu_n \end{array}\right] \tag{37}$$

where

$$\begin{cases} \epsilon_i = [d_i\ e_i\ f_i] \\ \sigma_i = \begin{bmatrix} -\sigma & 0 & 0 \\ 0 & -\sigma & 0 \\ 0 & 0 & -\sigma \end{bmatrix} \\ \nu_i = [a_i\ b_i\ c_i] \end{cases}$$

are defined in Appendix B.1 in [5].

4 SOLVING THE REDUCED SYSTEM

Solving the reduced system means to solve the linear system using the technique of dimension reduction. Partitioning the linear system, $J\Delta\mathbf{X} = \mathbf{E}$, according to the partition in (4) gives

$$\begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{X}_1 \\ \Delta\mathbf{X}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{E}_1 \\ \mathbf{E}_2 \end{bmatrix}$$

Since J_{11} is a unit upper-triangular matrix and is non-singular, we can obtain

$$\begin{bmatrix} J_{11} & J_{12} \\ 0 & J_R \end{bmatrix} \begin{bmatrix} \Delta\mathbf{X}_1 \\ \Delta\mathbf{X}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{E}_1 \\ \mathbf{E}_R \end{bmatrix}$$

For the independent corrections $\Delta\mathbf{X}_2$, we solve

$$\boxed{J_R\Delta\mathbf{X}_2 = \mathbf{E}_R} \tag{38}$$

by LU factorization where

$$\begin{cases} J_R = J_{22} - J_{21}J_{11}^{-1}J_{12} \\ \mathbf{E}_R = \mathbf{E}_2 - (J_{21}J_{11}^{-1})\mathbf{E}_1 \end{cases} \tag{39}$$

The dependent corrections $\Delta\mathbf{X}_1$ are obtained by backward substitutions:

$$\Delta\mathbf{X}_1 = J_{11}^{-1}(\mathbf{E}_1 - J_{12}\Delta\mathbf{X}_2) \tag{40}$$

4.1 Solving for Independent Corrections

The independent corrections $\Delta\mathbf{X}_2$ are obtained by solving the linear system $J_R\Delta\mathbf{X}_2 = \mathbf{E}_R$. The evaluation of J_R and $\mathbf{E}_R$ is accomplished by developing a set of formulae such that the sparsity of the linear system is completely utilized. From (4), we derive the following formulae for J_R and $\mathbf{E}_R$:

$$\begin{aligned} J_R = {}& {}^{r}F_{P^{v}} - ({}^{r}F_V)({}^{t}\dot{G}_{P^{v}}) - ({}^{r}\Psi_4)({}^{r}I^2_{P^{v}}) \\ & - ({}^{r}\Psi_5)({}^{r}I^3_{P^{v}}) - ({}^{r}\Psi_6)({}^{r}I^1_{P^{v}}) \end{aligned} \tag{41}$$

and

$$\mathbf{E}_R = {}^{r}\mathbf{F} - ({}^{r}F_V)\,{}^{t}\dot{\mathbf{G}} - ({}^{r}\Psi_4)\,{}^{r}\mathbf{I}^2 - ({}^{r}\Psi_5)\,{}^{r}\mathbf{I}^3 - ({}^{r}\Psi_6)\,{}^{r}\mathbf{I}^1 \tag{42}$$

where

$$\begin{cases} {}^{r}\Psi_4 = {}^{r}F_{S^{u}} - ({}^{r}F_V)({}^{t}\dot{G}_{S^{u}}) \\ {}^{r}\Psi_5 = {}^{r}F_{S^{v}} - ({}^{r}F_V)({}^{t}\dot{G}_{S^{v}}) \\ {}^{r}\Psi_6 = {}^{r}F_{P^{u}} - ({}^{r}F_V)({}^{t}\dot{G}_{P^{u}}) \end{cases} \tag{43}$$

When we compute J_R and $\mathbf{E}_R$ using the above formulae, several matrix multiplications will be performed. The special structures discussed in the previous section can be exploited to reduce computation cost.

4.2 Solving for Dependent Corrections

Once we obtain the independent corrections, $\Delta\mathbf{X}_2$, the dependent corrections, $\Delta\mathbf{X}_1$, can be computed by $J_{11}^{-1}(\mathbf{E}_1 - J_{12}\Delta\mathbf{X}_2)$. However, using this formula directly is not practical since J_{11} has a very simple structure with many zeros. In order to utilize the sparsity completely, a set of formulae are derived symbolically for the dependent corrections. They are as follows:

$$\Delta\mathbf{P}^{u}_{b} = {}^{r}\mathbf{I}^1 - ({}^{r}I^1_{P^{v}})\Delta\mathbf{P}^{v}_{b} \tag{44}$$

$$\Delta\mathbf{S}^{v}_{b} = {}^{r}\mathbf{I}^3 - ({}^{r}I^3_{P^{v}})\Delta\mathbf{P}^{v}_{b} \tag{45}$$

$$\Delta\mathbf{S}^{u}_{b} = {}^{r}\mathbf{I}^2 - ({}^{r}I^2_{P^{v}})\Delta\mathbf{P}^{v}_{b} \tag{46}$$

$$\begin{aligned} \Delta\mathbf{V}_b = {}& {}^{t}\dot{\mathbf{G}} - ({}^{t}\dot{G}_{S^{u}})\Delta\mathbf{S}^{u}_{b} - ({}^{t}\dot{G}_{S^{v}})\Delta\mathbf{S}^{v}_{b} \\ & - ({}^{t}\dot{G}_{P^{u}})\Delta\mathbf{P}^{u}_{b} - ({}^{t}\dot{G}_{P^{v}})\Delta\mathbf{P}^{v}_{b} \end{aligned} \tag{47}$$

$$\Delta\mathbf{R}_b = {}^{t}\mathbf{G} - ({}^{t}G_{P^{u}})\Delta\mathbf{P}^{u}_{b} - ({}^{t}G_{P^{v}})\Delta\mathbf{P}^{v}_{b} \tag{48}$$

Again, when we compute the dependent corrections using the above formulae, the special structures discussed in the previous section can be exploited.

5 LINEAR COMPUTATIONAL-COST SCHEME

Rewriting the full system (4) by

- combining $\mathbf{S}_b^u$ and $\mathbf{S}_b^v$ to form $\mathbf{S}$ in the original order
- combining $\mathbf{P}_b^u$ and $\mathbf{P}_b^v$ to form $\mathbf{P}$ in the original order
- combining ${}^r\mathbf{I}^1$ and ${}^r\mathbf{I}^3$ to form $\mathbf{I}$
- combining $-2\sigma\,{}^r\mathbf{I}^2$ (scaled by -2σ) and ${}^r\mathbf{F}$ to form $\mathbf{F}$
- re-arranging the columns and rows
- dropping the subscripts and superscripts for clarity and simplicity

gives

$$\begin{bmatrix} G_R & G_P & 0 & 0 \\ 0 & I_P & I_S & 0 \\ 0 & F_P & F_S & F_V \\ 0 & \dot{G}_P & \dot{G}_S & \dot{G}_V \end{bmatrix} \begin{bmatrix} \Delta\mathbf{R} \\ \Delta\mathbf{P} \\ \Delta\mathbf{S} \\ \Delta\mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{G} \\ \mathbf{I} \\ \mathbf{F} \\ \dot{\mathbf{G}} \end{bmatrix} \tag{49}$$

or

$$\begin{bmatrix} U & G_P & 0 & 0 \\ 0 & U & I_S & 0 \\ 0 & F_P & F_S & F_V \\ 0 & \dot{G}_P & \dot{G}_S & U \end{bmatrix} \begin{bmatrix} \Delta\mathbf{R} \\ \Delta\mathbf{P} \\ \Delta\mathbf{S} \\ \Delta\mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{G} \\ \mathbf{I} \\ \mathbf{F} \\ \dot{\mathbf{G}} \end{bmatrix} \tag{50}$$

where G_R and $\dot{G}_V$ were shown to be unit matrices. In order to make I_P a unit matrix, Gaussian elimination has to be applied to I_P, I_S, and $\mathbf{I}$ locally. An example for one body is given in Appendix B.2 in [5]. We intend to solve this linear system with a computational cost which is linearly proportional to the number of bodies in the system.

5.1 Basic Theory

Applying the technique of dimension reduction to (50) gives

$$\begin{bmatrix} U & G_P & 0 & 0 \\ 0 & U & I_S & 0 \\ 0 & 0 & J_F & 0 \\ 0 & 0 & 0 & J_R \end{bmatrix} \begin{bmatrix} \Delta\mathbf{R} \\ \Delta\mathbf{P} \\ \Delta\mathbf{S} \\ \Delta\mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{G} \\ \mathbf{I} \\ \mathbf{E}_F \\ \mathbf{E}_R \end{bmatrix} \tag{51}$$

where

$$\begin{cases} J_R = U - (\dot{G}_S - \dot{G}_P I_S)\, J_F^{-1} F_V \\ \mathbf{E}_R = \dot{\mathbf{G}} - (\dot{G}_P)\,\mathbf{I} - (\dot{G}_S - \dot{G}_P I_S)\, J_F^{-1}(\mathbf{F} - F_P\,\mathbf{I}) \end{cases} \tag{52}$$

and

$$\begin{cases} J_F = F_S - F_P I_S \\ \mathbf{E}_F = \mathbf{F} - (F_P)\,\mathbf{I} - (F_V)\,\Delta\mathbf{V} \end{cases} \tag{53}$$

Instead of solving (50) directly by LU factorisation, we solve

$$\boxed{J_R \Delta\mathbf{V} = \mathbf{E}_R} \tag{54}$$

to obtain $\Delta\mathbf{V}$; then we solve

$$\boxed{J_F \Delta\mathbf{S} = \mathbf{E}_F} \tag{55}$$

to obtain $\Delta\mathbf{S}$. The rest of corrections can be computed by

$$\begin{cases} \Delta\mathbf{P} = \mathbf{I} - (I_S)\,\Delta\mathbf{S} \\ \Delta\mathbf{R} = \mathbf{G} - (G_P)\,\Delta\mathbf{P} \end{cases} \tag{56}$$

5.2 Invertibility

In [6], we have shown that J_R is non-singular if J is non-singular. From (52), we have to find J_F in order to evaluate J_R. We would like to show that J_F is invertible when $\sigma \rightarrow \infty$.

The matrix J_F is diagonally blocked because it is computed by $F_S - F_P I_S$ where F_S, F_P, and I_S are all diagonally blocked matrices. Each block at the diagonal position in J_F is computed by its corresponding block in F_S, F_P, and I_S. Hence, we write

$$J_{F_i} = F_{S_i} - F_{P_i} I_{S_i} \tag{57}$$

for each diagonal block. From (B.12) in Appendix B given in [5], we find that I_{S_i} is a matrix scaled by $\frac{1}{\sigma}$. Hence, we may write

$$F_{P_i} I_{S_i} = \frac{1}{\sigma} \Upsilon_i$$

Also, from (34) we find that F_{S_i} may be written as

$$F_{S_i} = -\sigma(2I_i G_i) + \Gamma_i$$

Hence, we can write

$$J_{F_i} = -\sigma(2I_i G_i) + \Gamma_i - \frac{1}{\sigma} \Upsilon_i \tag{58}$$

If $\sigma \rightarrow \infty$, we have

$$J_{F_i} \approx -\sigma(2I_i G_i) \tag{59}$$

However, we have added one more equation into the F block as mentioned previously. This increases the number of equations in each block in F from three to four. The equation is ${}^r\mathbf{I}_i^2$ scaled by -2σ; that is,

$$-2\sigma\, {}^r\mathbf{I}_i^2 \equiv -2\sigma\, \mathbf{p}_i^T \mathbf{s}_i = 0 \tag{60}$$

The Jacobian corresponding to s is

$$-2\sigma\, \mathbf{p}_i^T \tag{61}$$

Combining it with (59) gives

$$J_{F_i} \approx -\sigma(2I_i \overset{+}{\mathbf{P}}_i^T) \tag{62}$$

Adding equation (60) into block F is equivalent to using Euler's equations in quaternion space [7]. We have proved that the coefficient matrix with respect to $\dot{\mathbf{s}}$ in Euler's equations for inertial torque in 4-space is a non-singular matrix [7]. This matrix is exactly the same as $2I_i(\overset{+}{\mathbf{P}}_i)^T$ in (62). Therefore, J_{F_i} as well as J_F are invertible if $\sigma \rightarrow \infty$.

5.3 Summary of Computational Cost

The detailed discussion of the linear-cost scheme was presented in [5]. In terms of multiplicative($\times$) and additive ($+$) operations, the computational cost required for solving the reduced linear system for the linear-cost scheme is summarized also in detail in [5] as Tables 2 to 6. Adding up the totals in the tables gives the grand total of operations required for the linear-cost scheme:

$$\begin{cases} \times: & 657n - 324 \\ +: & 15n^2 + 574n - 321 \end{cases} \tag{63}$$

This cost is not for obtaining one solution point; it is the cost of solving the reduced linear system for one Newton's iteration. The computational cost for reaching a solution point depends on output step size and the number of iterations.

6 IMPLEMENTATION AND RESULTS

A program, called LINPEN (LINear-cost scheme for simulating an n-body PENdulum), was coded to simulate a pendulum. The program is equipped with modules for reporting various physical quantities such as displacement, velocity, and acceleration and was run on a VAX 750 computer. The motion of a user-specified pendulum can be displayed on a GRINNELL display terminal.

Each body is drawn as a standardized 3-D polygon and is projected in perspective. on the terminal screen according to its simulated position and orientation. An example of simulating a 3-body pendulum for ten continuous positions is illustrated in Figure 1. The figure looks exactly the same as displayed on the display terminal.

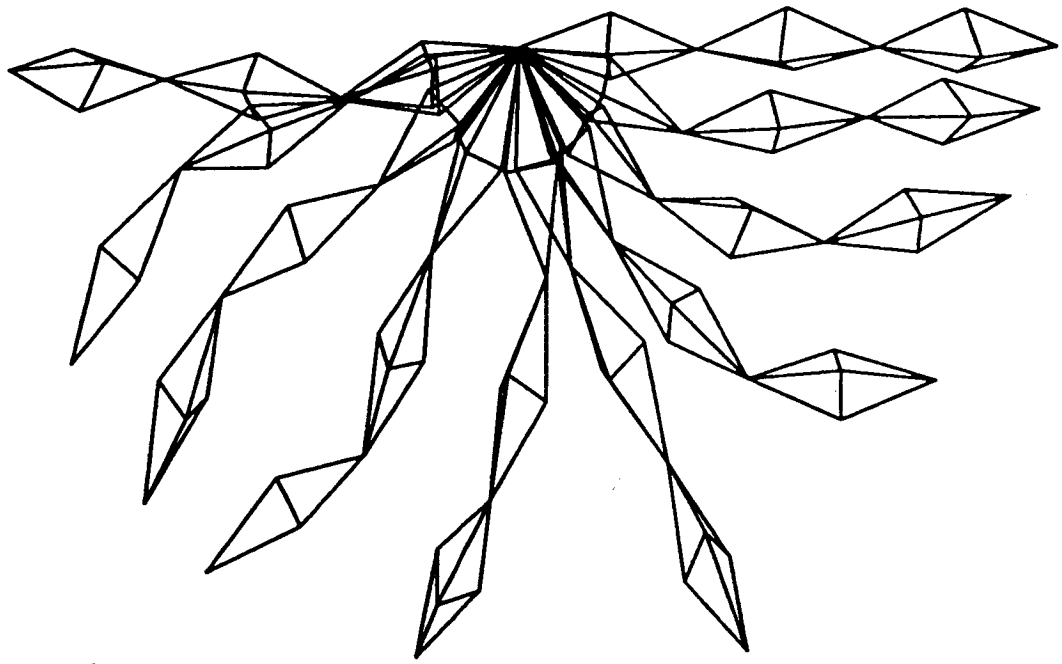

Figure 1: Simulation of a 3-body pendulum.

6.1 Comparative Study in Computational Cost

For each solution point, a linear system, $J\,\Delta\mathbf{X} = \mathbf{E}$, may have to be solved by LU factorization several times. There are four schemes available to accomplish this:

- solving the full linear system by LU factorization where J is approximated by numerical difference
- solving the full linear system by LU factorization where J is evaluated by the exact Jacobian matrix of $\mathbf{E}$
- solving the reduced linear system by LU factorization where J is evaluated by the exact Jacobian and is further reduced to its optimal size using the technique of dimension reduction
- solving the reduced linear system by LU factorization where J is evaluated by the exact Jacobian and is further reduced to its optimal size using the linear-cost scheme

For each iteration, the analytical count of operations required to solve the full system, reduced system, and the linear-cost system with exact Jacobian is tabulated in Table 1. Although the additive operations for the linear-cost system is $O(n^2)$, we may consider the cost linear because the amount of time required to perform a multiplication or division on a computer is about the same and is considerably greater than that required to perform an addition or subtraction.

The simulation of an n-body pendulum was run for the four systems. For each system, the program was run ten times from one body to ten bodies. One hundred solution points were generated, for each run, with an output time-step of 0.01 seconds and a tolerance of 10^{-7}. Total CPU time for solving four systems is calculated in terms of CPU time per residual-call or CPU time per Jacobian-call. Total CPU time per residual-call versus the number of bodies is plotted in Figure 2.

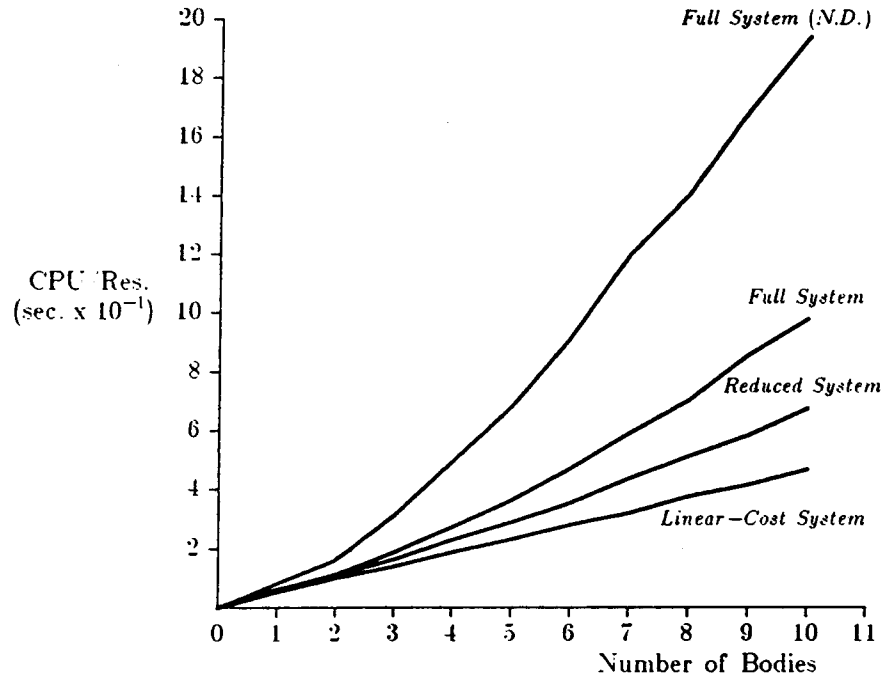

Figure 2: Computational cost in terms of CPU time for four systems.

An N-Body Pendulum Grand Total of Operations for Three Systems with Exact Jacobian Matrix		
System	×	+
Full System	$915n^3 + 196n^2 - 5n$	$915n^3 + 196n^2 - 19n$
Reduced System	$33n^3 + 129n^2 + 50n$	$25n^3 + 51n^2 + 25n$
Linear-Cost System	$657n - 324$	$15n^2 + 574n - 321$

Table 1: Analytical count of operations required for solving three systems with analytical Jacobian matrices.

7 REMARKS

The mathematical model of an n-body pendulum with spherical joints and its solution method have been presented. The mathematical model derived here is a mixed system of differential and algebraic equations (DAE's) in implicit form. A model of state-space equations can be derived if we do further complex substitutions. However, in the form of DAE's the equations of motion provide more sparsity, and this sparsity can be exploited when numerical solution methods are applied. The modeling and formulation of an n-body pendulum is the first study, for its similarity to a robotic manipulator in structure and for its simplicity in equations.

We also present four solution methods for solving the equations of motion of this n-body pendulum. The first method solves the linear system, $\boldsymbol{J}\,\Delta\mathbf{X} = \mathbf{E}$, directly by LU factorization where $\boldsymbol{J}$ is approximated by numerical difference. The second method solves the linear system directly by LU factorization, but $\boldsymbol{J}$ is evaluated by its analytical Jacobian matrix with some structural consideration.

The third method employs the technique of dimension reduction to transform the full system to its reduced system, $\boldsymbol{J}_R\,\Delta\mathbf{V} = \mathbf{E}_R$, such that LU factorization is performed on the reduced system only. This technique utilizes the sparsity of the system completely and saves a considerable number of computations. The four methods also employ the technique of dimension reduction to reduce the size of the system. However, they further exploit the structure of the system such that the reduced Jacobian generated possesses a very special structure which can be utilized to solve the system in a nearly-linear computational cost.

The comparative study in computational cost for these four systems in the paper gives strong evidence of the success of the theory developed.

References

[1] J. C. K. Chou, K. Singhal, and H. K. Kesavan. Multi-body systems with open chains: Graph-theoretic model. *Mechanism and Machine Theory*, 21(3):273–284, 1986.

[2] W. W. Armstrong. Recursive solution to the equations of motion of an n-link manipulator. *Proceeding of 5th World Congress on Theory of Machines and Mechanisms*, 2:1343–1346, July 1979.

[3] L. R. Petzold. Differential/algebraic equations are not ODE's. *SIAM J. Sci. Stat. Comput.*, 3(3):367–384, September 1982.

[4] L. R. Petzold. A description of DASSL: A differential/algebraic system solver. *Scientific Computing*, pages 65–68, North-Holland, 1983.

[5] Jack C. K. Chou. *Modeling, Formulation, and Solution Scheme for an N-Body Pendulum.* Technical Report No. TR-89008, Program in Engineering Science, Erik Jonsson School of Engineering and Computer Science, University of Texas at Dallas, Richardson, Texas, 1989.

[6] Jack C. K. Chou. *Computer-Aided Design Methods for Three-Dimensional Constrained Mechanical Systems.* Ph.D. Dissertation, Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada, 1988.

[7] Jack C. K. Chou. *Quaternions, Finite Rotation, and Dynamics.* Technical Report No. TR-89007, Program in Engineering Science, Erik Jonsson School of Engineering and Computer Science, University of Texas at Dallas, Richardson, Texas, 1989.