Parallel Implementation of a Low Order Algorithm for Dynamics of Multibody Systems on a Distributed Memory Computing System

Accepted for publication in the Journal *Engineering with Computers*

S. Duan [1] and K.S. Anderson [2]
Department of Mechanical Engineering, Aeronautical Engineering, and Mechanics
Rensselaer Polytechnic Institute
Troy, NY 12180-3590

Abstract

In this paper, a new hybrid parallelizable low order algorithm, developed by the authors for multibody dynamics analysis is implemented numerically on a distributed memory parallel computing system. The presented implementation can currently accommodate the general spatial motion of chain systems, but key issues for its extension to general tree and closed loop systems are discussed. Explicit algebraic constraints are used to increase coarse grain parallelism and to study the influence of the dimension of system constraint load equations on the computational efficiency of the algorithm for real parallel implementation using the Message Passing Interface (MPI). The equation formulation parallelism and linear system solution strategies which are used to reduce communication overhead are addressed. Numerical results indicate that the algorithm is scalable, that significant speed-up can be obtained, and that a quasi logarithmic relation exists between time needed for a function call and numbers of processors used. This result agrees well with theoretical performance predictions. Numerical comparisons with results obtained from independently developed analysis codes have validated the correctness of new hybrid parallelizable low order algorithm and demonstrated certain computational advantages.

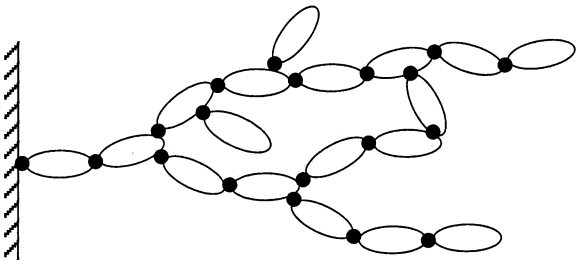
1 Introduction

How to effectively and efficiently exploit the concurrent nature of multibody system (MBS) dynamic behavior simulation has been receiving increasing attention since the first parallel MBS algorithm developed by Kasahara *et al* in 1987 [1]. A variety of formulations and dynamics simulation algorithms have been put forward by individuals whose interests span in a wide range of applications including robotics, vehicular dynamics, biomechanics, and aerospace. In 1988 Bae and Haug applied their parallel algorithm to an off-road vehicle with a suspension system that had eight closed loops [2]. Also in 1988, Lee presented a global $O(n^3)$(the number of computations required per temporal integration step increase a cubic function of the number of degrees of freedom n) algorithm for the simulation of the behavior of large flexible space structures [3]. In 1990, Anderson applied his parallelizable $O(n)$ algorithm to an eight appendage spacecraft with multi-degree of freedom joints [4]. Sharf in 1993 presented a parallel procedure with application to a manipulator system [5]. Eichberger in 1994 introduced a new parallel $O(n)$ residual algorithm and applied it to the analysis of motion of an off-road vehicle [6][7]. And in 1993 and 1995, Fijany *et al* produced their parallel procedures for spacecraft manipulator arm applications [8][9]. Each of these works have from their inception been dedicated to exploiting the concurrent nature of many aspects of multibody systems dynamics analysis, ideally yielding increases in simulation speed.

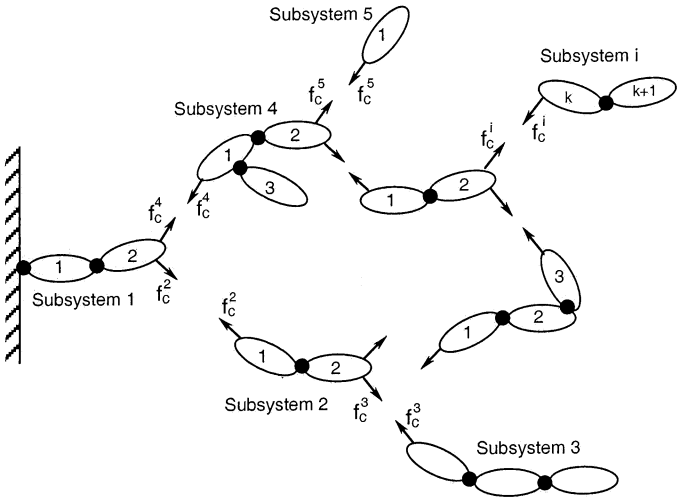
Recently, Anderson and Duan have developed a parallelizable *state space-full descriptor* low order algorithm to improve computational efficiency of MBS [10] [11][13]. The MBS model is constructed through the separation of certain key system interbody joints so that largely independent multibody subchain systems are formed. These subchains in turn interact with one another through associated unknown

[1] Doctoral Graduate Student
[2] Assistant Professor

constraint loads $\underline{f_c}$ at those separated joints as shown in figure 1. The added parallelism is obtained through this separation and the explicit determination of associated constraint loads by parallel iterative methods at a coarse grain level. In other words, an efficient sequential dynamics analysis procedure is



(A) General Multibody System



(B) One of many possible associated subchain systems

Figure 1: A multibody system and associated set of subsystems (subchains)

carried out to form and solve equations of motion within each of the subchains and parallel strategies are used to form and solve constraint load equations between subchains concurrently. The resulting parallel algorithm may be considered a hybridization of different methods at three levels: *i)* A hybridization between parallel computations and a sequential $O(n)$ dynamics analysis procedure; *ii)* A hybridization between descriptor form and state space form dynamic system formulations; and, *iii)* A hybridization between direct and iterative solution schemes at each integration step for the solution of system state derivatives and constraint loads.

The system equations of motion and constraint equations which result from this procedure have the following form respectively

$$\underline{M}(t, \underline{q}) \cdot \dot{\underline{u}} = \underline{K}(t, \underline{q}, \underline{u}, \underline{f_c})$$

$$\Rightarrow \quad \underline{\dot{u}} = \underline{RHS}(t, \underline{q}, \underline{u}, \underline{f_c}), \tag{1a}$$

$$\underline{\Gamma}(t, \underline{q}) \cdot \underline{f_c} = \underline{\Psi}(\underline{q}, \underline{u}, t). \tag{1b}$$

In this formulation, the system mass matrix $\underline{M}$ is not explicitly formed. Instead the $O(n)$ dynamic analysis algorithm used considers $\underline{M}$ implicitly, and effectively produces the equations of motion in the form of (1a) directly. In equations (1a) and (1b), $\underline{q}$ are the generalized coordinates used to define the configuration of the MBS, $\underline{u}$ are generalized speeds which characterize the motion of the system, and $\underline{\dot{u}}$ are the time derivatives of these generalized speeds (generalized accelerations) which are to be determined. The matrix $\underline{RHS}$ (Right Hand Side) consists of all applied forces, stiffness terms, constraint forces, plus that portion of inertia forces and torques associated with Coriolis and centripetal accelerations. In general, the elements of matrix $\underline{RHS}$ are nonlinear functions of the system state and time. $\underline{\Gamma}$ is the constraint load coefficient matrix and $\underline{\Psi}$ is composed of nonlinear functions of the state variables. Finally $\underline{f_c}$ are the unknown constraint load measure numbers which must exist at cut joints between subchains so that the subsystems each behave correctly as part of the original uncut system.

Systems of equation (1a) and (1b) have characteristics of both the state space form and descriptor form. As is common with standard state space formulation, relative generalized coordinates are used and the resulting system mass matrix $\underline{M}$, if it were actually formed, would be of small dimension and densely populated. Additionally, as with descriptor formulations, absolute coordinates are used and the resulting constraint load coefficient matrix $\underline{\Gamma}$ is generally sparse. Once the values of constraint load measure numbers $\underline{f_c}$ are determined in algebraic constraint load equation (1b), these values are then substituted into (1a), which is a set of ordinary differential equations (ODE), to be solved for the generalized accelerations $\underline{\dot{u}}$. So the equations (1a,1b) represent a set of differential algebraic equations (DAE). This special form favors the use of hybrid direct-iterative solution schemes of linear equations. The recursive sequential $O(n)$ procedures takes advantage of direct methods and is used to form and solve equations of motion (1a), and form constraint load equation (1b) for each individual subchain. Due to the sparseness of constraint load coefficient matrix $\underline{\Gamma}$, iterative methods may be efficiently used to solve for the constraint loads in parallel.

However, on the negative side, iterative methods can have convergence difficulties and the number of iterations required for convergence depends highly on the eigenvalue spectra of the system of equations and the quality of the initial estimate of the solution. Many iterative methods need a good initial guess to reduce iterative time. Due to these difficulties and uncertainties, care must be exercised to insure that intelligent (cost and time effective) parallel solution of (1b) is realized. Toward this end, each block row of $\underline{\Gamma}$ in (1b), which is associated with the degrees-of-freedom (DOF) of a cut joint, is assigned to one processor and an iterative method, such as Parallel Preconditioned Conjugate Gradient (PPCG) method with Jacobi iterative method as a preconditioner, is used. In this way, each processor can iterate individually to obtain part of $\underline{f_c}$ starting from the initial solution estimate. If a good initial solution estimate is not available, because no information exists on which to form a legitimate estimate may be made (e.g. $t = 0$, or because of discontinuous forcing), then a direct method such as LU decomposition may be used to solve constraint equations (1b). Except when discontinuities in forcing occurs, the state variables from the converged solution of the previous time step (i.e. a *zero order hold* [[14]]) are used as the initial solution estimate of the current time step. If the system is sufficiently well behaved, then the quality of this initial estimate may be improved markedly using a higher order approximation such as a *first* or *second order hold.*

This paper is organized as follows. In section 2, parallel issues of the algorithm are presented. Starting with the system of equations formed by the hybrid parallelizable lower order algorithm, section 3 addresses how to combine parallel strategies with the recursive sequential $O(n)$ dynamic analysis procedure to form and solve the equations of motion, while reducing the associated communication overhead. Then the parallelization of system constraint load equations will be presented in section 4. In section 5, the new parallelizable algorithm, which has be coded in C and MPI, will be applied to a multibody chain. The parallel implementation and performance of this study case on an IBM SP2 distributed memory parallel

computing system will be presented. Finally, discussions and some concluding remarks will be given in section 6. The details of the derivation of the hybrid parallelizable lower order algorithm are presented in references [10] [11]. Notational conventions used in this paper are same as those used in reference [11].

2 Parallel Issues Presented in Algorithm

Consider a system as shown in figure 1. For the purpose of this implementation, a system of cut subchains is used. After separation of certain key joints, constraint forces $\vec{f}_c^{\,i}$ and constraint moments $\vec{l}_c^{\,i}$ appear between the terminal body of the subchain i and the base body of the subchain $i+1$ to insure the system of subchains continuing to behave as the original interconnected systems. Then each subchain may be assigned to a processor of a parallel computing system as shown in figure 2. This physical parallel model

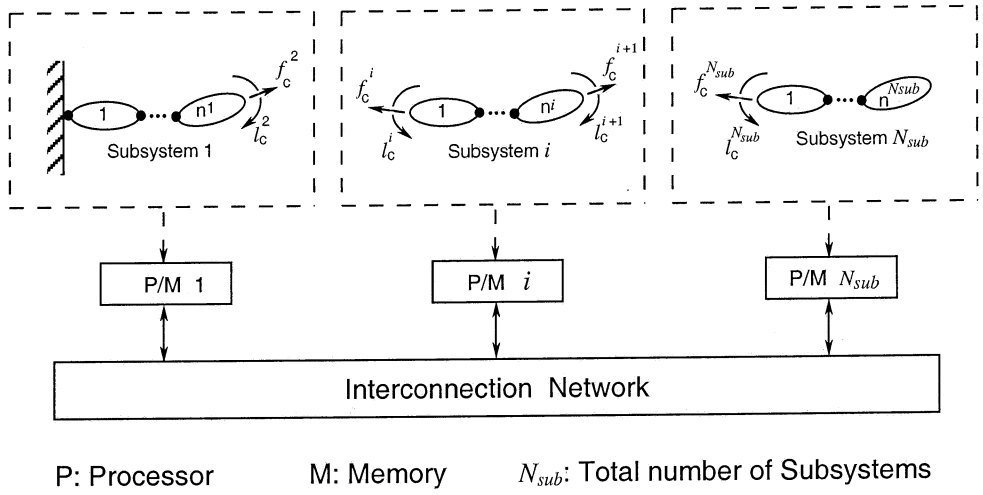


Figure 2: Parallelism of Physical System of Subchains

is mathematically represented by equation (1a) and (1b).

There are six key computing tasks in this parallel model. These principal tasks are: *i)* The formation of all kinematic quantities associated(1a); *ii)* The formation and subsequent solution of those kinetic portions of the equation of motion (1a) associated with all forces other than the unknown constraint loads; *iii)* The formation and subsequent solution of those kinetic portions of the equation of motion (1a) associated only with the still unknown constraint loads; *iv)* The formation/assembly of constraint load equations (1b); *v)* The solution of constraint load equations (1b) for the constraint load measure numbers; and, *vi)* The determination of state derivative values and their subsequent temporal integration.

Data parallelism, control parallelism, and precedence constraints or precedence relations which do not favor parallelism exist between certain of these tasks. Precedence constraints exist between the determination of all kinematic quantities associated with task *i)* and all remaining tasks. Also, one would expect that some form of precedence constraints should exist between the formation of equations (1a) and their subsequent solution for $\underline{\dot{u}}$, mentioned in both tasks *ii)* and *iii)*. Indeed, one would expect here that a set of equations cannot be solved until have they have been formed. The use of the $O(n)$ dynamic formulation blurs the distinction between the formulation of the equations of motion and their solution for $\underline{\dot{u}}$. Moreover, the procedure reduces the number of required operations by effectively decomposing and solving

the equations of motion as they are being formed, thus melding the independent formulation and solution tasks associated with traditional dynamic modeling formulation into a single task. Thus tasks *ii)* and *iii)* can proceed in parallel, but the construction of the constraint load equations (1b) in task *iv)*, must follow task *iii)*. Also, the constraint load measure numbers $\underline{f}_c$ associated with task *v)*, which must follow *iv)*, must be determined before state derivatives values of task *vi)* may be explicitly determined and temporally integrated.

Additionally, multiple functional units (subchains) apply almost the same operations prescribed in tasks *i)-iv)* simultaneously to elements of a set of data (inertia, forces, kinematic, and geometric quantities) without necessity of communication. A k-fold increase in the number of functional units (subchains) leads to an k-fold increase in the throughput of the system during all stages except those associated with some parts of task *iv)* and all of task *v)*. In other words, the algorithm may take advantage of data parallelism for the formation and solution of much of equations (1a) and (1b). Ideally, the algorithm is scalable until minimum functional unit, which is each single separated body, is reached. Therefore, a control-data hybrid parallelism is produced in the algorithm. These features of precedence constraints and control-data hybrid parallelism are roughly presented in figure 3.

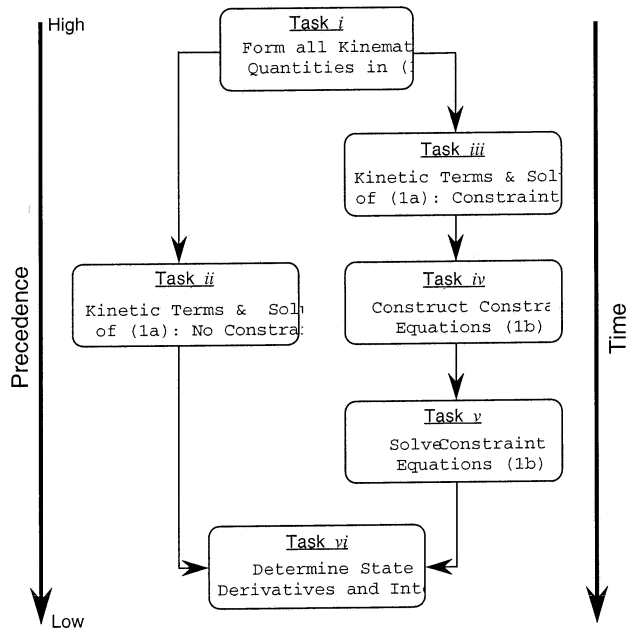


Figure 3: Precedence and Parallelism between Tasks

3 Parallel Implementation of Equations of Motion

As mentioned above, there are six tasks associated with the formulation and solution of the equations (1a) and (1b). In this section, the algorithmic formulations for the parallel determination and solution strategies used for implementation of the equation of motion (1a) will be presented.

In general when one describes the dynamics of a complex multibody system, it becomes necessary to keep track of quantities associated with key points J_k of body B_k belonging to subsystem i, as measured by an observer in some reference frame N. To aid in this description the rigorous notation shown in figure 4 is adopted.

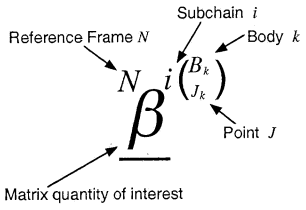


Figure 4: Notational convention

Based on Kane's dynamic formulations [16], generalized speeds u^{ik} associated with the of k-th degree-of-freedom of subchain i may be chosen to describe motion of subchain i for a system shown in figure 1 ($k = 1, 2, ..., \nu^i$ and $i = 1, ..., Nsub$). ν^i is the total number of degrees of freedom (DOF) of subchain i and $Nsub$ is the total number of subchains. Then as defined in reference [10][11], generalized accelerations can be expressed in the form

$$\dot{u}^{ik} = \eta^{ik} + \zeta^{ik} \qquad (k = 1, 2, ..., \nu^i) \tag{2}$$

where the scalar quantity η^{ik} is that portion of the generalized acceleration which is associated with the unconstrained motion of the system, while ζ^{ik} is that portion of the generalized acceleration which is explicit in the unknown constraint loads.

Quantity ζ^{ik} can be recursively determined by the relations

$$\zeta^{ik} = \underline{D_b}^{iB_k} \cdot \underline{f}_c^{i6} + \underline{D_t}^{iB_k} \cdot \underline{f}_c^{i(\nu^i+1)} \qquad (k = 1, 2, ..., \nu^i). \tag{3}$$

with

$$\underline{D_t}^{iB_k} = \frac{\left(\underline{\mathcal{P}}^{i\left(\begin{smallmatrix}J_k\\B_k\end{smallmatrix}\right)}\right)^T}{M^i_{kk}} \cdot \left[\underline{\tau}_t^{ik} - {}^{iB_k}\underline{\hat{I}}^{i\left(\begin{smallmatrix}J_k\\B_k\end{smallmatrix}\right)} \cdot \left({}^{iB_{k-1}}\underline{S}^{iB_k}\right)^T \cdot \underline{\bar{D}_t}^{iB_{k-1}}\right] \qquad (k = 1, 2, ..., \nu^i) \tag{4}$$

and

$$\underline{D_b}^{iB_k} = \frac{\left(\underline{\mathcal{P}}^{i\left(\begin{smallmatrix}J_k\\B_k\end{smallmatrix}\right)}\right)^T}{M^i_{kk}} \cdot \left[\underline{\tau}_b^{ik} - {}^{iB_k}\underline{\hat{I}}^{i\left(\begin{smallmatrix}J_k\\B_k\end{smallmatrix}\right)} \cdot \left({}^{iB_{k-1}}\underline{S}^{iB_k}\right)^T \cdot \underline{\bar{D}_b}^{iB_{k-1}}\right] \qquad (k = 1, 2, ..., \nu^i) \tag{5}$$

where

$$\underline{\bar{D}_t}^{iB_k} = \left({}^{iB_{k-1}}\underline{S}^{iB_k}\right)^T \cdot \underline{\bar{D}_t}^{iB_{k-1}} + \underline{\mathcal{P}}^{i\left(\begin{smallmatrix}J_k\\B_k\end{smallmatrix}\right)} \cdot \underline{D_t}^{iB_k} \qquad (\underline{\bar{D}_t}^{iB_0} = 0) \tag{6}$$

$$\underline{\bar{D}_b}^{iB_k} = \left({}^{iB_{k-1}}\underline{S}^{iB_k}\right)^T \cdot \underline{\bar{D}_b}^{iB_{k-1}} + \underline{\mathcal{P}}^{i\left(\begin{smallmatrix}J_k\\B_k\end{smallmatrix}\right)} \cdot \underline{D_b}^{iB_k} \qquad (\underline{\bar{D}_b}^{iB_0} = 0) \tag{7}$$

While quantity η^{ik} can be determined by using the recursive relations

$$\eta^{ik} = \frac{\left(\underline{\mathcal{P}}^{i\left(\begin{smallmatrix}J_k\\B_k\end{smallmatrix}\right)}\right)^T}{M^i_{kk}} \cdot \left[-{}^{iB_k}\underline{\hat{I}}^{i\left(\begin{smallmatrix}J_k\\B_k\end{smallmatrix}\right)} \cdot \left({}^{iB_k}\underline{S}^{iB_{k+1}}\right)^T \cdot \underline{\tilde{\eta}}^{i(k-1)} + \underline{\hat{F}}^{iB_k}\right] \qquad (k = 1, 2, ..., \nu^i) \tag{8}$$

where

$$\underline{\tilde{\eta}}^{ik} = \left({}^{iB_k}\underline{S}^{iB_{k+1}}\right)^T \cdot \underline{\tilde{\eta}}^{i(k-1)} + \underline{\mathcal{P}}^{i\binom{J_k}{B_k}} \cdot \eta^{ik} \qquad (k = 1, 2, ..., \nu^i) \tag{9}$$

with boundary condition $\underline{\tilde{\eta}}^{i0} = \underline{0} \in \mathcal{R}^{6\times 1}$.

In equation (3), quantities $\underline{D_t}^{iB_k}$ and $\underline{D_b}^{iB_k}$ are shift-triangularized and backsubstituted constraint Boolean matrices, while quantities $\underline{f}_c^{i6}$ and $\underline{f}_c^{i(\nu^i+1)}$ are constraint load components acting on base body and terminal body of subchain i, respectively, as shown in figure 2. In equation (4)-(8), matrix ${}^{iB_k}\underline{\hat{I}}^{i\binom{J_k}{B_k}}$ is the composite inertia matrix of body k of subchain i, and is recursively determined by

$${}^{iB_k}\underline{\hat{I}}^{i\binom{J_k}{B_k}} = {}^{iB_k}\underline{I}^{i\binom{J_k}{B_k}} + {}^{iB_k}\underline{\mathcal{T}}^{iB_{k+1}} \cdot {}^{iB_{k+1}}\underline{\hat{I}}^{i\binom{J_{k+1}}{B_{k+1}}} \cdot \left({}^{iB_k}\underline{S}^{iB_{k+1}}\right)^T \qquad \in \mathcal{R}^{6\times 6}. \tag{10}$$

Matrix $\underline{\mathcal{P}}^{i\binom{J_k}{B_k}}$ is the partial velocity (or *free modes of motion* [12]) matrix associated with DOF of joint k of subchain i. The matrix ${}^{iB_{k-1}}\underline{S}^{iB_k}$ appearing in (6)-(9) is a linear transformation or "shift" matrix, which shifts forces and inertias from the inboard joint of body k to an equivalent force/moment system acting on the inboard joint of body $k-1$. Scalar quantity M_{kk}^i in (4,5,8), is the diagonal element of the system mass matrix $\underline{M}$ associated with equation (1a) affiliated with the DOF k of subsystem i. This quantity may be recursively determined from

$$M_{kk}^i = \left(\underline{\mathcal{P}}^{i\binom{J_k}{B_k}}\right)^T \cdot {}^{iB_k}\underline{\hat{I}}^{i\binom{J_k}{B_k}} \cdot \underline{\mathcal{P}}^{i\binom{J_k}{B_k}}. \tag{11}$$

While the matrix ${}^{iB_{k-1}}\underline{\mathcal{T}}^{iB_k}$ is the *triangularization operation* matrix which recursively triangularizes the equations as they are being formed and is determined from

$${}^{iB_{k-1}}\underline{\mathcal{T}}^{iB_k} = {}^{iB_k}\underline{S}^{iB_{K+1}} \cdot \left\{\underline{U} - \frac{1}{M_{(k)(k)}^i} \cdot {}^{iB_{k-1}}\underline{\hat{I}}^{i\binom{J_k}{B_k}} \cdot \underline{\mathcal{P}}^{i\binom{J_k}{B_k}} \cdot \left(\underline{\mathcal{P}}^{i\binom{J_k}{B_k}}\right)^T\right\} \in \mathcal{R}^{6\times 6}. \tag{12}$$

Matrices $\underline{\tau_b}^{ik}$ and $\underline{\tau_t}^{ik}$ in(3)(5) are the shift-triangularization matrices, which shift constraint measure numbers $\underline{f_c}^{i6}$ and $\underline{f_c}^{i(\nu^i+1)}$ from their original acting point on the base body and terminal body, respectively, to the proximal joint point of body k during triangularization procedure of the hybrid $O(n)$ algorithm. Finally, the quantity $\underline{\hat{F}}^{iB_k}$ in (8) is the composite force matrix of body k of subchain i, and is determined recursively from

$$\underline{\hat{F}}^{iB_k} = \underline{F}^{iB_k} + {}^{iB_k}\underline{\mathcal{T}}^{iB_{k+1}} \cdot \underline{\hat{F}}^{iB_{k+1}} \qquad \in \mathcal{R}^{6\times 1}. \tag{13}$$

Within subchain i, all of quantities in equation (2)-(13) except the constraint load measure numbers are formed in a sequential recursive manner using the hybrid $O(n)$ procedure [10][11]. Since each subchain is assigned to a processor as shown in figure 2, formations of these quantities associated with each individual subchain are completely independent. In the other words, they can be produced on each processor in parallel at a coarse grain level without necessity of message passing between processors or subchains. State variables, however, which describe relative motion between subchains, and constraint load measure numbers are coupled between adjacent subchains. These quantities play a key role of liaison between subchains so some modest communication costs will be expended for them. To reduce communication costs associated with these quantities, a parallel strategy of boundary data overlapping between subchains or processors is introduced. This approach is similar to that used in the fields of finite difference methods for solving Laplace's equation by using a *red-black decomposition* [17]. To do this, an imaginary body is introduced and attached to terminal body of subchain 1 to $Nsub - 1$ as shown in figure 5. The imaginary body takes the state variables between subchains. These state variables between subchains may be obtained by using the difference between state of the terminal body of subchain i and state of base body of subchain

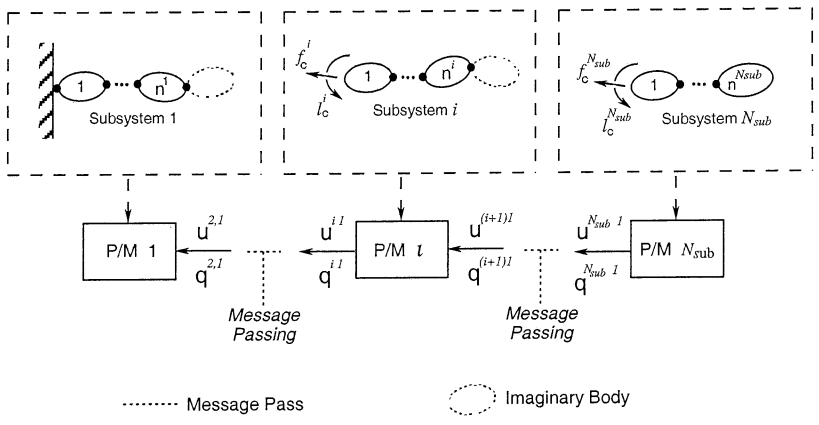


Figure 5: Message Passing Using Boundary Data Overlap

i's distal subchain, projected on to the *free mode of motion* of the actual connecting joint. Then these differences, which are state variables values between subchains, are assigned to the imaginary body. In this way, only processors associated with adjacent subsystems exchange information through message passing. If message passing interface (MPI) is used, then only simple point to point communication is needed to accomplish this. In other words, the state variables of base body of subchain/processor i will be sent to subchain/processor $i-1$, and subchain/processor $i-1$ will receive these messages from subchain/processor i at the beginning of each integration time step as indicated in figure 5.

Equations (2)-(13) are algorithmic formulations of equations of motion (1a) for the whole of system ($i = 1, ..., Nsub$ and $k = 1, ..., \nu^i$). Since a state space sequential $O(n)$ dynamics analysis procedure is used, the formation and solution of equations of motion of each subchain can be performed simultaneously. Once the constraint load measure numbers $\underline{f}_c^{i6}$ and $\underline{f}_c^{i(\nu^i+1)}$ are determined, current generalized accelerations speeds can be obtained through the use of these equations concurrently on each processor. Further, both the generalized speeds and generalized accelerations may be integrated on each processor in parallel to obtain updated state variables for next time step.

4 Parallel Implementation of Constraint Equation

There are two key tasks which should be considered for parallel implementation of the constraint load equations. The first is the parallel formation of the constraint load equations, and then the parallel solution of these constraint equations.

4.1 Forming Constraint Equation in Parallel

In section 2 and 3, it has been shown that formation of constraint load equations and formation of equations of motion are carried out concurrently. As shown in references [10][11], the constraint load equations, which define constraint conditions of separated joint between subchain i and subchain $i+1$, can be express in the form

$$-(\sum_{k=1}^{\nu^i} \underline{G}_1^{ik} \cdot \underline{D_b}^{iB_k}) \cdot \underline{f}_c^{(i-1)n^{i-1}} + [(\sum_{k=1}^{\nu^i} \underline{G}_1^{ik} \cdot \underline{D_t}^{iB_k}) + (\sum_{k=1}^{6} \underline{G}_2^{(i+1)k} \cdot \underline{D_b}^{(i+1)B_k})] \cdot \underline{f}_c^{in^i} -$$
$$(\sum_{k=1}^{6} \underline{G}_2^{(i+1)k} \cdot \underline{D_t}^{(i+1)B_k}) \cdot \underline{f}_c^{(i+1)n^{i+1}} = (\sum_{k=1}^{6} \underline{G}_2^{(i+1)k} \cdot \eta^{(i+1)1}) - (\sum_{k=1}^{\nu^i} \underline{G}_1^{i1} \cdot \eta^{ik}) - \underline{H}^i \qquad (14)$$

with

$$\underline{G}_1^{ik} = \underline{L}_1^{ik} \cdot \underline{\mathcal{P}}^{i\binom{J_k}{B_k}}, \qquad (15)$$

$$\underline{L}_1^{ik} = \underline{L}_1^{i(k+1)} \cdot \left({}^{iB_k}\underline{S}^{iB_{(k+1)}}\right)^T, \tag{16}$$

$$\underline{L}_1^{i(\nu^i+1)} = \underline{\mathcal{O}}^{i(\nu^i+1)}, \tag{17}$$

and

$$\underline{G}_2^{(i+1)k} = \underline{L}_2^{(i+1)k} \cdot \underline{\mathcal{P}}^{(i+1)\binom{J_k}{B_k}} \qquad (k < 7), \tag{18}$$

$$\underline{L}_2^{(i+1)k} = \underline{L}_2^{(i+1)(k+1)} \cdot \left({}^{(i+1)B_k}\underline{S}^{(i+1)B_{(k+1)}}\right)^T \qquad (k < 6), \tag{19}$$

$$\underline{L}_2^{(i+1)6} = \underline{\mathcal{O}}^{i(\nu^i+1)}, \tag{20}$$

and

$$\underline{H}^i = \underline{\mathcal{O}}^{i(\nu^i+1)} \cdot [\left({}^{iB_{\nu^i}}\underline{S}^{iB_{\nu^i}}\right)^T \cdot \underline{\mathcal{A}}_t^{iB_{\nu^i}} + {}^{iB_{\nu^i}}\underline{\mathcal{A}}_t^{iB_{\nu^i+1}} - \underline{\mathcal{A}}_t^{(i+1)B_1}]. \tag{21}$$

Equations (14)-(21) are algorithmic formulations of constraint equation (1b) for the entire subchain system. In equation (14), quantities $\underline{f}_c^{(i-1)n^{i-1}}$, $\underline{f}_c^{in^i}$ and $\underline{f}_c^{(i+1)n^{i+1}}$ are the constraint load measure numbers associated with the separated joints between subchain $i-1$ and i, subchain i and $i+1$, and subchain $i+1$ and $i+2$, respectively. Matrices $\underline{G}_1^{ik}$ and $\underline{G}_2^{i1}$ are those portions of the constraint coefficient matrix associated with outboard joint of terminal body and inboard joint of base body, if a subchain is separated at both ends. These quantities play the role of implicitly shifting the DOF's of the entire subchain to the separated joints and project these DOF's onto the constraint subspace associated with the separated joints. Matrix $\underline{\mathcal{O}}^{i(\nu^i+1)}$ in equation (17) is the orthogonal complement to the free mode of motion matrix $\underline{\mathcal{P}}^{i\binom{J_{\nu^i+1}}{B_{\nu^i+1}}}$. The joint orthogonal complement $\underline{\mathcal{O}}^{i(\nu^i+1)}$ defines the constraint subspace associated with the separated outboard joint on terminal body of subchain i. Matrices $\underline{L}_1^{ik}$ and $\underline{L}_2^{ik}$ in equations (15)-(20) are intermediate quantities, which are introduced for the convenient expression of recursive relations in an algorithmic way. In equation (21), quantity $\underline{\mathcal{A}}_t^{iB_{\nu^i}}$ is the *acceleration remainder term* [16], which represents all kinematic acceleration terms of the terminal body of subchain i, which are not explicit in the elements of $\underline{\dot{u}}$. Similarly, the quantity ${}^{iB_{\nu^i}}\underline{\mathcal{A}}_t^{iB_{\nu^i+1}}$ is the acceleration remainder term matrix, associated with all of the terms of relative acceleration matrix not explicit in $\dot{u}$s of imaginary body ν^i+1 relative to the terminal body ν^i of subchain i.

If equation (14), which is associated with a separated joint i, is assigned to a processor, then $Nsub$ processors are needed to form constraint equation and $Nsub-1$ processors are needed to solve constraint equation for a chain system in parallel since inboard joint of base body of subchain 1 is fixed on Newtonian frame as shown in figure 2 and 3. With this in mind, from (14) it is clear that there is some modest message passing needed between adjacent processors or subchains. For the sake of convenient use of parallel strategies, these message passings are sorted into two groups. One group are messages which are passed for the formation of constraint load equations (1b), such as quantities $\underline{G}_2^{(i+1)1}$, $\underline{D_b}^{(i+1)B_1}$, $\eta^{(i+1)k}$, $\underline{\mathcal{A}}_t^{(i+1)B_1}$ and so on. The other group are messages, which are passed for the solution of constraint equations, such as $\underline{f}_c^{(i-1)n^{i-1}}$ and $\underline{f}_c^{(i+1)n^{i+1}}$.

In the hybrid parallelizable $O(n)$ algorithm [10] [11], quantities $\underline{\mathcal{A}}_t^{(i+1)B_1}$, $\underline{G}_2^{(i+1)k}$, $\underline{D_b}^{(i+1)B_1}$ and $\eta^{(i+1)k}$ are formed at different stages. The acceleration remainder matrix $\underline{\mathcal{A}}_t^{(i+1)B_1}$ is a kinematic quantity produced during task *i)* shown in figure 3, while quantities $\underline{G}_2^{(i+1)k}$, $\underline{D_b}^{(i+1)B_1}$ and $\eta^{(i+1)k}$ are determined in task *iii)*. Once determined, these quantities are not needed until the assembly of the constraint equations (14) in

task *iv)*. Consequently, these quantities may all be easily saved then used as needed since only modest storage are needed. This offers the direct benefit that each individual processor can largely perform the hybrid $O(n)$ procedure independently and in a highly data parallel manner through four stages of the procedure without any communication. Some message passing strategies, such as grouping message passing or buffered message passing, can be used in this context to reduce overall costs associated with time of starting up communication. At the stage of assembling constraint load equation (1b), these quantities are passed from subchain/processor i to subchain/processor $i-1$. Since only adjacent subchains/processors need such communication, a relatively simple message passing strategy, as indicated in figure 6 is needed for the formation of constraint load equation (1b).

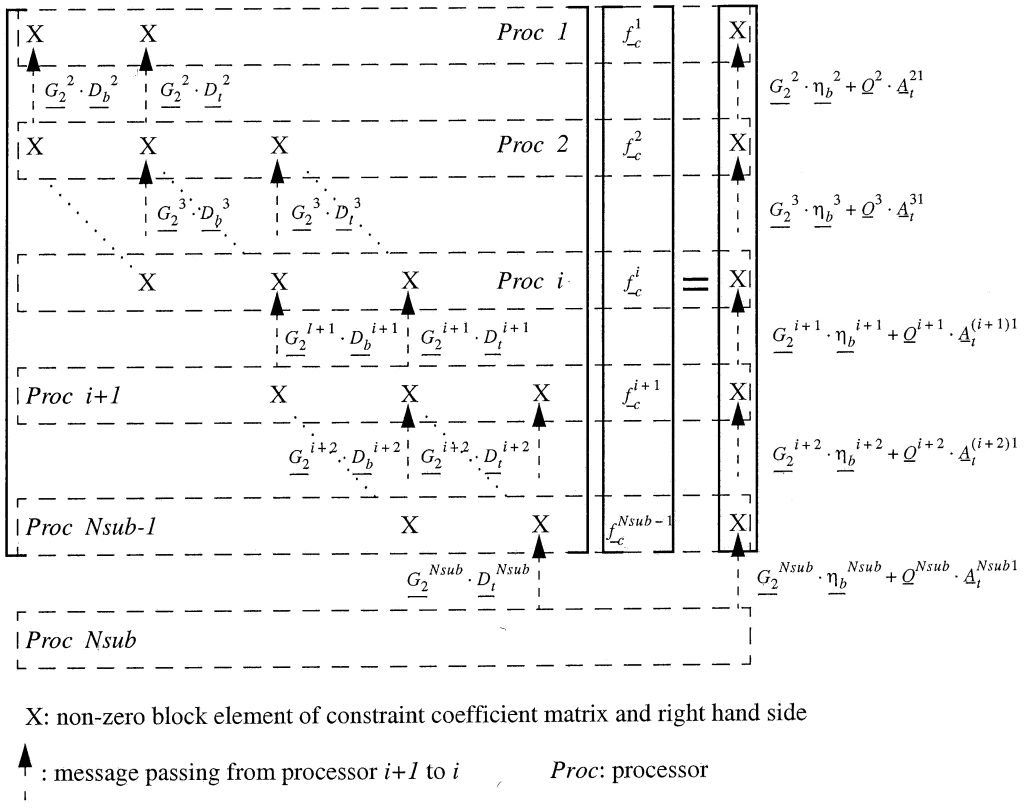


Figure 6: Group Message Passing for Formation of Constraint Equation (1b)

If a more general system topology is considered (e.g. one possessing multiple branches and/or closed loops) then the relation between processors becomes somewhat more involved, being governed by the relation for processor/subsystem i

$$-(\sum_{k=1}^{\nu^i} \underline{G_1}^{ik} \cdot \underline{D_b}^{iB_k}) \cdot \underline{f_c}^{jn^j} + [(\sum_{k=1}^{\nu^i} \underline{G_1}^{mk} \cdot \underline{D_t}^{mB_k}) \cdot \underline{f_c}^{mn^m} + (\sum_{k=1}^{6} \underline{G_2}^{lk} \cdot \underline{D_b}^{lB_k})] \cdot \underline{f_c}^{ln^l} -$$

$$(\sum_{k=1}^{6} \underline{G_2}^{(l)1} \cdot \underline{D_t}^{(l)B_1}) \cdot \underline{f_c}^{(l)n^l} = (\sum_{k=1}^{6} \underline{G_2}^{lk} \cdot \eta^{lk}) - \sum_{k=1}^{\nu^i} (\underline{G_1}^{mk} \cdot \eta^{mk}) - \underline{H}^i, \qquad (22)$$

i = Subchain currently under consideration, (i),

$$
\begin{aligned}
j &= \text{Proximal subchain to subchain } i,\ (j = pr[i]),\\
k &= \text{Body (DOF) being considered,}\\
l &= \text{Distal subchains to subchain } i,\ (l = dist[i]),\\
m &= \text{Distal subchains to subchain } j,\ (m = dist[j]).
\end{aligned}
$$

MPI was chosen to accomplish the presented parallel computing strategies. Two outstanding features of MPI [18] are that it is portable and independent from parallel computing systems, as well as easy to implement. There are three mechanisms for grouping individual data items into a single message in MPI [18]. These methods are: *i)* the count parameter, *ii)* derived data types, and *iii)* MPI_Pack/MPI_Unpack. Each of these has its own advantages and disadvantages. Since quantities $\underline{A_l}^{(i+1)B_1}$, $\underline{G_2}^{(i+1)k}$, $\underline{D_b}^{(i+1)B_1}$, $\eta^{(i+1)k}$, and $\underline{\mathcal{O}}^{i(\nu^i+1)}$ all have the same data type, the simplest method is the use of the *count parameter*, to group and send these messages. To do so, an intermediate array is formed on each distal processor to be used to keep all these quantities consecutively addressed. A *count parameter* is used to count total length of message. Once the grouped message is received by each proximal processor, different segments of this message will be distributed and assigned to the appropriate matrix quantities on that proximal processor.

4.2 Solving Constraint Load Equations in Parallel

Once the constraint load equations are formed, a parallel iterative method can be used to solve them for the unknown constraint load measure numbers $\underline{f_c}$. For current systems, a special parallel preconditioned conjugate gradient solver (PPCG) has been developed to solve constraint equations (1b). One of features of this PPCG solver is its adaptability to the changes of topology of multibody systems. The detailed discussion about this PPCG solver will be given in another paper. A flow chart indicating the parallel implementation of a function call to hybrid $O(n)$ algorithm is presented in figure 7.

5 Numerical Results of Parallel Implementation of a Multibody Chain

The algorithm associated with this work is coded in C with MPI, and has been run on Rensselaer's IBM SP2 distributed memory parallel computing system. The numerical experiments reported here have been designed to demonstrate:

1. The validity of the parallelizable hybrid lower order algorithm.
2. The computational efficiency of the algorithm as a function of the numbers of cuts (processors used) in the simulation of the system dynamic behavior.

For this purpose, the modeling and simulation of a 16 body chain has been chosen as shown in figure 8. Each body of the chain is identical, and bodies are connected to each other by simple revolute joints. The properties of each body are: mass $m^k = 1.0$ kg; inertial $I^k = [1,0,0;0,1,0;0,0,1]$ kg $\cdot$ m^2, position vector $\bar{s}^k = [0,-2.0,0]$ m; position vector $\bar{r}^k = [0,-1.0,0]$ m. The initial conditions of the system are represented by generalized coordinates $q^k = 0.1745$ rad (k=1,...,12) and $q^k = -0.1745$ rad (k=13,...,16), Generalized speeds $u^k = 0.0\ \frac{\text{rad}}{\text{sec}}$.

The numbering of each body/joint increases sequentially from the base body/joint 1 to terminal body/joint 16, and five different cases of joint separation are applied to this chain system.

case 1. 0 cut (pure sequential $O(n)$ procedure),

case 2. 1 cut (chain separated into 2 subchains at joint 9),

case 3. 3 cuts (chain separated into 4 subchains at joint 5, 9 and 13),

case 4. 7 cuts (chain separated into 8 subchains at odd number of joint starting with 3),

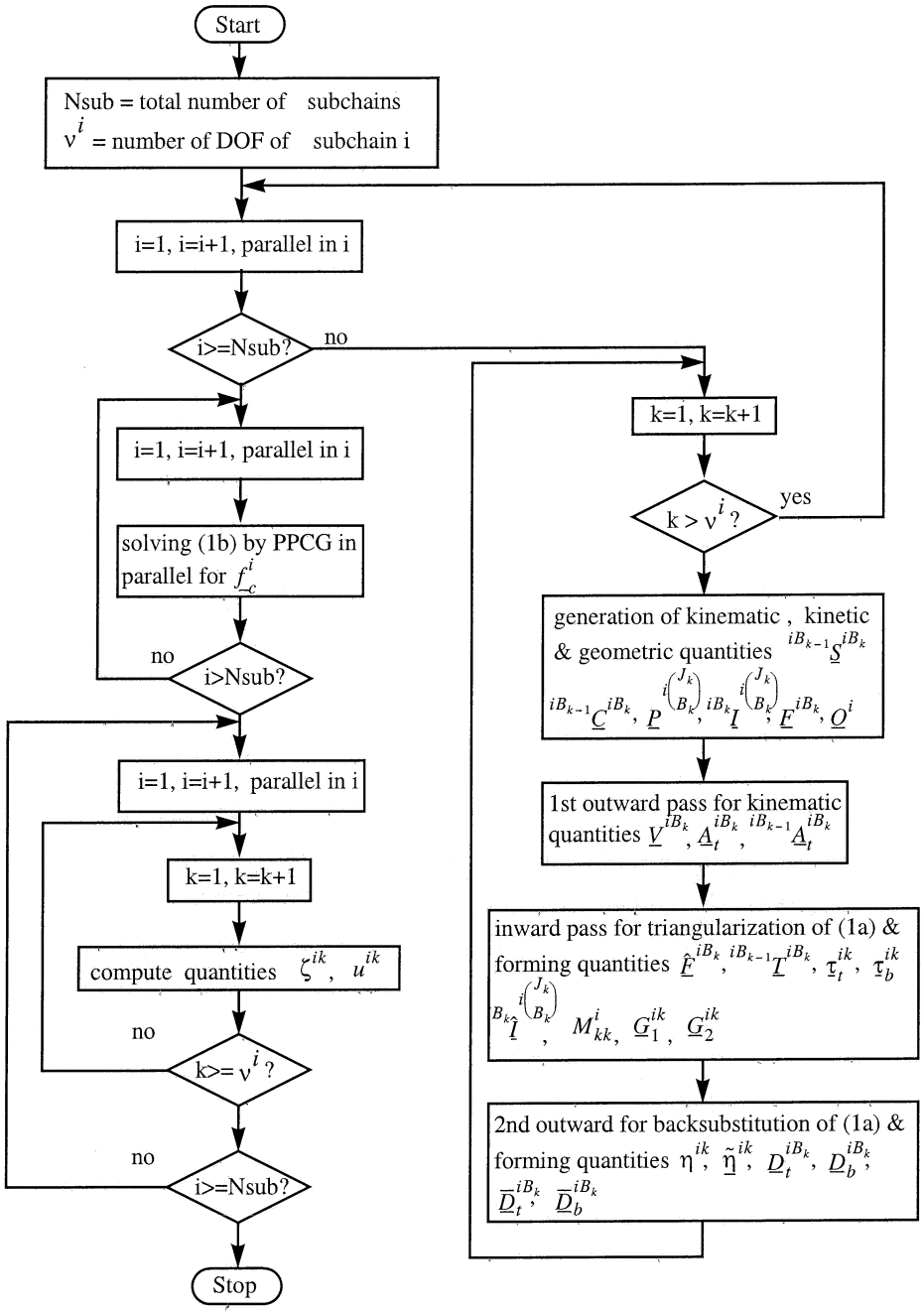
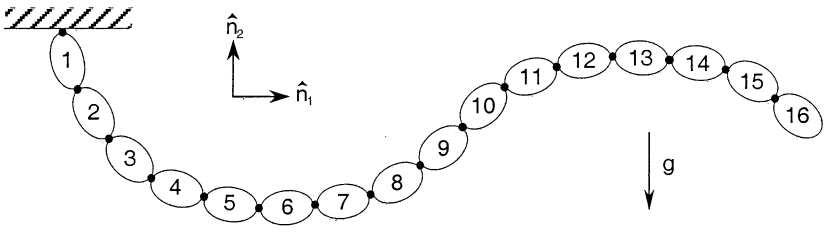


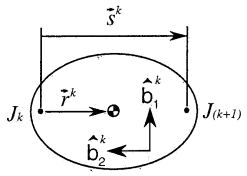
Figure 7: Flow Chart of Hybrid $O(n)$ Algorithm

case 5. 15 cuts (chain separated into 16 subchains at all joints except joint 1).

The time histories of the generalized coordinates q^k are shown in figure 9 and agree perfectly with those produced by the commercial dynamics analysis software *AUTOLEV*. Motion traces associated with each body of each subchain for case 3 are shown in figure 10. From figure 10, it can be seen that the state variables associated with each of the four subchains are updated in parallel. Due to constraints



(A) Multibody Chain System



$\hat{n}_j$ = unit vectors fixed in Newtonian frame (j=1,2,3)

$\hat{b}_j^k$ = unit vectors fixed in body k (j=1,2,3)

$\vec{s}^k$ = position vector from proximal joint J_k to distal joint $J_{(k+1)}$ of body k

$\vec{r}^k$ = position vector from proximal joint J_k to mass center of body k

(B) Geometric parameters of body k

Figure 8: A 16 Body Chain

imposed between subchains, the motion of outboard tip of each proximal subchain must be exactly same as the motion of inboard tip of each distal subchain. If the motion trace of each subchain is superposed on another, the motion trace of entire chain can be obtained as shown in "Entire Chain" portion of figure 10.

Figure 11 indicates the time cost of a function evaluation of the hybrid parallelizable

$O(n)$ algorithm as a function of the number of processor used. The experimentally obtained timing data agrees well with the theoretically anticipated computational cost of

$$\text{Cost} \approx C_1\frac{n}{N_p} + C_2\frac{nm}{N_p^2} + m^\gamma(C_3 + C_4\log_2(N_p - 1)) \tag{23}$$

n : Total number of system degrees of freedom

m : Total number of system constraints

N_p : Number of processors used

γ : Parameter describing iterative solution scheme performance

C_i : Coefficients $(i = 1, \ldots, 4)$.

The nature of the algorithm is such that as a greater the number of joints are separated ($\approx N_p - 1$), the subsystems become smaller and less of a computational load exists on each processor. Thus, an overall system saving in computational time costs will arise from distributing the load over additional processors if the savings achieved offset the added expenses associated with communication overhead and the iterative solver. If the number of iterations varies only slightly with the increasing of dimension of constraint equation or number of cuts, then communication overhead will mainly determine the computational efficiency of the algorithm.

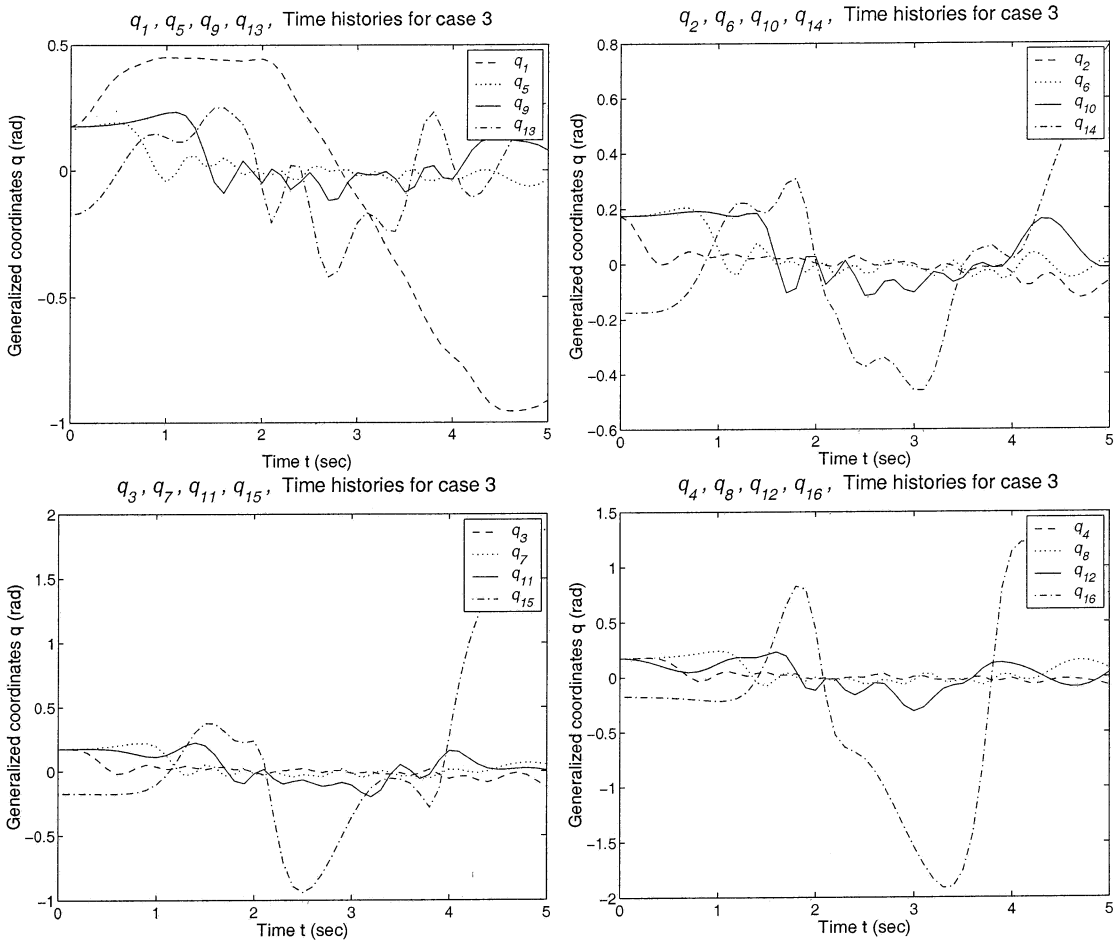


Figure 9: Time Histories of Generalized Coordinates

Since a Parallel Preconditioned Conjugate Gradient method is used, the calculation of some norm values are needed at each integration time step. MPI uses a bitree communication architecture to realize the determination of the values of these norms. Therefore, a logarithmic relation, such as that indicated in (23), should appear between numbers of cuts made and time of function evaluation needed. Figure 11 shows how the fitting curve associated with the theoretical number of operations (23) agrees with computational times obtained from controlled timing runs.

For case 3, the hybrid algorithm was implemented involving 3 cut joints with both sequential and parallel versions of the resulting simulation code being produced. These codes were in turn run both sequentially on a single processor and concurrently on four processors of the same parallel computing system. This results in a factor of *Speedup* ≈ 3.75 for parallel implementation relative to sequential implementation of the algorithm for this same case. The penalty here is that the number of iterations required in the iterative solution of (1b) increases as the number of algebraic constraints m ($\approx$proportional to the number of processors in this formulation) raised to the power γ, where for this system and the preconditioning used, gamma was found to be $\gamma \approx 0.739$. Indeed, the majority of the CPU time associated with this simulation was spent in the solution of the constraint load equations (1b). Consequently, any increase in the number of iterations required to solve of the constraint load measure numbers has a very direct affect

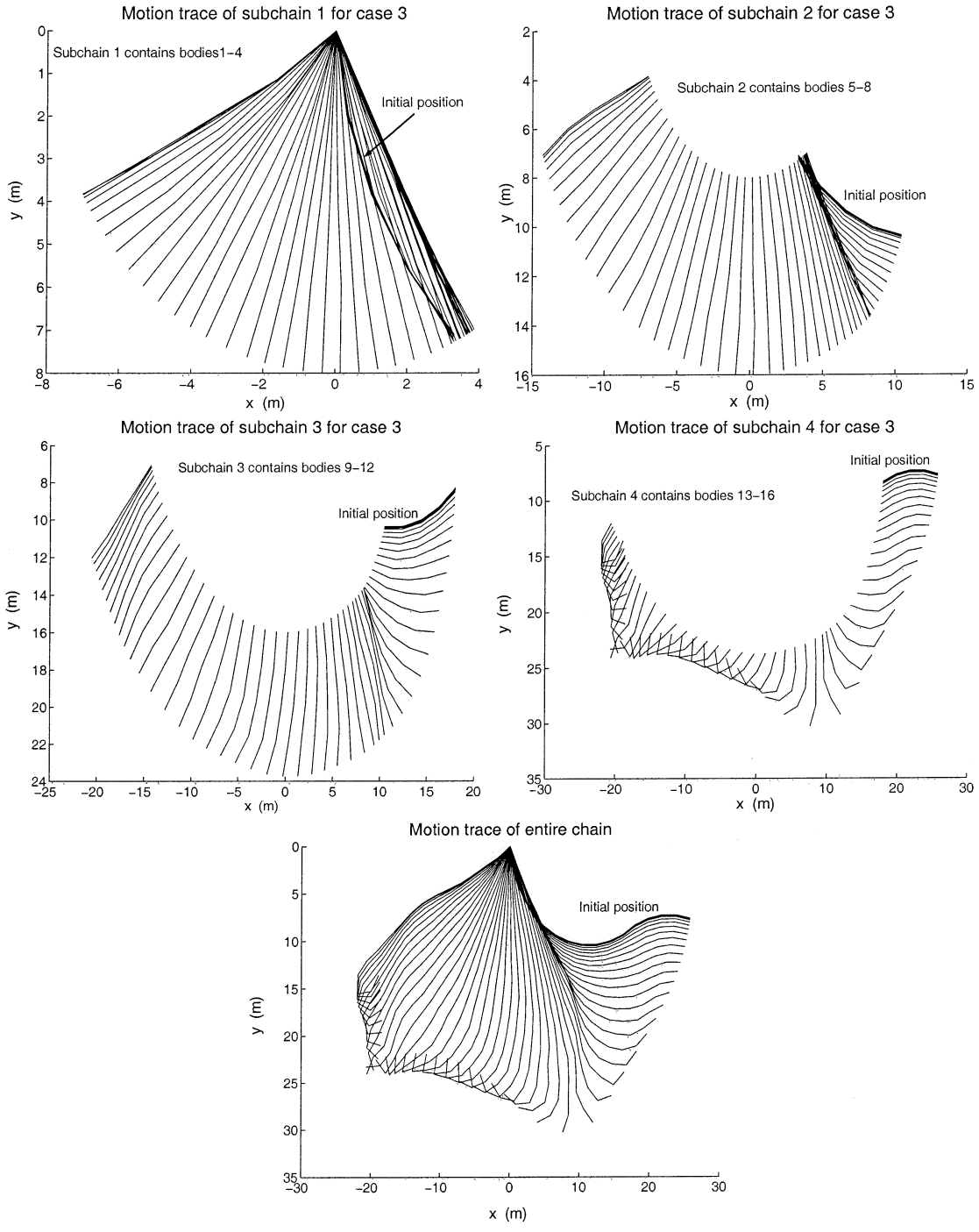


Figure 10: Motion Trace of Entire Chain and Each Subchain of Case 3

on the CPU time required to perform this simulation.

The number of iterations required in the solution of (1b) depends in a variety of factors including the eigenvalue spectra (and thus increase the condition number)of the constraint load coefficient matrix Γ,

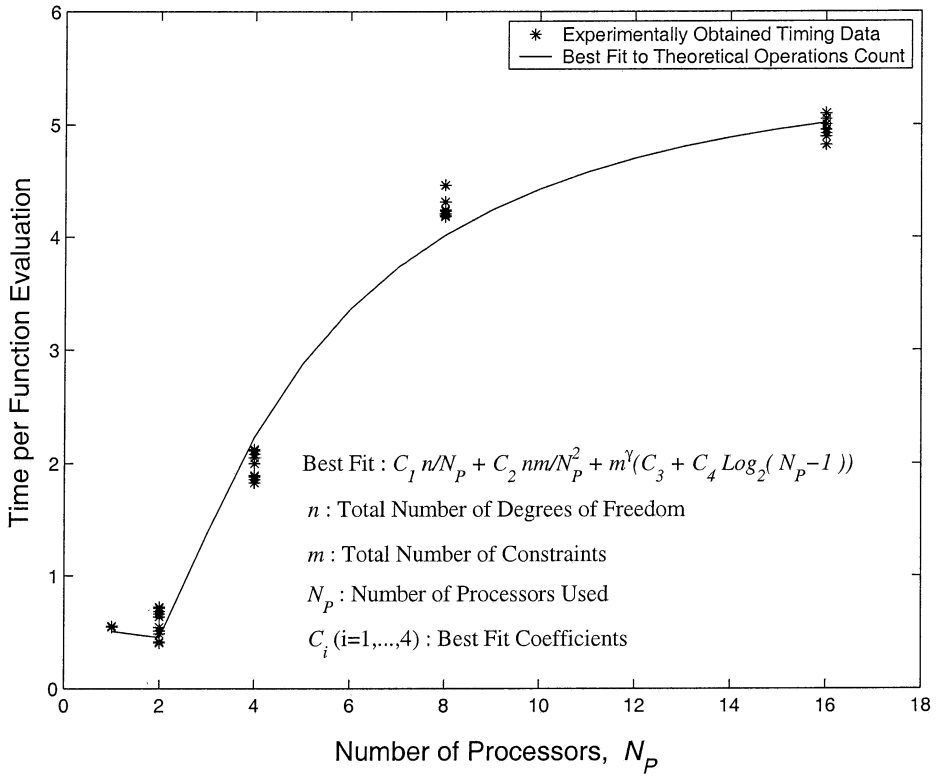


Figure 11: Best fit between theoretical cost and experimental data

the effectiveness of the preconditioning, the quality of the initial estimate for the solution values, and the convergence criteria used. The eigenvalue spectra is a function of the system being considered, but for a given physical system becomes more broad with the increased dimension of Γ, and with the existence of branches in the topology of the physical system. The performance of the preconditioning also greatly influences the rate of convergence, but the cost of determining superior preconditioning may be high and must also be considered. Also, the number of iterations required can be greatly reduced with high quality initial estimates for the solution value. If the system is reasonably well behaved, very good initial estimates can be obtained using a second order extrapolation (*second order hold*). Finally, the tighter the convergence tolerance, the more iterations which can be expected.

The results indicated in figure 11 indirectly demonstrate an advantage of using this hybrid *state-space* $O(n)$ - *full descriptor* algorithm. Specifically, this algorithm can adjust the dimension of the constraint load coefficient matrix Γ appearing in (1b), to the minimum necessary to achieve the desired (time optimal) level of coarse grain parallelism. The code associated with a particular processor has no constraints other than what is necessary for parallelism purposes and is very efficient. Thus the number of processors used ($\approx$proportional to the dimension of Γ) can be easily adjusted to produce a simulation which is optimum with respect to minimizing run time.

6 Summary and Conclusions

Parallelism and parallel implementation of a hybrid parallelizable $O(n)$ algorithm have been presented in this paper. A hybrid data-control parallelism is developed to carry out six computational tasks. These tasks are parallel determination of all kinematic quantities associated with each subsystem; the formation and solution of equation of motion (1a)(constrained and unconstrained portions); formation and solution

of constraint load equations (1b); and the determination and integration of state variables. These tasks are further hybridized at three different ways resulting in: A hybridization of state space and descriptor form dynamics analysis formulations; A hybridization between sequential $O(n)$ dynamics analysis procedure and parallel strategies; and, A combination of direct and iterative linear equation solution methods to achieve high overall computational efficiency. Due to these hybridizations, the algorithm can offer advantages of both the sequential $O(n)$ procedure and parallel computation. Specifically, benefits of the algorithm lie in its suitability for parallel implementation for the explicit determination of the constraint load measure numbers, which can be obtained using parallel iterative techniques. Further these loads may be used as a liaison between subchains in the formation of constraint load equations. This is in turn used in the formation and solution of equations of motion, which are performed sequentially using $O(n)$ procedure on each single processor, but concurrently between processors using data-control parallelism. This intelligent sequential and concurrent hybridization offers considerable flexibility. Therefore, the algorithm can easily accommodate the available number of processors while maintaining high efficiency. An $O[\frac{n}{N_p} + \frac{nm}{N_p^2} + \frac{m^\gamma}{N_p} + m^\gamma log_2(N_p - 1)]$ [19] performance will be achieved with N_p processors for a chain system with n degrees of freedom and m constraints due to cutting of interbody joints. The iterative solver performance parameter γ depends on the effectiveness of the preconditioner, the quality of the solution initial estimate, the eigenvalue spectra of the constraint force coefficient matrix, and the convergene criteria used.

The algorithm has been validated and implemented in both sequential and parallel computer simulations. Through the use of different numbers of joint separations and theoretical work it is concluded that simulation time should show a logarithmic dependence with the numbers of cut joints. Also, a factor 3.75 speed up of parallel implementation of the algorithm with 3 cut joints on 4 processors was be obtained, relative to sequential implementation of same case on a single processor of the same parallel computing system. Finally, iterative solver performance and improvement is key to making real gains in simulation speed.

Acknowledgment

Authors wish to acknowledge and thank the National Science Foundation for its support of this work through award No. ECS-9596237.

References

[1] Kasahara, H.; Fujii, H.; Iwata, M. (1987) "Parallel Processing of Robot Motion Simulation", *Proc. IFAC 10th World Conference*, Munich, FRG.

[2] Bae; D.S., Kuhl, J.G.; Haug,E.J. (1988) "A recursive Formation for Constrained Mechanical System Dynamics: Part III, Parallel Processing Implementation", *Mechanisms, Structures, and Machines*, Vol. 16, pp. 249-269.

[3] Lee, S. S. (1988) "Symbolic Generation of Equation of Motion for Dynamics/Control Simulation of Large Flexible Multibody Space Systems," *Ph.D Dissertation*, University of California, Los Angeles.

[4] Anderson, K. S. (1990) "Recursive Derivation of Explicit Equations of motion for Efficient Dynamic/Control Simulation of Large Multibody Systems" *Ph.D. Dissertation*, Stanford University.

[5] Sharf, I.; D'Eleuterio, G.M.T. (1993) "An Iterative Approach to Multibody Simulation Dynamics Suitable for Parallel Implementation." *Journal of Dynamic Systems, Measurement and Control*, Vol. 115, pp. 730-735.

[6] Eichberger, A. (1994) "Transputer-Based Multibody System Dynamic Simulation, Part I: The Residual Algorithm-A Modified Inverse Dynamic Formulation", *Mechanics Of Structures And Machines*, Vol. 22, No. 2, pp. 211-237.

[7] Eichberger, A. (1994) "Transputer-Based Multibody System Dynamic Simulation, Part II: The Residual Algorithm-A Modified Inverse Dynamic Formulation", *Mechanics Of Structures And Machines*, Vol. 22, No. 2, pp. 239-261.

[8] Fijany, A.; Bejczy, A. K. (1993) "Parallel Computation of Forward Dynamics of Manipulators", *JPL New Technology Report NPO-18706, NASA Technical Brief*, Vol. 17, No. 12, Item #82.

[9] Fijany, A.: Sharf,I.; D'Eleuterio, G.M.T. (1995)"Parallel O(log N) Algorithms for Computation of Manipulator Forward Dynamics." *IEEE Transactions on Robotics and Automation*, Vol. 11, No. 3, pp. 389-400.

[10] Anderson, K. A.; Duan, S. (2000) "Highly Parallelizable Low Order Algorithm for the Dynamics of Complex Multi Rigid Body Systems", *Journal of Guidance, Control, and Dynamics*to appear March-April Issue.

[11] Anderson, K. A.; Duan, S. (1999) "A Hybrid Parallelizable Low Order Algorithm for Dynamics of Multi-Rigid-Body Systems: Part I, Chain Systems", *Mathematical and Computer Modelling*, vol. 30, pp.193-215.

[12] Roberson, R. E.; Schwertassek (1988) *Dynamics of Multibody Systems*, Springer-Verlag, Berlin.

[13] Anderson, K. A.; Duan, S. (1998) "A New Order N - Order N^3 Hybrid Parallelizable Algorithm for Multi-Rigid-Body Dynamics", Submitted to ASME Journal *Applied Mechanics*, Oct. 1998.

[14] Franklin, G.F.; Powell, J.D. (1981) *Digital Control of Dynamic Systems*, Addison-Wesley, Menlo Park, CA, 1981.

[15] Saad, Y. (1996) *Iterative Methods for Sparse Linear Systems*, PWS, Boston.

[16] Kane, T.R.; Levinson, D.A. (1985) *Dynamics: Theory and Application*, McGraw Hill, NY.

[17] Flaherty, Joseph E. (1997) Class notes of Computational Science and Engineering, Rensselaer Polytechnic Institute, Spring 1997.

[18] Pacheco, P. S. (1997) *Parallel Programming with MPI*, Morgan Kaufmann Publishers, Inc., San Francisco, CA.

[19] Golub, G. H. (1993) *Matrix Computations*, Second Edition, Johns Hopkins, Baltimore.