Parallel Implementation of Hybrid Direct-Iterative Algorithm for Multibody Dynamics via Krylov Subspace Methods on IBM 1350 Cluster

Shanzhong (Shawn) Duan, *Member, IAENG,* and Yogesh Patel

***Abstract*—In this paper, a new hybrid direct-iterative algorithm for multibody dynamics is discussed. The implementation of the algorithm on IBM 1350 cluster is then presented. Some Krylov subspace methods available in Aztec library are utilized to solve constraint equations for real parallel implementation using MPI. The parallel simulation results are further compared with the results produced from different iterative solvers. Integration between the new algorithm and Aztec will make the algorithm accommodate various multibody systems easily while keeping its high simulation speed.**

***Keywords* — hybrid direct-iterative algorithm, Krylov subspace method, multibody system (MBS), parallel computing**

I. Introduction

With the development of various parallel computer systems, the techniques provided by these systems and the concurrent nature of MBS have been receiving increasing attention. A few parallel algorithms for computer simulation of dynamical behaviors of MBS have been put forward by individuals whose interests may lie in various fields since the first parallel MBS algorithm developed by Kassahara, Fujii, and Iwata in 1987 [1]. In 1988, Bae, Kuhl and Haug [2] applied the recursive algorithms developed by them to a shared memory multiprocessor system to realize parallel computing of MBS. Lee [3] presented a global $O(n^3)$ algorithm from its inception to develop the equations of motion in such a way as to allow the exploitation of the concurrent processing to the maximum degree on a multiple instruction multiple data (MIMD) machines. Based on his previous work [4] in 1990, Anderson [5] demonstrated that significantly greater parallelism could be realized with a modified form of the $O(n)$ algorithm. Fijany and Bejczy [6] showed that the more traditional $O(n^3)$ methods provide the highest degree of parallelism in 1993. Sharf and D'Eleuterio presented a new parallel solution procedure for dynamic simulation of multibody chains by solving joint constraint forces via various iterative parallel methods [7]. Based on parallel strategies for constraint equations in [7], Fijany, Sharf, and D'Eleuterio [8] implemented Constraint-Force algorithm (CF) on parallel computing system with $O(N)$ processors for solving both equations of motion and constraint equations. Anderson and Duan presented a new direct-iterative algorithm (HDIA) for multibody chain systems [9] and implemented it on IBM SP2 and SGI ONYX [10]. In their work, MBS is constructed so that largely independent multibody subchain systems are formed. They specifically applied the HDIA to a pure chain system and used a 16 body chain for parallel implementation of their HDIA approach with various joint cutting on IBM SP2 and SGI ONYX with MPI-supported environment. The work presented in this paper is an integration of HDIA with Krylov methods used in Aztec iterative solvers [11] to extend computational power and application abilities of HDIA.

S. Z. Duan is a faculty member with the South Dakota State University, College of Engineering, Mechanical Engineering Department, Brookings, SD 57007 USA (the corresponding author: phone: 605-688-5930; fax: 605-688 -5878; e-mail: shawn.duan@ sdstate.edu).
Y. Patel is a graduate student with the South Dakota State University, College of Engineering, Mechanical Engineering Department, Brookings, SD 57007 USA

II. Hybrid Iterative-Direct Algorithm (HDIA)

In HDIA approach, 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 constraint loads f_c at those separated joints as shown in figure 1. The added parallelism is obtained through this separation and the explicit determination of f_c by parallel iterative methods at a coarse grain level. Each branch is assigned to a processor of a parallel computing system for parallel simulation. An efficient sequential O(n) procedure is carried out to form and solve equations of motion within each processor, while parallel strategies are used to form and solve constraint equation between the processors concurrently. The O(n) method is embedded with direct method for formation and solution of equations of motion so that once the equations of motion is formed they have been solved. The parallel strategies take advantages provided by iterative method. Thus HDIA takes a combination of direct and iterative approaches.

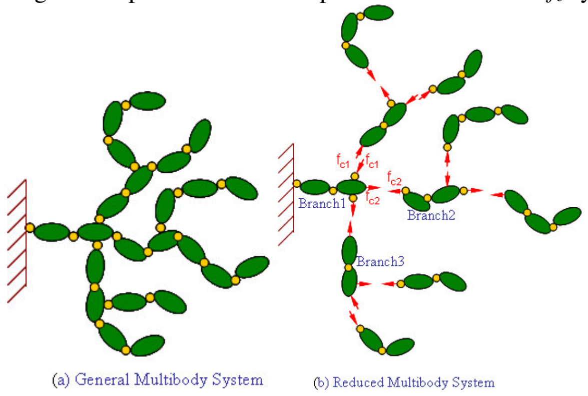


Figure 1: A MBS & Its Associated Subchains

In HDIA, the equations of motion and constraint equation have the following form, respectively

$$\underline{M}(t,\underline{q})\cdot\dot{\underline{u}}=\underline{R}(t,\underline{q},\underline{u},\underline{f}_c)$$

$$\Rightarrow \dot{\underline{u}}=\underline{RHS}(t,\underline{q},\underline{u},\underline{f}_c) \quad \text{(1a)}$$

$$\underline{\Gamma}(t,\underline{q})\cdot\underline{f}_c=\underline{\Psi}(t,\underline{q},\underline{u}) \quad \text{(1b)}$$

In equation (1a) and (1b), $\underline{M}$ is the system mass matrix implicitly formed by using the sequential *O(n)* procedure. $\underline{q}$

are the generalized coordinate column-matrix used to define the configuration of MBS, $\underline{u}$ are generalized speed column-matrix that characterizes the motion of the system, and $\dot{\underline{u}}$ are the generalized accelerations that are to be determined. The matrix $\underline{RHS}$ is right hand side matrix which contains all of external loads, body loads, constraint forces, and as well as inertia forces and torques associated with centripetal and Coriolis accelerations. Generally, the elements of $\underline{RHS}$ are nonlinear functions of the system state variables and time. The matrix $\underline{\Gamma}$ is the constraint coefficient matrix and the matrix $\underline{\Psi}$ consists of nonlinear functions of state variables and time. Finally, $\underline{f_c}$ is the unknown constraint force measuring numbers that must exist at separated joints between subchains of the reduced system, which makes the reduced system behave like the original system.

There are five computational tasks associated with forming and solving equation (1a) and (1b) as shown in figure 2. Next HDIA associated with these five computational tasks will be discussed and presented.

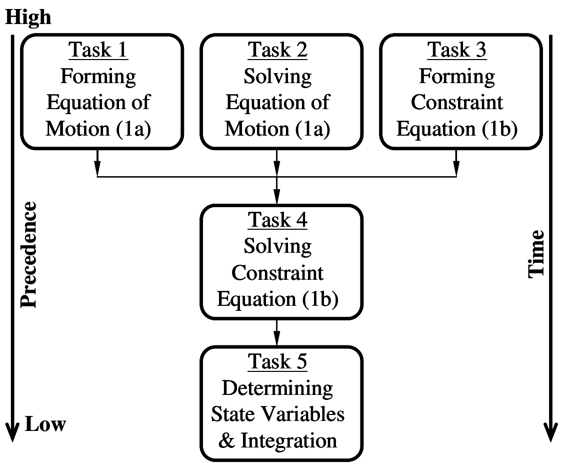

High
Task 1 Forming Equation of Motion (1a)
Task 2 Solving Equation of Motion (1a)
Task 3 Forming Constraint Equation (1b)
Task 4 Solving Constraint Equation (1b)
Task 5 Determining State Variables & Integration
Precedence
Time
Low

Figure 2: Precedence and Parallelism between Tasks

In the following mathematical representation of the HDIA algorithm, Kane's equations [12] are taken as axiomatic. The detail derivation of general formulations of the HDIA method is presented in references [9, 13, 14]. A notation convention with key joints J_k of body B_k belonging to subchain i may be used as shown in figure 3.

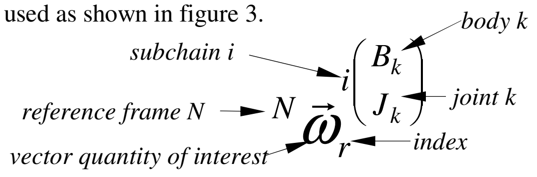


Figure 3: An Example for Notation Convention

A. Formation and Solution of Equations of Motion

In HDIA, generalized accelerations in equations of motion (1a) can be expressed in the form

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

where scalar quantity ν^i is degree of freedom (DOF) of subchain i, scalar quantity ζ^{ik} is the generalized acceleration that results from the unknown constraint forces and scalar quantity η^{ik} is the generalized acceleration from all other forcing terms. Both of them can be determined using the following recursive formulations

$$\eta^{ik} = \frac{\left(\underline{P}^{i\binom{J_k}{B_k}}\right)^T}{M^i_{kk}} \cdot \left[-{}^{iB_k}\hat{\underline{I}}^{i\binom{J_k}{B_k}} \cdot \left({}^{iB_{k-1}}\underline{S}^{iB_k}\right)^T \cdot \underline{\tilde{\eta}}^{i(k-1)} + \hat{\underline{F}}^{iB_k} \right] \tag{3}$$

with

$$\underline{\tilde{\eta}}^{ik} = \left({}^{iB_{k-1}}\underline{S}^{iB_k}\right)^T \cdot \underline{\tilde{\eta}}^{i(k-1)} + \underline{P}^{i\binom{J_k}{B_k}} \cdot \eta^{ik} \qquad (\underline{\tilde{\eta}}^{ik} = 0) \tag{4}$$

and

$$\zeta^{ik} = \underline{\mathcal{D}}_b^{iB_k} \cdot \underline{f}_c^{\,i6} + \underline{\mathcal{D}}_t^{iB_k} \cdot \underline{f}_c^{\,i(\upsilon^i+1)} \tag{5}$$

with

$$\underline{D}_t^{iB_k} = \frac{\left(\underline{P}^{i\binom{J_k}{B_k}}\right)^T}{M^i_{kk}} \cdot \left[\underline{\tau}_t^{ik} - {}^{iB_k}\hat{\underline{I}}^{i\binom{J_k}{B_k}} \cdot \left({}^{iB_{k-1}}\underline{S}^{iB_k}\right)^T \cdot \overline{\underline{D}}_t^{iB_{k-1}} \right] \tag{6}$$

and

$$\underline{D}_b^{iB_k} = \frac{\left(\underline{P}^{i\binom{J_k}{B_k}}\right)^T}{M^i_{kk}} \cdot \left[\underline{\tau}_b^{ik} - {}^{iB_k}\hat{\underline{I}}^{i\binom{J_k}{B_k}} \cdot \left({}^{iB_{k-1}}\underline{S}^{iB_k}\right)^T \cdot \overline{\underline{D}}_b^{iB_{k-1}} \right] \tag{7}$$

where

$$\overline{\underline{D}}_t^{iB_k} = \left({}^{iB_{k-1}}\underline{S}^{iB_k}\right)^T \cdot \overline{\underline{D}}_t^{iB_{k-1}} + \underline{P}^{i\binom{J_k}{B_k}} \cdot \underline{D}_t^{iB_k} \quad (\overline{\underline{D}}_t^{iB_0} = 0) \tag{8}$$

and

$$\overline{\underline{D}}_b^{iB_k} = \left({}^{iB_{k-1}}\underline{S}^{iB_k}\right)^T \cdot \overline{\underline{D}}_b^{iB_{k-1}} + \underline{P}^{i\binom{J_k}{B_k}} \cdot \underline{D}_b^{iB_k} \quad (\overline{\underline{D}}_t^{iB_0} = 0) \tag{9}$$

In equation (5), $\underline{D}_t^{iB_k}$ and $\underline{D}_b^{iB_k}$ are shifting triangularized and backsubstituted constraint Boolean matrix, while $\underline{f}_c^{\,i6}$ and $\underline{f}_c^{\,i(\upsilon^i+1)}$ are constraint load components applying to base body and terminal body of subchain i respectively. In formula (3), (6), and (7), ${}^{iB_k}\hat{\underline{I}}^{i\binom{J_k}{B_k}} \in R^{6\times6}$ is the articulated inertia matrix of body k of subchain i, and scalar term M^i_{kk} is the diagonal element of the mass matrix $\underline{M}$ associated with equation (1a). Both of them can be recursively determined by

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

$$M^i_{kk} = \left(\underline{P}^{i\binom{J_k}{B_k}}\right)^T \cdot {}^{iB_k}\underline{I}^{i\binom{J_k}{B_k}} \cdot \underline{P}^{i\binom{J_k}{B_k}} \tag{11}$$

Matrix $\underline{P}^{i\binom{J_k}{B_k}}$ is the partial velocity matrix associated with DOF of joint k of subchain i, and matrix ${}^{iB_k}\underline{S}^{iB_{k+1}}$ is a linear transformation matrix that 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. Matrices $\underline{\tau}_t^{ik}$ and $\underline{\tau}_b^{ik}$ in equations (6) and (7) are the shift-triangularization matrices that shift constraint measure numbers $\underline{f}_c^{i6}$ and $\underline{f}_c^{i(v^i+1)}$ from their original acting point on the base body and terminal body to the proximal joint of body k during triangularization of O(n) algorithm. $\hat{\underline{F}}^{iB_k} \in R^{6\times1}$ in equation (3) is the articulated body force matrix of body k of subchain i, and is determined recursively from

$$\hat{\underline{F}}^{iB_k} = \underline{F}^{iB_k} + {}^{iB_k}\underline{T}^{iB_{k+1}} \cdot \hat{\underline{F}}^{iB_{k+1}} \tag{12}$$

where ${}^{iB_{k-1}}\underline{T}^{iB_k} \in R^{6\times6}$ is the triangularization operation matrix that recursively triangularizes the equations as they are being formed, and is determined from

$${}^{iB_{k-1}}\underline{T}^{iB_k} = \left({}^{iB_{k-1}}\underline{S}^{iB_k}\right)\cdot\left\{\underline{U} - \frac{1}{M_{kk}^i}\cdot{}^{iB_k}\underline{I}^{i\binom{J_k}{B_k}} \cdot \underline{P}^{i\binom{J_k}{B_k}} \cdot \left(\underline{P}^{i\binom{J_k}{B_k}}\right)^T\right\} \tag{13}$$

Through equations (2) – (13), equations of motion (1a) associated with a MBS is formed and solved at a computing cost of $O\left(\dfrac{n}{N_p}\right)$ [10], where n is number of DOF of the MBS, and N_p is number of processors of a parallel computing system.

B. Formation of Constraint Equations

Formulation of constraint equation (1b) is carried out by enforcing constraints at separating joints (Nsub-1 joints. Nsub: total number of subchains) through following equations

$$-(\underline{\Gamma})_{i(i-1)} \cdot \underline{f}_c^{(i-1)(v^{(i-1)}+1)} + (\underline{\Gamma})_{ii} \cdot \underline{f}_c^{i(v^i+1)} - (\underline{\Gamma})_{i(i+1)} \cdot \underline{f}_c^{(i+1)(v^{(i+1)}+1)} = \underline{\Psi}_i$$

$$\text{(i=1, 2,...., Nsub-1)} \tag{14}$$

where

$$(\underline{\Gamma})_{i(i-1)} = \left[\underline{G}_1^{i1} \quad \cdots \quad \underline{G}_1^{iv^i}\right]\cdot\begin{bmatrix}\underline{D}_b^{iB_1}\\ \vdots \\ \underline{D}_b^{iBv^i}\end{bmatrix}, \tag{15}$$

$$(\underline{\Gamma})_{ii} = \left[\underline{G}_1^{i1} \quad \cdots \quad \underline{G}_1^{iv^i}\right]\cdot\begin{bmatrix}\underline{D}_t^{iB_1}\\ \vdots \\ \underline{D}_t^{iBv^i}\end{bmatrix} +$$

$$\left[\underline{G}_2^{(i+1)1} \quad \cdots \quad \underline{G}_2^{(i+1)6}\right]\cdot\begin{bmatrix}\underline{D}_b^{(i+1)B_1}\\ \vdots \\ \underline{D}_b^{(i+1)B_6}\end{bmatrix}, \tag{16}$$

$$(\underline{\Gamma})_{i(i+1)} = \left[\underline{G}_2^{(i+1)1} \quad \cdots \quad \underline{G}_2^{(i+1)6}\right]\cdot\begin{bmatrix}\underline{D}_t^{(i+1)B_1}\\ \vdots \\ \underline{D}_t^{(i+1)B_6}\end{bmatrix}, \tag{17}$$

and

$$\underline{\Psi}_i = \left[\underline{G}_2^{(i+1)1} \quad \cdots \quad \underline{G}_2^{(i+1)6}\right]\cdot\begin{bmatrix}\eta^{(i+1)1}\\ \vdots \\ \eta^{(i+1)6}\end{bmatrix} -$$

$$\left[\underline{G}_1^{i1} \quad \cdots \quad \underline{G}_1^{iv^i}\right]\cdot\begin{bmatrix}\eta^{i1}\\ \vdots \\ \eta^{iv^i}\end{bmatrix} + \underline{O}^{i(v^i+1)}\cdot\left(\underline{A}_t^{(i+1)B_6} - \underline{A}_t^{iB_{v^i}}\right) \tag{18}$$

with

$$\underline{G}_1^{ik} = \underline{L}_1^{ik} \cdot \underline{\mathrm{P}}^{i\binom{J_k}{B_k}}, \tag{19}$$

$$\underline{L}_1^{ik} = \underline{L}_1^{i(k+1)} \cdot ({}^{iB_k}\underline{S}^{iB_{(k+1)}})^T, \tag{20}$$

and

$$\underline{L}_1^{i(v^i+1)} = \underline{O}^{i(v^i+1)}. \tag{21}$$

Similarly

$$\underline{G}_2^{(i+1)k} = \underline{L}_2^{(i+1)k} \cdot \underline{\mathrm{P}}^{(i+1)\binom{J_k}{B_k}} \quad (k < 7), \tag{22}$$

$$\underline{L}_2^{ik} = L_2^{(i+1)(k+1)} \cdot ({}^{(i+1)B_k}\underline{S}^{(i+1)B_{(k+1)}})^T \quad (k < 6), \tag{23}$$

and

$$\underline{L}_2^{(i+1)6} = \underline{\mathrm{O}}^{i(v^i+1)}. \tag{24}$$

Equations (14)-(24) are algorithmic formulations of equation (1b) for constraint measure numbers f_c associated with entire subchain system. In equation (14), $\underline{f}_c^{(i-1)(v^{(i-1)}+1)}$, $\underline{f}_c^{i(v^i+1)}$, and $\underline{f}_c^{(i+1)(v^{(i+1)}+1)}$ are the constraint load measure numbers of 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 $\underline{G}_2^{ik}$ are those portions of the constraint coefficient matrix associated with the 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 of the entire subchain to the separated joints and project these DOF onto the constraint subspace associated with the separated joints. $\underline{\mathrm{O}}^{i(v^i+1)}$ is the orthogonal complement matrix to the free mode of motion matrix (partial velocity matrix) $\underline{P}^{i\binom{J_k}{B_k}}$. The joint orthogonal complement $\underline{\mathrm{O}}^{i(v^i+1)}$ defines the constraint subspace associated with the separated outboard joint on terminal body of subchain i. Matrices $\underline{L}_1^{ik}$ and $\underline{L}_2^{ik}$ are intermediate quantities that are introduced for convenient expression of recursive relations algorithmically. $\underline{A}_t^{iB_{v^i}}$ in equation (18) is the acceleration remainder terms of terminal body of subchain i, which are not explicit in the element of $\dot{\underline{u}}$. Quantity $\underline{A}_t^{(i+1)B_6}$ in the same equation is the acceleration remainder term matrix associated with all of the terms of the relative acceleration matrix not explicit in $\dot{u}s$ of imaginary body $\nu^i + 1$ relative to the terminal body ν^i of subchain *i*.

Through equations (14) – (24), constraint equation (1b) associated with the reduced MBS is formed at a computing cost of $O\left(\frac{nm}{N_p^2}\right)$ [10], where *m* is number of constraints for the reduced subchain system.

III. SOLUTION OF CONSTRAINT EQUATION

For the work presented in reference [10], a simple in-house Parallel Preconditioned Conjugate Gradient (PPCG) iterative solver was used to solve constraint equation (1b) in parallel. PPCG methods work very well for solution of a large sparse, symmetric, and positive definite system of linear equations. In reality constraint matrix Γ seldom has such nice properties. To extend power and application ability of HDIA approach, other Krylov subspace methods such as Generalized Minimal Residual (GMRES), Biconjugate Gradient (BiCG), and Quasi -Minimal Residual (QMR) may be explored and integrated with HDIA. These nonstationary iterative methods have their advantages for solution of a large-sized sparse linear system. For example, by comparison GMRES performs well for non-symmetric systems and QMR works for non-positive definite systems [15]. Nonstationary iterative solvers in Aztec has been used to integrate with HDIA.

A. *Aztec Nonstationary Iterative Method Library*

There are a few massively parallel iterative solver libraries, such as Aztec, PETsc, ScaLAPACK etc. for solving sparse linear systems. Aztec Library has been chosen to integrate with HDIA for solving constraint equation (1b) in parallel because Aztec has a similar coding environment as the HDIA codes and it is ease of availability.

Aztec is a parallel iterative library for solving linear systems, which is both easy-to-use and efficient. It is written in ANSI-standard C language. The source code of Aztec is available freely from the Sandia National Laboratory website. Aztec has been adopted to a number of parallel machines and platforms: workstation clusters (DEC, SGI, SUN, LINUX, etc.), Cray T3E, Intel TeraFlop, Intel Paragon, IBM SP2, nCUBE2 as well as other parallel platforms. There are a number of Krylov subspace methods such as CG, CGS, BiCGSTAB, GMRES, and TFQMR available in Aztec. These Krylov methods may be used in conjunction with various preconditioners. Point & block Jacobi, Gauss-Seidel, and least-squares polynomials are samples of preconditioners in Aztec [11].

B. *Integration between HDIA and Aztec Iterative Solvers*

Aztec can work with user-supplied matrix-vector product routines or two specific sparse matrix formats (in which case Aztec provides the matrix-vector product) - a point entry modified sparse row (MSR) format or a block-entry variable block row (VBR) format. These two formats have been generalized for parallel implementation and, as such, are referred to as "distributed" yielding DMSR and DVBR references. To invoke Aztec from an external program through a user-supplied matrix-vector product interface, the following steps are required:

1. Describe the machine (serial/parallel)
2. Initialize the matrix and vector data structures
3. Choose the iterative method, preconditioner and the convergence criteria
4. Initialize the right hand side and initial guess
5. Supply the preconditioner
6. Supply the left hand side implicitly through the matrix-vector product routine
7. Invoke the solver

The flow chart in figure 4 shows the integration between HDIA and iterative solvers in Aztec.

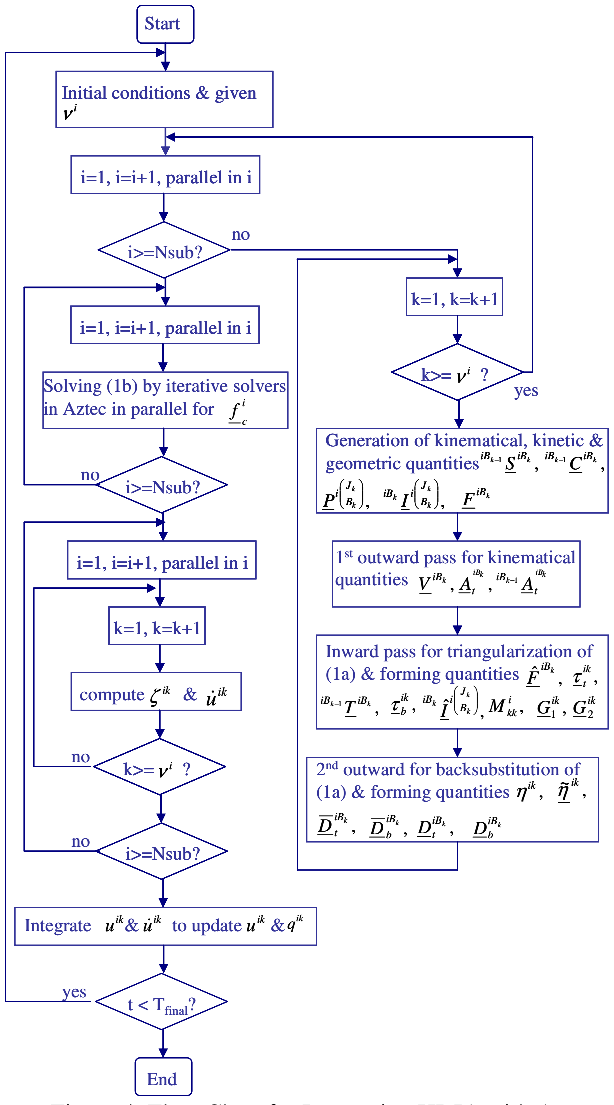

Start
Initial conditions & given ν^i
i=1, i=i+1, parallel in i
i>=Nsub?
no
k=1, k=k+1
k>= ν^i ?
yes
Generation of kinematical, kinetic & geometric quantities ${}^{iB_{k-1}}\underline{S}^{iB_k}$, ${}^{iB_{k-1}}\underline{C}^{iB_k}$, $\underline{P}^i\binom{J_k}{B_k}$, ${}^{iB_k}\underline{I}^i\binom{J_k}{B_k}$, $\underline{F}^{iB_k}$
1st outward pass for kinematical quantities $\underline{V}^{iB_k}$, $\underline{A}_t^{iB_k}$, ${}^{iB_{k-1}}\underline{A}_t^{iB_k}$
Inward pass for triangularization of (1a) & forming quantities $\hat{\underline{F}}^{iB_k}$, $\underline{\tau}_t^{ik}$, ${}^{iB_{k-1}}\underline{T}^{iB_k}$, $\underline{\tau}_b^{ik}$, ${}^{iB_k}\hat{\underline{I}}^i\binom{J_k}{B_k}$, M_{kk}^i, $\underline{G}_1^{ik}$, $\underline{G}_2^{ik}$
2nd outward for backsubstitution of (1a) & forming quantities η^{ik}, $\tilde{\underline{\eta}}^{ik}$, $\overline{\underline{D}}_t^{iB_k}$, $\overline{\underline{D}}_b^{iB_k}$, $\underline{D}_t^{iB_k}$, $\underline{D}_b^{iB_k}$
i=1, i=i+1, parallel in i
Solving (1b) by iterative solvers in Aztec in parallel for $\underline{f}_c^i$
no
i>=Nsub?
i=1, i=i+1, parallel in i
k=1, k=k+1
compute ζ^{ik} & $\dot{u}^{ik}$
no
k>= ν^i ?
no
i>=Nsub?
Integrate u^{ik} & $\dot{u}^{ik}$ to update u^{ik} & q^{ik}
yes
t < T$_{final}$?
End

Figure 4: Flow Chart for Integrating HDIA with Aztec

The Message-Passing Interface (MPI) is chosen for parallel computing communication. MPI is the most widely used parallel communication interface [16]. It has a library of subprograms that can be called within either C, C++, C# or Fortran programs. Two outstanding features of MPI are its portability and its independence from specific computing systems.

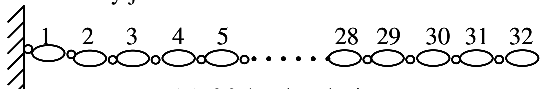
Through the integration between HDIA and iterative solvers in Aztec, and use of MPI, a computational cost of

$O\left(\frac{m^{\gamma}}{N_p}+m^{\gamma}\log_2(N_p-1)\right)$ for the solution of constraint equation (1b) can be achieved, where γ is iteration index $0<\gamma<1$.

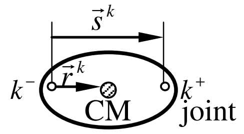
IV. PARALLEL IMPLEMENTATION

The algorithm is coded in C and MPI, and integrated with iterative solvers in Aztec. The algorithm has been run on IBM 1350 cluster, a distributed memory parallel computing system at South Dakota State University (SDSU). The simulation results reported here will demonstrate: 1. the validity of the integration of HDIA with Aztec. 2. Comparison of results produced by various solvers and joint separation cases.

For this purpose, a 32 body chain with identical bodies connected by simple revolute joints has been used for the modeling and simulation as shown in figure5. The properties of each body are: mass m^k=1kg; inertial I^k=[1,0, 0; 0, 1, 0; 0, 0, 1] kg.m^2; position vector $\vec{s}^{\,k}$ =[0, -2, 0] m; position vector $\vec{r}^{\,k}$ =[0, -1, 0] m. The initial conditions of the system are generalized speeds u^k=0.0 (rad/s) (k=1, … , 32), generalized coordinates q^k =0.1745 rad (k=1,…, 8, 17,…, 24) and q^k =-0.1745 rad (k= 9,…, 16, 25,…, 32). The numbering of each body/joint increases sequentially from the base body/ joint *1* to terminal body/joint.



(a) 32 body chain



(b) geometric parameters of body k

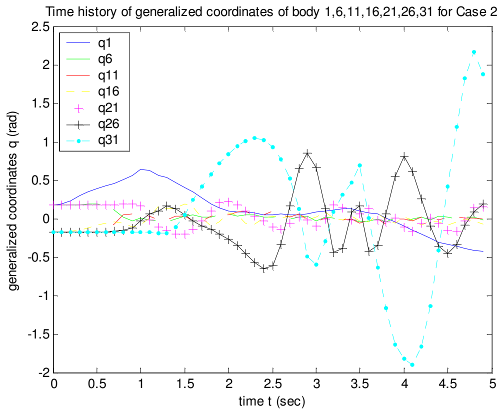
Figure 5: A 32 Body Chain Example

Table 1 shows that six different cases of joint separation are applied to this chain system. For example, case 3 has 3 joint separations (cuts), which results in 4 subchains. Each subchain is assigned to 1 processor for parallel computing and each processor contains 8 bodies. Case 1 has no cut, which is purely sequential. In case 6, each joint from 2 to 32 is cut and each processor contains one body only.

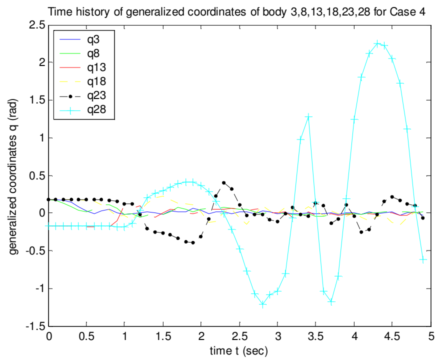
Table 1: Cases with Various Joint Separations

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
#. of processors	1	2	4	8	16	32
# of bodies /processor	32	16	8	4	2	1
#. of cuts	0	1	3	7	15	31

Figure 6 (a) and (b) show time histories of the generalized coordinates q_1, q_6, q_{11}, q_{16}, q_{21}, q_{26},and q_{31} of case 2, and q_3, q_8, q_{13}, q_{18}, q_{23}, and q_{28} of case 4. These time history curves agree with those produced by commercial dynamical analysis software Autolev. In figure 7, motion trace simulations associated with each body of each subchain for case 3 are given. From figure 7, it can be seen clearly that configurations of four subchains are updated in parallel. Due to constraints imposed between subchains being communicated correctly during concurrent simulation, the motion of outboard tip of each proximal sub-
chain is exactly same as the motion of inboard tip of each distal subchain. If the motion trace of each subchain in figure 7 (a), (b), (c), (d) is superposed on another, the motion trace of the entire subchain can be obtained as in figure 7 (e).



(a) Time histories of q_1, q_6, q_{11}, q_{16}, q_{21}, q_{26},and q_{31} of case 2



(b) Time histories of q_3, q_8, q_{13}, q_{18}, q_{23}, and q_{28} of case 4

Figure 6: Time Histories of Generalized Coordinates

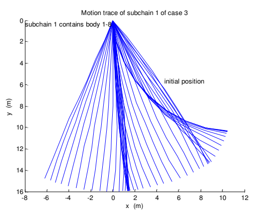
Table 2 shows average time cost of a function evaluation of six cases with application of five Krylov subspace methods in Aztec to each case. Take case 4 as an example. From table 1 it is clear that there are 7 cuts and 8 processors for case 4. Each processor contains 4 bodies. In table 2, average time cost of a single function evaluation is 1.06e-3 seconds for CG, 1.17e-3 seconds for CGS, 1.17e-3 seconds for TGQMR, 1.15e-3 seconds for BiCGSTAB, and 1.1e-3 seconds for GMRES respectively.

Table 2: Average Time Cost of a Function Evaluation

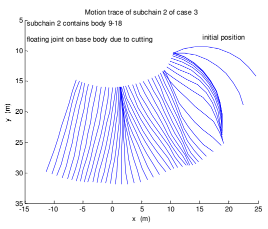
	Time in Seconds				
Krylov Method	CG	CGS	TFQMR	BiCGSTAB	GMRES
Case 1	4.7e-4	4.75e-4	4.7e-4	4.65e-4	4.7e-4
Case 2	5.4e-4	5.6e-4	5.7e-4	5.75e-4	5.55e-4
Case 3	9.2e-4	9.1e-4	9.35e-4	9.25e-4	9.3e-4
Case 4	1.06e-3	1.17e-3	1.17e-3	1.15e-3	1.1e-3
Case 5	1.23e-3	1.83e-3	1.73e-3	1.765e-3	1.54e-3
Case 6	2.34e-3	2.49e-3	2.51e-3	2.465e-3	2.39e-3

The time data from table 2 agree well with the theoretical estimation of computational cost as follows [10]:

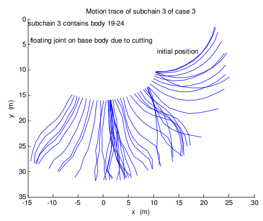
$$Cost \approx \frac{n}{N_p}+\frac{nm}{N_p}+\frac{m^{\gamma}}{N_p}+m^{\gamma}\log_2(N_p-1).$$



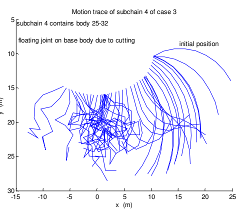
(a) subchain 1



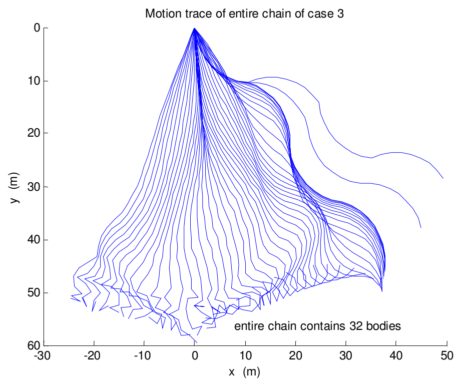
(b) subchain 2



(c) subchain 3



(d) subchain 4



(e) the entire chain

Figure 7: Motion Trace of Case 3

V. Conclusions

A new HDIA procedure and its parallel implementation on IBM 1350 cluster via integration with various nonstationary iterative solvers in Aztec have been introduced in this paper. Joint separation is utilized to distributed computing loads in parallel so that five computational tasks associated with formation/solution of equation (1) and consequent integration for updating state variables can be carried out efficiently for parallel simulation. An efficient sequential O(n) method is utilized within each processor in parallel for formulation and solution of equations of motion (1a), and parallel techniques are applied between processors for formation and solution of constraint equation (1b). A combination of direct and iterative linear equation solution methods is created to achieve high overall computational efficiency. Thus the algorithm can fully offer advantages of both the sequential O(n) procedure and parallel computation. A computational complexity of

$$O\left(\frac{n}{N_p}+\frac{nm}{N_p}+\frac{m^{\gamma}}{N_p}+m^{\gamma}\log_2\left(N_p-1\right)\right)$$

performance can be achieved with N_p processors for a chain system with n degrees of freedom and m constraints due to separation of interbody joints. The index γ is the parameter for iterative solver performance. It depends on effectiveness of the preconditioner, the quality of the solution initial guess, the eigenvalue spectra of the constraint force coefficient matrix, and the convergence criteria used.

The integration of the HDIA with Krylov subspace iterative solvers in Aztec enhances the application power of HDIA so that it can handle various properties that constraint coefficient matrix in equation (1b) may have. Both the algorithm and its integration with Aztec have been validated and implemented on IBM 1350 with MPI. Various joint separation cases and iterative solvers have been used to show flexibility of HDIA.

Acknowledgment

Authors wish to acknowledge and thank financial support from SD NASA EPSCoR Research Initiative fund and SDSU Research Start-up grant.

References

[1] H. Kassahara, H. Fujii, and M. Iwata, "Parallel Processing of Robot Motion Simulation", *Proc. IFAC 10th World Conference,* Munich, FRG, July, 1987.

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

[3] S. S. Lee, "Symbolic Generation of Equation of Motion for Dynamics/Control Simulation of Large Flexible Multibody Spaces Systems", *Ph. D. Dissertation*, University of California at Los Angeles, 1988, *UMI No. 8814809*

[4] K. S. Anderson, 1990. "Recursive Derivation of Explicit Equations of Motion for Efficient Dynamic/Control Simulation of Large Multibody Systems." *Ph.D. Dissertation*, Stanford University. UMI No. 9108778.

[5] K. S. Anderson, "An Efficient Modeling of Constrained Multibody Systems for Application with Parallel Computing," *Zeitschrift fur Angewante Mathematic un Mechanik*, Vol. 73, No. 6, 1993, pp. 520-528.

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

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

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

[9] K. S. Anderson and S. Z. Duan, "A Highly Parallelizable Low Order Algorithm for Dynamics of Multi-Rigid-Body Systems: Part I, Chain Systems", *Journal Mathematical and Computer Modeling*, Vol. 30, 1999, pp. 193-215

[10] Duan, S. Z. and K. S. Anderson, "Parallel Implementation of a Low Order Algorithm for Dynamics of Multibody Systems on a Distributed Memory Computing System", *Journal Engineering with Computers*, Vol. 16, No. 2, 2000, pp. 96-108.

[11] R.S. Tuminaro, M. Heroux, S.A. Hutchinson, and J.N. Shadid, *Official aztec user's guide version 2.1.* Massively Parallel Computing Research Laboratory, Sandia National Laboratories, Albuquerque, NM, 1999.

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

[13] S. Z. Duan, and Y. Patel, "A Hybrid Parallelizable Algorithm for Computer Simulation of the Motion Behaviors of a Branched Multibody System." *Proceedings of 2006 ASME International Conference and Exposition*. Chicago. November 5-10, 2006.

[14] S. Z. Duan, and J. A. Ries, "Efficient Parallel Computer Simulation of the Motion Behaviors of Closed-loop Multibody Systems" *Proceedings of 2007 ASME International Mechanical Engineering Congress and Exposition*, November 11-15, 2007, Seattle, Washington.

[15] Y. Saad, *Iterative Methods for Sparse Linear Systems,* PWS, Boston, 2000.

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