Rotational Maneuvers of Flexible Spacecraft," *J. of Guidance*, Vol. 3, No. 6, Nov.-Dec. 1980, pp. 578-585.

13 Turner, J. D., and Chun, H. M., "Optimal Distributed Control of a Flexible Spacecraft During a Large-Angle Maneuver," *J. of Guidance*, Vol. 7, No. 3, May-June 1984, pp. 257-264.

14 Yang, S. M., "Dynamics of Mechanical Systems with Couple Rigid Body and Flexible Body Motion," *Proceedings of the 60th Shock and Vibration Symposium*, Vol. II, Nov. 1989, pp. 71-91.

15 Yang, S. M., and Mote, C. D., "Stability of Non-Conservative Linear Discrete Gyroscopic System," *J. of Sound and Vibration*, Vol. 147, No. 3, June 1991, pp. 453-464.

16 Yang, S. M., "Modal Analysis of Structures with Holonomic Constraints," Accepted for publication in *AIAA Journal*, 1992.

An Iterative Approach to Multibody Simulation Dynamics Suitable for Parallel Implementation

I. Sharf[1] and G. M. T. D'Eleuterio[2]

This paper presents a new solution procedure for simulation dynamics of multibody systems. The method is applicable to open chains with general interbody constraints. It is based on obtaining an explicit solution for the joint constraint forces by means of iterative techniques. We show that the algorithm possesses a parallel structure which matches the topology of the system. Numerical results for an anthropomorphic manipulator indicate that the conjugate-gradient Jacobi iteration is computationally most efficient. An estimate for the parallel efficiency of this scheme is obtained by combining the theoretical bound for parallel complexity with an approximate overhead cost associated with parallel implementation.

1 Introduction

The simulation of multibody systems has occupied the attention of more than just a few researchers over the past quarter century. Myriad formulations and algorithms have been put forward (Walker and Orin, 1982; Featherstone, 1983; Hughes, 1985; Bae and Haug, 1987; Golla, 1987; Rodriguez, 1987; D'Eleuterio, 1992), all in an effort to devise yet more efficient or more general simulation procedures. Until recently, the development and implementation of simulation-dynamics (as well as inverse-dynamics) algorithms have been confined to serial-architecture computers.

Recent developments in distributed computing and the advent of *parallel* computers have given dynamicists cause to explore the multibody dynamics problems from a new perspective. The merits of existing procedures are accordingly being re-evaluated in the light of parallelism. Parallel processing has, moreover, given us the opportunity to develop algorithms anew with a parallel structure specifically in mind.

In this paper, we present a novel algorithm for the simulation dynamics of open multibody chains that is suitable for parallel implementation. The algorithm differs from much of the current work in the field in two respects. First, it was developed with an eye to parallelism from the outset, rather than attempting to parallelize an existing serial procedure. Second, we have sought a high-level, or coarse-grain, parallelism instead of low-level, or fine-grain, parallelism. The result is an algorithm which possesses a *macroparallel* structure, as it exploits the inherent structure in the problem. Thus, one processing unit is dedicated to each body in the chain.

The algorithm is founded on an iterative solution for the constraint forces which, indeed, renders the algorithm's macroparallelism possible. Moreover, only nearest-neighbor communication is required among processors. The algorithm has been validated on a serial computer and the results reveal that the computational cost of the algorithm is only marginally worse than $O(N)$ in serial implementation. However, the algorithm is almost completely macroparallelizable, and hence the computational time for an N-body chain on an appropriately architectured N-processor system would increase only slightly with N.

2 Previous Work

A number of investigators have looked into a parallel approach to the *inverse-dynamics* problem. Most of this work has been based on the *recursive* Newton-Euler equations. The recursive procedure, however, has a serial structure and hence its potential for *high-level* parallelism is very limited. With this realization, considerable effort has been concentrated on developing various methods to achieve some degree of concurrency in the inverse dynamics calculation at the low-operation level. This approach was followed by Lathrop (1985), Binder and Herzog (1986), Lee and Chang (1986), and more recently Hashimoto and Kimura (1989).

There has been relatively little work on the development of parallel methods for *simulation* (*forward*) dynamics of manipulators. Unlike the case of inverse dynamics, parallelism in simulation dynamics has been investigated mainly in the context of global methods of solution.

Kasahara et al. (1987) made the first contribution to parallel processing of robot dynamics simulation. They parallelize all the computations by subdividing them into tasks. The method chosen for evaluation of the dynamics equations is Method 3 of Walker and Orin (1982) and each equation of this algorithm is treated as a task. To solve the linear system for accelerations, Kasahara et al. employ the Gauss-Jordan procedure. Here, a task involves dividing operations and update operations for each row. Finally, the integration of each variable is also treated as a task. One of the unique aspects of this work is the application of two scheduling algorithms, first proposed by Kasahara and Narita (1984), for assigning the tasks onto parallel processors. Kasahara et al. have also implemented their procedure on an actual multiprocessor system, thus demonstrating the efficiency of the simulation.

Lee and Chang (1988) employed Methods 3 and 4 of Walker and Orin (1982) to develop two parallel simulation procedures by utilizing a number of parallel algorithms designed for SIMD computers. Parallelizing Method 3 involves using the techniques of Lathrop (1985) and Lee and Chang (1986) for evaluating the inertia matrix and the bias (force) vector. To solve for accelerations, Lee and Chang have parallelized Cholesky factorization for implementation on a VLSI array processor. The resultant *parallel composite-rigid-body* algorithm has a time complexity of $O(N)$ with $O(N^2)$ processors. The second method consists of parallelizing the computations that have to be executed at each step of the conjugate-gradient iterative procedure. The theoretical time bound achieved for the resulting *parallel conjugate-gradient* method is $O(N)$ for multiplication operations and $O(N \log_2 N)$ for addition operations.

Fijany and Bejczy (1989) have developed several algorithms for parallel computation of the inertia matrix. All of these are based on the *composite rigid-body spatial inertia* method which is a modification of Walker and Orin's (1982) Method 3. The

[1]University of Victoria, Victoria, British Columbia, Canada V8W 3P6.

[2]Institute for Aerospace Studies, University of Toronto, Downsview, Ontario, Canada M3H 5T6.

Contributed by the Dynamic Systems and Control Division of THE AMERICAN SOCIETY OF MECHANICAL ENGINEERS. Manuscript received by the DSCD Division December 12, 1989; revised manuscript received May 1993. Associate Technical Editor: K. Youcef-Toumi.

parallel procedures are derived by application of the recursive doubling algorithm to the homogeneous recurrence relations, in terms of which the inertia matrix is evaluated. Fijany and Bejczy also discuss the mapping of the parallel schemes onto suitable processor arrays and evaluate their communication costs and efficiency.

Finally, we acknowledge the work of Hwang et al. (1988) for developing a parallel processing algorithm based on their own recursive dynamics formulation. For open chains these authors present a task graph to generate inertia matrix and force vector. The resulting methodology amounts to subdividing inherently serial computations in order to achieve some degree of concurrency. A very important feature of this work, however, is that it can be applied to trees and closed-loop configurations. Here, Hwang et al. exploit the natural parallelism of the physical system by carrying out the computations in parallel along the independent branches of the spanning tree-structure.

3 Theoretical Background

In this section, we present the underlying theory for our simulation procedure. For details of the dynamical formulation, readers are directed to Sincarsin and Hughes (1990).

Equations of Motion. The most natural starting point for the development of a dynamical formulation for a chain of bodies. $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_N$, is the motion equations for each body of the system. These are, in essence, the equations of Newton and Euler applied to a rigid body $\mathcal{B}_n$, which can be compactly written as

$$\mathfrak{M}_n \dot{\mathfrak{v}}_n = \mathfrak{J}_{n+1,n}^T \mathbf{f}_{n+1}^n - \mathbf{f}_n^{n-1} + \mathbf{f}_{n,\text{ext}} + f_{n,I} \tag{1}$$

Here,

$$\mathfrak{v}_n \triangleq \begin{bmatrix} \mathbf{v}_n \\ \boldsymbol{\omega}_n \end{bmatrix} \tag{2}$$

is the *generalized* absolute velocity (cf. twist velocity) of $\mathcal{B}_n$ at O_n, $\mathbf{v}_n$ is the translational velocity of $\mathcal{O}_n$ and $\boldsymbol{\omega}_n$ is the rotational velocity of $\mathcal{B}_n$ with respect to inertial space. Accordingly, $\mathfrak{M}_n$ is the generalized mass matrix, $\mathbf{f}_{n+1}^n$ is the (generalized) force acting on $\mathcal{B}_{n+1}$ from $\mathcal{B}_n$, $\mathbf{f}_{n,\text{ext}}$ denotes the external forces not resulting from interbody effects and $\mathbf{f}_{n,I}$ represents the nonlinear inertial effects. The matrix $\mathfrak{J}_{n+1,n}$ is a generalized transformation matrix from $\mathcal{B}_n$ to $\mathcal{B}_{n+1}$ and is dependent upon configuration.

We proceed further by expressing the interbody force $\mathbf{f}_n^{n-1}$ as a combination of the specified control forces $\mathbf{f}_{n,c}$ and the unknown constraint forces $\mathbf{f}_{n,\square}$, acting at joint $\mathcal{J}_n$:

$$\mathbf{f}_n^{n-1} = -\mathcal{P}_n \mathbf{f}_{n,c} - \mathcal{Q}_n \mathbf{f}_{n,\square} \tag{3}$$

where $\mathcal{P}_n$ is a projection matrix and $\mathcal{Q}_n$ is its orthogonal complement.

Kinematical Constraints. The motion equations must be complimented by the appropriate geometrical and kinematical constraints. Thus, we express the generalized velocity of $\mathcal{B}_n$ in terms of that of its inboard neighbor $\mathcal{B}_{n-1}$ and the joint velocity as

$$\mathfrak{v}_n = \mathfrak{J}_{n,n-1} \mathfrak{v}_{n-1} + \mathcal{P}_n \mathfrak{v}_{n\gamma} \tag{4}$$

where $\mathfrak{v}_{n\gamma}$ denotes the free joint rates. We analogously can write down the expression relating the accelerations:

$$\dot{\mathfrak{v}}_n = \mathfrak{J}_{n,n-1} \dot{\mathfrak{v}}_{n-1} + \dot{\mathfrak{J}}_{n,n-1} \mathfrak{v}_{n-1} + \mathcal{P}_n \dot{\mathfrak{v}}_{n\gamma} + \dot{\mathcal{P}}_n \mathfrak{v}_{n\gamma} \tag{5}$$

It will be more convenient, however, to rewrite the kinematical constraint Eq. (5) in a form where the independent joint acceleration, $\dot{\mathfrak{v}}_{n\gamma}$, has been eliminated. This can be done by using the orthogonal complement $\mathcal{Q}_n$ to give

$$\mathcal{Q}_n^T(\dot{\mathfrak{v}}_n - \mathfrak{J}_{n,n-1} \dot{\mathfrak{v}}_{n-1}) = \mathcal{Q}_n^T(\dot{\mathfrak{J}}_{n,n-1} \mathfrak{v}_{n-1} + \dot{\mathcal{P}}_n \mathfrak{v}_{n\gamma}) \tag{6}$$

since $\mathcal{Q}_n^T \mathcal{P}_n = \mathbf{O}$. The above kinematical constraints will be essential to our derivation.

At this point, we have all the equations necessary for a complete description of the motion of the chain. Equations (1) (combined with (3)) and Eqs. (6) for $n = 0, \ldots, N$ are sufficient to solve for the unknowns in the problem—the accelerations, $\dot{\mathfrak{v}}_n$, and the constraint forces, $\mathbf{f}_{n,\square}$.

4 Solution Procedure

The method proposed here involves obtaining an explicit solution for the joint constraint forces. It was originally presented by Baumgarte (1972) for a system of particles, and later extended by Roberson (1978) to rotating rigid bodies, in the context of the constraint stabilization problem.

Equations for Constraint Forces. To solve for constraint forces, let us rewrite Eqs. (1) combined with (3) in the global form:

$$\mathfrak{M}_\Sigma \dot{\mathfrak{v}}_\Sigma = \mathfrak{J}_\Sigma^{-T}(\mathcal{P}\mathbf{f}_c + \mathcal{Q}\mathbf{f}_\square) + \mathbf{f}_{\Sigma,\text{ext}} + \mathbf{f}_{\Sigma,I} \tag{7}$$

where

$$\mathfrak{M}_\Sigma \triangleq \text{diag}\{\mathfrak{M}_0, \mathfrak{M}_1, \ldots, \mathfrak{M}_N\}$$

$$\dot{\mathfrak{v}}_\Sigma \triangleq \text{col}\{\dot{\mathfrak{v}}_0, \dot{\mathfrak{v}}_1, \ldots, \dot{\mathfrak{v}}_N\}$$

$$\mathbf{f}_{\Sigma,\text{ext}} \triangleq \text{col}\{\mathbf{f}_{0,\text{ext}}, \mathbf{f}_{1,\text{ext}}, \ldots, \mathbf{f}_{N,\text{ext}}\} \tag{8}$$

and the rest of the quantities are defined accordingly. Also, the kinematical constraint Eqs. (6) can be assembled into

$$\mathcal{Q}^T \mathfrak{J}_\Sigma^{-1} \mathfrak{v}_\Sigma = \mathcal{Q}^T[-\dot{\mathfrak{J}}_\Sigma^{-1} \mathfrak{v}_\Sigma + \dot{\mathcal{P}} \mathfrak{v}] \tag{9}$$

Substituting for $\dot{\mathfrak{v}}_\Sigma$ from (7) into (9) and rearranging yields a global system of linear equations:

$$\mathbf{A}\mathbf{f}_\square = \mathbf{b} \tag{10}$$

where

$$\mathbf{A} = \mathcal{Q}^T \mathfrak{J}_\Sigma^{-1} \mathfrak{M}_\Sigma^{-1} \mathfrak{J}_\Sigma^{-T} \mathcal{Q} \tag{11}$$

and

$$\mathbf{b} = -\mathcal{Q}^T \mathfrak{J}_\Sigma^{-1} \mathfrak{M}_\Sigma^{-1}[\mathfrak{J}_\Sigma^{-T} \mathcal{P}\mathbf{f}_c + \mathbf{f}_{\Sigma,\text{ext}} + \mathbf{f}_{\Sigma,I}] + \mathcal{Q}^T[-\dot{\mathfrak{J}}_\Sigma^{-1} \mathfrak{v}_\Sigma + \dot{\mathcal{P}} \mathfrak{v}] \tag{12}$$

The solution of the above system gives the values for the unknown constraint forces.

Special Features of the Method. First, let us distinguish the present solution procedure from the two major classes of existing algorithms, global and recursive. In the global formulation, the motion Eqs. (1) are assembled into the global system and the constraint forces are eliminated using a global projection matrix (Hughes, 1985). In a recursive procedure (Sharf and D'Eleuterio, 1988), the constraint forces are effectively removed by carrying out inward recursion from the tip of the chain to the base body. This is followed by outward recursion whereby one solves for accelerations $\dot{\mathfrak{v}}_{n\gamma}$ for each body in succession. The common feature of both classes of algorithms is that one never needs to calculate the constraint forces. Our methodology on the other hand involves solving for the constraint forces explicitly, which makes the present procedure amenable to parallelization. The inherent parallelism moreover possesses a *macrostructure* as shall be shown.

We first observe that the system matrix $\mathbf{A}$ of (11) has a block tridiagonal structure where the block partitioning corresponds to the body partitioning of the chain. This special form of $\mathbf{A}$ allows us to write out (10) as:

$$-\mathbf{L}_n \mathbf{f}_{n-1,\square} + \mathbf{D}_n \mathbf{f}_{n,\square} - \mathbf{U}_n \mathbf{f}_{n+1,\square} = \mathbf{b}_n \quad n = 0, \ldots, N \tag{13}$$

with the proviso that the constraint forces $\mathbf{f}_{-1,\square}$ and $\mathbf{f}_{N+1,\square}$ vanish, since $\mathcal{B}_0$ and $\mathcal{B}_N$ are terminal bodies. In the above,

$-\mathbf{L}_n$, $\mathbf{D}_n$, $-\mathbf{U}_n$ are the three rowblocks of $\mathbf{A}$ and $\mathbf{b}_n$ is defined in accordance with the "body" partitioning of $\mathbf{A}$ so that

$$\mathbf{b} = \text{col}\{\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_N\} \tag{14}$$

Note that the tridiagonal structure of the constraint force matrix is a reflection of the physical topology of the chain: any body $\mathcal{B}_n$ (except for the base and the tip) is connected to only two bodies—its outboard and inboard neighbors.

The matrix $\mathbf{A}$ is symmetric positive-definite, which is a very attractive property for a system of linear equations. Moreover, the tridiagonal structure of $\mathbf{A}$ ensures that it has property $\mathfrak{A}$ and is consistently ordered (in the sense of Hageman and Young, 1981). These characteristics of the system (10) make it particularly amenable to an iterative solution. In fact, it is the application of *iterative* techniques that allows us to develop a macroparallel procedure to solve for the constraint forces.

The foregoing suggests that in a parallel implementation, a computer processor can be dedicated to each body in the chain. The tridiagonal structure of the matrix $\mathbf{A}$ implies that communication of the constraint forces is required between adjacent processors only, which makes the interconnection network very simple. The calculation of the block matrices $\mathbf{L}_n$, $\mathbf{D}_n$ as well as the vectors $\mathbf{b}_n$ can be carried out on each body-dedicated processor for $n = 0, \ldots, N$ in parallel. In addition, the knowledge of the constraint forces allows us to evaluate the accelerations $\dot{\mathbf{v}}_n$ concurrently for all bodies of the chain.

Solution Techniques for Constraint Forces. The kernel of the present method is the determination of the joint constraint forces, $\mathbf{f}_{n,\square}$, by solving the linear system of Eqs. (10). The solution of this system can be achieved by a variety of *direct* and *iterative* techniques. Although there has been some development on parallel direct solvers like, for example, parallel Gaussian elimination (Nelken and Oxley, 1987), we believe that these methods are essentially sequential processes. Hence, parallelizing direct algorithms requires very careful scheduling of subtasks and synchronization which in turn leads to a considerable overhead cost. Also, the performance of parallel direct solvers is very much dependent on a computer architecture. The existing algorithms achieve their optimal computation time if implemented on a computer with a specific architecture and number of processors. Otherwise, the efficiency deteriorates quite dramatically.

In light of the above comments, it appears that iterative methods offer more flexibility and are more versatile. We have chosen the iterative route because an iterative solution for the joint constraint forces fits well into the macroparallel structure of the present simulation dynamics algorithm. The following section describes five representative iterative schemes that we have applied to the solution of the system (10).

Basic Iterative Methods. We start with the three traditional methods: Jacobi, Gauss-Seidel and SOR (successive over-relaxation). The Jacobi iteration applied to the constraint force Eqs. (13) can be written as:

$$\mathbf{D}_n\mathbf{f}_{n,\square}^{(i+1)} = \mathbf{b}_n + (\mathbf{L}_n\mathbf{f}_{n-1,\square}^{(i)} + \mathbf{U}_n\mathbf{f}_{n+1,\square}^{(i)}) \tag{15}$$

The Gauss-Seidel iteration has the form,

$$\mathbf{D}_n\mathbf{f}_{n,\square}^{(i+1)} = \mathbf{b}_n + (\mathbf{L}_n\mathbf{f}_{n-1,\square}^{(i+1)} + \mathbf{U}_n\mathbf{f}_{n+1,\square}^{(i)}) \tag{16}$$

and SOR the form,

$$\mathbf{D}_n\mathbf{f}_{n,\square}^{(i+1)} = \omega(\mathbf{b}_n + \mathbf{L}_n\mathbf{f}_{n-1,\square}^{(i+1)} + \mathbf{U}_n\mathbf{f}_{n+1,\square}^{(i)}) + (1-\omega)\mathbf{D}_n\mathbf{f}_{n,\square}^{(i)} \tag{17}$$

where ω is the over-relaxation parameter and $0 < \omega < 2$ for convergence. All of these schemes can be written (in global form) as the *basic* iterative method, following Hageman and Young (1981),

$$\mathbf{f}_{\square}^{(I=1)} = \mathbf{G}\mathbf{f}_{\square}^{(i)} + \mathbf{k} \tag{18}$$

where $\mathbf{G}$ is the iteration matrix and $\mathbf{k}$ is a known vector. The rate of convergence of the methods of the form (18) depends on the spectral radius of $\mathbf{G}$, $\rho(\mathbf{G})$. In particular, the approximate number of iterations, I, required to achieve convergence to an accuracy ϵ is given by

$$I = \frac{\ln(\epsilon)}{\ln[\rho(\mathbf{G})]}$$

Since the matrix $\mathbf{A}$ is consistently ordered, there exists a definite relationship between the convergence rates of Jacobi, Gauss-Seidel and SOR schemes. To be specific, Gauss-Seidel iteration converges twice as fast as the Jacobi, while SOR converges $2\zeta^{-1}$ as fast, with $0 < \zeta \leq 1$ depending on the value of ω.

From expression (15), it is easy to see that Jacobi iteration is well suited for parallel processing. This is so because the new iterate for the constraint forces, $\mathbf{f}_{\square}^{(i+1)}$ (or $\mathbf{f}_{n,\square}^{(i+1)}$, $n = 0, \ldots, N$), depends only on the old value, $\mathbf{f}_{\square}^{(i)}$. Therefore, the updates for $\mathbf{f}_{n,\square}$ can be carried out concurrently for all the bodies in the chain. By contrast, the Gauss-Seidel and SOR methods require using, in part, the current values of $\mathbf{f}_{\square}$ at each iteration. These algorithms are partially sequential and hence not suitable for a macroparallel implementation.

Table 1 Polynomial acceleration procedures

Chebyshev semi-iterative method
$\mathbf{D}_n\mathbf{f}_{n,\square}^{(i+1)} = \bar{\rho}_{i+1}(\mathbf{b}_n + \mathbf{L}_n\mathbf{f}_{n-1,\square}^{(i)} + \mathbf{U}_n\mathbf{f}_{n+1,\square}^{(i)}) + (1-\bar{\rho}_{i+1})\mathbf{D}_n\mathbf{f}_{n,\square}^{(i-1)}$
$\bar{\rho}_1 = 1,\ \bar{\rho}_2 = \left(1 - \frac{1}{2}\bar{\mu}^2\right)^{-1},\ \bar{\rho}_{i+1} = \left(1 - \frac{1}{4}\bar{\mu}^2\bar{\rho}_i\right)^{-1},\ \bar{\mu} = \rho(\mathbf{B})$
Conjugate gradient Jacobi method
$\mathbf{D}_n\mathbf{f}_{n,\square}^{(i+1)} = \rho_{i+1}[\gamma_{i+1}(\mathbf{b}_n + \mathbf{L}_n\mathbf{f}_{n-1,\square}^{(i)} + \mathbf{U}_n\mathbf{f}_{n+1,\square}^{(i)}) + (1-\gamma_{i+1})\mathbf{D}_n\mathbf{f}_{n,\square}^{(i)}] + (1-\rho_{i+1})\mathbf{D}_n\mathbf{f}_{n,\square}^{(i-1)}$
$\rho_{i+1} = \left(1 - \frac{\gamma_{i+1}}{\gamma_i}\frac{\mathbf{z}^{(i)},\mathbf{D}\mathbf{z}^{(i)})}{\mathbf{z}^{(i-1)},\mathbf{D}\mathbf{z}^{(i-1)})}\frac{1}{\rho_i}\right)^{-1},\ \gamma_{i+1} = \frac{(\mathbf{z}^{(i)},\mathbf{D}\mathbf{z}^{(i)})}{(\mathbf{z}^{(i)},\mathbf{z}^{(i)})}$
$\mathbf{D}\mathbf{z}^{(i)} = \mathbf{b} - \mathbf{A}\mathbf{f}_{\square}^{(i)}$

Accelerated Iterative Methods. We now describe two well developed *polynomial acceleration* procedures as applied to the Jacobi method: *Chebyshev* acceleration and *conjugate-gradient* acceleration. The advantage of the resulting schemes, Chebyshev semi-iterative (J-SI) and conjugate-gradient Jacobi (J-CG), is that they retain the parallel structure of the Jacobi iteration and have improved rates of convergence. The iterative expressions for the constraint forces corresponding to these two accelerated procedures are presented in Table 1. We note that the conjugate-gradient procedure has been expressed as a three-term nonstationary iterative method, rather than in its standard form. This is done in order to expose the macroparallel structure of the algorithm.

The rate of convergence of the Chebyshev semi-iterative procedure is generally comparable to that achieved by Gauss-Seidel iteration. Unlike the basic iterative methods and the J-SI scheme, the rate of convergence of the conjugate-gradient acceleration procedure depends not only on the spectral radius of $\mathbf{G}$, but also on the number and the particular distribution of the eigenvalues in the spectrum. In the absence of rounding errors, the *classical* conjugate-gradient method is guaranteed to converge to the exact solution of the system in a finite number of iterations, that being bounded by the order of the system. However, considerably faster convergence rates have been achieved with the conjugate-gradient acceleration procedure in the solution of *sparse* linear systems (Axelsson, 1985).

5 Computer Simulation

We have presented a number of techniques for obtaining a

solution to the joint constraint forces, which in itself is part of the algorithm to determine the accelerations. Our main objective, however, is to simulate the motion of a multibody chain as it evolves in time. Thus, the accelerations, $\dot{\mathbf{v}}_n$, obtained according to the motion equations must be integrated to give velocities $\mathbf{v}_n$ and, from those, the joint velocities $\mathbf{v}_{n\gamma}$ can be integrated for the joint coordinates. We observe that the integration of $\dot{\mathbf{v}}_n$ and $\mathbf{v}_{n\gamma}$ can be carried out in parallel for $n = 0, \ldots, N$ using any one of the explicit integration formulas. However, for our immediate purpose—that of numerically validating the proposed algorithm on a serial computer—we have chosen to use a commercially available package LSODE. Some special issues had to be addressed in developing the computer implementation of the formulation and to these we devote the following comments.

Convergence Criterion. Any iterative scheme that is carried out in finite precision must be terminated according to some convergence criterion. The convergence test that we have implemented in our simulation is of the form: If

$$\frac{\|\mathbf{f}_{\square}^{(i)} - \mathbf{f}_{\square}^{(i-1)}\|}{\|\mathbf{f}_{\square}^{(i)}\|} \leq \epsilon$$

then set $\bar{\mathbf{f}}_{\square} = \mathbf{f}_{\square}^{(i)}$ and stop the iteration; otherwise, calculate the next iterate, $\mathbf{f}_{\square}^{(i+1)}$. We have used the overbar in $\bar{\mathbf{f}}_{\square}$ to differentiate the solution obtained by using an iterative procedure from the true solution of the system, $\mathbf{f}_{\square}$.

The error ϵ admissible in the value of the constraint forces must be specified in accordance with the accuracy of the integration. When integrating with a package like LSODE, the user is required to prescribe a value TOL which is used by the routine for error control and step-size selection. Since ϵ introduces a perturbation of the same order in the values of accelerations $\dot{\mathbf{v}}_n$, it seems only reasonable to have $\epsilon \leq$ TOL (it makes no sense to require the result of integration of $\dot{\mathbf{v}}_n$ to be more accurate than its value.) Also, the choice $\epsilon \leq$ TOL will ensure that the accelerations $\dot{\mathbf{v}}_n$ have a time history profile that is "continuous" to within TOL. Otherwise, the noise introduced by the error perturbation will cause the integration procedure to become extremely inefficient. Based on extensive numerical testing, we have found that $\epsilon \approx$ TOL/10 usually yields the best overall performance of the simulation.

Initial Guess for the Constraint Forces. The iterative scheme to solve for $\mathbf{f}_{\square}$, must be started with some initial guess. Since the constraint forces must be determined at every time step in the simulation, it is natural to employ the constraint forces calculated at previous time steps for making an initial guess to start the iteration at a current time step. The most straightforward implementation of this idea is to set

$$\mathbf{f}_{\square}^{(0)}(t_k) = \bar{\mathbf{f}}_{\square}(t_{k-1}) \tag{19}$$

i.e., the initial guess for $\mathbf{f}_{\square}$ at time t_k is equal to the value at the previous time t_{k-1}. This notion can be extended to produce more sophisticated initial guesses by using the values of $\bar{\mathbf{f}}_{\square}$ at any number of previous time steps, combined with an appropriate extrapolation scheme. In our computer simulation, we have implemented a cubic polynomial extrapolation. This seems to be a reasonable trade-off between the improvement in the initial guess and the additional computational cost incurred by the extrapolation procedure.

6 Numerical Results and Discussion

In this section, we discuss the performance of our procedure when applied to the simulation of a rigid-link manipulator. The numerical experiments reported here demonstrate: (i) the relative performance of different iterative schemes; and (ii) computational efficiency as a function of the number of bodies in the system.

For all of the following results, the manipulator depicted in Fig. 1 was used. It is a 3-link anthropomorphic manipulator with a total of five degrees of freedom. Its geometric and inertial properties can be found in Sharf and D'Eleuterio (1988). Simulation results are presented for two maneuvers—planar (Maneuver I) and three-dimensional (Maneuver II) also described in the aforementioned reference. The numerical integration was performed at TOL $= 10^{-6}$ and, consistent with the comments of Section 5, the convergence tolerance for the iterative procedures was set to 10^{-7}.

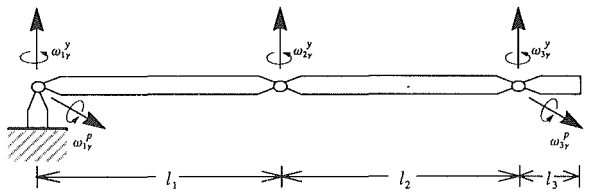

Fig. 1 Manipulator system

Table 2 Simulation statistics for $T = 10$ s, TOL $= 10^{-6}$, $\epsilon = 10^{-7}$

	Manueuver I/Maneuver II		
Algorithm	% accuracy at T	average # of iterations	CPU (s)
PA-Jacobi	0.05/0.02	26/1483	12.2/470
PA-J-SI	0.03/0.08	16/694	10.0/164
PA-J-CG	0.05/0.05	7/12	8.8/8.8
Gauss-Seidel	0.03/0.04	14/197	9.0/63
SOR	0.03/0.05	10/150	8.6/39

Table 2 contains simulation statistics for five iterative solutions discussed in Section 4, three of these denoted by PA represent ≈100% parallelizable methods. For completeness, we have also included the results for Gauss-Seidel and SOR methods, the latter being implemented with a nearly optimal value for ω. The CPU statistics in Table 2 were obtained by running the simulations on a serial computer, the SUN 3/60 workstation and hence, do not represent the optimal run time for the parallel algorithms. Nevertheless, the runs on a single processor allow us to make a relative comparison of different iterative techniques.

As can be seen from Table 2, there is little variation in the speeds of the iterative algorithms for the first maneuver. The situation is radically different for Maneuver II where the conjugate-gradient procedure is clearly the most efficient. Such a variation between the J-CG and the other techniques arises because the spectral radius of the Jacobi iteration matrix **B**, and as a consequence that of J-SI, Gauss-Seidel and SOR is very close to unity for this particular trajectory. The performance of these methods deteriorates accordingly as a large number of iterations is required to achieve convergence. Based on a series of numerical experiments conducted with different configurations and trajectories, we conclude that the J-CG method yields consistently the most efficient results. We also note that all iterative schemes provide similar accuracy in the solution.

In the context of multibody dynamics problem, the general performance of a solution algorithm should be assessed by determining its dependence on the number of links in the chain. Here, we investigate this relationship numerically for the best among the iterative techniques considered, the J-CG method. For the purpose of comparison with another class of solution procedures, an analogous set of results was obtained with the recursive (R) method.

The numerical data for chains made up of rigid bodies with one-degree-of-freedom joints are depicted in Figs. 2(a) and 2(b). In the former, we have plotted the CPU times averaged over 1000 solutions against the number of bodies in the chain

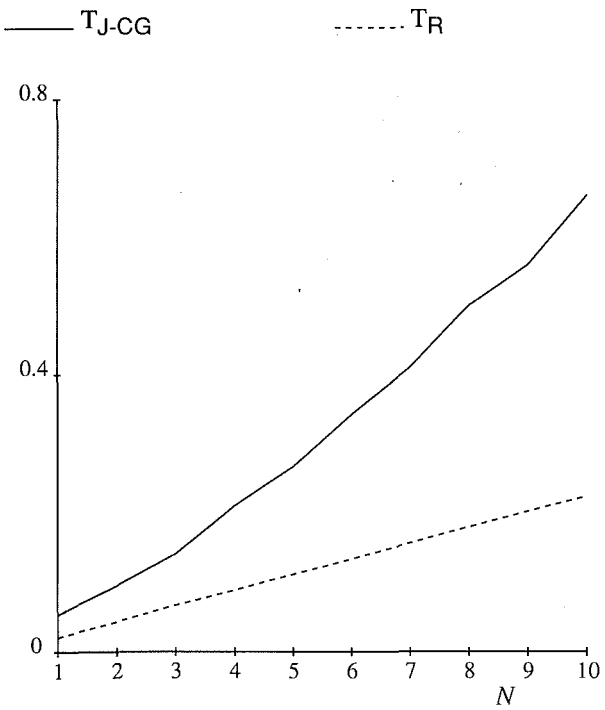

Fig. 2(*a*) Sun 3/60 CPU time versus *N*

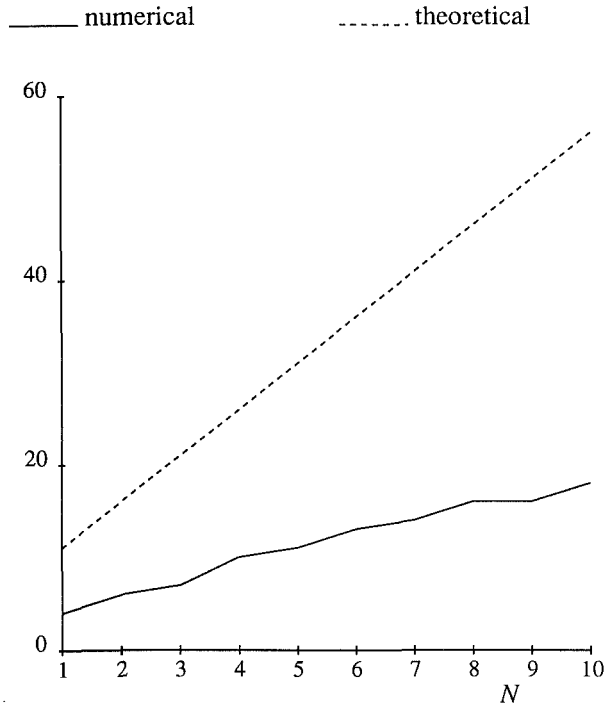

Fig. 2(*b*) Number of iterations for J-CG versus *N*

for the two aforementioned procedures. Figure 2(*b*) shows the number of iterations for the J-CG method. Here, the solid line represents the average value obtained from our simulation, while the dashed line is the theoretical limit which is equal to the dimension of the matrix **A**. It is evident that the practical implementation of the J-CG iteration in the simulation is considerably more efficient. This improvement results from the use of good initial guesses and a judiciously chosen value for the iteration convergence tolerance ϵ.

From Fig. 2(*a*), the CPU time of J-CG procedure can be approximated by an $O(N^{\alpha})$ function with $\alpha = 1.12$. One might be tempted to describe this computational complexity as being

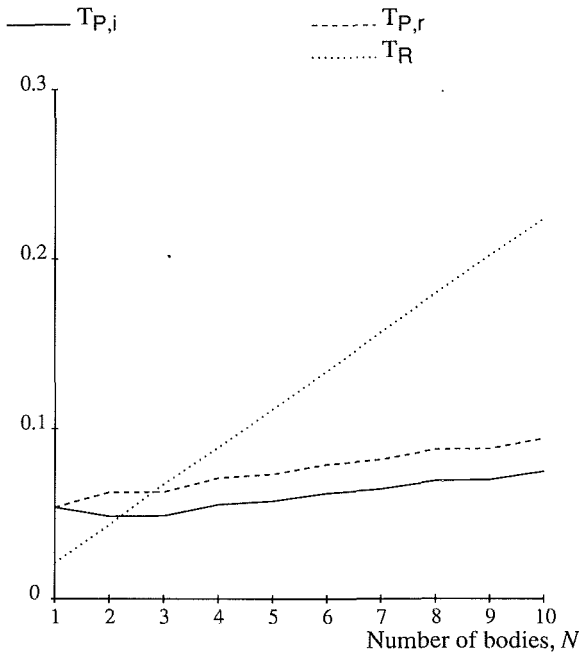

Fig. 3 Computational speeds of J-CG and recursive algorithms

paralinear (meaning "beyond linear") in the number of bodies. Similar results were obtained for systems of rigid bodies with two degrees of freedom per joint. The corresponding value of α for the solution algorithm based on the J-CG iteration is 1.17. These results are consistent with the findings of Axelsson (1985) who reports that, for sparse matrices, a conjugate-gradient procedure possesses a computational complexity of $O(N^{\beta})$, where β ranges from about 1.17 to 1.25 depending on the system. We also observe that Fig. 2(*a*) displays the familiar $O(N)$ performance of the recursive procedure.

To complete this section, we would like to *estimate* the parallel efficiency of the algorithm developed in this paper. To this end, we first determine the theoretical lower bound for the parallel complexity which represents the ideal performance of the algorithm. It is calculated with:

$$T_{P,i} = \frac{(T_{CPU} - T_{serial})}{N} + T_{serial} \tag{20}$$

where T_{CPU} is the CPU time for one solution for the accelerations, T_{serial} denotes the time taken by the part of the algorithm which is not macroparallelizable, and the subscript $(\bullet)_{P,i}$ stands for "parallel, ideal." The quantity $(T_{CPU} - T_{serial})$ measures the CPU for the macroparallel part of the procedure and, accordingly, has been divided by the number of processors N. This presumes that each processor contributes equally to the algorithm's macroparallel part which will largely be true if each joint has the same number of degrees of freedom. In any case, $T_{P,i}$ does serve as a lower bound.

Next, we improve on the estimate of (20) by accounting for the overhead costs that are incurred when a parallel algorithm is implemented on a real parallel processing system. For this purpose, we use the results of Kasahara et al., (1987) and allocate approximately 30% of the parallel processing time to the associated overhead. With this additional cost, we suggest a more realistic estimate for the parallel speed of our solution procedures. It is:

$$T_{P,r} = 1.3 \times \frac{(T_{CPU} - T_{serial})}{N} + T_{serial} \tag{21}$$

where the parallel execution time has been multiplied by a factor of 1.3.

Figure 3 shows the plots of $T_{P,i}$ and $T_{P,r}$ calculated using

the CPU data for the J-CG algorithm, previously shown in Fig. 2(*a*). From Fig. 3, one can observe that the predicted realistic speed of the J-CG method increases only marginally with the number of bodies (and processors). The shallow growth in run time due to the slightly nonlinear nature of T_{CPU} and the linear, though small, component T_{serial}. In the same plot, we include the CPU time for the recursive algorithm, T_R. Comparison of $T_{P,r}$ and T_R demonstrates the potential speed-up that can be achieved by exploiting macroparallelism in the simulation dynamics problem. We emphasize, however, that the parallel efficiency presented for the J-CG solution is only a preliminary estimate. A definitive statement on the performance of all the algorithms introduced in this paper awaits implementation on a suitably-designed parallel-processing system.

7 Concluding Remarks

In this paper, we have developed a new solution procedure for the simulation dynamics of open chain manipulator systems. The procedure as described here applies to rigid-link manipulators. However, its extension to flexible-body chains as well as tree configurations can be accommodated. The advantage of the proposed method lies in its suitability for parallel implementation as the solution for the constraint forces can be obtained using parallel iterative techniques and the accelerations of the bodies can also be evaluated concurrently. Moreover, knowledge of the system constraint forces in itself proves essential for certain applications such as manipulator design.

The proposed algorithm has been implemented and validated in a serial-computer simulation. We have also applied a number of iterative schemes to solve for the constraint forces and conclude that the conjugate-gradient acceleration of the Jacobi method, J-CG, gives the best results. Numerical tests on a serial computer indicate that the execution time of the solution algorithm with the conjugate-gradient Jacobi iteration varies paralinearly with the number of links. An estimate for the parallel efficiency of this procedure demonstrates the potential for achieving more efficient dynamics simulation of multibody systems by exploiting the high-level parallelism available in the problem.

Acknowledgments

The development of the computer simulation has been conducted at McGill Research Center for Intelligent Machines. This research has been funded by the Natural Sciences and Engineering Research Council of Canada, the Institute for Space and Terrestrial Science and the Institute for Robotics and Intelligent Systems.

References

Axelsson, O., 1985, "A Survey of Preconditioned Iterative Methods for Linear Systems of Algebraic Equations," *BIT*, Vol. 25, pp. 166–187.

Bae, D. S., and Haug, E. J., "A Recursive Formulation for Constrained Mechanical System, Part I—Open Loop," *Mechanics of Structures and Machines*, Vol. 15, No. 3, 1987.

Baumgarte, J., 1972, "Stabilization of Constraints and Integrals of Motion in Dynamical Systems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 1, pp. 1–16.

Binder, E. E., and Herzog, J. H., 1986, "Distributed Computer Architecture and Fast Parallel Algorithms in Real-Time Robot Control," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-16, No. 4, pp. 543–549.

D'Eleuterio, G. M. T., 1992, "Dynamics of Elastic Multibody Chains: Part C—Recursive Dynamics," *Int. J. Dynamics and Stability of Systems*, Vol. 7, No. 2, pp. 61–89.

Featherstone, R., 1983, "The Calculation of Robot Dynamics Using Articulated-Body Inertias," *Int. J. Robotics Research*, Vol. 2, 1, pp. 13–30.

Fijany, A., and Bejczy, A. K., 1989, "A Class of Parallel Algorithms for Computation of the Manipulator Inertia Matrix," *Proc. IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, pp. 1818–1826.

Golla, D. F., 1987, "MDSF Theoretical Model Report: A New Recursive Solution of the Motion Equations for a Chain of Elastic Bodies," Spar-SS-TM.116.

Hageman, L. A., and Young, D. M., 1981, *Applied Iterative Methods*, Academic Press, New York.

Hashimoto, K., and Kimura, H. K., 1989, "A New Parallel Algorithm for Inverse Dynamics," *Int. J. Robotics Research*, Vol. 8, No. 1, pp. 63–76.

Hughes, P. C., 1985, "Multibody Dynamics for Space Station Manipulators: Dynamics of a Chain of Elastic Bodies," Dynacon Report SS-2.

Hwang, R. S., Bae, D. S., and Haug, E. J., 1988, "Parallel Processing for Real-Time Dynamic System Simulation," *Advances in Design Automation*, Kissimmee, FL, pp. 509–517.

Kasahara, H., and Narita, S., 1984, "Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing," *IEEE Trans. Comput.*, Vol. C-33, No. 11, pp. 1023–1029.

Kasahara, H., Fujii, H., and Iwata, M., 1987, "Parallel Processing of Robot Motion Simulation," *Proc. IFAC 10th World Congress*, Pergamon Press.

Lathrop, L. H., 1985, "Parallelism in Manipulator Dynamics," *Int. J. Robotics Res.*, Vol. 4, No. 2, pp. 80–102.

Lee, C. S. G., and Chang, P. R., 1986, "Efficient Parallel Algorithm for Robot Inverse Dynamics Computations," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-16, No. 4, pp. 532–542.

Lee, C. S. G., and Chang, P. R., 1988, "Efficient Parallel Algorithm for Robot Forward Dynamics Computation," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 18, No. 2, pp. 238–251.

Nelken, I., and Oxley, D., 1987, "Parallel Gaussian Elimination on an Optically Interconnected Data Flow Computer," *Mathematics and Computers in Simulation*, Vol. 29, pp. 515–529.

Roberson, R. E., 1978, "Constraint Stabilization for Rigid Bodies: an Extension of Baumgarte's Method," *IUTAM Symposium on Dynamics of Multibody Systems*, Munich, 1977, Berlin-Heidelberg-New York, Springer, pp. 274–289.

Rodriguez, G., 1987, "Kalman Filtering, Smoothing and Recursive Robot Arm Forward and Inverse Dynamics," *IEEE J. Robotics and Automation*, Vol. RA-3, No. 6, pp. 624–639.

Sharf, I., and D'Eleuterio, G. M. T., 1988, "Computer Simulation of Elastic Chains Using a Recursive Formulation," *Proc. IEEE International Conference on Robotics and Automation*, Philadelphia, PA, pp. 1539–1547.

Sincarsin, G. B., and Hughes, P. C., 1990, "Dynamics of Elastic Multibody Chains: Part A—Body Motion Equations," *Int. J. Dynamics and Stability of Systems*, Vol. 4, No. 3, pp. 209–226.

Walker, M. W., and Orin, D. E., 1982, "Efficient Dynamic Computer Simulation of Robot Mechanisms," ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 104, pp. 205–211.