An Efficient Solution Method for Multibody Systems with Loops Using Multiple Processors

Tushar K. Ghosh, Luong A. Nguyen
National Security Solutions
L-3 Communications
Houston, Texas, USA

Leslie J. Quiocho
Software, Robotics and Simulation Division
NASA Johnson Space Center
Houston, Texas, USA

***Abstract*--This paper describes a multibody dynamics algorithm formulated for parallel implementation on multiprocessor computing platforms using the divide-and-conquer approach. The system of interest is a general topology of rigid and elastic articulated bodies with or without loops. The algorithm divides the multibody system into a number of smaller sets of bodies in chain or tree structures, called "branches" at convenient joints called "connection points", and uses an Order-N ($\mathcal{O}$ (N)) approach to formulate the dynamics of each branch in terms of the unknown spatial connection forces. The equations of motion for the branches, leaving the connection forces as unknowns, are implemented in separate processors in parallel for computational efficiency, and the equations for all the unknown connection forces are synthesized and solved in one or several processors. The performances of two implementations of this divide-and-conquer algorithm in multiple processors are compared with an existing method implemented on a single processor.**

Keywords-Multibody dynamics; multiple processors; divide-and-conquer algorithm; computational efficiency; Order-N.

I. INTRODUCTION

The Software, Robotics and Simulation Division (SRSD) of the NASA Lyndon B. Johnson Space Center provides mathematical modeling and simulation to support engineering analyses and crew training activities for the center. The division currently uses a mass matrix-based formulation to simulate the dynamics of on-orbit robotic manipulators implemented in a single processor of the simulation host computer. The SRSD has recently taken up the task of developing real time and non-real time simulation models for space exploration vehicles and mechanisms which consist of a large number of rigid and flexible bodies configured as structures with multiple branches, both with and without loops. In the pursuit of this task the division is in the process of developing and evaluating generic multibody dynamics code using parallel processing that would also allow the necessary flexibility and control for use in different simulations. This paper presents an approach that was considered and evaluated.

II. PREVIOUS WORK

Dynamics simulation of multibody systems using algorithms that can be computed in parallel has been an active research area for many years. Fijani et al. [1] developed the first known algorithm that can be parallelized to derive a time $\mathcal{O}(\log N)$ algorithm with $\mathcal{O}(N)$ processors. Featherstone [2] proposed a divide-and-conquer algorithm (DCA) $\mathcal{O}(\log N)$ formulation for chains of articulated bodies which was also generalized to handle tree and loop configurations [3]. Anderson and Duan [4] considered a hybrid direct and iterative solution scheme that allows a substantially higher degree of parallelization than normally obtainable with conventional recursive $\mathcal{O}(N)$ procedures. Their method is applicable to any general system of rigid bodies which may contain arbitrary joint types, multiple branches and/or closed loops. Based on the DCA approach by Featherstone which may not be computationally efficient in the presence of a modest number of processors, Critchley and Anderson [5] demonstrated that efficiency can be improved by breaking the original DCA system into subsystems where faster sequential techniques can be applied.

In this paper, an alternate formulation is developed and implemented for solving the forward dynamics of a general system of multiple flexible/rigid bodies with constrained motion which can be implemented in multiple processors. The DCA techniques are also adopted in the final phase of the solution to avoid a large square matrix inversion operation.

III. FORMULATION

Figure 1 shows a multibody system consisting of 18 bodies and having two loops. The bodies are connected by joints having up to six degrees of freedom. The connections between bodies 3 and 8 and between bodies 11 and 14 are considered as closures of loops 0 and 1 respectively. Body 0 (also called the base body of the system) is the body chosen according to convenience, it is tracked with respect to an inertial frame.

Figure 2 shows the same system divided into four branches with the loops cut at convenient points. After loop connections are cut and replaced with equal and opposite forces that maintain the loops, each branch assumes a tree structure. The branch containing the base body of the system is labeled Branch 0. A branch adjacent to another branch in the direction

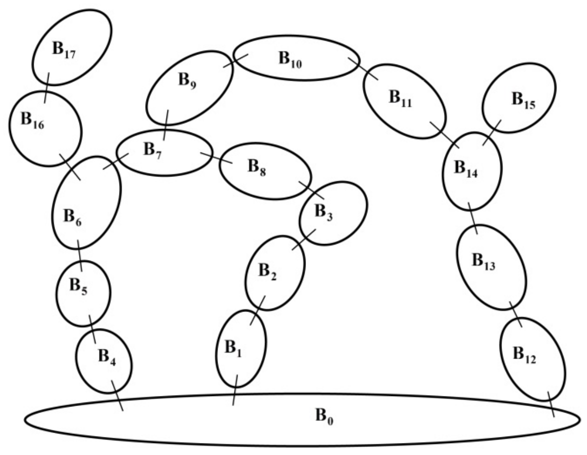

Figure 1. Multibody System Configuration

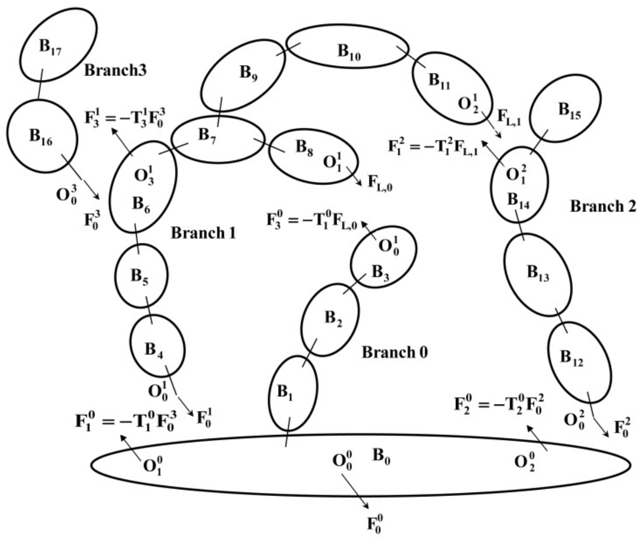

Figure 2. Multibody System Divided into Branches with Loops Cut

of Branch 0 is called its previous branch. Other adjacent branches to a branch are called its child branches. The body of a branch that connects to the previous branch is labeled the base body or Body 0 of the branch. Connections at a cut loop are called loop connections. Other connections between branches are called branch connections. Connection points are defined as points where two branches meet. Connection point 0 of a branch is located on its base body at the inboard end of the proximal joint and connects rigidly with the connection point on the previous branch.

The connection points and spatial connection forces between the branches that act at the connection points are shown in Figure 2 as O_i^j and F_i^j respectively where j is the index for the branch and i is the index for connection point of the branch. The connection forces are among the unknowns to be solved for. The connection forces between two branches are equal and opposite. They are expressed in the respective branch reference frames and therefore a frame transformation is needed, as shown in the figure, when the two are used in an equation. T_j^k is the 6x6 spatial transformation matrix needed to pre-multiply a spatial vector expressed in branch j coordinate frame in order to express it in that of the branch k frame.

For the connection points at a cut loop, one end is called the parent, and the other end is called the follower. They may be arbitrarily assigned, but in the present formulation, where one connection point is considered fixed on one body and the other may move on the connecting body, the former is called the parent and the latter is called the follower. The spatial force experienced by the parent connection point is defined as the corresponding loop force and is labeled $F_{L,i}$ where i is the loop index. The spatial force experienced by the follower connection point is the negative of the force on the parent point with appropriate frame transformation.

Since at a connection point the two connection forces are equal and opposite, there is only one unknown spatial force at any connection. Our goal is to determine the quantities F_0^j and $F_{L,i}$ for j = 1 to N_b -1 and i = 0 to N_ℓ -1, where N_b and N_ℓ are the number of branches and loops of the system. If needed, $F_0^{\mathbf{0}}$ is computed afterwards within the solution for dynamics of Branch 0.

A. Equations of A Branch

Treating each branch as an independent multibody system, its equations may be derived using known methods like the traditional mass matrix or Order-N with two modifications. The first modification would have the connection forces F_i^j appear explicitly in the expressions for the derivatives of the generalized speeds of the system. In the usual formulation of multibody system equations these forces are treated as known quantities. The second modification involves derivation of the expressions for the acceleration of connection points in terms of connection forces using the solution obtained for the time derivatives of generalized speeds and kinematics. These modifications involve some effort but are straight forward and are not being reported in this paper. The spatial acceleration of a connection point is obtained as

$$A_i^j = h_i^j + \sum_{k=0}^{N_j-1} H_{i,k}^j F_k^j \qquad (1.1)$$

where A_i^j is the spatial inertial acceleration of the connection point i of branch j, h_i^j is a 6x1 array made of known forces, generalized coordinates and generalized speeds of the branch, $H_{i,k}^j$ is a 6x6 matrix made of the generalized coordinates of the branch, and N_j is the number of connection points of the branch.

B. *Solution Method 1: Simultaneous Determination of Connection Forces Via Connection Matrix*

In this method we create a connection equation for the system and solve for all connection forces at once. We can see that $F_k^j = F_0^j$ when k = 0, $F_k^j = -T_n^j F_0^n$ when the connection point k has a child branch n connected to it, $F_k^j = F_{L,r}$ when the connection point k is the parent connection point of loop r, and $F_k^j = -T_s^j F_{L,r}$ when the connection point k is the follower connection point of loop r with the parent located on branch s. The parameters n, r and s for any branch j and its connection point k are determined in the initialization pass of the simulation. With the above information and defining N_b and N_ℓ as the number of branches and number of loops respectively, we can write:

$$A_i^j = h_i^j + \sum_{k=1}^{N_b-1} M_{i,k}^j F_0^k + \sum_{r=0}^{N_\ell-1} M_{i,N_b+r}^j F_{L,r} \qquad (1.2)$$

where

$M_{i,k}^j = H_{i,0}^j$ when k = 0,

$= -H_{i,k}^j T_n^j$ when connection point k is attached to the child branch n

= 0 Otherwise

$M_{i,N_b+r}^j = H_{i,k}^j$ when connection point k is the parent of loop r

$= -H_{i,k}^j T_s^j$ when connection point k is the follower point of loop r, with parent on branch s

= 0 Otherwise

1) *Loop Spatial and Constraint Force*

Loop closure points may allow degrees of freedom in rotation and/or translation. Accordingly, the spatial force at loop r parent point may be expressed by the equation

$$F_{L,r} = P_{L,r}\hat{F}_{L,r} + P_{F,r}\hat{F}_{F,r} \qquad (1.3)$$

where $P_{L,r}$ and $P_{F,r}$ are $[6 \times N_{c,r}]$ and $[6 \times (6 - N_{c,r})]$ matrices made of columns of spatial unit vectors in the constrained and free directions respectively, where $N_{c,r}$ is the number of constraints in the loop r. $\hat{F}_{L,r}$ is the array of $N_{c,r}$ constraint forces to be determined and $\hat{F}_{F,r}$ is the array of $6 - N_{c,r}$ forces in the free directions. In the current derivation it is assumed that $\hat{F}_{F,r}$ is fully known, even though it may be formulated to be related to $\hat{F}_{L,r}$.

2) *Branch Connection Equations*

Consider the connection between a branch i and its previous branch p. Let m be the connection point of branch p attached to connection point 0 of branch i. The spatial accelerations of the two points must be equal, i.e., $A_0^i = T_p^i A_m^p$. Using Equation (1.2) and (1.4) we then get the equation for connection between two branches:

$$\sum_{k=1}^{N_b-1} \hat{M}_{i,k} F_0^k + \sum_{r=0}^{N_\ell-1} \hat{M}_{i,N_b+r} F_{L,r} = \hat{h}_i$$

$$\text{for } i = 1 \text{ to } N_b - 1 \qquad (1.4)$$

where

$$\hat{M}_{i,k} = T_p^i M_{m,k}^p - M_{0,k}^i$$

$$\hat{M}_{i,N_b+r} = (T_p^i M_{m,N_b+r}^p - M_{0,N_b+r}^i) P_{L,r}$$

$$\hat{h}_i = h_0^i - T_p^i h_m^p + \sum_{r=0}^{N_\ell-1} [M_{0,N_b+r}^i - T_p^i M_{m,N_b+r}^p] P_{F,r} \hat{F}_{L,r}$$

3) *Loop Connection Equations*

Consider the loop j. Let the branch id and connection node id of the parent node be n and m respectively and those of the follower node be s and r respectively. Accelerations of the parent node and follower nodes in the constrained directions contained in the matrix $P_{L,j}$ must match. This condition results in the following equation

$$P_{L,j}^T [T_s^n A_r^s - A_m^n] + \dot{P}_{L,j}^T \Delta V_j + P_{L,j}^T [K_j \Delta X_j + C_j \Delta V_j] = 0 \qquad (1.5)$$

where ΔV_j and ΔX_j are spatial velocity and position of the follower with respect to the parent constrained point expressed in parent branch frame. $P_{L,j}^T [K_j \Delta X_j + C_j \Delta V_j]$ is the Baumgarte's stabilization term [6] for the constraint, K_j and C_j are the stabilization constants. ΔV_j , ΔX_j and $\dot{P}_{L,j}$ are obtained from kinematic equations. Using Eqs. (1.2) and (1.3) in Eq. (1.5) one gets the following loop connection equations.

$$\sum_{k=0}^{N_b-1} \hat{M}_{N_b+j,k} F_0^k + \sum_{r=0}^{N_\ell-1} \hat{M}_{N_b+j,N_b+r} F_{L,r} = \hat{h}_{N_b+j}$$

$$\text{for } j = 0 \text{ to } N_\ell - 1 \qquad (1.6)$$

where

$$\hat{M}_{N_b+j,k} = P_{L,j}^T [M_{m,k}^n - T_s^n M_{r,k}^s]$$

$$\hat{M}_{N_b+j,N_b+k} = P_{L,j}^T [M_{m,N_b+k}^n - T_s^n M_{r,N_b+k}^s] P_{L,k}$$

$$\hat{h}_{N_b+j} = P_{L,i}^T [T_s^n h_r^s - h_m^n] + \dot{P}_{L,j}^T \Delta V_j + P_{L,j}^T (K_j \Delta X_j + C_j \Delta V_j)$$
$$- P_{L,j}^T \sum_{k=1}^{N_\ell-1} [M_{m,N_b+k}^n - T_s^n M_{r,N_b+k}^s] P_{F,k} \hat{F}_F^k$$

Equations (1.4) and (1.6) can be written in one compact equation

$$\hat{M} F_c = \hat{h} \qquad (1.7)$$

where F_c is an array containing F_0^k and $\hat{F}_{L,r}$. $\hat{M}$ is a symmetric positive definite matrix which is called the connection matrix. This equation corresponds to Eq. (35) of

[4]. After Equation (1.7) is solved, F_0^k and $\hat{F}_{L,r}$ are extracted from F_c, and spatial parent loop force $F_{L,r}$ is generated using Equation (1.3). The forces F_0^k and $F_{L,r}$ are used to generate the equal and opposite forces exerted on the other ends of the connections, completing computation of all forces that act on the individual branches. The connection forces are then used in the individual group equations to generate their solutions.

C. *Solution Method 2: Sequential Determination of Connection Forces*

Eq. (1.5) requires inversion of a large square matrix. Due to overheads created by the Portable Operating System Interface for Unix (POSIX) thread function calls, parallelizing the solution of this equation for improving its computational efficiency may not always be successful. We could instead determine the connection forces sequentially using the divide-and-conquer concept for multiple articulated bodies described in [2] to achieve better computational efficiency. As proof of concept, a system with five branches described in the next section was considered. Eq. (1.1) can be rewritten for each of the 5 branches {0,1,2,3,4} of the system as an articulated body with two handles (single or multiple joints) as follows (In this method all quantities are expressed in the coordinate frame of the base body of branch 0).

For branch 0,

$$\hat{a}_1^{\{0\}} = \hat{h}_1^{\{0\}} + \hat{H}_{11}^{\{0\}}\hat{F}_1^{\{0\}} + \hat{H}_{12}^{\{0\}}\hat{F}_2^{\{0\}} \tag{2.1}$$

$$\hat{a}_2^{\{0\}} = \hat{h}_2^{\{0\}} + \hat{H}_{21}^{\{0\}}\hat{F}_1^{\{0\}} + \hat{H}_{22}^{\{0\}}\hat{F}_2^{\{0\}} \tag{2.2}$$

where,

$$\hat{a}_1^{\{0\}} = A_2^0 \qquad \hat{h}_1^{\{0\}} = h_2^0$$

$$\hat{H}_{11}^{\{0\}} = H_{2,2}^0 \quad \hat{H}_{12}^{\{0\}} = \begin{bmatrix} H_{2,1}^0 & H_{2,3}^0 \end{bmatrix} \quad \hat{H}_{22}^{\{0\}} = \begin{bmatrix} H_{1,1}^0 & H_{1,3}^0 \\ H_{3,1}^0 & H_{3,3}^0 \end{bmatrix}$$

$$\hat{F}_1^{\{0\}} = F_2^0 \qquad \hat{F}_2^{\{0\}} = \begin{bmatrix} F_1^0 \\ F_3^0 \end{bmatrix}$$

For branch 1,

$$\hat{a}_1^{\{1\}} = \hat{h}_1^{\{1\}} + \hat{H}_{11}^{\{1\}}\hat{F}_1^{\{1\}} + \hat{H}_{12}^{\{1\}}\hat{F}_2^{\{1\}} \tag{2.3}$$

$$\hat{a}_2^{\{1\}} = \hat{h}_2^{\{1\}} + \hat{H}_{21}^{\{1\}}\hat{F}_1^{\{1\}} + \hat{H}_{22}^{\{1\}}\hat{F}_2^{\{1\}} \tag{2.4}$$

where,

$$\hat{a}_1^{\{1\}} = \begin{bmatrix} A_0^1 \\ A_1^1 \end{bmatrix} \qquad \hat{h}_1^{\{1\}} = \begin{bmatrix} h_0^1 \\ h_1^1 \end{bmatrix} \qquad \hat{H}_{11}^{\{1\}} = \begin{bmatrix} H_{0,0}^1 & H_{0,1}^1 \\ H_{1,0}^1 & H_{1,1}^1 \end{bmatrix}$$

$$\hat{H}_{12}^{\{1\}} = \begin{bmatrix} H_{0,2}^1 & H_{0,3}^1 & H_{0,4}^1 \\ H_{1,2}^1 & H_{1,3}^1 & H_{1,4}^1 \end{bmatrix} \qquad \hat{H}_{22}^{\{1\}} = \begin{bmatrix} H_{2,2}^1 & H_{2,3}^1 & H_{2,4}^1 \\ H_{3,2}^1 & H_{3,3}^1 & H_{3,4}^1 \\ H_{4,2}^1 & H_{4,3}^1 & H_{4,4}^1 \end{bmatrix}$$

$$\hat{F}_1^{\{1\}} = \begin{bmatrix} F_0^1 \\ F_1^1 \end{bmatrix} \qquad \hat{F}_2^{\{1\}} = \begin{bmatrix} F_2^1 \\ F_3^1 \\ F_4^1 \end{bmatrix}$$

Similarly, expressions for the handle accelerations can be obtained for branches 2, 3 and 4. For a rigid connection which is the case for all five connection points of the system of Figure 3, the connection forces at the interface points between branches 0 and 1 are equal and opposite, i.e., $\hat{F}_2^{\{0\}} = -\hat{F}_1^{\{1\}}$, and the accelerations of the branches at these points are related by

$$\hat{a}_1^{\{1\}} = \hat{a}_2^{\{0\}} \tag{2.5}$$

the equations (2.1) through (2.4) can be used to combine the branches 0 and 1 into a subsystem {0,1} defined by

$$\hat{a}_1^{\{0,1\}} = \hat{h}_1^{\{0,1\}} + \hat{H}_{11}^{\{0,1\}}\hat{F}_1^{\{0,1\}} + \hat{H}_{12}^{\{0,1\}}\hat{F}_2^{\{0,1\}} \tag{2.6}$$

$$\hat{a}_2^{\{0,1\}} = \hat{h}_2^{\{0,1\}} + \hat{H}_{21}^{\{0,1\}}\hat{F}_1^{\{0,1\}} + \hat{H}_{22}^{\{0,1\}}\hat{F}_2^{\{0,1\}} \tag{2.7}$$

where

$$\hat{H}_{11}^{\{0,1\}} = \hat{H}_{11}^{\{0\}} - \hat{H}_{12}^{\{0\}}W^{\{0,1\}}\hat{H}_{21}^{\{0\}} \tag{2.8}$$

$$\hat{H}_{22}^{\{0,1\}} = \hat{H}_{22}^{\{1\}} - \hat{H}_{21}^{\{1\}}W^{\{0,1\}}\hat{H}_{12}^{\{1\}} \tag{2.9}$$

$$\hat{H}_{21}^{\{0,1\}} = \hat{H}_{21}^{\{1\}}W^{\{0,1\}}\hat{H}_{21}^{\{0\}} = \left(\hat{H}_{12}^{\{0,1\}}\right)^T \tag{2.10}$$

$$\hat{h}_1^{\{0,1\}} = \hat{h}_1^{\{0\}} - \hat{H}_{12}^{\{0\}}G^{\{0,1\}} \tag{2.11}$$

$$\hat{h}_2^{\{0,1\}} = \hat{h}_2^{\{1\}} + \hat{H}_{21}^{\{1\}}G^{\{0,1\}} \tag{2.12}$$

$$W^{\{0,1\}} = \left(\hat{H}_{11}^{\{1\}} + \hat{H}_{22}^{\{0\}}\right)^{-1} \tag{2.13}$$

$$G^{\{0,1\}} = W^{\{0,1\}}\left(\hat{h}_2^{\{0\}} - \hat{h}_1^{\{1\}}\right) \tag{2.14}$$

$$\hat{F}_1^{\{0,1\}} = \hat{F}_1^{\{0\}} \tag{2.15}$$

$$\hat{F}_2^{\{0,1\}} = \hat{F}_2^{\{1\}} \tag{2.16}$$

These connection forces can be computed using the formula

$$\hat{F}_1^{\{1\}} = W^{\{0,1\}}\hat{H}_{21}^{\{0\}}\hat{F}_1^{\{0\}} - W^{\{0,1\}}\hat{H}_{12}^{\{1\}}\hat{F}_2^{\{0\}} + G^{\{0,1\}} \tag{2.17}$$

when the external forces at the handles of the subsystem {0,1} and the internal active joint forces are known. Note that this computation only requires the inversion of a 12 x 12 matrix. Continue with combining branches 3 and 4 into the subsystem {3,4}, then combining the subsystem {0,1} with the branch {2} into the subsystem {0,1,2}, and finally combining the subsystems {0,1,2} and {3,4} into the big system {0,1,2,3,4} to solve for the connection forces across the interface of these subsystems. The computation of all connection forces in terms of matrix inversions for this approach turns out to include one 6 x 6 operation and three 12 x 12 operations which require less computer time than the original 42 x 42 matrix inversion operation.

IV. TEST MODEL AND IMPLEMENTATION

A conceptual Multi-Mission Space Exploration Vehicle (MMSEV) similar to one shown in Fig. 3 is simulated using the C programming language on a 3.07 GHz i386 computer platform with eight processors. This system consists of a crew module modeled as a rigid body and three identical articulating legs and two identical manipulator arms attached to the crew module. Each leg and manipulator has seven revolute joints to provide rotation in the sequence roll, yaw, pitch, pitch, pitch, yaw and roll. Each leg and manipulator arm has two long elements in the middle, which are called booms. The booms are considered flexible in the flexible model of the system. Booms in legs and the manipulators have eight and four flexible degrees of freedom respectively. Other elements of legs and manipulators are considered rigid. The ends of two manipulator arms rigidly connect to a rigid payload (not shown in the figure). The free ends of legs are rigidly anchored to the planet or the asteroid, which for the purpose of simulation is arbitrarily considered as a rigid body. Fictitious mass properties were used for evaluation of the simulation.

The system is divided into five branches. Branch 0 consists of the base body, and one leg. It has 13 rigid and 16 flex degrees of freedom (dofs). Branch 1 consists of the second leg and the crew module. Branch 2 consists of the third leg. They have 7 rigid and 16 flex DOFs each. The two manipulators constitute Branches 3 and 4. Each of them branches has 7 rigid and 8 flex DOFs. The system has a total of 41 rigid and 64 flex degrees of freedom. Equations of individual branches are implemented using the Order-N approach. Each computation cycle of the system consists of six consecutive steps. Step 1 is a forward pass to determine the position and velocity states of the bodies of each branch starting with the base body, using the generalized coordinates and generalized speeds of the branch. Step 2 is a backward pass, starting with the outermost bodies, to implement the dynamics equations. Step 3 determines the quantities $H_{i,k}^{j}$ and h_{i}^{j} of Equation (1.1). Step 4 generates the connection forces using Method 1 or Method 2. Step 5 determines the time derivatives of the generalized speeds and integrates them to update the generalized speeds and generalized coordinates of the branch. Steps 1, 2 and 3 were executed together in 5 processors in parallel, one for each branch. Step 4 in Method 1 was performed in two sub-steps 4a and 4b. In Step 4a the matrix $\hat{M}$ is generated in 7 separate processors, one for each row. Step 4b solves Eq. (1.5) using one or several processors. Step 4 could also be performed using Method 2. Recall that equation (2.17) provides the formula for computing the connection forces between branches {0} and {1}. This formula can be similarly obtained for the connection forces between other branches/subsystems. Step 4 in Method 2 then computes sequentially the connection forces across the interface between subsystems {0,1,2} and{3,4}, {3} and {4}, {0,1} and {2}, and finally between {0} and {1} with the exception of the connection forces between {3} and {4} which can be calculated in parallel with other connection forces after the computation of the connection forces between {0,1,2} and {3,4}.

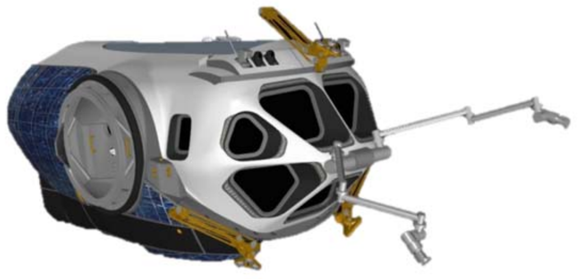
Fig. 3 A Conceptual Multi-mission Space Exploration Vehicle

Step 5 was executed in 5 processors, one for each branch. The parallel implementation was done using standard POSIX threads utility function calls.

V. RESULTS AND DISCUSSIONS

Table I shows the total computation times for each branch and loop. They were obtained by adding computation times for different segments of the code corresponding to the five steps mentioned earlier, from a code without parallelization. In the parallelized code computations were distributed such that one processor was assigned to one group or one loop only. Since different amounts of computations are involved in different groups and loops the computation was not equally distributed between the processors. However, distributing computations otherwise would have resulted in a very complicated and difficult to maintain code. Table II shows other relevant timing data, with and without parallelization. All times are in seconds and for one computation cycle. Rigid and Flex models refer to cases where the booms are considered rigid and flexible, respectively.

It was found that the speed-up ratio, defined as the ratio of actual time for one cycle of computations with parallelization to that without parallelization, was 1.64 and 1.72 for rigid and flex models respectively. Even though 5 and 7 processors were used in different parts of the code, these numbers were low for three reasons: (1) a significant amount (22.9% for rigid, 17.34% for flex) of code could not be parallelized, (2) computations could not be distributed evenly for coding considerations and (3) parallelization overhead involved in the POSIX function calls. Since most of the time 5 processors were used, a speed up of at least 2.60 was expected, for example, for the rigid model using Amdahl’s Law [7]. The primary reason for getting a lower number is the uneven distribution of computations to the processors; computation in processor 1was significantly more than others because branch 1 has more connection points.

The total non-parallelized computation time presented in Table II was obtained by subtracting the sum of branch and loop times from the total cycle time. The expected time with parallelization was determined by adding the maximums for branch time and loop time to non-parallelized computation time.

Table III presents the timing data for Step 4 implemented by Methods 1 and 2. It may be seen that Method 2 did not reduce the computation time for Step 4 for this problem.

TABLE I. TIMING DATA (SECONDS) FOR PARALLELIZED CODE SEGMENTS

	Rigid Model	**Flex Model**
Branch 0	1.115e-04	1.873e-04
Branch 1	2.579e-04	4.343e-04
Branch 2	9.434e-05	1.727e-04
Branch 3	8.616e-05	1.249e-04
Branch 4	8.421e-05	1.237e-04
Loop 0	1.068e-05	1.110e-05
Loop 1	8.466e-06	8.634e-06
Loop 2	4.779e-06	4.786e-06
Total for Parallelized Computations	65.808e-05	106.744e-5

TABLE II. TIMING DATA (SECONDS) FOR ONE CYCLE FOR METHOD 1

	Rigid Model	**Flex Model**
Total, without parallelization	85.37e-05	129.13e-05
Total, with parallelization, Method1	52.00e-05	75.00e-05
Speed-Up Ratio	1.64	1.72
Total non-parallelized Computations	19.56e-05	22.39e-05
Expected total with parallelization	46.42e-05	66.93e-05
Parallelization overhead	5.58e-05	8.07e-05
Solution of Eq. (3.10)	6.58e-05	6.59e-05

TABLE III. TIMING DATA (SECONDS) FOR STEP 4 FOR METHODS 1 & 2

	Rigid Model	**Flex Model**
Method 1: step 4a	7.200e-05	7.203-05
Method 1: step 4b	6.583-05	6.595-05
Total for Method 1	1.378e-04	1.380e-04
Total for Method 2	1.650e-04	1.640e-04

Attempts to speed up solution of Eq. (1.5) by parallelization were not successful. This was because the overhead involved in in the parallelization of Choleski solution exceeded the saving in computation for this problem. Since the time taken for this step was small compared to the total cycle time, the improvement would not have been significant anyway.

TABLE IV. Timing Results (seconds) for Different Methods

	Rigid	**Flex**
Mass Matrix (existing code)	57.40e-05	406.3e-05
Method 1 without parallelization	85.37e-05	129.1e-05
Method 1 with parallelization	52.00 e-05	75.00e-05
Method 2 with parallelization	53.60 e-05	76.54e-05

Table IV presents the comparison of computing cycle times for methods presented here and the existing mass-matrix based computation. The computation time for Method 1 where the system is modeled as a single branch without parallelization is also presented.

VI. ACKNOWLEDGMENT

This work was performed under a contract with NASA Johnson Space Center.

VII. REFERENCES

1. Fijany, A., Sharf, I., D'Eleuterio, G.M.T.: Parallel $\mathcal{O}(N)$ algorithms for computation of manipulator forward dynamics, IEEE Transactions on Robotics and Automation 11 (3) (1995) 389–400.
2. Featherstone, R.: A divide-and-conquer articulated body algorithm for parallel $\mathcal{O}(\log N)$ calculation of rigid body dynamics. Part 1: basic algorithm. Int. J. Robot. Res. **18**, 867–875 (1999)
3. Featherstone, R.: A divide-and-conquer articulated body algorithm for parallel $\mathcal{O}(\log N)$ calculation of rigid body dynamics. Part 2: trees, loops, and accuracy. Int. J. Robot. Res.**18**, 876–892 (1999)
4. Anderson, K.S., Duan, S.: Highly parallelizable low-order dynamics simulation algorithm for multi-rigid-body systems. J. Guid. Control Dyn. **23**, 355–364 (2000)
5. Critchley, J., Anderson, K., Binani, A.: An efficient Multibody Divide and Conquer Algorithm and Implementation. Journal of Computational and Nonlinear Dynamics **4/021004**, 1-10 (2009)
6. Baumgarte, J: Stabilization of constraints and integrals of motion in dynamical systems. Computational Methods in Applied Mechanics and Engineering. **1** 1-16 (1972).
7. Amdahl, Gene M: Validity of the single processor approach to achieving large-scale computing capabilities. AFIPS Conference proceedings **30**, 438-485 (1967)