Parallel Methods for Dynamic Simulation of Multiple Manipulator Systems

Scott McMillan P. Sadayappan† David E. Orin

Department of Electrical Engineering
†Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210

Abstract

In this paper, efficient dynamic simulation algorithms for a system of m *manipulators, cooperating to manipulate a large load, are developed; and their performance, using two possible forms of parallelism on a general-purpose parallel computer, is investigated. One form, temporal parallelism, is obtained with the use of parallel numerical integration methods [1]. A speedup of 3.78 on four processors of a CRAY Y-MP8 was achieved with a parallel four-point block predictor-corrector method for the simulation of a four manipulator system. These multi-point methods suffer from reduced accuracy, and when comparing these runs with a serial integration method, the speedup can be as low as 1.83 for simulations with the same accuracy. To regain the performance lost due to accuracy problems, a second form of parallelism is employed. Spatial parallelism allows most of the dynamics of each manipulator chain to be computed simultaneously. Used exclusively in the four processor case, this form of parallelism in conjunction with a serial integration method results in a speedup of 3.1 on four processors over the best serial method. In cases where there are either more processors available or fewer chains in the system, the multi-point parallel integration methods are still advantageous despite the reduced accuracy because both forms of parallelism can then combine to generate more parallel tasks and achieve greater effective speedups. This paper also includes results for these cases.*

1. Introduction

With increases in the structural complexities of today's systems, such as multiple manipulators, multilegged vehicles, and flexible robots, parallelization of the dynamic algorithms for these systems must be considered in an effort to improve computational rates. With significant speedups over previous implementations, real-time performance of graphic animation would make man-in-the-loop remote control of these systems feasible [2]. And with super-real-time simulation (computing seconds of motion in milliseconds), an entirely new approach to on-line robotic control using predictive simulation for planning is within range [3]. One promising area of research which is striving to achieve these computational rates focuses on the use of parallel algorithms.

In previous work, fine-grain parallel algorithms have been developed for robot dynamics computations. Some require the use of special-purpose architectures to implement the fine-grain parallelism of the computations required for a single-chain system [4, 5, 6, 7]. Others decompose the algorithm into groups of concurrent tasks that are scheduled on a number of

tightly coupled parallel processors to produce speedup [8, 9]. All of these algorithms, while designed to perform well on specific parallel architectures, are usually much less efficient when implemented on available general-purpose parallel machines with a limited number of processors. These systems were not designed to handle the large amount of interprocessor communication and synchronization that is required by the fine-grain tasks.

In our work, parallel algorithms which run efficiently on existing general-purpose parallel computers are investigated. In the initial stage of this research, the dynamic algorithms required to simulate a single, open-chain robotic manipulator were effectively parallelized [1]. The approach used a parallel block predictor-corrector integration method to achieve a temporal parallelism which enabled the forward dynamics problem to be computed for multiple points in time simultaneously. In this paper, the work is extended to simulate systems of multiple manipulators that are cooperating to manipulate a large load. In this case, a second form of parallelism which corresponds to the system's structural parallelism is explored. Called spatial parallelism, it allows most of the dynamics of the individual chains to be computed in parallel as well.

This work shows that there are various costs associated with the two forms of parallelism which affect their efficiency. With temporal parallelism, the accuracy of the parallel integration methods is lower than the corresponding serial methods. As a result, smaller stepsizes must be used, and hence, more computation must be performed with the temporal parallel methods to achieve the same accuracy. With spatial parallelism, most of the dynamics may be computed in parallel; however, a serial portion of the algorithm which computes the dynamics of the common load reduces the overall efficiency of this parallelism as well. The advantage of these methods is that they may be combined by implementing the spatial parallelism *within* each parallel integration task to gain even greater speedups. Our results show the effects of reduced efficiency of both methods and how they are combined to gain the greatest speedups on varying numbers of processors.

In the following section, the algorithm to simulate the multiple manipulator system is developed. In this development, the dynamics used in the previous work for a single manipulator are extended and the spatial parallelism in this algorithm is presented. In Section 3, the block predictor-corrector integration methods are presented which provide various amounts of temporal parallelism in the simulation problem. Section 4 discusses how both forms of parallelism are combined and implemented on a general-purpose parallel computer. This algorithm is then implemented on the CRAY Y-MP8/864, and the speedup results are given in Section 5 for various configurations of the simulation system including spatial parallelism only, temporal parallelism only, and a combination on various numbers of the CRAY's processors. Finally, a summary and conclusions are presented in Section 6.

2. Parallel Algorithm for Multiple Manipulator Dynamic Equations

In this section, an efficient dynamic simulation algorithm for a multiple, closed-chain manipulator system is developed, and the spatial parallelism in the algorithm is investigated. The system contains m manipulators, each with N degrees of freedom, that are

rigidly grasping a common load, called the reference member. Each simple, closed chain (manipulator) is governed by the dynamic equations of motion for a single chain. Numbering the chains in the system 1 through m, the equation for each chain is given as follows:

$$\tau_k = \mathbf{H}_k(\mathbf{q}_k)\ddot{\mathbf{q}}_k + \mathbf{C}_k(\mathbf{q}_k, \dot{\mathbf{q}}_k) + \mathbf{G}_k(\mathbf{q}_k) + \mathbf{J}_k^T(\mathbf{q}_k)\mathbf{f}_k \quad \text{for all } k = 1, \ldots, m, \tag{1}$$

where

τ_k	$N \times 1$ vector of torques/forces of the joint actuators,
$\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k$	$N \times 1$ vectors of joint positions, rates, and accelerations,
$\mathbf{H}_k$	$N \times N$ symmetric, positive definite inertia matrix,
$\mathbf{C}_k$	$N \times 1$ vector specifying centrifugal and coriolis effects,
$\mathbf{G}_k$	$N \times 1$ vector specifying the effects due to gravity,
$\mathbf{J}_k$	$6 \times N$ Jacobian matrix, and
$\mathbf{f}_k$	6×1 force exerted by the tip of chain k on the reference member.

The "for all" in Eq. (1) indicates that the equation may be computed for all chains in parallel provided the required quantities are known. Since we are interested in the solution to the Forward Dynamics (or simulation) problem, the state of the system, consisting of joint positions and rates, and the input joint torques/forces are given. The joint accelerations for each chain must then be computed. Lilly and Orin [10] present an efficient serial algorithm for Forward Dynamics which is the basis for the parallel implementation used in this paper.

By grouping some terms from Eq. (1) and solving for the joint accelerations, we obtain the following equation:

$$\ddot{\mathbf{q}}_k = \ddot{\mathbf{q}}_{k_{open}} - \mathbf{H}_k^{-1}\mathbf{J}_k^T\mathbf{f}_k \quad \text{for all } k = 1, \ldots, m, \tag{2}$$

where $\ddot{\mathbf{q}}_{k_{open}}$ is the vector of joint accelerations for chain k if it were not contacting the reference member causing the tip force to be zero. This is called its open-chain solution which was implemented in [1]. Note that this quantity, as well as $\mathbf{H}_k^{-1}\mathbf{J}_k^T$, depends only on the system state and input joint torques/forces and may be computed from known quantities.

An equivalent operational-space formulation for the dynamics of a closed-chain manipulator can be found by using the following relationship:

$$\dot{\mathbf{x}}_k = \mathbf{J}_k\dot{\mathbf{q}}_k, \tag{3}$$

where $\dot{\mathbf{x}}_k$ is the Cartesian velocity of the tip of the manipulator. Taking the time derivative of both sides yields the equation for the closed-chain acceleration of the tip:

$$\ddot{\mathbf{x}}_k = \dot{\mathbf{J}}_k\dot{\mathbf{q}}_k + \mathbf{J}_k\ddot{\mathbf{q}}_k, \tag{4}$$

and substituting from Eq. (2) yields the following:

$$\ddot{\mathbf{x}}_k = \ddot{\mathbf{x}}_{k_{open}} - \mathbf{\Lambda}_k^{-1}\mathbf{f}_k, \quad \text{for all } k = 1, \ldots, m, \tag{5}$$

where

$$\ddot{\mathbf{x}}_{k_{open}} = \dot{\mathbf{J}}_k\dot{\mathbf{q}}_k + \mathbf{J}_k\ddot{\mathbf{q}}_{k_{open}}, \text{ and} \tag{6}$$

$$\Lambda_k = \left(\mathbf{J}_k \mathbf{H}_k^{-1} \mathbf{J}_k^T\right)^{-1}. \tag{7}$$

The quantity, $\ddot{\mathbf{x}}_{k_{open}}$, is the acceleration of the tip of chain k if it were open, and Λ_k is the operational space inertia matrix [11]. Both quantities may be computed given the current state of the system and the input. The physical interpretation of Λ_k is an inertial quantity which combines the dynamic properties of the chain and projects them to its tip. It is the physical resistance that is "felt" when a force is exerted on the tip, and it defines the relationship between the force that is applied to the tip, and the resulting acceleration of the tip of the chain, $\ddot{\mathbf{x}}_k$. The advantage to using these operational space quantities is that the matrix sizes are constant regardless of the number of degrees of freedom in the chain.

With the chains attached with a fixed grip to the reference member, the accelerations of the tips of each chain, at an appropriate point attached to the end-effector, are the same and are equal to the reference member acceleration, $\mathbf{a}_r$. As a result, Eq. (5) can be rewritten as follows:

$$\mathbf{a}_r = \ddot{\mathbf{x}}_{k_{open}} - \Lambda_k^{-1} \mathbf{f}_k \qquad \text{for all } k = 1, \ldots, m. \tag{8}$$

Now the dynamic behavior of the reference member can be determined using a spatial force balance equation. This states that the sum of the spatial forces exerted by the tips of the chains and any other external forces (including gravity) is related to the acceleration of the reference member through its inertia. This may be written as follows:

$$\sum_{k=1}^{m} \mathbf{f}_k + \mathbf{g}_r = \mathbf{I}_r \mathbf{a}_r + \mathbf{b}_r, \tag{9}$$

where

$\mathbf{g}_r$ 6×1 vector specifying the force of gravity exerted on the reference member,
$\mathbf{a}_r$ 6×1 acceleration vector of the reference member,
$\mathbf{I}_r$ 6×6 spatial inertia of the reference member [10],

and the last term, $\mathbf{b}_r$, is a spatial vector specifying the bias force due to the spatial velocity of the reference member.

In this work, we also assume that the chains are not in singular configurations, and consequently, all of the Λ_k^{-1} matrices are non-singular. Therefore, the unknown force terms, $\mathbf{f}_k$, may be isolated in Eq. (8) and substituted into Eq. (9). After collecting terms, the following equation results:

$$\left[\mathbf{I}_r + \sum_{k=1}^{m} \Lambda_k\right] \mathbf{a}_r = \left[\mathbf{g}_r - \mathbf{b}_r + \sum_{k=1}^{m} \Lambda_k \ddot{\mathbf{x}}_{k_{open}}\right]. \tag{10}$$

This equation states that the sum of all the inertias of the system multiplied by the reference member acceleration is equal to the sum of all the forces that are exerted on the reference member. The acceleration term, $\mathbf{a}_r$, may now be determined from this linear system of

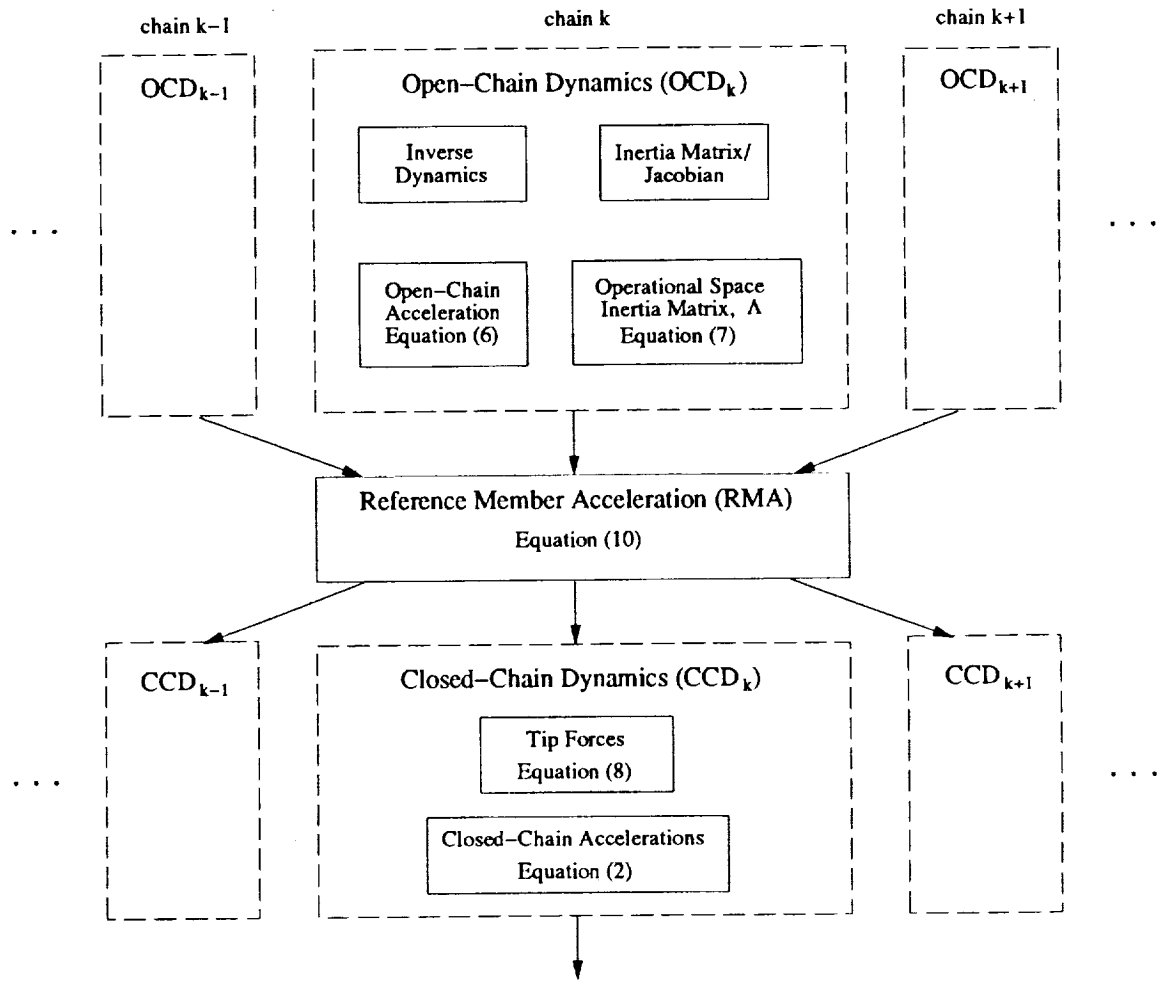


Figure 1: Closed-Chain Dynamics Algorithm.

equations using any linear system solver (in this case Cholesky decomposition can be used). Finally $\mathbf{a}_r$ is substituted back into Eq. (8) to determine the tip forces, $\mathbf{f}_k$, on each chain and this can then be used to determine the joint accelerations from Eq. (2).

The flowchart in Figure 1, outlines the steps in the algorithm to compute the desired accelerations. This constitutes the "derivative computation" which uses the state information that is provided by a numerical integration routine. The additional computation required for the closed chain dynamics that were not present in the open-chain algorithm used in [1] is indicated with the appropriate equation numbers except for the computation of the manipulator Jacobian. With slight modification to the inertia matrix routine that was used in [1], this matrix may be computed as well. The algorithm for this computation can also be found in [12]. Note that the Open-Chain Dynamics (OCD) and Closed-Chain Dynam-

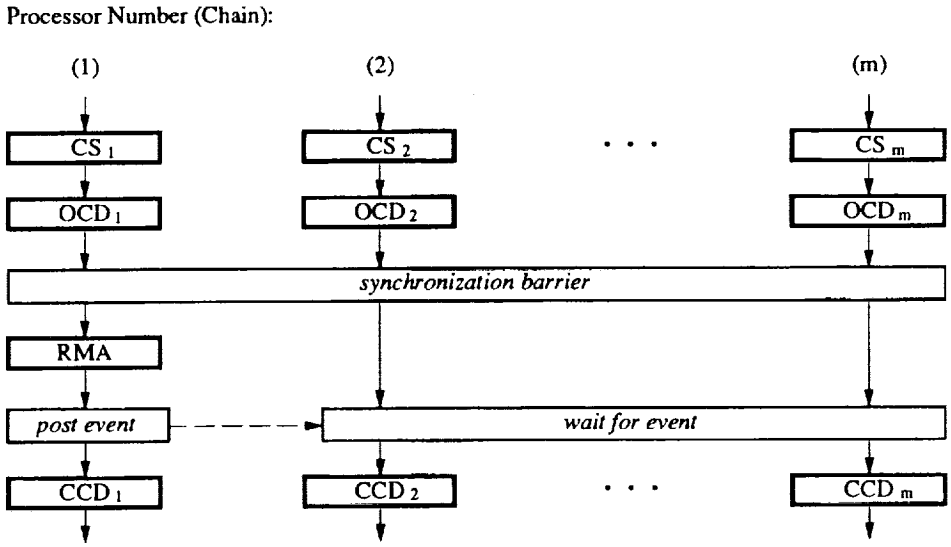


Figure 2: Spatial Parallelism in Multiple-Chain Dynamic Equations (Serial RMA Computation).

ics (CCD) blocks are repeated for each chain in the system, while the Reference Member Acceleration (RMA) is performed once for a given derivative evaluation. This implies that the OCD and CCD computations for each chain may be executed in parallel.

Figure 2 shows how this algorithm, along with a numerical integration method, would be implemented on a general-purpose parallel computer. The CS (Compute State) block consists of the integration method that is used to compute the state of the required chain, and will be discussed in the next section. A processor synchronization is required after the parallel OCD computation in order to collect the chains' operational space inertia matrices and open-chain accelerations before the serial RMA computation is performed. In Figure 2 this is shown as a *barrier* which holds the parallel tasks which have completed the OCD section until all m of them have finished. After the serial computation is completed by one processor, it signals the other processors to continue with the parallel CCD computations. This is accomplished by posting an *event* signal for which the other processors are waiting.

This computation can be modified to remove the event synchronization as shown in Figure 3. This is accomplished by allowing each task to maintain its own "copy" of the reference member information and perform the same computation on all of them. While this introduces redundant computation, it can actually reduce the wallclock time because a costly synchronization has been removed and the same time that was spent waiting for one processor to perform the RMA calculation is now used to perform it on all of the processors. This algorithm is then used within the framework of various parallel integration methods that are described in the next section.

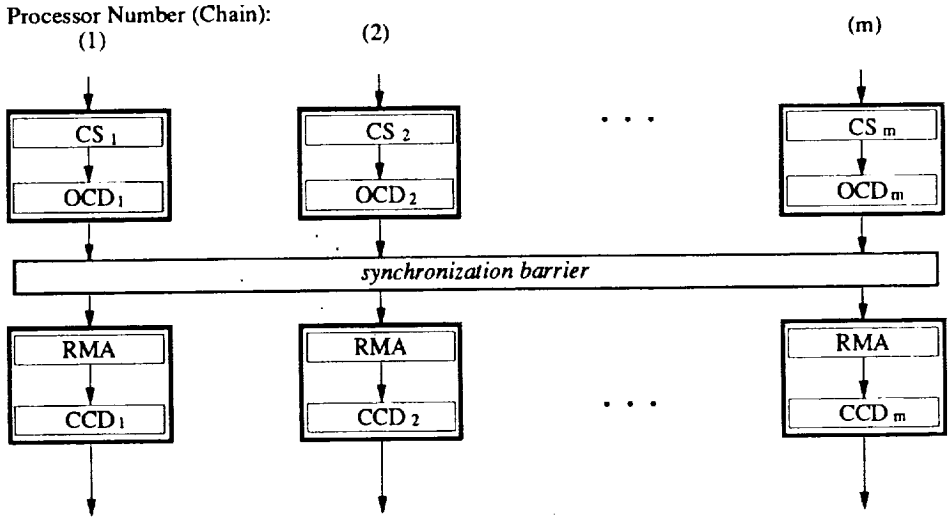


Figure 3: Spatial Parallelism in Multiple-Chain Dynamic Equations (Redundant Parallel RMA Computation).

3. Parallel Integration Methods

Many different algorithms exist to perform numerical integration. One set of algorithms which has shown in the past to be readily parallelizable are predictor-corrector methods. The standard serial algorithm, commonly abbreviated PECE, consists of two pairs of steps which correspond to the letters of the abbreviation. The steps consist of Predicting the next state and Evaluating the derivative based on this prediction, and then Correcting the state with a corresponding derivative Evaluation. The derivative evaluations required in both steps to determine the accelerations were presented in the last section, and the methods for predicting and correcting the states are described here.

In this paper, fifth-order methods are used because they provide an adequate tradeoff between accuracy, which tends to increase with order, and stability, which tends to decrease with order. Figure 4 shows the quantities needed and computed in both steps of the serial method. This method, usually called PECE5, will be referred to as B1PC5 in the rest of this paper to indicate a one-point block method which utilizes fifth-order predictor-corrector formulas. The predictor step, shown in Figure 4(a), uses a linear combination of five past derivative values, $\mathbf{f}_{-4}, \ldots, \mathbf{f}_0$ (hence it is fifth-order) and the state at the most recent point in time, $\mathbf{y}_0$ to predict the state of the system (joint and reference member positions and rates) at the next point in time, $\mathbf{y}_1^p$. With this, the algorithm in the previous section can be used to compute the derivative (joint and reference member accelerations) at this point, $\mathbf{f}_1^p$, which is based on the predicted state. To correct the state in the second step of the method, the quantities shown in Figure 4(b) are used. In this step, the "oldest" derivative value is dropped and the linear combination used to compute the corrected state, $\mathbf{y}_1^c$, now includes the new predicted derivative value. The same state quantity, $\mathbf{y}_0$, is used rather

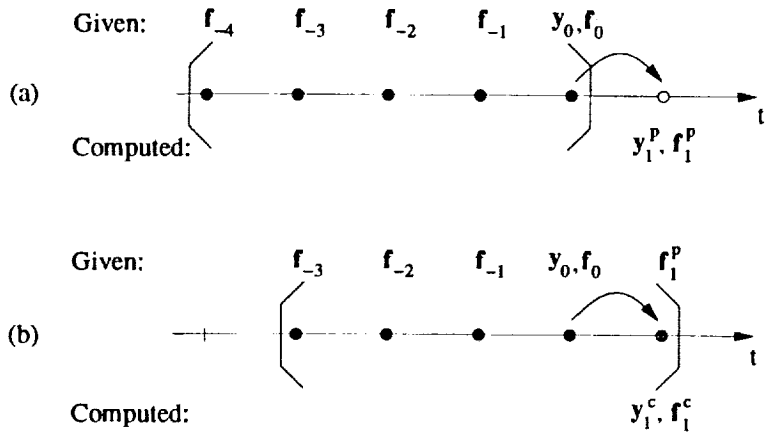


Figure 4: Serial B1PC5 Integration: (a) predictor step, (b) corrector step.

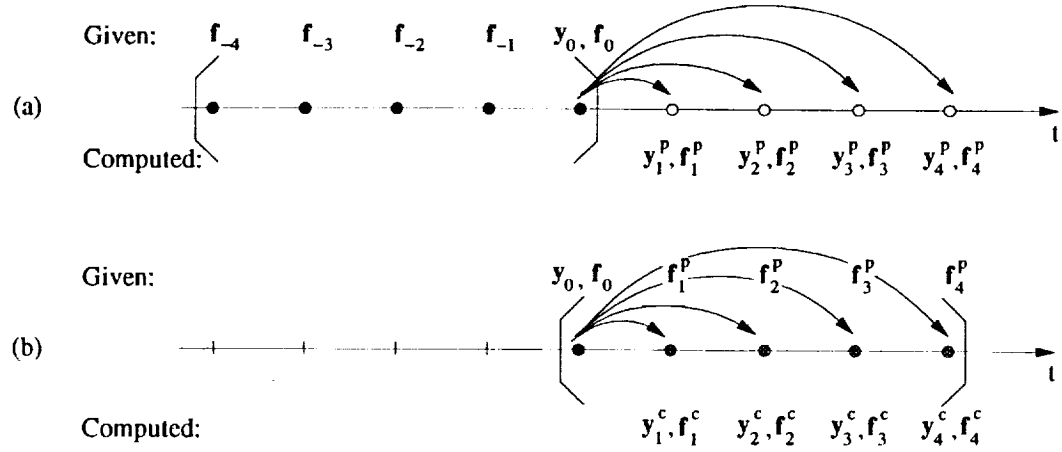


Figure 5: Four-Point Parallel B4PC5 Integration: (a) predictor step, (b) corrector step.

than the predicted one for reasons of stability and accuracy of the method. Finally, the corrected derivative value is computed using $\mathbf{y}_1^c$. Then in the next iteration of the method, the values are shifted and four old derivative values and the new corrected derivative and state values are used.

A parallel version of this method called the block predictor-corrector method was first formulated by Birta and Abou-Rabia [13] and used in our previous work [1]. The fifth-order version of this method, B4PC5, is shown in Figure 5. In this method, a block of four points are computed in parallel during a single iteration. Each processor is responsible for computing the required quantities at a single point in the block. In the predictor step shown in Figure 5(a), each processor computes the predicted value for its point using the same information (but with a different linear combination) in parallel. Then each uses

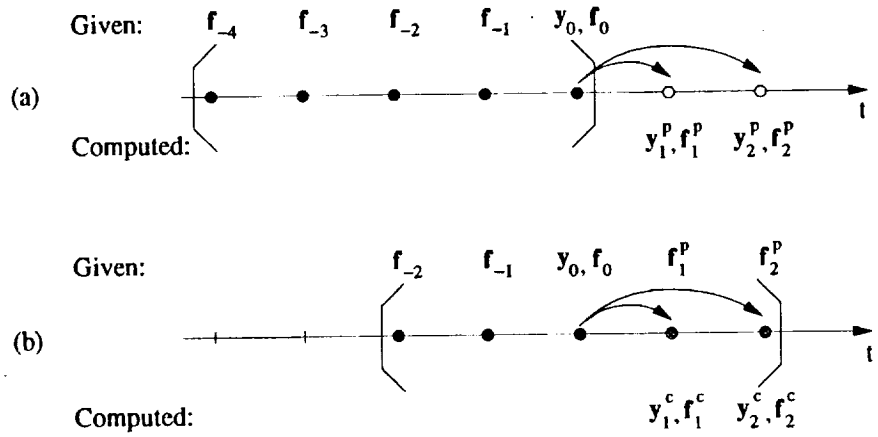


Figure 6: Two-Point Parallel B2PC5 Integration: (a) predictor step, (b) corrector step.

the dynamics algorithm to compute the derivative at its point in parallel as well. In the corrector step, an entire block of old derivative values are discarded in favor of the predicted ones as shown in Figure 5(b). Once the processors have swapped the derivative information from the predictor step, correction and subsequent derivative evaluation can again be done in parallel.

The B4PC5 method offers a way to simulate the system using four processors in parallel. Because the method extrapolates farther from the known values in the predictor step, it is subject to larger errors for a given stepsize. Also, when both forms of parallelism are combined in the next section, the number of parallel tasks may exceed the number of processors available. For these reasons, we developed a two-point parallel method called B2PC5 which lies between the two methods described thus far in terms of parallelism and accuracy. This method is shown in Figure 6. Instead of predicting four points as in the B4PC5, it only predicts two new points and thus can be implemented in parallel on two processors. The coefficients for both steps of this method were computed using the method of undetermined coefficients so that the resulting method is fifth-order as well.

The structure of the temporal parallelism for a single block of the parallel integration methods presented in this section is shown in Figure 7. The PE (compute Predicted state and Evaluate the derivative) and CE (compute Corrected state and Evaluate the derivative) blocks in this figure make up the predictor and corrector steps in these methods along with the derivative evaluations described in the previous section. There are p parallel tasks which correspond to the number of block points computed by the method during a single iteration. A single column of blocks represents the computation of the solution of the entire m-chain system for a specific point in time on one of p processors. Barriers are used to synchronize these parallel tasks after each derivative evaluation so that state and derivative information may be exchanged. An event is also used so that a small amount of serial processing may be performed by a single processor before starting the next integration iteration.

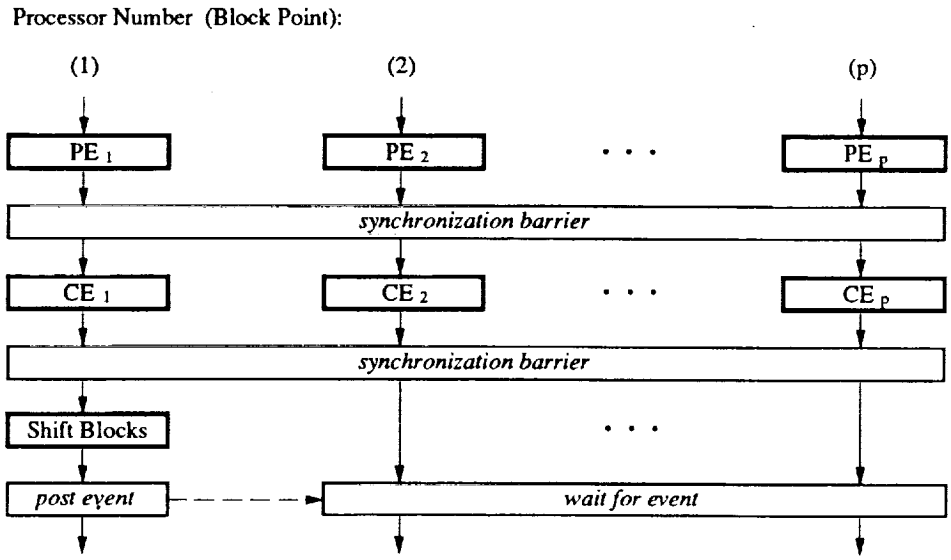


Figure 7: Temporal Parallelism in Dynamic Simulation.

4. Implementation of Combined Parallelism

Both forms of parallelism, spatial and temporal, can be combined by introducing the spatial parallelization of the derivative evaluations in Figure 3 into the individual PE/CE computational blocks of Figure 7. In this case, the CS (Compute State) blocks of Figure 3 become the predictor or corrector equations of the desired integration method. A modification is then made to decrease the required synchronization between all of the tasks to improve the performance. This modification is made to the barriers and events associated with the temporal parallelism. Because the integration of the state of each chain depends only on its own values at all of the block points, the temporal barriers are broken apart so that each one only synchronizes with the processors associated with the same chain. In some sense, these temporal synchronization points have been spatially parallelized. In addition, the serial Shift Blocks task can also be spatially parallelized.

Figure 8 shows the resulting implementation which consists of as many as mp parallel tasks. Also included in this figure are the synchronization commands that are used in the implementation on the CRAY Y-MP8/864. The simulation code was written in FORTRAN (Version 5.0 running under UNICOS Release 6.0) using macrotasking commands to implement the parallelism. The BARSYNC commands correspond to the synchronization barriers and the argument supplied to this command corresponds to the number of parallel tasks that must be synchronized by it. Finally, the EVPOST-EVWAIT commands provide the mechanism by which certain processors are suspended while others perform serial operations.

In order to examine the effects of various configurations on the parallelism (taking the

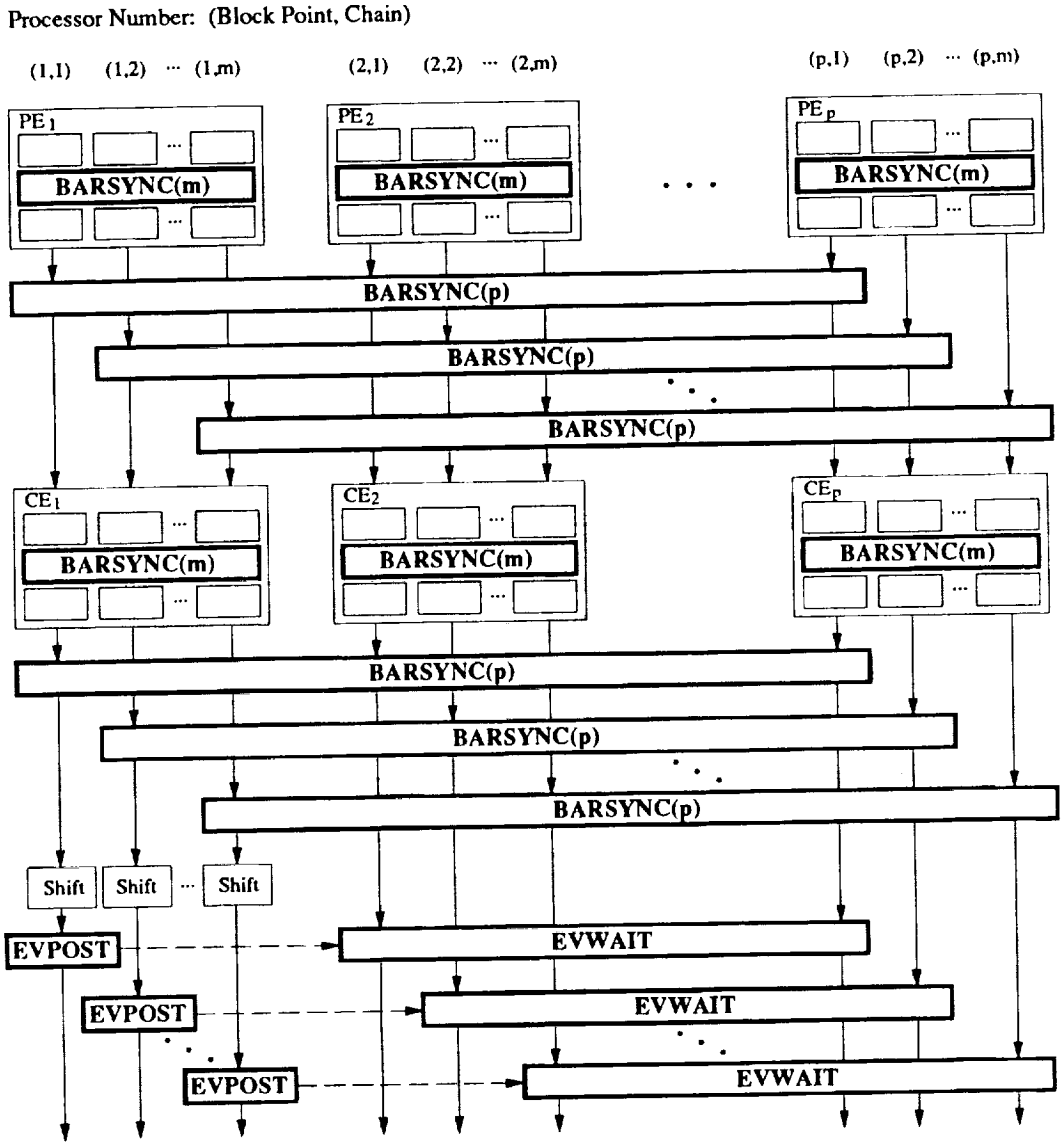


Figure 8: Structure of Combined Spatial and Temporal Parallelism.

number of integration block points, number of chains, and number of physical processors into account), a mechanism was included in the implementation which allows the user to specify the number of desired parallel tasks along both the temporal and spatial dimensions of the simulation regardless of the number of chains and block points. These numbers are denoted by $\tilde{m}$ and $\tilde{p}$, respectively, and are used in place of m and p in Figure 8 when referring to the number of actual parallel computational blocks. With this mechanism, for example, the required computation could be paired off and a single processor could be

Table 1: Speedup Results for B4PC5.

# Processors Requested	Chain Tasks, $\tilde{m}$	Integration Tasks, $\tilde{p}$	T (sec.)	S_p
1	1	1	0.241	1.00
4	1	4	0.0637	3.78
	2	2	0.0668	3.61
	4	1	0.0731	3.30
8	2	4	0.0390	6.18
	4	2	0.0417	5.78

responsible for the computations of two chains (or two block points, or both) instead of just one. This would allow the computation of a larger system, in which mp exceeded the number of processors, to be efficiently mapped to smaller parallel machines. In order to maintain the load balance, however, the specified partitions in both dimensions should be integer divisors of the total number of block points and chains, respectively. Results for various partitions of the computation using this method are given in the next section.

5. Results

With the parallel algorithm described in the last section, a system consisting of four PUMA 560 manipulators was simulated. A test trajectory with a duration of one second was generated which consisted of lifting a 4kg object 0.8 meters straight up. Then an appropriate joint torque profile was computed which, when applied to the joints of the manipulators, would produce the desired motion. These torques were used as input into the simulator, and the error in the final position of the reference member was used as a measure of accuracy for the various integration methods. Only this value was reported for brevity and also its accuracy was representative of the accuracy of the rest of the states in the system.

The simulation was then executed using the B4PC5 integration method on 1, 4, and 8 of the Y-MP's eight processors. Using the mechanism for partitioning the computation among existing processors, the block point and chain computations were partitioned singly (one chain and one block point per processor), in pairs (two chains and two blocks points), all together, or any useful combination thereof. A fixed integration stepsize was used and varied over a number of runs so that a profile of execution time versus error was produced. To get a fair comparison of relative performance between the different integration methods in later experiments, an estimate of the execution time, T, required to achieve an error of 10^{-6} meters was reported in Table 1.

Examining these execution times for a given number of processors, it can be seen that they increased as the number of temporal parallel tasks, $\tilde{p}$, decreased. When $\tilde{p}$ is less than four, the full amount of temporal parallelism provided by this integration method

Table 2: Accuracy Performance for Various Configurations.

# Processors Requested	Chain Tasks, $\tilde{m}$	Integration Tasks, $\tilde{p}$	Block Size	T (sec.)	S_p	S_a	S_T
1	1	1	4	0.241	1.00	0.485	0.485
			2	0.152	1.00	0.770	0.770
			1	0.117	1.00	1.00	1.00
4	1	4	4	0.0637	3.78	0.485	1.83
	2	2	4	0.0668	3.61	0.485	1.75
			2	0.0444	3.42	0.770	2.63
	4	1	4	0.0731	3.30	0.485	1.60
			2	0.0463	3.28	0.770	2.53
			1	0.0377	3.10	1.00	3.10
8	2	4	4	0.0390	6.18	0.485	3.00
	4	2	4	0.0417	5.78	0.485	2.80
			2	0.0295	5.15	0.770	3.97

is not being fully utilized. As a result, increased numbers of redundant reference member acceleration (RMA) calculations are being performed by each processor in an effort to avoid the extra synchronization point that was discussed at the end of Section 2. Consequently, the best speedups due to parallelization, S_p, were achieved by using the greatest amount of temporal parallelism which was as high as 3.78 on four processors. Runs were also made while requesting all eight of the Y-MP's processors and a speedup of 6.18 was still achieved despite an inability to gain dedicated use of these processors in our multi-user environment.

When $\tilde{p}$ was less than four, simulations using an integration method with a smaller block size, such as B2PC5 or serial B1PC5, were also tried. Because these methods compute fewer points per iteration, the cost that is incurred is an increased number of iterations (and hence, parallel task synchronizations) than the B4PC5 method for a given integration stepsize. However, the overall efficiency of these methods increased because they were more accurate. This resulted in fewer iterations and less execution time for a given amount of error. The complete set of results using the various integration block sizes as well as parallel configurations is shown in Table 2.

In this table, S_p is the speedup for a given method over the serial runtime using the same integration method. Since there is an accuracy loss associated with the larger block methods a speedup due to algorithmic changes, S_a, is also reported. This corresponds to the amount of time relative to the B1PC5 method that is required by the methods to compute the trajectory with a given error. Based on the serial results for the three integration methods, the traditional serial method, B1PC5, took less than half the time to simulate the trajectory to the desired accuracy as the B4PC5, and the performance of B2PC5 fell in between. The last column shows the total effective speedup of the various configurations. This is based on the time required for the method to simulate the trajectory to the desired accuracy as compared to the best serial time which was exhibited by the B1PC5 method,

and it is equal to the product of S_p and S_a. When compared with the best integration methods the S_p speedups obtained with the B4PC5 method must therefore be cut in about one-half, and the B2PC5 speedups are reduced by approximately 23%.

Therefore, the best effective speedups were obtained using the integration method with the smallest block size while partitioning the computation so that the number of parallel tasks was equal to the number of processors. Since the system had four chains, the serial B1PC5 method was the best on four processors and the resulting speedup was 3.1. On eight processors, where the serial method could not be used and still have eight parallel tasks, the B2PC5 method showed the greatest speedup at 3.97. From these results it would appear that the B4PC5 has no advantage. However, if a system with a smaller number of chains is to be simulated, this method would allow a greater number of parallel tasks to be generated than the other methods and would most likely exhibit the greatest speedup.

6. Summary and Conclusions

In this paper, two approaches for achieving effective parallelization for dynamic simulation on a general-purpose parallel computer were presented. One approach that was discussed in [1] for parallel simulation of a single chain system was based on temporal parallelism achieved with the use of a parallel numerical integration method. In this paper, the work has been extended to include multiple chain systems which introduce a second form of parallelism. Called spatial parallelism, the form comes from the ability to compute the dynamics of individual chains in the system simultaneously.

Various ways to use both forms of parallelism to the greatest advantage were investigated. The greatest effective speedup from these methods was gained by partitioning the computation into as many load balanced parallel tasks as possible while using the integration method with the smallest block size. This implies that the greatest amount of spatial parallelism, and the most accurate integration methods should be employed.

With the general rule in mind, our results for the simulation of a four chain system showed that the greatest speedup on four processors of the CRAY Y-MP8 was 3.1. This was achieved with spatial parallelism only, and the use of the serial predictor-corrector integration method. Even greater speedup was achieved on eight processors when full spatial parallelism was used. In this case, a two-point parallel integration method was used to achieve the desired amount of parallel tasks. And it appears that the four-point integration method would be beneficial if sixteen processors are available.

An additional benefit to these forms of parallelism is that they do not preclude any of the previous work mentioned in the introduction that dealt with the fine-grain parallel algorithms for computation of the robot dynamics quantities. The special-purpose architectures required could be set up in parallel and used in conjunction with the methods discussed in this paper. The resulting combination of parallel computation could be thought of as occurring in three dimensions, and shows promise for even greater speedups.

7. References

[1] S. McMillan, D. E. Orin, and P. Sadayappan, "Towards Super-Real-Time Simulation of Robotic Mechanisms Using a Parallel Integration Method," *IEEE Transactions on Systems, Man, and Cybernetics*, January/February 1992.

[2] L. Conway, R. Volz, and M. Walker, "Tele-Autonomous Systems: Methods and Architectures for Intermingling Autonomous and Telerobotic Technology," in *Proc. of 1987 IEEE Int. Conf. on Robotics and Automation*, (Raleigh, NC), pp. 1121–1130, 1987.

[3] D. E. Orin, "Dynamical Modelling of Coordinated Multiple Robot Systems," in *Report of Workshop on Coordinated Multiple Robot Manipulators* (A. J. Koivo and G. A. Bekey, eds.), (San Diego, CA), Jan. 1987.

[4] M. Amin-Javaheri and D. E. Orin, "Systolic Architectures for the Manipulator Inertia Matrix," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 18, pp. 939–951, November/December 1988.

[5] C. S. G. Lee and P. R. Chang, "Efficient Parallel Algorithms for Robot Forward Dynamics Computation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, pp. 238–251, March/April 1988.

[6] P. Sadayappan, Y. L. C. Ling, K. W. Olson, and D. E. Orin, "A Restructurable VLSI Robotics Vector Processor Architecture for Real-Time Control," *IEEE Trans. on Robotics and Automation*, vol. 5, pp. 583 - 599, October 1989.

[7] A. Fijany and A. K. Bejczy, "Techniques for Parallel Computation of Mechanical Manipulator Dynamics. Part II: Forward Dynamics," *Control and Dynamic Systems*, vol. 40, pp. 357–410, 1991.

[8] C. L. Chen, C. S. G. Lee, and E. S. H. Hou, "Efficient Scheduling Algorithms for Robot Inverse Dynamics Computation on a Multiprocessor System," in *IEEE Int. Conf. on Robotics and Automation*, (Philadelphia, PA), pp. 1146–1151, 1988.

[9] J. Y. S. Luh and C. S. Lin, "Scheduling of Parallel Computation for a Computer-Controlled Mechanical Manipulator," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-12, pp. 214 - 234, March/April 1982.

[10] K. W. Lilly and D. E. Orin, "Efficient Dynamic Simulation for Multiple Chain Robotic Mechanisms," in *Proc. 3rd Annual Conf. on Aerospace Computational Control* (D. E. Bernard and G. K. Man, ed.), vol. 1, (Pasadena, CA), pp. 73–87, December 1989.

[11] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE Journal of Robotics and Automation*, vol. RA-3, pp. 43–53, February 1987.

[12] K. W. Lilly and D. E. Orin, "Alternate Formulations for the Manipulator Inertia Matrix," *The International Journal of Robotics Research*, The MIT Press, vol. 10, pp. 64–74, February 1991.

[13] L. G. Birta and O. Abou-Rabia, "Parallel Block Predictor-Corrector Methods for ODE's," *IEEE Trans. on Computers*, vol. C-36, pp. 299–311, March 1987.

OVERVIEW OF MULTIBODY DYNAMICS ANALYSIS CAPABILITIES

Hon M. Chun and James D. Turner

Fifth Annual
NASA/NSF/DoD Workshop
on
Aerospace Computational Control

Santa Barbara, California
August 17-19, 1992

Photon Research Associates, Inc.
1033 Massachusetts Avenue
Cambridge, MA 02138

Photon Research Associates, Inc.

PRESENTATION OVERVIEW

- Historical Overview
- Dynamic Interactions
- Recent Advances
- Current Capabilities
- Unresolved Issues

PRA
Photon Research Associates, Inc.

HISTORICAL OVERVIEW

MECHANISMS

Loop Topology
2D Motion
Lagrangian Equations
Joint Libraries

SPACECRAFT

Tree Topology
3D Motion
Newton/Euler Equations
General Joints

Tree & Loop Topologies
2D & 3D Motion
Various Formulations
Joint Libraries and General Joints

Fast Algorithms
Parallel Computers
Symbolic Equations

Photon Research Associates, Inc.

SPACECRAFT COMPLEXITY

SPACECRAFT

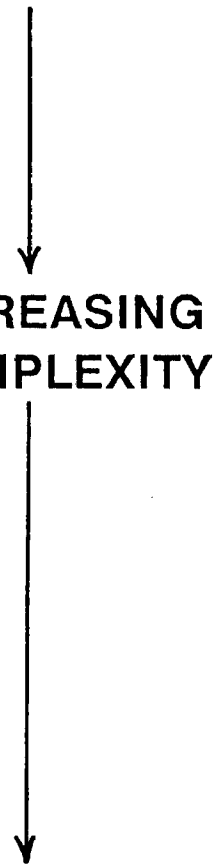
GRAVITY GRADIENT BOOMS

NUTATION DAMPERS

SPINNING PARTS OF MOMENTUM EXCHANGE DEVICES

SOLAR ARRAYS

ARTICULATED SUBSTRUCTURES



Aerospace interest in stabilization and control of s/c attitude, using gravity-gradient and momentum exchange devices fostered the development of equations of motion for multibody systems.

PRA
Photon Research Associates, Inc.

DYNAMIC INTERACTIONS

- Environmental - Thermal Solar Magnetic Gravity
  Plumes Atmosphere
- Active Control - Attitude (Thrusters, Wheels, Gas Jets)
  Structural Optical Maneuver
  Vibration Thermal Deployment
  Joint motion (e.g. manipulator arm, radiometer rotation)
- Rigid/Flex Coupling
- On-Board Disturbances - Fluid Slosh Spinning Devices
  Thermal Astronaut Motion
  Docking Cryogenic Coolers

Photon Research Associates, Inc.

DYNAMIC INTERACTIONS (CONT'D)

- Changing System Parameters -
  - Mass Flow
  - Deployment (Length and Elasticity Change)
  - Aging/Damage
  - Depletion of Consumables

PRA
Photon Research Associates, Inc.

RECENT ADVANCES

- ORDER (N) ALGORITHMS, REAL-TIME ALGORITHMS & OTHER ALGORITHMIC SPEED-UPS
- PARALLEL PROCESSING
- SYMBOLIC FORMULATION
- USER-FRIENDLY GRAPHICAL INTERFACES
- HIGH-LEVEL SIMULATION ENVIRONMENT
- JOINT MODELS - CLEARANCES, JOINT FLEXIBILITY, MULTIPLE CONTACT
- SIMPLE IMPACT MODELS
- SENSITIVITY ANALYSIS - RECURSIVE LINEARIZATION

Photon Research Associates, Inc.

RECENT ADVANCES (CONT'D)

- GEOMETRIC STIFFENING (FIRST ORDER CORRECTIONS)
- COMPONENT MODE SHAPES
- VERIFICATION

PRA
Photon Research Associates, Inc.

CURRENT CAPABILITIES

ANALYSIS	CAPABILITY	COMMENTS
NUMBER OF BODIES	MANY	• FAST ALGORITHMS • PARALLEL PROCESSING • SYMBOLIC CODE GENERATION
STRUCTURAL MODELS	FROM FINITE ELEMENT, ANALYTICAL, OR CAD	• SMALL ELASTIC DEFLECTIONS
SYSTEM PARAMETERS	CONSTANT	• NO TIME-VARYING MODE SHAPES AND FREQUENCIES • NO MASS AND INERTIA CHANGES DUE TO EXPENDABLE FUELS
DOCKING IMPACT	RIGID - LIMITED FLEXIBLE - NONE	• RESEARCH PROBLEM • IMPULSE FORMULATION
DEPLOYMENT	LIMITED	• SEQUENTIAL ALGORITHMS
CONTROL	GENERAL ALGORITHMS	• INTERFACES WITH CONTROL DESIGN/ANALYSIS SOFTWARE • GENERAL INPUTS BY USER
THERMAL	QUASI-STATIC	• NO RAPID-TRANSIENT HEATING • NEEDED FOR ENVIRONMENTAL, ON-BOARD, AND THREAT MODELING

Photon Research Associates, Inc.

CURRENT CAPABILITIES (CONT'D)

ENVIRONMENTAL DISTURBANCES	GRAVITY GRADIENT MAGNETIC FIELD SIMPLE ATMOSPHERE MODELS	• NO SOLAR RADIATION MODELS • NO PLUME IMPINGEMENT MODELS
DISTRIBUTED PARAMETER SENSING AND CONTROL	NONE	• RESEARCH TOPIC
NONLINEAR JOINT MODELS	LIMITED	• MUCH WORK DONE RECENTLY • JOINT-DOMINATED STRUCTURES DIFFICULT TO DEAL WITH
WAVE MOTION MODELS	NONE	• PROHIBITIVELY LARGE MODELS REQUIRED • PROBLEM MAY GO AWAY WITH ADVANCES IN ALGORITHMS AND COMPUTER HARDWARE • PDE MODELING BEST
PASSIVE DAMPING	NONE (IN-HOUSE PROPRIETARY CODES PERHAPS)	• FINITE ELEMENT MODELING FOR VISCOELASTIC MATERIAL IS A CURRENT RESEARCH TOPIC • PDE MODELS PROVIDE A MORE DIRECT MODELING • FREQUENCY, STRAIN AND TEMPERATURE DEPENDENCE DIFFICULT TO MODEL

Photon Research Associates, Inc.

CURRENT CAPABILITIES (CONT'D)

FLUID-MECHANICAL INTERACTION	SIMPLE PENDULUM MODELS	• FLUID SLOSH
SPECIAL APPLICATIONS	BIOMECHANICS RAIL VEHICLES	• SPECIALIZED USER-FRIENDLY INTERFACES • SPECIALIZED KINEMATICS
HIGH LOAD SYSTEMS	GEOMETRIC STIFFENING	• HIGH ANGULAR RATES • LARGE LOADS DUE TO INERTIAL, APPLIED, OR CONSTRAINT FORCES
DESIGN SENSITIVITY	HELP IN OPTIMIZING SYSTEM PARAMETERS	
LINEARIZATION	COMPUTED ABOUT SPECIFIED OPERATING POINT	• CONTROL DESIGN • SYSTEM EIGENVALUES • TRANSFER FUNCTIONS

UNRESOLVED ISSUES/FUTURE WORK

Photon Research Associates, Inc.

- EXPERIMENTAL VALIDATION AND CROSS VERIFICATION
- BETTER JOINT CHARACTERIZATION
- COMPUTATIONAL EFFICIENCY
- USER INTERFACE (GRAPHICS, VIRTUAL REALITY)
- INTERCONNECTIONS WITH OTHER CODES - FLUIDS, THERMAL
- DEPLOYMENT DYNAMICS WITH MASS FLOW, CHANGING MODE SHAPES AND FREQUENCIES
- IMPACT DYNAMICS, PLASTIC DEFORMATION AND BUCKLING MODELS

Photon Research Associates, Inc.

UNRESOLVED ISSUES/FUTURE WORK (CONT'D)

- SENSITIVITY ANALYSIS FOR OPTIMIZATION OF CONTROL, LINKAGE DIMENSIONS, MASS, INERTIA, ETC.
- SYSTEM IDENTIFICATION
- CONTROL OF FLEXIBLE MULTIBODY SYSTEMS
- KINEMATIC SYNTHESIS - DESIGN LINKAGES TO FOLLOW A CERTAIN TRAJECTORY