Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea · May 21-26, 2001

Efficient dynamic simulation of robotic systems with hierarchy

Joel M. Esposito Vijay Kumar

Mechanical Engineering and Applied Mechanics
University of Pennsylvania
jme,kumar@grip.cis.upenn.edu

Abstract

In this paper multirate numerical integration techniques are introduced as a tool for simulating robotic systems. In contrast with traditional simulation techniques where a single global time step is used, multirate methods seek a gain in efficiency by using larger step sizes for the slow varying components and smaller step sizes for components with rapidly changing solutions. We argue that many robotic systems inherently posses different time scales, and therefore can benefit from multirate techniques. We have developed a multirate version of the popular Adams Predictor Corrector methods, which has a variety of modern features. We present results on the accuracy, stability and efficiency of the algorithm along with simulation results.

1 Introduction

Traditionally, when simulating a system of equations numerically, synchronization across all components is required therefore a single global time step is used. While the time step may change adaptively as the simulation proceeds, at each step the selection of step size is determined by the most ill-behaved component of the system. For example, in a system with many slow changing components and only one high frequency oscillatory component, a time step of 0.1 sec may be perfectly acceptable for the slow subsystems, however a global time step of 0.0001 sec must be used to accommodate the oscillatory component. The idea behind multistep numerical integration methods is, reduce the computational effort required by using the largest possible integration time step for *each* component of the system resulting in an asynchronous integration scheme. The idea dates back to the 1960's, [1] seems to have introduced it; [2] and others developed it further in the 80's. More recently, other variants of the method have been developed, ([3], [4], [5]). Areas of application include simulating integrated circuits and molecular and stellar dynamics.

Many robotic systems exhibit multiple time scales and hierarchy in the dynamics. In many cases, the hierarchy comes from the controller [6], [7]. As a motivating example, we will consider the notion of *hierarchically abstracted systems* presented in [8]; where a complex control system is decomposed into a sequence of increasingly simplified abstract systems. The idea behind this technique is that a fully detailed model may be impractical for long range planning and optimization so some simplification is done at a higher level; while at a lower level increasingly detailed system models are used and inputs are calculated to track the high level outputs. For example, in a typical robotic system one may have a planning module running at 1 Hz, an inverse kinematic solver running at 100 Hz, while the motor control loop may run at 10kHz. In such systems, a three level hierarchical description results in the following set of equations:

$$\dot{x}_1 = u_1 \quad (1)$$
$$\dot{x}_2 = f_2(x_2)u_2 \quad (2)$$
$$\dot{x}_3 = f_3(x_3) + g(x_3)u_3. \quad (3)$$

The first level is the trajectory generation or the planning level where the dynamic system is abstracted with one or more integrators. At a lower level, the kinematics and therefore the Jacobian $f_2(x_2)$ become important. Finally, at the lowest level, the rigid body dynamics are described by an affine system. Coupling is introduced through the feedback terms. If, for example the goal of eq.(2) is to track trajectories produced by eq.(1), u_2 may depend on both x_1 and x_2. Likewise, the input in eq.(3) may take the form $u_3(x_1, x_2, x_3)$ to track eq.(2). If one so desired, a fourth level could even be added to the model where the inputs are motor currents and actuator nonlinearities, such as motor saturation and deadzone effects, are accounted for. While planning based on eq.(1) may yield sub-optimal motion plans, the complexity of the planning problem decreases dramatically. Such approaches to planning and control are becoming more common as the complexity of robotic systems increases.

In robotic systems with contacts between nominally rigid bodies, it is also natural to see dynamics with multiple time scales [9]. The dynamics of contact in-

teraction evolve at a time scale that is determined by the very high stiffness of the contacting rigid bodies [10, 11]. For example, impacts between rigid bodies may last less than several microseconds and may necessitate integration rates of hundreds of nanoseconds [12], while the slowly evolving rigid body dynamics driven by position controllers may require integration rates of milliseconds. A typical two-level hierarchy in such systems has the following set of equations:

$$\ddot{x_1} = f_1(x_1, \dot{x_1}) \quad (4)$$

$$\ddot{x_2} = f_2(x_1, \dot{x_1}, x_2, \dot{x_2}). \quad (5)$$

Here x_1 represents the gross rigid body motion while x_2 models the compliant contact state.

More generally, the techniques presented in this paper are applicable to hierarchical systems of the following closed loop form:

$$\begin{aligned} \dot{x}_1 &= f_1(x_1) \quad (6)\\ \dot{x}_2 &= f_2(x_1, x_2)\\ &\vdots\\ \dot{x}_n &= f_n(x_1, \ldots, x_n). \end{aligned}$$

Closed loop systems in the form of eq.(6) appear in the control system literature under the name of *cascaded systems*. Of course when a particular equation in the above system is said to depend on the variables $x_j, \ldots, x_k$ this is the maximal set of variables it may depend on. Also note that the variables $x_1, \ldots x_n$ may be vectors, representing subsystems.

There has been a growing trend to study robotic systems, and other controlled systems as hierarchies [13], [8], [14], [15]. This trend has been paralleled in the computer aided design community with the introduction of several packages capable of modeling and simulating hierarchical control systems (see for example [16] or [17]). However, despite the seemingly natural connection, multirate methods have not appeared in the robotics literature. The goal of this paper is to introduce a new multirate method which is well suited to simulating robotic systems and analyze its performance. The paper is organized as follows: Sect. 2 presents the simulation algorithm; Sect. 3 looks at the accuracy, stability and efficiency of the method; Sect. 4 describes the implementation briefly and illustrates the use of the technique to simulate a mobile robot; finally in Sect. 5 we suggest some other application areas.

2 The algorithm

While multirate methods can and have been extended to systems of equations with structures other than lower triangular form, they are best suited to equations such as eq.(6). It is assumed that: if $i < j$, in some sense, x_i evolves *slower* than x_j; *and* the f_i's are sufficiently continuous and satisfy all the criteria to ensure the existence and uniqueness of the solution to the differential equation.

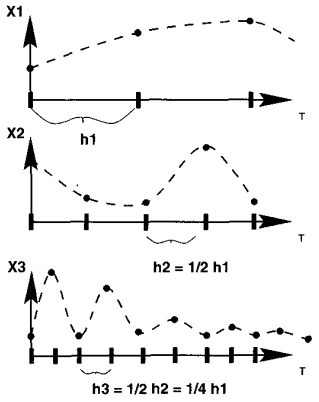

Figure 1: Multirate technique with $r_1 = r_2 = 1/2$.

In this section we develop a multirate version of an m^{th} order predictor-corrector (PC) pair. Multirate versions of other numerical integration schemes such as, BDF or extrapolation methods, have appeared elsewhere in the literature. The multirate Adams method differs from tradition version in that the systems is integrated asynchronously one component at a time. For each component a different time step may be used. Here h_i is the step size for the for the ith equation; and $h_i \geq h_j$ for $i < j$. The integer ratio, r_i of step sizes for adjacent levels in the hierarchy is given by the relation $h_i = r_i h_{i+1}$ for $i = 1, \ldots, n-1$, as seen in Fig. 1.

The decision of which equation to integrate at a given time is made according to the *slowest-first* criterion which was shown in [2] to have superior performance versus the *fastest first methods*. The slowest first criterion selects to advance the equation which has been simulated through the *least* amount of time so far. Since the step sizes are all integer multiples of one another, often there will be several equations that fulfill the criterion. In the case of such a tie we integrate the one with the smallest index (the slowest equation) first. For a review of the traditional version of the algorithm see [18] for example. The multirate algorithm proceeds as follows:

1. Select the equation to be integrated according the slowest first rule.
2. Compute the predicted value $x_i^P(t_{k+1})$ using values computed in previous steps

$$x_i^P(t_{k+1}) = x_i(t_k) + h_i \sum_{j=1}^{m} \beta_j f_i\left(x_1(t_{k+1-j}), \ldots, x_i(t_{k+1-j})\right)$$

where the β's are constants (see below).

3. Evaluate the derivative at the predicted point:

$$f_i^P = f_i\left(x_1(t_{k+1}), \ldots, x_{i-1}(t_{k+1}), x_i^P(t_{k+1})\right)$$

4. Since different equations are being integrated at different rates, values for x_1 through x_{i-1} used in the above step may not have been computed exactly at the mesh point (t_{k+1}), in which case they must be derived from interpolation polynomials.

5. Apply the corrector equation

$$x_i(t_{k+1}) = x_i(t_k) + h_i\left\{\beta_0^* f_i^P + \sum_{j=1}^{m-1} \beta_j^*\, f_i\left(x_1(t_{k+1-j}), \ldots, x_i(t_{k+1-j})\right)\right\}$$

where β^*'s are constants (see below).

6. In order to accommodate coupling, construct an interpolation polynomial for x_i which is valid from t_k to t_{k+1} so that component x_i can be estimated by inferior levels at off-mesh points

7. Evaluate the derivative using the corrected value of x_i for future use:

$$f_i(t_{k+1}) = f_i\left(x_1(t_{k+1}), \ldots, x_i(t_{k+1})\right)$$

In this algorithm the β's and β^*'s are weights associated with the particular predictor corrector method, but are generally selected such that, if the solution $x(t)$ was a polynomial of order $< m$, the solution would be replicated exactly (see [18]). After completing the above sequence of steps, the truncation error is estimated using Milene's method and the step size h_i is adjusted accordingly.

Steps 1, 4, and 6 are not present in the traditional version of the algorithm. Step 1 implements the slowest first rule; Steps 4 and 6 are needed since the right hand sides in eq.(6) are partially coupled. Due to the lack of synchronization, values of the slow variables, needed to evaluate the right hand sides of the fast variables, may not have been computed at time t_{k+1}. Thus interpolants are used to approximate these values at off-mesh points.

3 Analysis

For the sake of brevity the results are presented here without formal proofs. The interested reader is referred to [19] for the full derivations. As in most numerical analysis, we will perform the error analysis *locally* in Section 3.1 assuming the required past values of the x_i's and f_i's used in the PC method are *exact*, then look at the errors introduced after one step. This local error analysis is then extended by simply showing that the error propagation is stable in Section 3.2, rather than analyzing the accumulated error directly. An efficiency analysis follows in Section 3.3.

3.1 Integration error analysis

In this section we analyze the accuracy of the method compared to traditional simulation techniques. Such an analysis is important (1) to determine the expected performance of the method, (2) provide a method to control and estimate truncation errors and hence select an appropriate step size at runtime, and (3) provide a basis for selecting acceptable interpolating functions. It is easily shown that the error in multirate simulation, e_M can be expressed as the sum of errors associated with the traditional PC method e_T and an additional term which accounts for the fact that interpolated values are used to accommodate coupling e_I, (i.e. $e_M = e_T + e_I$). The error associated with the traditional implementation of a m-step PC method is known to be $O(h^m)$, therefore it suffices to derive only e_I. In particular we would like to show that the contributions to the overall error from e_I are small compared with e_T and hence the multirate method essentially has the same performance characteristics as the traditional method.

Terms contributing to e_I are introduced only during the corrector step since it is that step which relies on interpolating the slow variables, the predictor step produces the same result regardless of whether traditional or multirate techniques are used. Let $\hat{x}$ indicate an interpolated quantity while $\bar{x}$ indicates the value resulting from simply applying the traditional technique. After various algebraic manipulations, we get

$$e_I = h_i\beta_o\left\{ f_i\left(\bar{x}_1(t_k), \ldots, \bar{x}_{i-1}(t_k), x_i^P(t_k)\right) - f_i\left(\hat{x}_1(t_k), \ldots, \hat{x}_{i-1}(t_k), x_i^P(t_k)\right)\right\}$$

Assuming $[\hat{x}_j(t_k) - \bar{x}_j(t_k)]$ is small we can expand the second term about $\bar{x}_j(t_k)$. We then have, to first order

$$e_I = h_i\beta_o\left\{ \sum_{q=1}^{i-1} \frac{\partial f_i}{\partial x_q}[\bar{x}_q(t_k) - \hat{x}_q(t_k)] + O([\hat{x}_q(t_k) - \bar{x}_q(t_k)]^2)\right\} \quad (7)$$

where the partial derivative is evaluated at $\bar{x}_q(t_k)$. This term is effectively a measure of how sensitive our method is to errors in estimating f_i due to inexact interpolation of slow variables. Note that this term is multiplied with h_i which is typically small. Since the error contribution from e_I should not dominate e_T, we select $\hat{x}_q$ to be m^{th} order polynomials. Thus the interpolation errors vary with the m^{th} power of the interpolation interval, h_q. Recall that the larger step

sizes of higher layers in the hierarchy can be expressed in terms of smaller step sizes as $h_{i-1} = r_{i-1}h_i$ and in general $h_q = \prod_{l=q}^{i-1} r_l h_i$. Using these relations,

$$e_M \approx \underbrace{C_1 h_i^m}_{e_T} + \underbrace{C_2 h_i^{m+1} \beta_o \sum_{q=1}^{i-1} \frac{\partial f_i}{\partial x_q} \left[\left(\prod_{l=1}^{q-1} r_l \right)^m - 1 \right]}_{e_I} \quad (8)$$

Thus the dominant error term, e_T, is the same as that of the traditional Adams PC method. Note that as the step size of the slower equations approach that of the fastest equations (i.e. $r_l \to 1$), the e_I term vanishes and the error expression reduces to that of the traditional technique. It also tells us that any arbitrary choice of interpolants is not acceptable. Had linear interpolants, for example, been used the e_I term would be $O(h^2)$ rather than $O(h^{m+1})$; and hence, the dominant contributer to the overall error.

3.2 Stability

The stability of the method is more difficult to analyze since traditional techniques for determining the stability of numerical operators fail for multirate methods. Fortunately, when the slow components are not coupled to the fast components, which is the case in eq.(6), the stability of the multirate Adams scheme is the same as the stability of the traditional Adams scheme [2]. This is essentially because the interpolation error is always bounded; whereas, in the case of slow to fast coupling, extrapolation is needed to evaluate the right hand side and the errors may become unbounded. The stability characteristics of the traditional Adams methods are excellent (though they are not suitable for stiff problems).

3.3 Efficiency

Recall that the motivation for introducing multirate methods in robotics was to increase the simulation efficiency. Despite the fact that larger step sizes can be used, multirate simulation techniques are not *always* more efficient than traditional techniques since there is some overhead associated with constructing and evaluating the interpolation polynomials. A detailed and lengthly efficiency analysis appears in [19], but for the purpose of illustration we make a number of simplifying assumptions here: a fixed step size (as a function of time), the right hand sides of each of the subsystems $(f_1, \ldots f_n)$, are equally expensive to evaluate, the difference in step sizes in the various levels of the hierarchy in the modular simulation is constant (the step size at any given level is twice the size of the step size being used one level below and half that of one level above).

We are interested in computing the speed up ratio, E, which reflects how much faster (or slower) the multirate simulation is vs. the global simulation. It is computed by dividing the computational cost required to simulate a given system with the multirate technique by the cost of the traditional simulation. A small value of E implies a great efficiency gain is possible with multirate simulation.

In the traditional simulation, all components are integrated using the maximum acceptable step size for the *fastest* component, h. Ignoring minor overhead and the irregularities introduced in the start up process, the steps required to simulate one component through one step h (associated computational costs denoted in parenthesis) are: predict (PC), evaluate right hand side (RHS), correct (PC) and reevaluate right hand side (RHS). For the multirate simulator, assume that the fastest component is integrated at the same rate as in the global simulator, h, and that at each higher level the step size is double that of the previous. At each step at level i in the multirate simulator all the steps of traditional method are computed in addition to evaluating the interpolation polynomials of the (n-i) slower variables (PE), and generating the interpolation coefficients (CG). However since larger step sizes are used, fewer total steps need to be taken.

Tallying the required computation per step and computing the speed-up ratio yields

$$E = c_1 + c_2 \frac{CG}{(RHS + PC)} + c_3 \frac{PE}{(RHS + PC)}. \quad (9)$$

with

$$c_1 = \frac{2 - \frac{1}{2^n - 1}}{n} \quad c_2 = \frac{c_1}{2} \quad c_3 = \frac{2n - 4 + \frac{1}{2}^{n-2}}{2n}$$

each of the coefficients is strictly a function of the number of subsystems n and the ratio of step sizes across the hierarchy (in this case assumed to be 2^n); While the terms they multiply are specific to both the implementation and the particular set of ODE being integrated (RHS).

In our implementation with a 5^{th} order Adams method: $CG \approx 40$ flops (*floating point operations*); $PC = 11$; $PE = 8$ flops. Figure 2 shows the predicted efficiency of the method calculated with eq.(9), as a function of the complexity of evaluating the right hand sides of ode's (RHS) for 3, 4, and 5 level hierarchies. Note that the point at which the multirate simulation becomes faster (when $E \leq 1$), is low enough (15-25 flops per RHS) that many robotic systems will fall in that category. For example, systems controlled using feedback linearization require inverting a matrix: $RHS \approx 40$, systems which require computing inverse kinematics: $RHS \approx 200$, or systems which require computing rigid body contact models: $RHS \approx 2000$. Observe that the benefits of multirate simulation become more pronounced as the number of subsystems

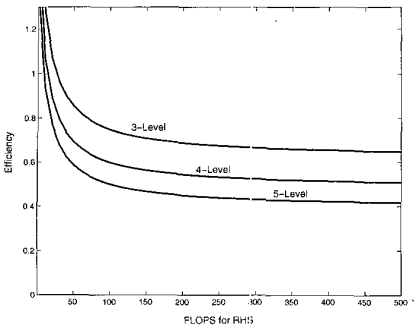

Figure 2: Efficiency as a function of the cost of evaluating the right hand side.

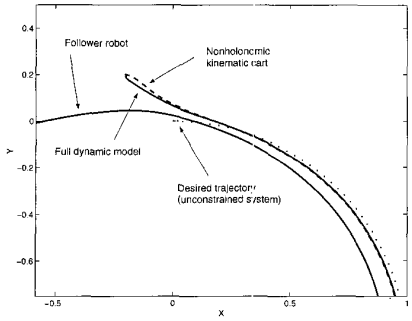

Figure 3: Simulation of the hierarchically controlled cart. The (x, y) coordinates are plotted at each level.

increases. It also should be mentioned that, while the ratio of step sizes of neighboring levels in the hierarchy used to compute Fig. 2 is 2, the efficiency gain increases as this ratio increases.

4 Implementation

The algorithm has been implemented in Matlab, using a 5^{th} order PC pair. It supports automatic step size selection, both across time and across levels of the hierarchy, to monitor and control truncation error to a user defined tolerance as the simulation proceeds. The user may specify groups of variables, called subsystems, that should be computed at the same rate. The method will be part of the simulation suite for the CHARON modeling language [17].

In Section 1, the idea of hierarchical control was motivated. As an example problem we simulated a hierarchical model of a standard differential drive cart. Equation (1) which is essentially a holonomic cart model was used for trajectory generation (level-1); a nonholonomic model of the cart is written in the form of eq.(2) and the controller calculates wheel velocities at level-2 to track the output of the trajectory generator at level-1; finally, at level-3 torques are computed so that the full dynamic model of the cart tracks the

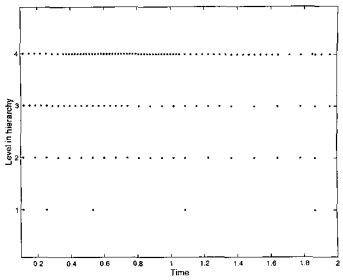

Figure 4: Illustration of the frequency at which the different levels in the hierarchy are integrated.

constrained kinematic reference trajectory. In addition we also added a forth level of in the hierarchy which models a second dynamic cart attempting to follow the first cart at a prespecified distance. Figure 3, shows the results of simulating this system with the tool presented in this paper. The position coordinates of eqs. (1, 2, 3) are plotted.

In this particular case we compared the performance of our (nonoptimized)code vs. that of a popular built in Matlab ode solver(*ode45*), using the same error tolerances. The multirate code required only 27% of the number of floating point (about 4 times faster) operation used by the built-in solver. Figure 4 shows the frequency with which the multirate algorithm integrates the various levels in the hierarchy. As expected the first level in the hierarchy is integrated with a large steps size ($h_1 \approx 0.6$) while the lowest level is integrated with a much smaller step size to capture the fast changing dynamics ($h_4 \approx 0.015$). These step sizes are selected dynamically to maintain a desired level of estimated integration error while maximizing the efficiency of the simulation. Recall that in the traditional simulation all components would have been integrated with a global stepsize corresponding to the fastest component ($h \approx 0.015$). In general the speed up factor is a function of several parameters, especially the desired integration accuracy. Figure 5 depicts this relationship. For this example the efficiency gain is more dramatic than the conservative estimate indicates in Fig. 2. This is because, while the analysis was performed assuming the ratio of stepsizes used in neighboring layers in the hierarchy was two, this particular example permitted ratios as high as 8. Notice that the benefits of multirate simulation increase as the error tolerances become tighter.

5 Conclusions

In this paper we have introduced multirate predictor corrector techniques for simulating robotic systems possessing multiple time scales. The method is shown to have accuracy on the same order as that of the traditional predictor-corrector pair and the same stability

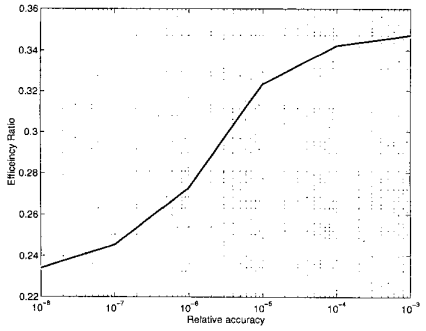

Figure 5: Experimental results for the mobile robot example on the efficiency of the multirate technique as a function of the integration tolerance.

properties, for the class of systems discussed here. The efficiency analysis indicates that a significant reduction of computation time is possible (a conservative analysis suggests 30-80% for some representative systems), when the system is sufficiently complex.

Other robotic systems that could potentially benefit from this type of simulation are: (1) Compliant contact models for rigid body simulation. The compliant contact point represents the fast dynamics while the gross motion is comparatively slow. (2) Large multiagent systems, such as platoons of automated highway vehicles; or simulations of many independent rigid bodies. Here one does not want to slow down the entire simulation when only two agents collide. (3) Hierarchically abstracted systems such as the one presented in this paper or [13]. This may be particularly useful in undulatory locomotion systems, as indicated in [15]. Future work will focus on developing techniques for systems where the right hand side of the differential equations changes discontinuously when certain *events* occur (*hybrid systems*). For example, in rigid body dynamics an event would be a change in contact states, whereupon the governing equations suddenly transition from unconstrained motion to rolling or sliding. Multirate techniques will hopefully allow one to simulate the bodies undergoing critical transitions at a finer level of detail (smaller time step), increasing the likelihood that these events are properly detected, without slowing down the integration of unaffected bodies.

Acknowledgments

The authors thank George Pappas, Aveek Das, and C.J. Taylor for several interesting discussions about hierarchical systems. We gratefully acknowledge support from DARPA grant ITO/MARS 130-1303-4-534328-xxxx-2000-0000, and a DoE GAANN grant.

References

[1] R.C.Rice, "Split Runge-Kutta methods for simultaneous equations," *Journal of Res. National Bureau of Standards*, vol. 64B, pp. 151–170, 1960.

[2] C.W.Gear and D.R.Wells, "Multirate linear multistep methods," *BIT*, vol. 24, pp. 484–502, 1984.

[3] J. Sand and S. Skelboe, "Stability of backward euler multirate methods and convergence of waveform relaxation," *BIT*, vol. 32, pp. 350–366, 1992.

[4] M.Gunther and P.Rentrop, "Multirate row methods and latency of electrical circuits," *Applied Numerical Mathematics*, vol. 13, pp. 83–102, 1993.

[5] C.Engstler and C.Lubich, "Multirate extrapolation methods for differential equations with different time scales," *Computing*, vol. 58, pp. 173–185, 1996.

[6] J. Albus, "The NIST real-time control system (RCS) an approach to intelligent systems research," *Special Issue of the Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, pp. 157–174, 1997.

[7] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. RA - 2, no. 1, pp. 14–23, March 1986.

[8] G.Pappas, G.Lafferriere, and S.Sastry, "Hierarchically consistant control systems," *IEEE transactions on automatic control*, vol. 45, no. 6, pp. 1144–1160, 2000.

[9] N. McClamroch, "A singular perturbation approach to modeling and control of manipulators constrained by a stiff envrionment," *Proceedings of the* 28^{th} *Conference on Decision and Control*, pp. 2407–2411, Dec. 1989.

[10] W.S.Howard and V.Kumar, "A minimum principle for the dynamic analysis of systems with frictional contacts," *IEEE International Conference on Robotics and Automation, Atlanta*, pp. 437–442, May 2-7, 1993.

[11] P. Song, M. Yashima, and V. Kumar, "Dynamic simulation for grasping and whole arm manipulation," *IEEE International Conference on Robotics and Automation*, pp. 1082–1087, May 2-7, 1993.

[12] J.C.Gerdes and V. Kumar, "An impact model for mechanical backlash for control system analysis," *American Control Conference, Seatle, Washington*, June 21–23, 1995.

[13] M.Engerstedt, "Behavior based robotics using hybrid automata," *Hybrid Systems Computation and Control: Third international workshop*, vol. 3, pp. 103–115, 2000.

[14] K.Gokbayrak and C.Cassandras, "Hybrid controlers for hierarchically decomposed systems," *Hybrid Systems Computation and Control: Third international workshop*, vol. 3, pp. 117–129, 2000.

[15] K.McIsaac and J.Ostrowski, "Steering algorithms for dynamic robotic locomotion systems," in *Workshop for the Algorithmic Foundations of Robotics*, 2000.

[16] J. X.Lui, T.J.Koo, B.Sinopoli, S.Sastry, and E.A.Lee, "A hierarchical hybrid system model and its simulation," *Proceedings of the* 38^{th} *Conference on Decision and Control*, pp. 2407–2411, 1999.

[17] R.Alur, R. Grosse, Y.Hur, V. Kumar, and I. Lee, "Modular specification of hybrid systems in charon," *Hybrid Systems Computation and Control: Third international workshop*, vol. 3, pp. 6–19, 2000.

[18] U. Ascher and L.Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations.* Philadelphia, PA: Society for Industrial and Applied Mathematics, 1998.

[19] J. Esposito and V. Kumar, "Multirate numerical integration methods for simulating robotic and hybrid systems," tech. rep., University of Pennsylvannia, 2000.