Computationally Efficient Multibody Simulations

Jayant Ramakrishnan
and
Manoj Kumar

Dynacs Engineering Company, Inc.
34650 US Hwy 19 North
Suite 301
Palm Harbor, FL 34684

Abstract

Computationally efficient approaches to the solution of the dynamics of multibody systems are presented in this work. The computational efficiency is derived from both the algorithmic and implementation standpoint. Order(n) approaches provide a new formulation of the equations of motion eliminating the assembly and numerical inversion of a system mass matrix as required by conventional algorithms. Computational efficiency is also gained in the implementation phase by the symbolic processing and parallel implementation of these equations. Comparison of this algorithm with existing multibody simulation programs illustrates the increased computational efficiency.

1 Introduction

Current multi-link mechanism control systems are based on inverse kinematic approaches. These approaches are used primarily because of the complexity and computational cost associated with the solution of the dynamics of such systems. Typical systems include robotic manipulators and mobile station servicing modules. In real-time control applications, a need exists for highly efficient dynamics solution algorithms (as opposed to kinematic) that will make the dynamic control of these mechanisms possible. The evolution of the formulation algorithm and the numerical solution methodology over the past decade to accomplish real-time control objectives is now presented.

TREETOPS developed in the mid-eighties was based on the minimum dimension formulation of the multibody equations of motion. Originally developed for bodies in a tree topology, kinematic relations were written for the sequence of joints in terms of relative coordinates. The dynamics of the multibody configuration were derived by projecting the translation and rotation equations along the generalized speeds. The generalized speeds were defined as the partial derivative of the expressions for body j translation and rotational velocities with respect to the degrees of freedom [1]. The algorithm resulted in a mass matrix of order(n) where (n) is the number of degrees of freedom. As the complexity of the multibody systems increased, the computational cost associated with this approach became prohibitively large (order of n^3).

The numerical order(n) approach was proposed in 1988 as a solution to the prohibitively high computational cost associated with the n^3 algorithms. Developed initially for chains of rigid bodies, the method was later extended to flexible bodies. The equations of motion are again formulated in minimum dimension (by the elimination of the non-working constraint forces) but now a frontal and back substitution approach is used. The inertia and active forces are shifted inboard to the core body for the solution of the equations and then the procedure is reversed to obtain outboard body variables. This frontal part and backsubstitution part result in the computational savings through inversions of mass matrices of much smaller dimension than n, the order of the system.

Symbolic processing of equations was the next step towards higher efficiency. A generic equation file was used to provide the inputs to a symbolic processor which eliminated unnecessary computations and generated a *configuration specific* simulation code. By parsing, layering and simplifying equations, an order of magnitude improvement over numerical implementation was achieved.

In 1990 , parallel implementation of the multibody dynamics algorithm was attempted on four Intel 860 chips connected to a host IRIS workstation. In the verification runs, for the class of problems tested (Large Space Structures, Space Station), a speed-up of more than two orders of magnitude was obtained.

The current efforts in this area are focusing on bringing this technology to fruition by refining and automating the implementation procedure.

Additionally, graphical user interfaces based on X-windows are being developed for pre and post processing. A symbolic programming language that supports a whole family of entities (partioned matrix operations, vector operations, etc.) is being developed to support the quick and painless generation of symbolic code for a variety of engineering applications. The concept behind these technology thrust areas is presented in this paper.

The paper is organized as follows. A flavor of the Order(n) approach is first presented. This is followed by a section on symbolic code generation. Issues and our past experience with the implementation of equations on a parallel hardware platform are then presented. Some of the practical problems solved using these simulations and performance comparisons are then presented followed by conclusions.

2 Algorithm Formulation

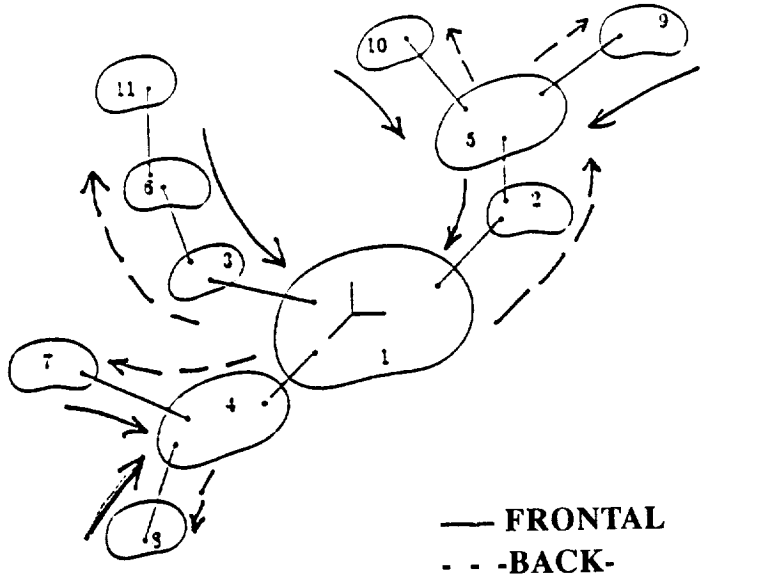
A multibody dynamic system is characterized by several bodies interconnected by joints that permit relative motion across them. Robots and spacecraft with articulated appendages such as solar arrays are typical examples for such systems. The first step in the study of such systems is the derivation of the equations of motion.

Early approaches to the dynamics formulation for multibody systems required the inversion of the system mass matrix for every integration step. Since the inversion of an $n \times n$ matrix involves operations of the Order(n^3), these are called Order(n^3) approaches. As the number of degrees of freedom (DOF) increases, this matrix inversion, for every integration step, becomes computationally expensive.

An Order (n) algorithm - so called because the computational burden increases only linearly with the number of bodies - presented earlier in [2] for systems containing rigid bodies demonstrated the achievable computational efficiency. Such an algorithm is attractive especially in on-line control schemes that consider system dynamics. The algorithm was extended in [3] systems containing flexible bodies.

2.1 System Description

A multibody system in a topological tree is shown in Figure 1. Body 1 is an arbitrarily selected reference body assumed to be connected to an imaginary



Body j	8	7	4	11	6	3	10	9	5	2
Body $L(j)$	4	4	1	6	3	1	5	5	2	1

Figure 1: General Tree Configuration and Body Pair Definition

inertially fixed body, numbered 0. For any other Body j, Body $L(j)$ is the adjacent body leading inward to Body 0 (or to the core body, Body 1). Body $L(j)$ is then defined as the body directly inboard of Body j. A kinematic joint between the body pair j and $L(j)$ allows relative motion between these bodies. Let NT_j and NR_j denote the number of translational and rotational DOF at j^{th} hinge, respectively.

2.2 Mathematical Formulation

The equations of motion are derived via Kane's method. The formulation and the corresponding solution algorithm are based on the kinematic relationships between body pairs j and $L(j)$. A joint between these bodies is defined between the q node on body j and the p node on body $L(j)$. Referring to Figure 2, we proceed as follows. The vector locating an elemental mass dm on Body j, in the inertial frame, is given by

$$\underline{R}_j = \underline{R}_f^{L(j)} + \underline{r}_p^{L(j)} + \underline{u}_p^{L(j)} + {}^{L(j)}\underline{y}^j - \left(\underline{r}_q^j + \underline{u}_q^j\right) + \underline{r}^j + \underline{u}^j \qquad (1)$$

where, $\underline{R}_f^j$ locates the body frame $\mathbb{b}_j$, $\underline{r}^j$ is a vector that defines the undeformed configuration of the elemental mass dm in $\mathbb{b}_j$, and $\underline{u}^j$ represents

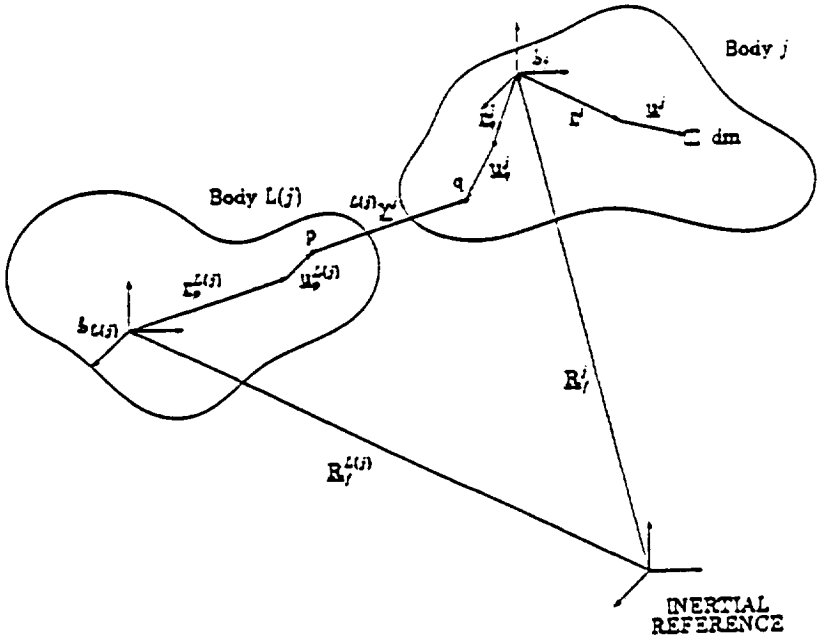


Figure 2: A Generic Hinge Between a Pair of Deformable Bodies

the elastic deformation (vector) experienced by dm. Using the method of assumed modes, the elastic deformation of body j can be expressed as the sum of the product of a set of assumed mode shape vectors $\underline{\phi}_\ell^j(\underline{r}^j)$ and their time-varying amplitudes $\eta_\ell^j(t)$ as:

$$\underline{u}^j = \sum_{\ell=1}^{NM_j} \underline{\phi}_\ell^j \left(\underline{r}^j\right) \eta_\ell^j(t) \tag{2}$$

where, NM_j denotes the number of retained modes for Body j.

The acceleration of the elemental mass dm is obtained by differentiating Eq.(1) twice with respect to time, as:

$$\begin{aligned} \ddot{\underline{R}}_j &= \ddot{\underline{R}}^j_{L(j)} + \dot{\underline{\omega}}^j_{L(j)} \times \left(\underline{r}^j + \underline{u}^j\right) + {}^{L(j)}\overset{\circ\circ}{\underline{y}}{}^j + \overset{\circ\circ}{\underline{u}}{}^j - \overset{\circ\circ}{\underline{u}}{}^j_q \\ &\quad + \left({}^{L(j)}\overset{\circ}{\underline{\omega}}{}^j_m - \overset{\circ\circ\prime}{\underline{u}}{}^j_q\right) \times \left(\underline{r}^j + \underline{u}^j - \underline{r}^j_q - \underline{u}^j_q\right) + \ddot{\underline{R}}^{rem}_j \end{aligned} \tag{3}$$

$$\begin{aligned} \ddot{\underline{R}}^j_{L(j)} &\triangleq \ddot{\underline{R}}^{L(j)}_f + \dot{\underline{\omega}}^{L(j)} \times \left(\underline{r}^{L(j)}_p + \underline{u}^{L(j)}_p + {}^{L(j)}\underline{y}^j - \underline{r}^j_q - \underline{u}^j_q\right) \\ &\quad + \overset{\circ\circ}{\underline{u}}{}^{L(j)}_p + \overset{\circ\circ\prime}{\underline{u}}{}^{L(j)}_p \times \left({}^{L(j)}\underline{y}^j - \underline{r}^j_q - \underline{u}^j_q\right), \end{aligned} \tag{4}$$

$$\dot{\underline{\omega}}^j_{L(j)} \triangleq \dot{\underline{\omega}}^{L(j)} + \overset{\circ\circ\prime}{\underline{u}}{}^{L(j)}_p \tag{5}$$

and $\underline{\ddot{R}}_j^{rem}$ in Eq.(3) represents the remainder term that contains only centrifugal and coriolis accelerations. Solid and open dots represent differentiation in the inertial and local frames, respectively. Eqs.(4) and (5) provide the recursive expressions needed for the Order(n) algorithm.

Now consider a leaf body, Body j in the tree topology. The total relative degrees of freedom associated with this body are: $NT_j + NR_j + NM_j$. For the modal degrees of freedom associated with this body one can obtain the equations of motion as:

$$\left[\mathcal{M}_\eta^j\right]\left\{\ddot{\eta}^j\right\}+\left[M_\eta^{jL(j)}\right]\begin{Bmatrix} \mathbb{b}_j \\ \mathbb{b}_j \end{Bmatrix}\cdot\begin{Bmatrix} \underline{\ddot{R}}^j_{L(j)} \\ \underline{\dot{\omega}}^j_{L(j)} \end{Bmatrix}+\left[M_\eta^{jj}\right]\begin{Bmatrix} \{\ddot{y}^j\} \\ \{\ddot{\theta}^j\} \end{Bmatrix}=\left\{f_{\eta^j}\right\}-\left\{f^*_{\eta^j}\right\}_R \tag{6}$$

The variables $\{\ddot{y}^j\}$ and $\left\{\ddot{\theta}^j\right\}$ denote the translational and rotational accelerations across joint j, and are of dimension NT_j and NR_j, respectively. The modal accelerations are denoted by $\{\ddot{\eta}^j\}$ which is of dimension NM_j. Similarly, the equations of motion associated with joint j^{th} DOF can be obtained, as:

$$\begin{aligned}\left[M_{y\theta}^{jL(j)}\right]\begin{Bmatrix} \mathbb{b}_j \\ \mathbb{b}_j \end{Bmatrix}\cdot\begin{Bmatrix} \underline{\ddot{R}}^j_{L(j)} \\ \underline{\dot{\omega}}^j_{L(j)} \end{Bmatrix} &+ \left[\mathcal{M}_{y\theta}^{jj}\right]\begin{Bmatrix} \{\ddot{y}^j\} \\ \{\ddot{\theta}^j\} \end{Bmatrix}+\left[\mathcal{M}_\eta^j\right]\left\{\ddot{\eta}^j\right\} \\ &= \begin{Bmatrix} \{f_y^j\} \\ \{f_\theta^j\} \end{Bmatrix}-\begin{Bmatrix} \{f^*_{jy}\}_R \\ \{f^*_{j\theta}\}_R \end{Bmatrix}\end{aligned} \tag{7}$$

The right hand side terms $\left\{f_{\eta^j}\right\}$, $\{f_y^j\}$ and $\{f_\theta^j\}$ represent the active force contributions and terms with $*$ contain the remainder terms in Eqs. (6) and (7). The Order (n) solution algorithm, consisting of a Frontal part and a Backsubstitution part, is as follows:

2.3 Order (N) Algorithm

2.3.1 Frontal Part

Starting with the leaf bodies in the tree topology, first the modal accelerations $\{\ddot{\eta}^j\}$ are solved for, in terms of the body j joint accelerations, and inboard body accelerations $\underline{\ddot{R}}^j_{L(j)}$ and $\underline{\dot{\omega}}^j_{L(j)}$, using Eq.(6). The result is then substituted in Eq.(7), and then the joint accelerations are solved for, solely

in terms of $\ddot{\underline{R}}^j_{L(j)}$ and $\dot{\underline{\omega}}^j_{L(j)}$. The recursive relations in Eq.(4) are then utilized to shift the inertia and active forces of Body j in terms of its inboard body DOF and the procedure is carried out for all the bodies in the tree topology, until the core body is reached. The core body accelerations are then obtained in terms of external forces. This completes the Frontal part.

2.3.2 Backsubstitution

The steps involved in this part are the reverse of the steps outlined above. Once the corebody accelerations are obtained, Eq.(4) is utilized to obtain the outboard body accelerations and, using a modified form of Eq.(4), in which the modal accelerations $\{\ddot{\eta}^j\}$ are eliminated, we obtain the joint accelerations $\{\ddot{y}^j\}$ and $\left\{\ddot{\theta}^j\right\}$. The body modal accelerations $\{\ddot{\eta}^j\}$ are then obtained using Eq.(6). This procedure is continued for all the bodies in the topology, starting with the bodies directly outboard of the core body.

The Frontal and Backsubstitution steps outlined above are also shown in Figure 1. Note that the matrix inversions required in setting up the functional evaluations needed for integration in the simulation correspond to individual joint DOF and the modal DOF, and thus much smaller than the system mass matrix. This is because, the matrix $\left[m^j_\eta\right]$ is of order $NM_j \times NM_j$ and the mass matrix $\left[\mathcal{M}^{jj}_{y\theta}\right]$ associated with the joint DOF is at the most a 6×6 matrix. Thus, it can be seen that substantial computational savings can be achieved using this algorithm, because the system mass matrix is never explicitly inverted.

3 Symbolic Processing

Symbolic processing of the equations of motion of a multi-body structure can result in a substantially more efficient simulation [4]. The increase in efficiency is achieved through simplifications that are possible because of special configuration characteristics as well as arithmetic and algebraic simplifications. The symbolic processing module described here receives its input from three sources:

(a) A configuration data file which describes the multi-body system being simulated, its topology and properties.

(b) A flexible body data file which contains data relating to the flexibility properties of each flexible body in the system.

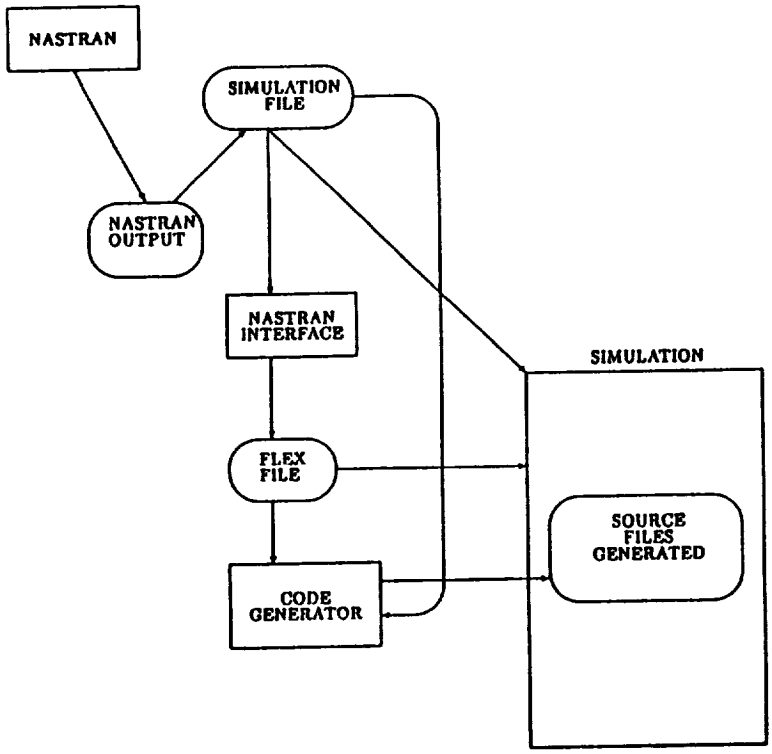


Figure 3: Context Diagram

(c) An equation file containing the equations of motion of a generic multi-flexible body system.

The output from the processor is a set of FORTRAN files containing an implementation of the specific set of equations of motion that are applicable to this multibody configuration. A context diagram is shown in Figure 3. The process involved in symbolically manipulating the equations of motion consists of the following sequence: parsing, layering, simplification, scalarization and code generation. These processes are described below.

3.1 Background

An equation consists of a left-hand-side and a right-hand-side. Equations can be represented in several forms. A convenient method of representation uses factors, terms and expressions. A factor is the smallest building block of an equation. The second building block of an equation is a term. A term may have a single factor as its element or a combination of factors separated by some operation between them. One or more of the terms when summed or multiplied together result in an expression. The left-hand-side of the equation consists of a single factor. The right-side of the equation is usually in the form of expression. If the factors are multiplied together to

make up a term, and the terms are summed together to form an expression, the equation is said to be in Sum of Products (SOP) form. If the factors are summed together to make up a term, and the terms are multiplied together to form an expression, the equation is said to be in Product of Sums (POS) form. The equations input to the symbolic manipulator are usually in matrix-vector form. The equations produced as a result of processing are in a scalar form. The different processing steps for each of the equations are parsing, reducing, layering, simplification and scalarization.

3.2 Parsing

Parsing is the translation of an algebraic expression from a form readable by humans to an internal form which can be easily manipulated by a computer. Once a set of equations representing a specific multibody model has been selected, they are "parsed" to generate the desired form of the equation so that they can be further processed.

The primary stage of the parsing process involves scanning each of the equation strings. The parser scans each of the equation strings and produces a stream of token representations. A brief description of the process of scanning and tokenization is contained below.

3.2.1 Scanning

The primary function of the scanning process is to read each input equation string and group the input characters into tokens. A token is basically an identifier. The approach used to scan the equation string could be either Top-Down, i.e., starting with the largest building block, or Bottom-Up, i.e., starting with the smallest building block. The method used here is the Top-Down method. The scanner first finds the first token (the LHS of the equation string). It inspects it to check for validity. An error message is sent if the token is not valid. Next the scanner looks for the separators of the LHS and the RHS.

Scanning of the RHS involving an expression is a slightly more complicated process. The scanner first starts out with the first expression. It then searches for the tokens that make up the expression, as well as the operators between the tokens. Once all the tokens have been parsed the scanner searches for the next expression in the list and carries out the same process until all of the expressions have been parsed. The next step is tokenizing.

3.2.2 Tokenizing

As the equation string is scanned, the tokens are inserted into their respective data structures. The information that needs to be stored for each of the tokens includes the names, their types (matrix, vector, scalar), dimensions, the pointer to the next token in the expression, and the operation between the two tokens. Once all the tokens defining the LHS and the RHS of an equation string have been created, they are linked together with the help of pointers to form the internal representation of the equation string.

3.3 Layering

This is a method by which a complex equation is split into a set of simpler equations. The method of splitting is selected in such a way that it results in the least number of operations (multiplies and adds) to be performed. An example given below demonstrates this process:

$$Z = A * B * C$$

where A is a matrix of size (1×2), B is a matrix of size (2×2) and C is a matrix of size (2×3).

The process of layering results in two equations

$$\begin{aligned} X_1 &= A * B \\ Z &= X_1 * C \end{aligned}$$

For the matrix sizes shown, if Z is computed explicitly without the use of intermediate variables, it would require 24 multiplies and 9 adds. Using the layering technique shown above, it would require only 10 multiplies and 5 adds.

3.4 Simplification

Once an equation is parsed and layered, it undergoes simplification to produce a minimal form of the equation. Simplification occurs at two stages. First the matrix-vector equation itself is simplified. Second the scalar equations describing the elements of the factor are simplified. The two stages are discussed in more detail below.

3.4.1 Matrix-Vector

The parser converts the matrix-vector algebraic equations to matrix-vector data structures. Simplifications of the equations involve operations such as the elimination of factors which are zero or have zero coefficients, identifying factors which are diagonal, etc. Basic rules of matrix-vector arithmetic follow.

3.4.2 Scalar

The scalar elements of each equation may allow simplification using the basic rules of scalar arithmetic. For space considerations, these rules are not presented here.

3.5 Scalarization

The process of generating the scalar elements of a matrix-vector equation is called scalarization. Scalarization of the factor that forms the left-hand-side of the equation results in multiple scalar equations, one for each element of that factor.

3.6 Code Generation

Finally, the parsed, layered, simplified and scalarized equation has to be converted into FORTRAN source code. This involves the conversion of the internal data structure into a string format, taking into account the various syntax rules of the FORTRAN language. This process is referred to as code generation.

4 Parallel Processing

The recursive nature of the Frontal solution algorithm makes it amenable to parallelization for a wide class of space structures. The availability of relatively inexpensive high-speed processors makes it possible to design and build parallel architectures at relatively low cost. A dedicated system with four Intel 860 processors was built to demonstrate the suitability of parallel architectures to the dynamics of multibody systems.

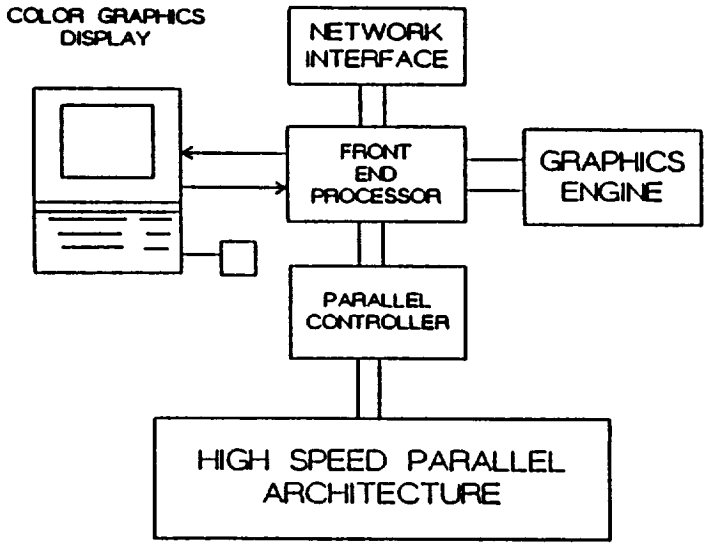


Figure 4: Hardware architecture

4.1 Architecture

The system consisted of a host machine on which all the graphical modeling, animation and all user interaction were performed and a dedicated parallel architecture on which the dynamics computations were performed. The host machine which acted as the front-end was a standalone SGI Personal Iris whereas the numeric intensive back-end consisted of a Sun SparcEngine hosting four Inter 860 processors on the VME bus (an individual 860 processor running at a 40 MHz rate is capable of a peak floating point performance of 80 MFLOPS). Details of the architecture are shown in Figure 4.

Communication between the processors was implemented using message passing. Message passing routines (send and receive) were implemented using memory shared over the VME bus.

4.2 Parallelization

The symbolic code generator discussed in the previous section was used to generate the parallelized software. The code generator read in the topology information and identified the segments of the topology which could be processed concurrently. The generated code reflected distribution of the

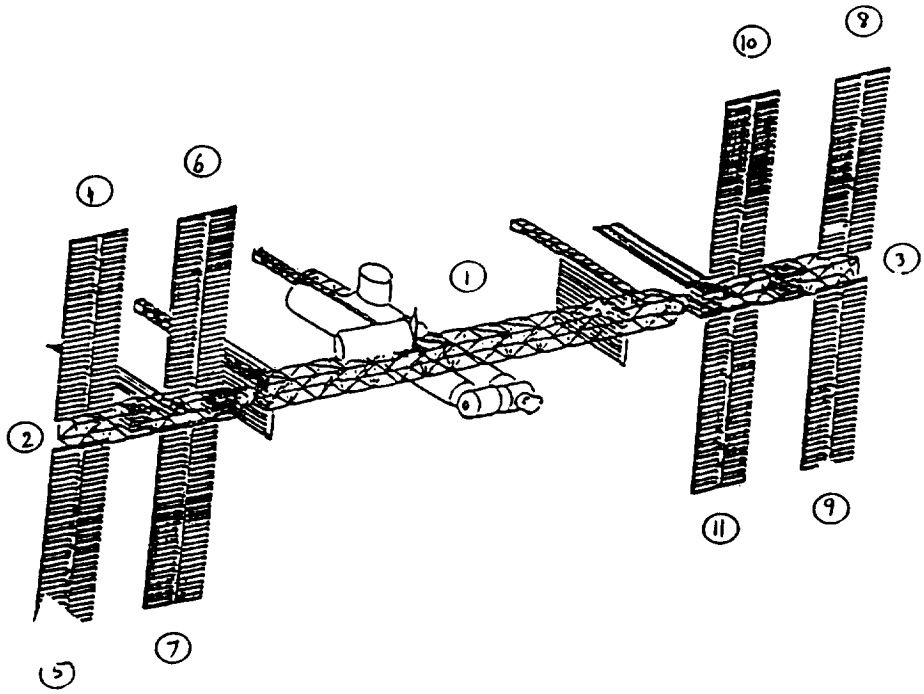


Figure 5: Assembly complete Space Station

code on the different processors and also included the messages to be passed between processors.

An 11 body (140 DOF) model of the Space Station has been chosen to demonstrate the parallelization process. Figure 5 shows the Assembly complete model of the Space Station as per the November 1989 model. This model of the Space Station has 8 photo-voltaic arrays, two power booms and the main truss (core body) making up the 11 body configuration. The 8 PV arrays are treated as leaf bodies and the frontal and kinematic computations (up to Eq. 7) for these bodies are computed simultaneously. This process is repeated recursively with the two power booms being processed simultaneously on two processors. Finally the core body accelerations are solved on a single processor and the backsubstitution is performed concurrently in the reverse sequence. The division of the frontal computations on to the four processors is shown in Figure 6.

5 Results

There are two sets of comparison results that are presented in this section. The first set is for rigid 7 body models of the Space Station and the second

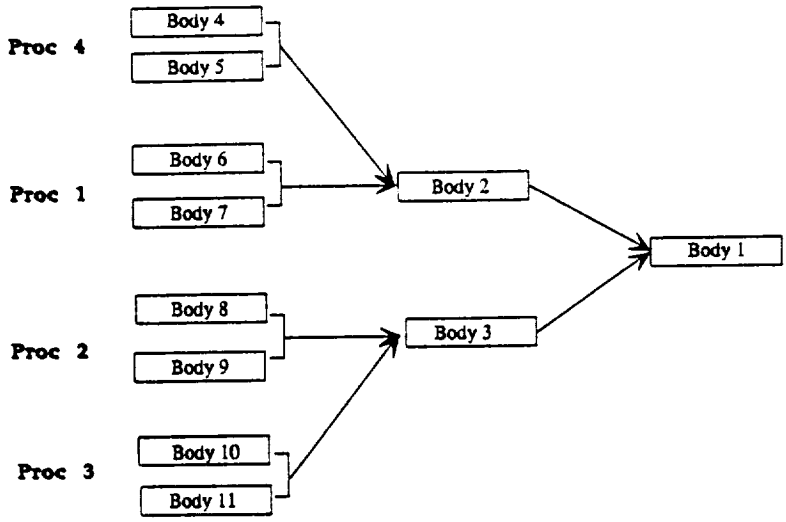


Figure 6: Division of computational load

set is for flexible 11 body models of the Station.

The 7 body model of the Space Station was integrated with the integrated Space Station Attitude Control System (SS-ACS) and several simulations were performed with varying degrees of freedom. This comparison primarily highlights the performance of the frontal solution algorithm discussed in this paper with Kane's approach as used in TREETOPS. The comparison results are presented in Figure 7.

The 11 flexible body (140 DOF) model of the Assembly complete Space Station has been used here to demonstrate the performance gain by the use of computationally efficient algorithms in combination with dedicated high speed parallel hardware. The dynamic model of the Space Station was combined with the baseline integrated SS-ACS. All the component modes in the bandwidth of the controller were retained. (The controller was running at 5 Hz whereas the highest component mode used was at 10 Hz). For this case, a total of approximately 40000 lines of FORTRAN code were generated. The complete non-linear multibody simulation for the 140 DOF system was performed for a complete orbit (90 minutes) of simulation time.

The performance comparison of the dedicated parallel processing system for the flexible body case with other commercially available hardware

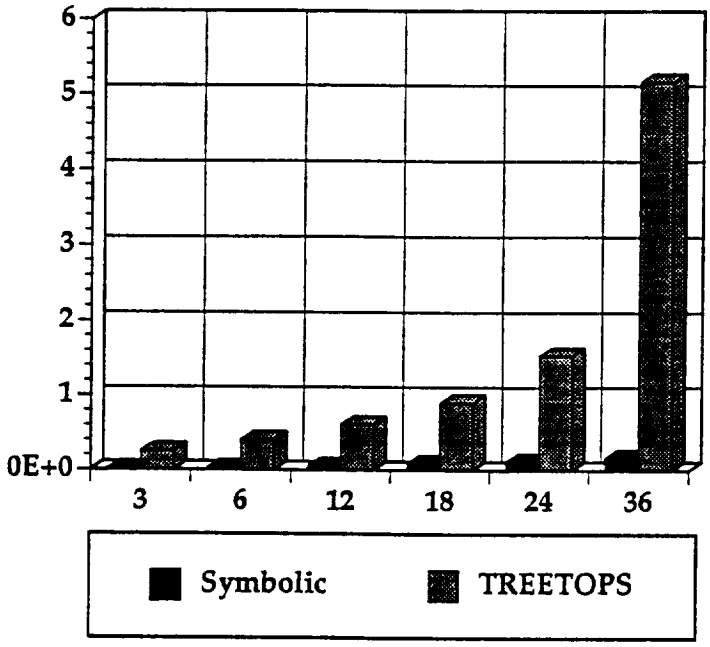


Figure 7: Rigid body comparisons

is shown below in Figure 8. Also shown are the comparisons to conventional approaches such as TREETOPS. These comparisons show that a simulation run that took over 315 hours using TREETOPS was completed in approximately 85 minutes, showing over two orders of magnitude improvement. This comparison is for the same problem with TREETOPS running on a single Intel 860 processor.

6 SUMMARY

The application of efficient algorithms to solve multibody dynamics problems has been presented in this work. While algorithms contribute to better solution strategies, efficient software implementation enhances the speed-up using these strategies further. In this work, the application of Order (n), symbolic processing approach on a parallel platform has been demonstrated. For the space station application considered in this work, a substantial performance improvement was obtained.

	Symbolic/O(n)	TREETOPS
System(4proc)	85	
System(1proc)	225	18900
SGI	675	
Sun-4	2025	
Vax-8850	2355	

Figure 8: Flexible body comparisons

References

[1] R. Singh, R. VanderVoort and P. Likins, *Dynamics of Flexible Bodies in a Tree Topology - A Computer Oriented Approach*, Journal of Guidance, Control and Dynamics, Vol. 8, No. 5, Sept-Oct 1985, pp. 584-590.

[2] R.P. Singh and B. Schubele, *Computationally Efficient Algorithm for Dynamics of Multi-Link Mechanisms*, AIAA GN&C Conference, Boston, MA, Aug. 1989.

[3] R.P. Singh, B. Schubele and S.S.K Tadikonda, *Efficient Dynamics Models for Flexible Multibody Systems Continuing Open Loop Topologies*, Accepted for presentation at the PACAM III Conference, Brazil, Jan 5-8, 1993.

[4] P.E. Nielan, *Efficient Computer Simulation of Motions of Multi-body Systems*, Dissertation, Stanford University, September 1986.