

# Parallel Multi-Step/Multi-Rate Integration of Two-Time Scale Dynamic Systems

Johnny T. Chang\*    Scott R. Ploen, Ph.D.<sup>†</sup>    Garrett A. Sohl, Ph.D.<sup>‡</sup>  
Bryan J. Martin, Ph.D.<sup>§</sup>

*Jet Propulsion Laboratory, Pasadena, CA*

Increasing demands on the fidelity of simulations for real-time and high-fidelity simulations are stressing the capacity of modern processors. New integration techniques are required that provide maximum efficiency for systems that are parallelizable. However many current techniques make assumptions that are at odds with non-cascadable systems. A new serial multi-step / multi-rate integration algorithm for dual-timescale continuous-state systems is presented which applies to these systems, and is extended to a parallel multi-step / multi-rate algorithm. The superior performance of both algorithms is demonstrated through a representative example.

## I. Introduction

As simulation has become an accepted tool for supporting development and validation of control systems, software, and devices, the demands placed on it have grown ever higher. High-fidelity and real-time simulation is becoming a requirement for performance validation, and the required fidelity of supported models is high. This has placed ever-increasing loads on simulation assets, and new approaches to increasing the aggregate computation capability of simulators is an active area of research. For continuous state-based simulations, there are three ways in which this can be done: increase the efficiency of state computations, parallelize the computations (either at the integrator or state computation level), and increase the efficiency of the integrator.

Most parallel/multirate integration techniques (*ref. list goes here*) make use of the assumption that a system is *cascaded* (see Figure 1). That is, they contain fast subsystems that can be reasonably assumed to not depend on the slower 'parent' systems, nor upon each other to first order. This assumption has proven to be quite valuable, in that it significantly simplifies the problem and produces useful solutions for those systems that conform. Examples of systems that can be classified as cascaded include those that are open loop in the fast subsystems, which is the case when these subsystems are treated as disturbances. Other examples are systems for which the fast subsystems are closed loop but independent of each other, and

\*Staff Engineer, Engineering and Science Directorate, Avionic Systems and Technology Division, Autonomy and Control Section, Simulation and Verification Group.

<sup>†</sup>Senior Staff Engineer, Engineering and Science Directorate, Avionic Systems and Technology Division, Autonomy and Control Section, Guidance and Control Analysis.

<sup>‡</sup>Staff Engineer, Engineering and Science Directorate, Avionic Systems and Technology Division, Autonomy and Control Section, Simulation and Verification Group.

<sup>§</sup>Senior Staff Engineer, Engineering and Science Directorate, Avionic Systems and Technology Division, Autonomy and Control Section, Simulation and Verification Group.

therefore their inputs to the slow subsystem can be treated as disturbances and have no significant effect on one another.

Figure 1. A typical cascaded system

If, however, the overall system contains either physical or control-based dependencies between the fast subsystems, the cascaded assumption is no longer valid. One example of this can be found in multi-spacecraft observations that require closed-loop control between spacecraft, typically found in formation-based interferometry missions. Figure 2 shows the block diagram of such a system. The dynamics of each spacecraft are clearly independent, even in the presence of closed-loop attitude control. At first glance this might indicate that the system state can be advanced independently for each spacecraft, exchanging states at occasional synchronization points. However, the fast subsystems work to combine light from a common observation onto a single sensor on one spacecraft. The control system that regulates the location of light on the final sensor ties the otherwise independent slow and fast dynamics of both systems tightly together, and normally forces computationally expensive small-timescale integration of the combined system dynamics for both spacecraft in order to yield accurate results.

Figure 2. A system which is not cascaded

A multi-step / multi-rate algorithm can greatly speed the simulation of such a system by separately computing the fast and slow systems close to their natural rates. This is efficient because slow subsystem states are often more expensive to compute on a per-step basis, due to increased complexity of the dynamics (flexible modes and other nonlinear effects are often not significant for the smaller, fast subsystems). Parallelizing the algorithm can produce significant additional gains.

If the time required to compute one step of the slow subsystem is  $T_S$ , and the time required to compute one step of the fast subsystem is  $T_F$ , then the total time required to advance the entire system using a serial algorithm can be written as:

$$T = N_S T_S + N_F T_F \quad (1)$$

where  $N_*$  is the corresponding number of state computations for the  $*$  subsystem. For a parallel algorithm, the total time required is instead

$$T = \max(N_S T_S, N_F T_F) + \delta \quad (2)$$

where  $\delta$  is the communication cost incurred by parallelizing the algorithm. Assuming  $\delta$  is small, and that the subsystems are well matched ( $N_S T_S \approx N_F T_F$ ), the parallel algorithm will be twice as efficient for a simple two-timescale system. In fact, for complex systems with more timescales or with more subsystems the speedup for a parallelized multi-rate algorithm can be much greater, limited only by the maximum computation cost of a single subsystem.

In the following sections we outline such an algorithm, which yields significant performance improvements while providing excellent results for non-cascaded systems.

## II. Previous Work

Mathematical models of a wide variety of physical, biological, and economic processes take the form of *initial value problems* (IVP's) consisting of a system of ordinary differential equations with the initial state (i.e., initial conditions) prescribed at a given initial time. As a result, numerical methods tailored for the efficient and accurate solution of IVP's have been the focus of extensive study. The classic text<sup>5</sup> provides a detailed treatment of both single-step and multi-step algorithms for IVP's. The more recent text<sup>17</sup> provides

valuable insights into the structure of IVP's and provides a comprehensive discussion of convergence, stability, and error estimation. The standard Adams family of predictor-corrector methods based on PECE (Predict-Correct-Estimate-Correct) are discussed in<sup>16</sup> and<sup>17</sup>. A thorough treatment of variable order and variable step-size integration algorithms is also provided. The survey article<sup>7</sup> and the introductory texts,<sup>8, 13</sup> and<sup>21</sup> also provide comprehensive treatments of solving IVP's numerically.

The traditional approach toward integrating coupled, high-order (i.e., systems with large state dimension), nonlinear IVP's has been to integrate the equations of motion of the system over a specified integration interval  $[t_o, t_f]$  using an integration step size based on the the highest frequency system mode of interest<sup>a</sup>. However, if the required time horizon  $t_f - t_o$  is large, and the system consists of many high frequency modes, numerical integration performed in this manner can be prohibitively slow and can diminish the utility of simulation in accessing the dynamic response of the system.

To address these issues, researchers have focused on developing *multi-rate*<sup>b</sup> algorithms to enable fast, accurate, numerical integration of nonlinear IVP's containing multiple time scales (e.g., mechanical systems containing both rigid and elastic modes,<sup>15</sup> molecular dynamics simulations,<sup>143</sup>). Specifically, multi-rate numerical integration algorithms are utilized to (1) decompose a given IVP into subsystems based on time-scale separation, and (2) automatically tune/adapt the integration step-size for each individual subsystem<sup>c</sup> to maintain solution accuracy. In this paper we will assume that the system of interest is a multibody mechanical systems (e.g., a spacecraft). As a result, the equations of motion admit a natural decomposition into elastic and rigid subsystems, and hence can be directly partitioned by the analyst into fast/slow subsystems.

We now provide a brief overview of previous work in the area of multi-rate integration algorithms. For a more comprehensive survey of the multi-rate numerical integration literature, the reader should consult<sup>4</sup> and<sup>18</sup>.

Multi-rate integration methods can be naturally divided into single-step (e.g., Runge-Kutta type) or multi-step (e.g., Adams type) algorithms. In an early paper Andrus<sup>1</sup> studied the integration of a system of ODE's partitioned into a slow and fast subsystem. Andrus assumed that a solution to the fast subsystem is available (e.g., closed-form solution) for use in integrating the slow states via a fourth order Runge-Kutta algorithm. In a later paper,<sup>2</sup> Andrus studies the stability of Runge-Kutta based multi-rate methods. In another early report,<sup>11</sup> Orailoglu developed the multi-rate software package MRATE for integrating IVP's by modifying the DIFSUB software (based on an Adams-type algorithms) developed by Gear. Wells and Gear<sup>22</sup> systematically studied the structure, stability, and convergence properties of multi-rate Adams predictor/corrector algorithms. The classification of multi-rate algorithms into *slowest first* vs. *fastest first* types was introduced in the doctoral thesis.<sup>22</sup> In the slowest first approach, the slow states are integrated first over a larger time-step, and then interpolated values of the slow components are used to integrate the fast states over a smaller time-step.

In,<sup>6</sup> Gear provides a general discussion on multi-rate integration algorithms and concludes that multi-rate, multi-step integrators provide the most efficient algorithms for many practical situations. Palusinski<sup>12</sup> developed an explicit multi-rate Runge-Kutta algorithm for a system partitioned into two time scales. His algorithm, based on a slowest-first approach, is used to investigate the dynamic response of an autopilot design.

The application of multi-rate numerical integration methods to the simulation of complex mechanical and multibody systems has been an area of increasing activity. As discussed above, most dynamic models of mechanical systems admit a natural decomposition into slow (rigid) and fast (elastic) dynamic subsystems<sup>d</sup>.

<sup>a</sup>Typically, ten samples per period of the highest frequency mode is sufficient.

<sup>b</sup>The term *multi-rate* should not be confused with the term *multi-step*. Multi-rate algorithms are designed to numerically integrate IVP's containing both fast and slow time scales, while multi-step algorithm refer to any algorithm (multi-rate or not) that use estimates of the solution (and its derivative) at several previous time-steps to construct an approximate solution at the current time-step.

<sup>c</sup>Here small integration time-steps are used to integrate fast dynamic modes, while larger time-steps are used to integrate slower dynamic modes.

<sup>d</sup>Although in many applications, the rigid modes influence the elastic modes and not vice-versa, flexible mechanical systems that spin at high rate are an exception due to spin-stiffening effects and should be treated with care.

Controller/sensor dynamics can also introduce wide time-scale separations in the simulation of closed-loop system behavior.

In,<sup>19</sup> multi-rate integration techniques for large order *linear* models with a wide dynamic bandwidth (defined as the ratio between the real parts of the largest and smallest system eigenvalues) are developed directly by partitioning the system state transition matrix. Latiffe<sup>10</sup> developed frequency domain techniques for interfacing fast/slow subsystems with applications to closed-loop control.

In the area of multibody dynamics, Buzdugan<sup>4</sup> and Kim<sup>9</sup> developed multi-rate integration algorithms based on the Nordsieck form of the Adams-Bashforth predictor/corrector.<sup>5</sup> Srinivasan,<sup>20</sup> developed a suite of multi-step, multi-rate integration algorithms for simulating flexible multibody systems. Solis<sup>18</sup> develops multi-rate integration techniques for DAE's (differential algebraic equations) describing multibody systems with dependent coordinates and/or closed-loops.

### III. Serial Multi-Step Integration Algorithm

In this section we review the basic Adams predictor/corrector method for numerical integration of the IVP

$$\frac{dx}{dt} = \dot{x} = f(t, x), \quad x(t_0) = x_0 \quad (3)$$

where  $x \in \mathbb{R}^{n \times 1}$ , and  $f : \mathbb{R} \times \mathbb{R}^{n \times 1} \mapsto \mathbb{R}^{n \times 1}$ .

The predictor part of the algorithm is formed by the Adams-Bashforth explicit multistep formula

$$x(t_m) = x(t_{m-1}) + ha^T \dot{X} \quad (4)$$

where  $x(t_m) \in \mathbb{R}^{n \times 1}$  denotes the approximate solution of the IVP at  $t_m = t_0 + mh$  ( $m = 1, 2, \dots, N$ ),  $h \in \mathbb{R}$  denotes the integration step-size,  $a = [0 \ a_1 I \ a_2 I \ \dots \ a_k I]^T \in \mathbb{R}^{n(k+1) \times n}$  are the coefficients of the Adams-Bashforth algorithm where  $I$  and  $0$  denote the  $n \times n$  identity and zero matrices respectively, and

$$\dot{X} = [\dot{x}(t_m) \ \dot{x}(t_{m-1}) \ \dot{x}(t_{m-2}) \ \dots \ \dot{x}(t_{m-k})]^T \in \mathbb{R}^{n(k+1) \times 1}$$

The coefficients  $a_i$  of various order Adams-Bashforth formulas are tabularized in.<sup>21</sup>

The corrector part of the algorithm is formed by the Adams-Moulton implicit multistep formula

$$x(t_m) = x(t_{m-1}) + hb^T \dot{X} \quad (5)$$

where  $b = [I \ b_1 I \ b_2 I \ \dots \ b_k I]^T \in \mathbb{R}^{n(k+1) \times 1}$  are the coefficients of the Adams-Moulton algorithm and  $I$  denotes the  $n \times n$  identity matrix. Note that  $x_m$  appears implicitly in equation (5) since the term  $b^T \dot{X}$  contains  $\dot{x}_m$  and  $\dot{x}_m = f(t_m, x_m)$ . The coefficients  $b_i$  of various order Adams-Moulton formulas are tabularized in.<sup>21</sup> Note that both the Adams-Bashforth and Adams-Moulton formulas are multi-step algorithms in that values of the solution and its derivative from multiple previous time-steps are used to construct the solution at the current time-step.

The idea behind the predictor/corrector algorithm is to use the explicit predictor formula (4) to provide an initial guess for developing an iterative solution of the implicit corrector formula (5). The iteration is continued until two successive outputs of the corrector agree to a specified tolerance. The stages in the predictor/corrector algorithm are commonly called PECE (Predict-Correct-Evaluate-Correct); See<sup>16</sup> or<sup>21</sup> for further information.

### IV. Serial Multi-Step/Multi-Rate Integration Algorithm

A dynamic system such as the planar spacecraft model described in the previous section is characterized by an initial value problem of the form given in equation (3). Here we assume the class of systems under study consists of multibody systems with both high frequency (e.g., elastic modes) and low frequency (e.g.,

rigid body modes) components. As a result, the state vector in equation (3) is partitioned into two parts; one consisting of slowly changing variables  $x_r$ , and the other consisting of rapidly changing variables  $x_e$ ; i.e.,

$$x = \begin{bmatrix} x_r \\ x_e \end{bmatrix} \quad (6)$$

We assume that the partitioning of the system is based on *a priori* knowledge of the system dynamics; e.g., the system is a spacecraft consisting of a rigid central body with flexible appendages. Thus, the IVP (3) can be expressed in terms of slow and fast subsystems as follows:

$$\dot{x}_r = f_r(t, x_r, x_e), \quad x_r(t_0) = x_{r0} \quad (7)$$

$$\dot{x}_e = f_e(t, x_r, x_e), \quad x_e(t_0) = x_{e0} \quad (8)$$

Here the subscript  $r$  denotes the rigid body modes (slow) and the subscript  $e$  denotes the elastic modes (fast). To integrate the slow and fast subsystems using different integration step-sizes, we have modified the classical predictor-corrector method discussed in a previous section. Specifically, fourth-order Adams-Bashforth and Adams-Moulton formulas were utilized.<sup>21</sup> As discussed earlier, multi-rate integration algorithms can be classified into several types depending on the order in which the integration is performed. The integration algorithm discussed here is based on a variation of the fastest first approach.<sup>20</sup> Coupling of the slow and the fast systems is done by interpolation. For each subsystem, the integration procedure based on Adams methods consists of prediction, evaluation, correction, and evaluation (PECE) stages. For a given slow integration step size  $H$  and a fast integration step size  $h$ , we first predict the value  $x_r(t)$  at  $t = t_k + H$ . We then iterate through a number of smaller steps  $h$  on the fast system where  $H = rh$ . The same PECE stages are repeated for each time-step of the fast system. The values of the slow system  $x_r(t)$  are determined at intermediate times  $t$  where  $t_k < t < t_k + H$  by interpolation and used in the correction stage for the fast system. The output from the fast system after  $r$  steps where  $H = rh$  is then utilized in the correction stage of the slow system. The complete multi-step, multi-rate integration algorithm based on the fourth-order Adams methods is presented below:

- **Step 1:** Adam-Bashforth (predictor) formula for the slow system.

This step produces an approximation of  $x_r(t)$  at  $t = t_k + H$ . The value of this approximation will be used by the fast system.

$$x_r^{PRED}(t_k + H) = x_r(t_k) + \frac{H}{24} \left[ 55\dot{x}_r(t_k) - 59\dot{x}_r(t_k - H) + 37\dot{x}_r(t_k - 2H) - 9\dot{x}_r(t_k - 3H) \right] \quad (9)$$

- **Step 2:** Adam-Bashforth (predictor) formula for the fast system.

The next few steps are the standard PECE stages of the fourth-order Adams-Bashforth and Adams-Moulton formulas applied to the fast system. The value of the slow state is coupled by interpolation in Step 3 below.

$$x_e^{PRED}(t_k + h) = x_e(t_k) + \frac{h}{24} \left[ 55\dot{x}_e(t_k) - 59\dot{x}_e(t_k - h) + 37\dot{x}_e(t_k - 2h) - 9\dot{x}_e(t_k - 3h) \right] \quad (10)$$

- **Step 3:** Interpolate the slow system to obtain  $x_r^{INT}(t_k + h)$ .

There are many ways to interpolate the the slow system. For example, linear interpolation or more complicated methods based on curve fitting techniques can be used.

- **Step 4: Evaluate.**

$$\dot{x}_e^{EVAL}(t_k + h) = f_e(t_k + h, x_r^{INT}(t_k + h), x_e^{PRED}(t_k + h)) \quad (11)$$

- **Step 5: Adams-Moulton (corrector) formula for the fast system.**

$$x_e^{COR(1)}(t_k + h) = x_e(t_k) + \frac{h}{24} \left[ 9\dot{x}_e^{EVAL}(t_k + h) + 19\dot{x}_e(t_k) - 5\dot{x}_e(t_k - h) + \dot{x}_e(t_k - 2h) \right] \quad (12)$$

- **Step 6: Evaluate.**

$$\dot{x}_e^{EVAL(1)}(t_k + h) = f_e(t_k + h, x_r^{INT}(t_k + h), x_e^{COR(1)}(t_k + h)) \quad (13)$$

- **Step 6a: Iterate the Corrector.**

Set  $x_e(t_k + h) = x_e^{COR(1)}(t_k + h)$ ,  $\dot{x}_e(t_k + h) = \dot{x}_e^{EVAL(1)}(t_k + h)$  and goto Step 7. Alternatively, perform another iteration to produce  $x_e^{COR(2)}$  based on  $x_e^{EVAL(1)}$ , and  $\dot{x}_e^{EVAL(2)}$  based on  $x_e^{COR(2)}$ , and then set  $x_e(t_k + h) = x_e^{COR(2)}(t_k + h)$ ,  $\dot{x}_e(t_k + h) = \dot{x}_e^{EVAL(2)}(t_k + h)$ , and then goto Step 7.

- **Step 7: Increment the fast states.**

Repeat Steps 2  $\rightarrow$  6 for  $t_k + 2h, t_k + 3h, \dots, t_k + rh$ , where  $rh = H$ .

- **Step 8: Adams-Bashforth evaluation step for the slow system.**

$$\dot{x}_r^{EVAL}(t_k + H) = f_r(t_k + H, x_r^{PRED}(t_k + H), x_e(t_k + rh)) \quad (14)$$

- **Step 9: Adams-Moulton (corrector) formula for the slow system.**

$$x_r^{COR}(t_k + H) = x_r(t_k) + \frac{H}{24} \left[ 9\dot{x}_r^{EVAL}(t_k + H) + 19\dot{x}_r(t_k) - 5\dot{x}_r(t_k - H) + \dot{x}_r(t_k - 2H) \right] \quad (15)$$

- **Step 10: Evaluate.**

$$\dot{x}_r(t_k + H) = f_r(t_k + H, x_r^{COR}(t_k + H), x_e(t_k + rh)) \quad (16)$$

Set  $\dot{x}_r(t_k + H) = x_r^{COR}(t_k + H)$ .

- **Step 11: Increment slow states and repeat.**

Repeat step 1 for  $t_k + 2H, t_k + 3H, \dots, t_k + NH$ .

Note that the multi-step Adams-Bashforth predictor for the slow and the fast systems in Steps 1 and 2 requires data from multiple previous time-steps. A one-step method, such as a fourth order Runge-Kutta algorithm, is utilized to compute the initial data required by the multi-step algorithm. Here the initial data was generated by using the step-size of the fast system  $h$ .

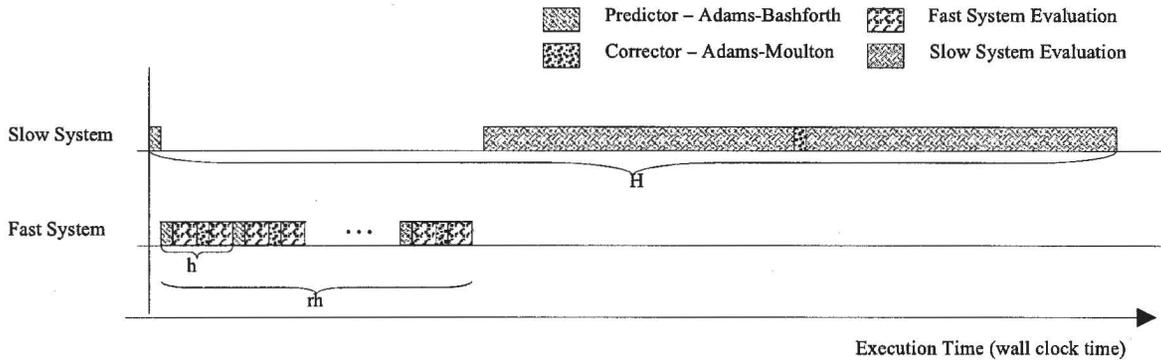


Figure 3. Execution order of the serial multi-step/multi-rate integration algorithm

## V. Parallel Multi-Step/Multi-Rate Integration Algorithm

Although we can observe significant benefit of partitioning the system into slow and fast components, we are limited to serial computations nevertheless. During the PECE stages, the slow system waits for the fast system to finish before coupling, and the fast system waits for the slow system to finish before moving onto the next integration step. The summary of this execution order is illustrated in Figure 3. The timing diagram shows the execution order of different PECE stages for one slow integration step  $H$ . The horizontal axis represents the execution time. The top line shows the PECE stages of the slow system, and the bottom line shows the PECE stages of the fast system. The system evaluation stages typically involve complicated derivative evaluations; therefore, they take significantly longer time to execute than the predictor and the corrector stages, which consist of only simple algebraic calculations. Note that there is a gap after the predictor stage of the slow system. No slow system computation is done in that gap since the fast system has to be integrated before moving onto the slow system. The situation is similar for the gap in the fast system case.

We realized a potential advantage for computing the slow and the fast system in parallel. Recent advances in multi-processor technology have made symmetric multi-processor systems widely available. We could easily compute the slow system and the fast system in parallel; one processor computes the slow system and the other computes the fast system. However, the execution order of the PECE stages have to be preserved nevertheless. In the parallel case, the gaps in the timing diagram of figure 3 would result in idle processing time. After observing the timing diagram more closely, we realize that the idle processing time can be utilized to perform additional computation, resulting in a more accurate integration that would not be possible with the serial version for the same given amount of time. The modified execution order diagram is shown in figure 4. The idle processing time in the slow system track can be used to perform one additional slow system evaluation. The outcome of this evaluation becomes the new predictor value for the later stages of the integration. The original Adam-Bashforth predictor is now only used by the first round of the fast system PECE stages. Since this additional evaluation is performed in parallel while the fast system value at integration step  $t = t_k + rh$  is being computed, the fast system value is not available at that time. In order to evaluate the slow system at  $t = t_k + H$ , we assume that the slow value at  $t = t_k + H$  stays constant, namely we use the slow value at  $t = t_k$  to evaluate  $\dot{x}_r(t_k + H)$ . We apply the Adams-Moulton corrector formula using this value to get  $x_r^{PRED'}(t_k + H)$ . We called this the preferred predictor value and will be used in the subsequent stages, since it involves derivative computation and is deemed to be a better prediction than the algebraic prediction of the Adam-Bashforth predictor formula.

In order to fill in the idle processing time of the fast system track, we compute another round of PECE stages for the fast system using the better predictor value  $x_r^{PRED'}(t_k + H)$ . The outcome of this computation

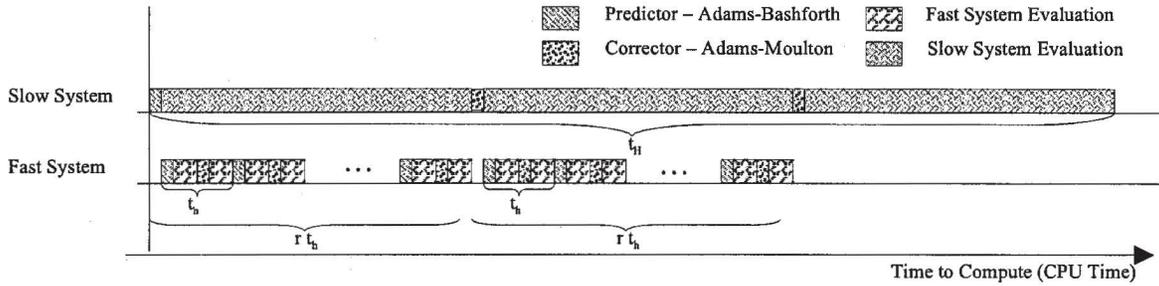


Figure 4. Execution order of the parallel multi-step/multi-rate integration algorithm

produces better result for the fast system, since it uses the improved predictor value  $x_r^{PRED'}(t_k + H)$ . The result is fed back into the slow system at the next integration step.

To summarize our new parallel multi-step/multi-rate integration algorithm, we outline the steps as follows:

For the slow system track:

- Step 1: Adam-Bashforth (predictor) formula for the slow system. This step makes a quick approximation of the values for the next step. The values from this approximation will be used by the fast system.

$$x_r^{PRED}(t_k + H) = x_r(t_k) + \frac{H}{24} \left[ 55\dot{x}_r(t_k) - 59\dot{x}_r(t_k - H) + 37\dot{x}_r(t_k - 2H) - 9\dot{x}_r(t_k - 3H) \right] \quad (17)$$

- Step 2: Evaluate using  $x_e(t_k)$

$$\dot{x}_r^{EVAL'}(t_k + H) = f_r(t_k + H, x_r^{PRED}(t_k + H), x_e(t_k)) \quad (18)$$

- Step 3: Adam-Moulton (corrector) formula for the slow system

$$x_r^{PRED'}(t_k + H) = x_r(t_k) + \frac{H}{24} \left[ 9\dot{x}_r^{EVAL'}(t_k + H) + 19\dot{x}_r(t_k) - 5\dot{x}_r(t_k - H) + \dot{x}_r(t_k - 2H) \right] \quad (19)$$

This becomes the new predictor for the later stages.

- Step 4: Sync up with the fast system track to get  $x_e(t_k + rh)$ , and evaluate

$$\dot{x}_r^{EVAL}(t_k + H) = f_r(t_k + H, x_r^{PRED'}(t_k + H), x_e(t_k + rh)) \quad (20)$$

- Step 5: Adam-Moulton (corrector) formula for the slow system

$$x_r^{COR}(t_k + H) = x_r(t_k) + \frac{H}{24} \left[ 9\dot{x}_r^{EVAL}(t_k + H) + 19\dot{x}_r(t_k) - 5\dot{x}_r(t_k - H) + \dot{x}_r(t_k - 2H) \right] \quad (21)$$

- Step 6: Evaluate

$$\dot{x}_r(t_k + H) = f_r(t_k + H, x_r^{COR}(t_k + H), x_e(t_k + rh)) \quad (22)$$

Set  $\dot{x}_r(t_k + H) = x_r^{COR}(t_k + H)$ .

Repeat step 1 for  $t_k + 2H, t_k + 3H, \dots, t_k + NH$ .

For the fast system track:

- Step 1: Adam-Bashforth (predictor) formula for the fast system. The next few steps are the standard PECE stages of the fourth order Adams-Bashforth and Adams-Moulton formulas applied to the fast systems. The slow system values are coupled by interpolation in step 2.

$$x_e^{PRED}(t_k + h) = x_e(t_k) + \frac{h}{24} \left[ 55\dot{x}_e(t_k) - 59\dot{x}_e(t_k - h) + 37\dot{x}_e(t_k - 2h) - 9\dot{x}_e(t_k - 3h) \right] \quad (23)$$

- Step 2: Interpolate the slow system  $x_r^{PRED}(t_k + H)$  to obtain  $x_r^{INT}(t_k + h)$ . There are many ways to interpolate the in between values of the slow system. Intuitively, one can linearly interpolate the slow state. More complicated methods can be performed by using curve fitting techniques.

- Step 3: Evaluate

$$\dot{x}_e^{EVAL}(t_k + h) = f_e(t_k + h, x_r^{INT}(t_k + h), x_e^{PRED}(t_k + h)) \quad (24)$$

- Step 4: Adam-Moulton (corrector) formula for the fast system

$$x_e^{COR(1)}(t_k + h) = x_e(t_k) + \frac{h}{24} \left[ 9\dot{x}_e^{EVAL}(t_k + h) + 19\dot{x}_e(t_k) - 5\dot{x}_e(t_k - h) + \dot{x}_e(t_k - 2h) \right] \quad (25)$$

- Step 5: Evaluate

$$\dot{x}_e^{EVAL(1)}(t_k + h) = f_e(t_k + h, x_r^{INT}(t_k + h), x_e^{COR(1)}(t_k + h)) \quad (26)$$

- Step 5a: Set  $x_e(t_k + h) = x_e^{COR(1)}(t_k + h)$ ,  $\dot{x}_e(t_k + h) = \dot{x}_e^{EVAL(1)}(t_k + h)$ , then move on to step 6.

- Step 5b: or perform another iteration to produce

$x_e^{COR(2)}$  based on  $x_e^{EVAL(1)}$ ,

$\dot{x}_e^{EVAL(2)}$  based on  $x_e^{COR(2)}$ ,

then set  $x_e(t_k + h) = x_e^{COR(2)}(t_k + h)$ ,  $\dot{x}_e(t_k + h) = \dot{x}_e^{EVAL(2)}(t_k + h)$ , then move on to step 6.

- Step 6: Repeat step 1 → 6a or step 6b for  $t_k + 2h, t_k + 3h, \dots, t_k + rh$ , where  $rh = H$ .

The second around of PECE stages will utilize the improved predictor value  $x_r^{PRED'}(t_k + H)$ .

- Step 7: Sync up with the slow system track to get  $x_r^{PRED'}(t_k + H)$ .

- Step 8: Adam-Bashforth (predictor) formula for the fast system. The next few steps are the standard PECE stages of the fourth order Adams-Bashforth and Adams-Moulton formulas applied to the fast systems. The slow system values are coupled by interpolation in step 9.

$$x_e^{PRED}(t_k + h) = x_e(t_k) + \frac{h}{24} \left[ 55\dot{x}_e(t_k) - 59\dot{x}_e(t_k - h) + 37\dot{x}_e(t_k - 2h) - 9\dot{x}_e(t_k - 3h) \right] \quad (27)$$

- Step 9: Interpolate the slow system  $x_r^{PRED'}(t_k + H)$  to obtain  $x_r^{INT}(t_k + h)$ . There are many ways to interpolate the in between values of the slow system. Intuitively, one can linearly interpolate the slow state. More complicated methods can be performed by using curve fitting techniques.
- Step 10: Evaluate

$$\dot{x}_e^{EVAL}(t_k + h) = f_e\left(t_k + h, x_r^{INT}(t_k + h), x_e^{PRED}(t_k + h)\right) \quad (28)$$

q

- Step 11: Adam-Moulton (corrector) formula for the fast system

$$x_e^{COR(1)}(t_k + h) = x_e(t_k) + \frac{h}{24} \left[ 9\dot{x}_e^{EVAL}(t_k + h) + 19\dot{x}_e(t_k) - 5\dot{x}_e(t_k - h) + \dot{x}_e(t_k - 2h) \right] \quad (29)$$

- Step 12: Evaluate

$$\dot{x}_e^{EVAL(1)}(t_k + h) = f_e\left(t_k + h, x_r^{INT}(t_k + h), x_e^{COR(1)}(t_k + h)\right) \quad (30)$$

- Step 12a: Set  $x_e(t_k + h) = x_e^{COR(1)}(t_k + h)$ ,  $\dot{x}_e(t_k + h) = \dot{x}_e^{EVAL(1)}(t_k + h)$ , then move on to step 6.
- Step 12b: or perform another iteration to produce  $x_e^{COR(2)}$  based on  $x_e^{EVAL(1)}$ ,  $\dot{x}_e^{EVAL(2)}$  based on  $x_e^{COR(2)}$ , then set  $x_e(t_k + h) = x_e^{COR(2)}(t_k + h)$ ,  $\dot{x}_e(t_k + h) = \dot{x}_e^{EVAL(2)}(t_k + h)$ , then move on to step 6.
- Step 13: Repeat step 7  $\rightarrow$  12a or step 12b for  $t_k + 2h, t_k + 3h, \dots, t_k + rh$ , where  $rh = H$ .

## VI. Implementation

In order to study the behavior of our numerical multi-rate integration algorithms, we devise a sample dynamic system with high and low frequency components. Our sample model is a 5 DOF planar spacecraft with two solar panels attached to the spacecraft body using 1 DOF joints. The spacecraft is limited to move in the planar x and y directions and only allowed to rotate on the xy plane. See figure 5. We can, therefore, characterize the system with 5 variables:

$$(x, y, \theta_B, \theta_R, \theta_L) \quad (31)$$

$\theta_R$  is the angle of the right solar panel with respect to its resting position, and  $\theta_L$  is the angle for the left solar panel.  $\theta_B$  is the rotation of the spacecraft. The solar panel joints have angular springs exerting torque proportional to  $\theta_R$  and  $\theta_L$  respectively. The equations of motion can be derived using the Lagrangian method. (Should I elaborate more here?)

The dynamics of the spacecraft is implemented using HYDRA, an architecture for building heterogeneous, distributed real-time simulations.<sup>23</sup> We assume a prior knowledge of the system and break the system into slow and fast subsystems. The slow system consists of the spacecraft body, and the fast system consists of the two solar panels. For the parallel case, HYDRA distributes the computation of the slow and the fast system dynamics onto two processors. HYDRA provides the communication mechanisms to synchronize the slow and the fast system.

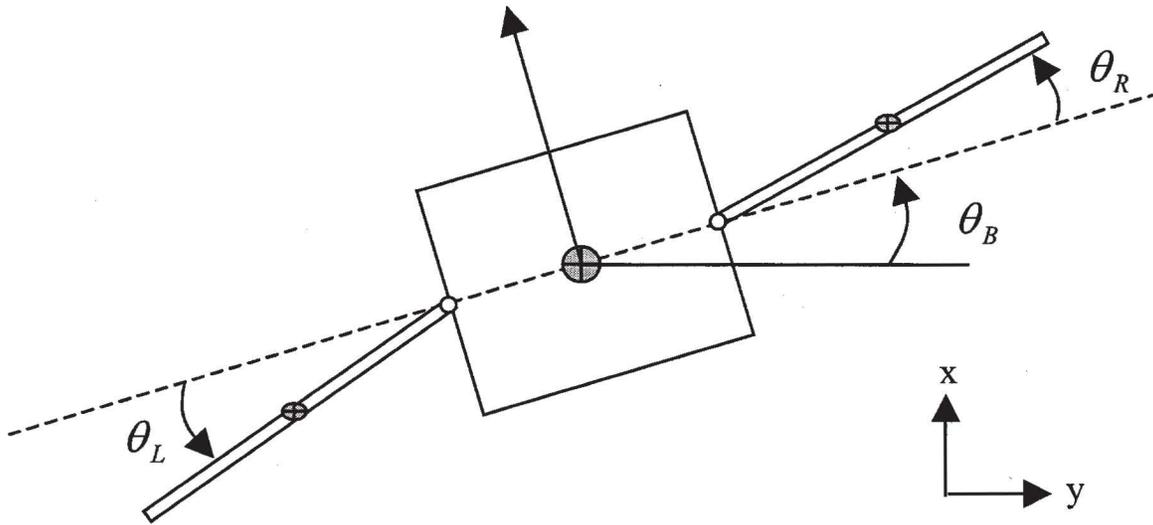


Figure 5. Planar Spacecraft Model

## VII. Results

We set the integration step size for the slow system to  $H = 0.01$ , and the fast system integration step size to  $h = 0.001$ . The right solar panel has an initial tilt degree of  $0.05$  Rad. The initial values for  $x, y, \theta_B$ , and  $\theta_L$  are all zero. We use the serial multi-step Adams predictor/corrector integration method as the basis of our comparisons. The single rate integration step size is set to  $0.001$ . This is used as the reference to evaluate against the accuracy of our serial and parallel multi-rate/multi-step integration algorithms. Note that our serial or parallel multi-rate/multi-step integration algorithm is iterative; at the end of each integration step, the integration process can be iterated again to improve the accuracy. For the purpose of our comparison, we only choose to perform one iteration for each integration step. This allows us to see the relative performance of our serial and parallel multi-rate/multi-step integration method with respect to the single-rate method. In figure 6, we can clearly see the benefit of parallel multi-rate/multi-step integration method over the serial multi-rate/multi-step integration method. With only one iteration for each integration step, the result from the serial multi-rate/multi-step method gradually diverges with respect to the reference. In contrast, the parallel multi-rate/multi-step method tracks the reference fairly well. Although the accuracy of the serial multi-rate/multi-step integration method can be improved by increasing the number of iterations, the timing requirements for real-time simulation might limit the number of iterations can be performed. In that case, the parallel method produces more accurate result in the same number of iterations. The plots for  $y, \theta_B, \theta_R$ , and  $\theta_L$  are shown in figure 7, 8, 9, and 10.

## VIII. Conclusion

We have extended the Adams predictor and corrector integration algorithm to incorporate multi-rate integration. We assume that the partitioning of the slow and the fast components can be performed based on a prior knowledge of the system. We extended the concept further and developed a novel parallel integration technique that utilizes two processors in a shared memory parallel computer. We conducted benchmarks on our sample spacecraft model under study and saw improvements in performance of our integration algorithms.

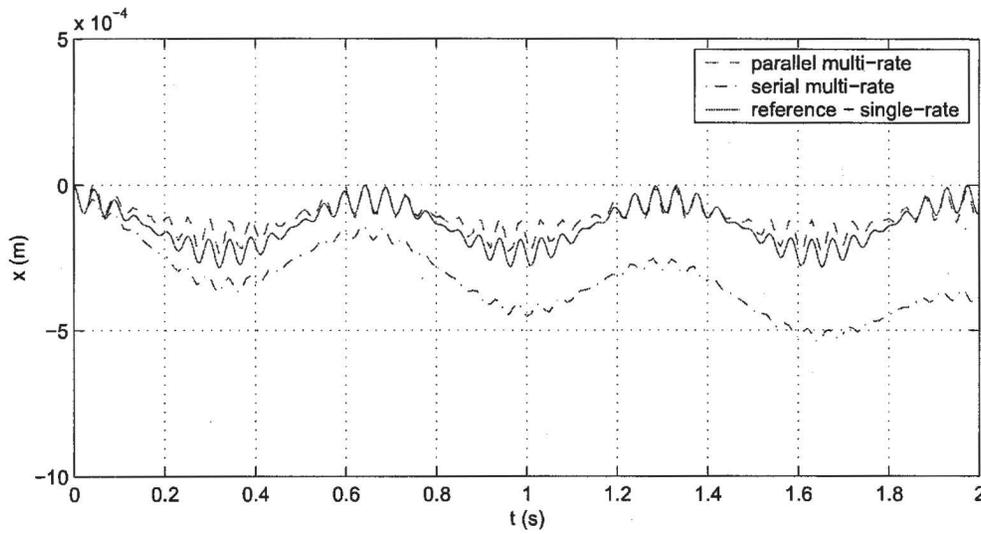


Figure 6. A comparison of the x position of the planar spacecraft with respect to time using different integration schemes.

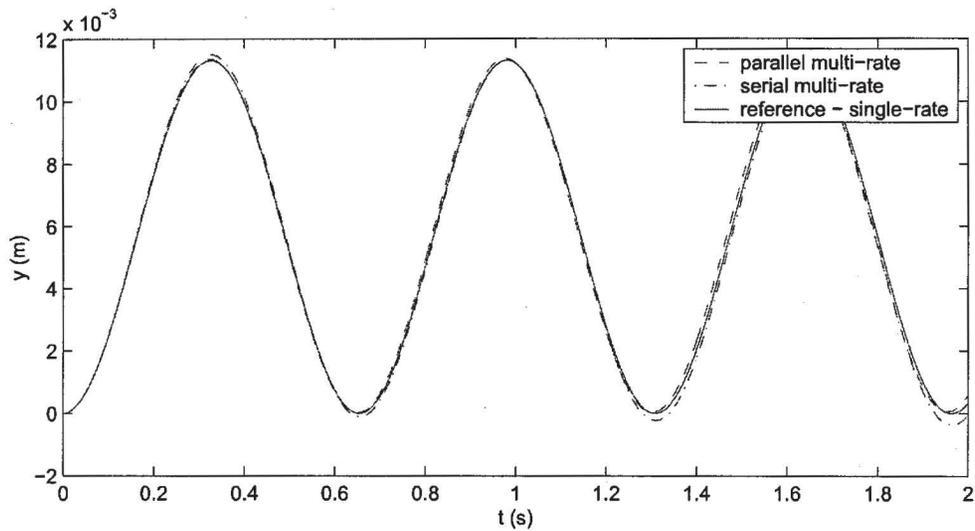


Figure 7. A comparison of the y position of the planar spacecraft with respect to time using different integration schemes.

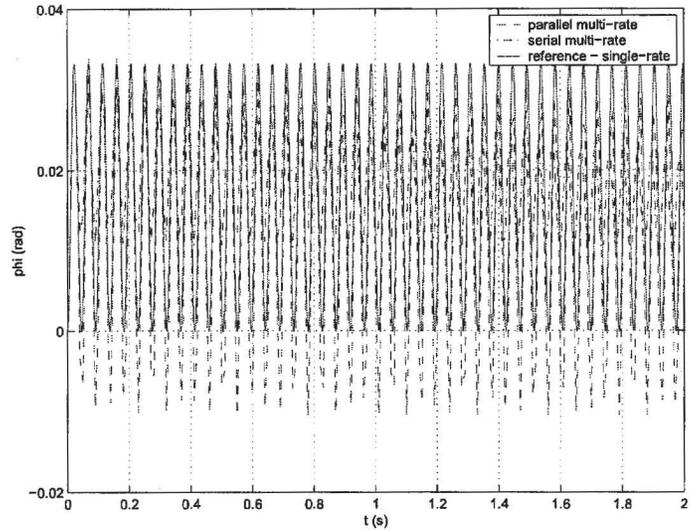


Figure 8. A comparison of  $\theta_B$  of the planar spacecraft with respect to time using different integration schemes.

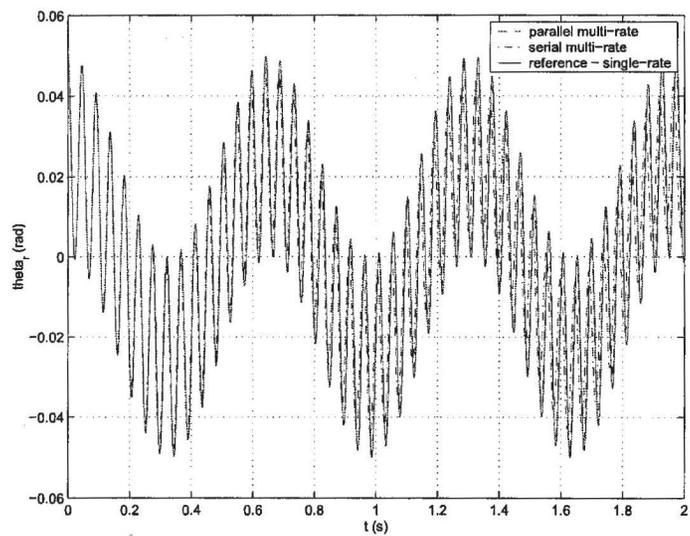


Figure 9. A comparison of  $\theta_R$  of the planar spacecraft with respect to time using different integration schemes.

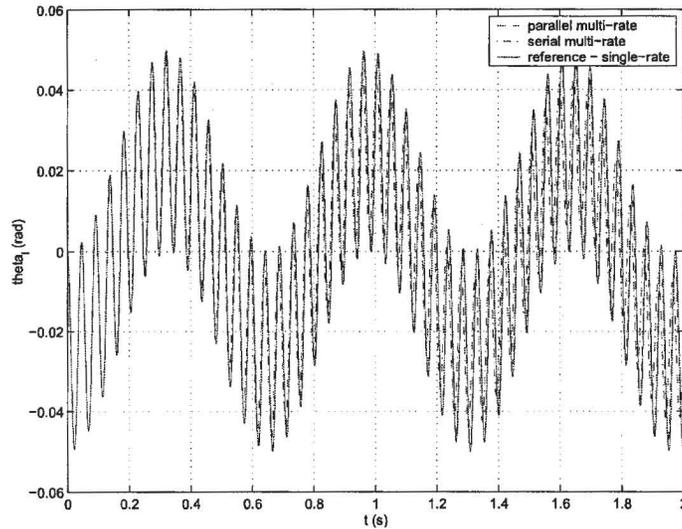


Figure 10. A comparison of  $\theta_L$  of the planar spacecraft with respect to time using different integration schemes.

## References

- <sup>1</sup>J.F. Andrus, "Numerical solution of systems of ordinary differential equations separated into subsystems", *SIAM J. Numerical Analysis*. Vol. 16, No. 4, pp. 605-611, 1979.
- <sup>2</sup>J.F. Andrus, "Stability of a multi-rate method for numerical integration of ODE's", *Computers Math. Applic.* Vol. 25, No. 2, pp. 3-14, 1993.
- <sup>3</sup>J.J. Biesiadeck, and R.D. Skeel, "Dangers of multiple time step methods", *J. Computational Physics*. Vol. 109, pp. 318-328, 1993.
- <sup>4</sup>L.I. Buzdugan, *Multirate Integration Algorithms for Real-Time Simulation of Mechanical Systems with Interacting Subsystems*. Ph.D Thesis, University of Iowa, 1999.
- <sup>5</sup>C.W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, Englewood Cliffs, 1971.
- <sup>6</sup>C.W. Gear, "The numerical solution of problems which may have high frequency components", *NATO-SAE Series, Computer Aided Analysis and Optimization of Mechanical System Dynamics*. E.J. Haug (Editor) Springer-Verlag, New York, 1984.
- <sup>7</sup>G.K. Gupta, R. Sacks-Davis, and P.E. Tischer, "A review of recent developments in solving ODE's", *Computing Surveys*. Vol. 17, No. 1, pp. 5-47, 1985.
- <sup>8</sup>R.W. Hamming, *Introduction to Applied Numerical Analysis*. McGraw-Hill, New York, 1971.
- <sup>9</sup>S.S. Kim, and J.S. Freeman, "Multirate integration for multibody dynamic analysis with decomposed subsystems", *Proc. 1999 ASME Design Engineering Technical Conferences*. Las Vegas, Nevada, 1999.
- <sup>10</sup>J. Laffitte, and R.M. Howe, "Interfacing fast and slow subsystems in the real time simulation of dynamic systems," *Transactions of the Society for Computer Simulation*. Vol. 14, No. 3, pp. 115-126, 1997.
- <sup>11</sup>A. Orailoglu, *A Multirate Ordinary Differential Equation Integrator*. Report UIUCDCS-R-79-959, Department of Computer Science, University of Illinois at Urbana-Champaign, 1979.
- <sup>12</sup>O.A. Palusinski, "Simulation of dynamic systems using multirate integration techniques," *Transactions of the Society for Computer Simulation*. Vol. 2, No. 4, pp. 257-273, 1986.
- <sup>13</sup>A. Ralston, and P. Rabinowitz, *A First Course in Numerical Analysis, Second Edition*. Dover (Reprint), New York, 2001.
- <sup>14</sup>S. Reich, "Multiple time scales in classical and quantum-classical molecular dynamics," *J. Computational Physics*. Vol. 151, pp. 49-73, 1999.
- <sup>15</sup>A.A. Shabana, *Dynamics of Multibody Systems. 2nd Edition*. Cambridge University Press, Cambridge, 1998.
- <sup>16</sup>L.F. Shampine, and M.K. Gordon, *Computer Solution of Ordinary Differential Equations*. W.H. Freeman, San Francisco, 1975.
- <sup>17</sup>L.F. Shampine, *Numerical Solution of Ordinary Differential Equations*. Chapman and Hall, New York, 1995.

<sup>18</sup>D. Solis, *Multirate Integration Methods for Constrained Mechanical Systems with Interacting Subsystems*. Ph.D Thesis, University of Iowa, 1996.

<sup>19</sup>K.C. Cou, and O.L. de Weck, "Fast time domain simulation for large order linear time-invariant systems," *44th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Conference, AIAA-2003-1532*. Norfolk, Virginia, 2003.

<sup>20</sup>M. Srinivasan, *Multirate Numerical Integration in Design and Analysis of Flexible Multibody Systems*. Ph.D Thesis, University of Iowa, 1984.

<sup>21</sup>A.H. Stroud, *Numerical Quadrature and Solution of Ordinary Equations: A Textbook for a Beginning Course in Numerical Analysis*. Springer-Verlag, New York, 1974.

<sup>22</sup>D.R. Wells, *Multirate Linear Multistep Methods for the Solution of Systems of Ordinary Differential Equations*. Ph.D Thesis, University of Illinois at Urbana-Champaign, 1982.

<sup>23</sup>B. Martin and G. Sohl "Hydra: High-Speed Simulation Architecture for Precision Spacecraft Formation Simulation," *AIAA Modeling and Simulation Technologies Conference and Exhibit, AIAA-2003-5380*. August 11-14, Austin, Texas.

End of File

