Proceedings of the 40th IEEE
Conference on Decision and Control
Orlando, Florida USA, December 2001

Multi-Agent Hybrid System Simulation

Joel Esposito†, Vijay Kumar†, and George J. Pappas‡
GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA
† Dept. of Mechanical Engineering, ‡ Dept. of Electrical Engineering
jme, kumar, pappasg @grip.cis.upenn.edu

Abstract

In this paper a technique is presented for simulating a set of hybrid systems (*agents*) whose continuous dynamics are decoupled; however coupling is introduced in the form of constraints (inequalities whose violation signifies the occurrence of a discrete event). We introduce a step size selection algorithm, motivated by the control theoretic concept of Input/Output linearization, which allows two or more agents to be integrated asynchronously using different step sizes when the state is away from the constraint. When the state approaches a constraint, the algorithm is able to bring the two local time clocks into synchronization and localize the time of constraint violation. The algorithm is guaranteed to never miss or overshoot the constraints, eliminating the need for simulation roll-back.

1 Introduction

Hybrid systems are systems of continuous ordinary differential equations(ODEs) whose right-hand sides switch based on the occurrence of discrete events. The trajectory of the hybrid system can be viewed as a concatenation of continuous flows and discrete jumps. A *multi-agent hybrid system* is a collection of interacting hybrid systems (individually referred to as *agents*). Simulation is an important tool for designing such systems; in addition, numerical approximation techniques are increasingly being used in approximate reachability computations, verification and other forms of automated analysis [1], [2], [3].

One particular class of multi-agent hybrid systems for which we would like to develop improved simulation techniques consists of groups of agents whose ODE's are decoupled while the guard depends on the state of both agents. Generically if x^i is the state of Agent i, where $i = 1, \ldots, N$

$$\dot{x}^i = f^i(x^i) \qquad g^{ij}(x^i, x^j) < 0 \tag{1}$$

where $i \neq j$ and $x^i \in R^{n_i}$ (resp. x^j), $f^i : R^{n_i} \to R^{n_i}$ (resp. x^j), and $g^{ij} : R^{n_i} \times R^{n_j} \to R$. Once $g^{ij}(x^i, x^j) \geq 0$, f_i and f_j would change. Examples of such systems abound. Automated highway systems, free-flight air traffic control, cooperative mobile robotic systems, and multi-body systems simulated for graphics applications all can be modeled this way – the dynamics of each vehicle, plane, or robot are decoupled; however, certain critical events which are relevant to the simulation (*e.g.* collisions) depend on the states of pairs of agents.

It is well known that, when simulating hybrid systems, a failure to detect an event can have disastrous results on the global solution due to the discontinuous nature of the problem. Documents detailing requirements for hybrid simulators list accurate event detection as a primary concern [4]. Event detection in hybrid systems is, in itself, a very difficult problem [5]. The proper way of simulating such a system is to terminate the numerical integration procedure as close as possible to the time of constraint violation; activate the appropriate mode switches and resets; and then restart the simulation with the new differential equations, using the event time as the new initial condition and the value of the state at the event as the new initial state. Therefore accurately detecting and locating the time at which the event occurred is critical to generating valid simulation results. To ensure proper event handling all existing hybrid system simulators must simulate each ODE in eq.(1) in a synchronized fashion, despite the fact that their dynamics are decoupled. This approach can lead to gross inefficiencies, especially when the number of agents is large, because it forces all of the ODEs to be simulated with the smallest acceptable step size. Indeed this has been mentioned as one of the reasons the SHIFT [6] simulator uses a fixed step size integration algorithm, despite the obvious disadvantages. Our ultimate goal is to develop a step size selection technique which is capable of simulating each agent asynchronously when the system is "far away" from the constraint surface, allowing each agent to be integrated with its own largest acceptable step size, increasing the overall efficiency of the simulation. As the system approaches the constraint, the relevant local time clocks automatically synchronize in order to properly detect and localize the occurrence of the surface crossing (*i.e.* the discrete event).

A variety of literature in fields closely related to this topic exists although it seems there has been little work specifically on this problem. For the "single-agent" case, early works [7], [8] recognized the need for special algorithms to locate discrete events. In [5] a technique for finding zero crossings of a polynomial constraint function is presented. A rigorous technique for event detection which uses interval arithmetic to develop an efficient "exclusion" test (a test which eliminates most intervals not containing an event from further consideration) is reported in [9]. Finally in [10] we present a control theoretic method which is guaranteed to correctly detect an event for certain classes of constraints, while guaranteeing that the ODE is never evaluated on the opposite side of the switching surface ($g(x) = 0$). This technique forms the basis for our current work and is reviewed in detail later.

2 Background

In this section we present some results leading up to the main contribution of this paper, given in Section 3. First, in Section 2.1, we review the details of *Linear Multistep Methods*– our prefered numerical integration technique. We then summarize our previous work [10] on event detection for hybrid system simulation for the single agent case in Section 2.2, since it provides the foundation for the multi-agent approach.

2.1 Linear multistep methods

Given the system $\dot{x} = f(x)$ and $x_0 = x(0)$, it is customary to denote the simulation data at the discrete time t_k as $x_k = x(t_k)$, and the value of the time derivative as $f_k = f(x_k)$. It is also convention to define the time step as $h_{k+1} = t_{k+1} - t_k$. Linear multistep methods (LMSM) are a popular set of numerical integration techniques, the most popular of which are the Adams methods. In such a method, the simulation is advanced by solving for x_{k+1} in terms of a weighted combination of past values of the state x and its derivative $f(x)$:

$$x_{k+1} = x_k + h_{k+1} \sum_{j=1}^{m} \beta_j f_{k-j+1}, \qquad (2)$$

where the weights, β_j, are selected such that the difference equation would *exactly* reproduce the analytical solution $x(t)$ if it were a polynomial of order m or lower. In general the accuracy of the method is proportional to $(h_{k+1})^m$. See any numerical analysis text for further details [11].

Often in text books, values of β are supplied as constants; however this is only the case when the step size is constant. In general, β is a rational polynomial function of the previous m step sizes. For our purposes it is important to note that, since the past step sizes are known, β_j is essentially an m^{th} order rational polynomial function of h_{k+1}. We emphasize this by writing $\beta_j(h_{k+1})$. Multistep methods are a natural choice for simulating hybrid systems because the polynomial expressions for β_j can be used as interpolants to approximate the solution at off-mesh points, which are exploited in event detection.

We also mention that most modern numerical integration software packages compute an ideal step size, h_{err} to keep the estimated local truncation error near some desired value. In addition an absolute minimum and maximum step size, $h_{\min}$ and $h_{\max}$ are typically specified. An automatic step size selection scheme takes all these factor into account. In the next section we discuss an algorithm for computing an ideal step size based on event considerations.

2.2 Control theoretic view of event detection

It is well known that systems of differential equations with nonlinear algebraic constraints can be transformed to a equivalent systems with linear constraints by appending a new state variable $z = g(x)$,

$$\begin{array}{lcl} \dot{x} = f(x) & \Leftrightarrow & \dot{x} = f(x),\ \dot{z} = \dfrac{\partial g}{\partial x} \cdot f(x) \\ g(x) < 0 & & z < 0. \end{array} \qquad (3)$$

Thus we only consider the case of linear constraints here; other cases, such as polynomial and more general nonlinear constraints, are handled directly in [10].

We treat the problem of selecting the step size to properly approach the event surface as a control system design problem in discrete time. For a linear multistep method the "system dynamics" are given by eq.(2) which implies the constraint dynamics are

$$g(x_{k+1}) = g_{k+1} = g(\, x_k + h_{k+1} \sum_{j=1}^{m} \beta_j f_{k-j+1}\,). \qquad (4)$$

In our control system analogy, we consider the constraint as an "output function". If the event function is linear, eq.(4) becomes

$$g_{k+1} = g_k + h_{k+1} \frac{\partial g}{\partial x} \sum_{j=1}^{m} \beta_j f_{k-j+1} \qquad (5)$$

which can be thought of as a Taylor series expansion in h_{k+1} about x_k. We treat h_{k+1}, in eq.(2) as an input. The problem is then to select a "feedback function" (a rule for selecting h_{k+1}) to force the constraint dynamics, eq.(5), to converge exponentially to the surface $g(x) = 0$ without overshooting it. Note that $\frac{\partial g}{\partial x} \sum_{j=1}^{m} \beta_j f_{k-j+1}$ is simply the Lie derivative $(L_{f\beta} g)_k$. Using the technique of Input/Output linearization (see [12]), select

$$h_{k+1} = \frac{(\gamma - 1) g_k}{\frac{\partial g}{\partial x} \sum_{j=1}^{m} \beta_j (h_{k+1}) f_{k-j+1}}, \qquad (6)$$

yielding the *difference* equation $g_{k+1} = \gamma g_k$, which has the solution $g_k = g_0 \gamma^k$ and converges exponentially to $g = 0$ as desired, provided $0 \leq \gamma < 1$. Note that if the denominator is zero the system is moving tangentially to the constraint surface and no event detection is needed.

As mentioned earlier, the β's for the Adams Method are only constant in the special case of constant step size. Since we are proposing to adjust the step size dynamically, the β's in the above discussion are not constant, but rather are rational polynomial functions of h_{k+1}. Computing the correct step size with eq.(6), for example, then entails finding the roots of a polynomial in h_{k+1}. The algorithm is as follows:

```
h = step_select(  $f_k, \ldots, f_{k-m}, h_{\max}, h_{\min},$
                  $h_{err}, \gamma, g$ ) {
  $Z = Roots\{hL_{f_{\beta(h)}}g - (\gamma - 1)g(x_k)\}$ ;
  $R = \{r \mid r \in Z,\ Im(r) = 0,\ r \geq 0\}$ ;
  if $R = \emptyset$
        $h_{k+1} = \infty$ ;
  else
        $h_{k+1} = \min_{r \in R}(r)$ ;
  end
  $h = \max[h_{\min}, \min(h_{\max}, \min[h_{k+1}, h_{err}])]$;
    terminate? }
```

For example in the case of two step Adams method $\beta_1 = (2h_k)/h_{k+1}$ and $\beta_2 = 1 - (2h_k)/h_{k+1}$. Substituting the expressions for β into eq.(6) and rearranging gives

$$Z = Roots[ah_{k+1}^2 + bh_{k+1} + c] \tag{7}$$

where $a = 1/2 \cdot h_k[\partial g/\partial x \cdot (f_k + f_{k-1})]$, $b = \partial g/\partial x \cdot f_k$ and $c = -(\gamma - 1)g(x_k)$. Eq.(7) must be solved for h_{k+1} at every time step.

This algorithm has several advantages over other existing algorithms. First, it is guaranteed to properly detect and locate the first event which occurs in an integration interval [10]. Secondly, since the gain is selected in such a way as to prevent overshoot, the simulation never proceeds past the switching surface $g(x) = 0$; hence it is never necessary to roll back the simulation. Often the motivation for modeling a system as a hybrid system is that a single model is not valid everywhere in the state space; due to model singularities, the simulation must switch between different ODEs. Not overshooting the constraint prevents accidental attempts to evaluate the right-hand side of the ODE at a model singularity. Finally, our step size selection scheme preserves the underlying accuracy and stability of the numerical integration method.

3 Multi-agent event detection

Extending the results of the previous section to the multi-agent case which was discussed in Section 1 is more complicated due to the fact that we want to allow the simulation for Agent 1 and Agent 2 to proceed at different rates.

Therefore we consider the simplest example of a multi-agent hybrid system:

$$\dot{x}^1 = f^1(x^1) \qquad \dot{x}^2 = f^2(x^2) \tag{8}$$

$$g(x^1, x^2) < 0 \tag{9}$$

We must define local times and time steps, t^1, h^1, and t^2, h^2 for each agent. The dynamics of the two systems and the

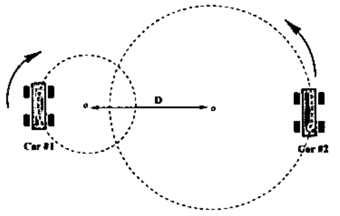


Figure 1: Two automated vehicles driving in spiral trajectories.

constraint are

$$x_{k+1}^1 = x_k^1 + h_{k+1}^1 \sum_{j=1}^{m} \beta_j(h_{k+1}^1) f_{k-j+1}^1 \tag{10}$$

$$x_{k+1}^2 = x_k^2 + h_{k+1}^2 \sum_{j=1}^{m} \beta_j(h_{k+1}^2) f_{k-j+1}^2 \tag{11}$$

$$g_{k+1} = g\left(x_k^1 + h_{k+1}^1 \sum_{j=1}^{m} \beta_j(h_{k+1}^1) f_{k-j+1}^1, \right. \\ \left. x_k^2 + h_{k+1}^2 \sum_{j=1}^{m} \beta_j(h_{k+1}^2) f_{k-j+1}^2 \right). \tag{12}$$

If we again assume the constraint function has been converted to a linear form, eq.(12) becomes

$$g_{k+1} = g_k + h_{k+1}^1 \frac{\partial g}{\partial x^1} \sum_{j=1}^{m} \beta_j(h_{k+1}^1) f_{k-j+1}^1 + \\ h_{k+1}^2 \frac{\partial g}{\partial x^2} \sum_{j=1}^{m} \beta_j(h_{k+1}^2) f_{k-j+1}^2. \tag{13}$$

Returning to our control system analogy, eq.(13) is a single output for a system with two inputs, h^1 and h^2, implying there is some freedom in the selection of the step sizes. It is important to point out that $g(x^1(t^1), x^2(t^2)) = 0$ does not necessarily correspond to a physical event unless $t^1 = t^2$. If $t^1 \neq t^2$ this implies that the two agents have passed though the same point in on the constraint surface *but at different local times*. Thus we introduce a second, fictitious, constraint which we term the synchronization constraint. Assuming for now that $t^1 > t^2$, define

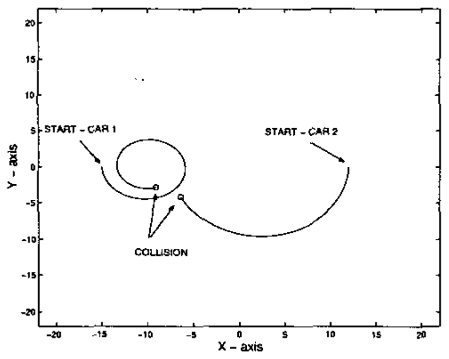
$$\tau(t^1, t^2) = t^2 - t^1 < 0 \tag{14}$$

$$\tau_{k+1} = \tau_k + h^2 - h^1. \tag{15}$$

Since $\tau = 0$ implies the two simulations are synchronized, it is a necessary condition for an event to have physical significance.

We simply treat eq.(14) as an additional constraint and use our Input/Output linearization technique, presented in the previous section to select a candidate step size for h^2 which would drive $\tau \to 0$

$$h_{k+1}^2 = (\gamma - 1)\tau_k + h_{k+1}^1, \tag{16}$$



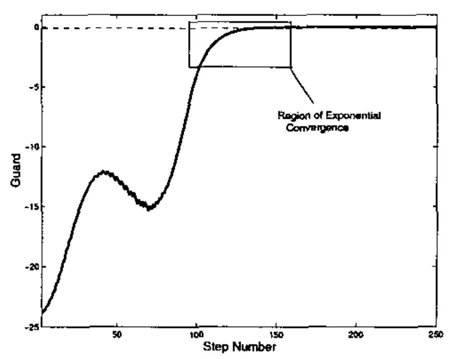
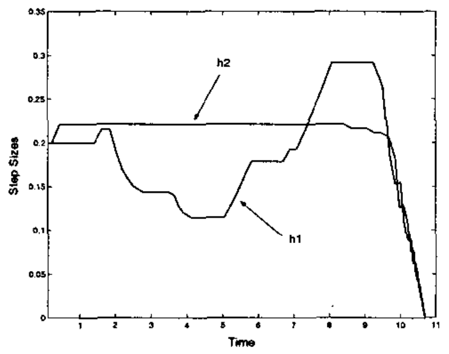


Figure 2: The trajectory of the two cars in the plane (*left*). D is small enough that they collide. The value of the constraint as a function of step number (*right*). The constraint converges to zero exponentially.



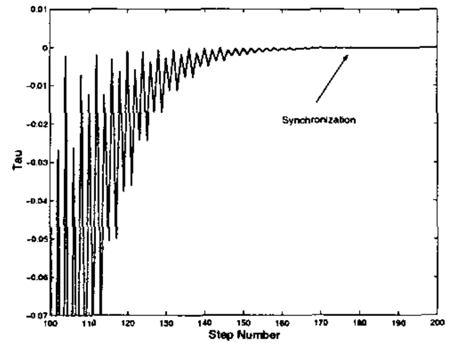


Figure 3: Step sizes for used in the first example (*left*). h^1 and h^2 are selected independently away from the constraint but are brought into synchronization when an event is impending. The value of τ (*right*), which is a measure of how out of synchronization the two local time clocks are.

this causes t^2 to synchronize with t^1. Using this constraint to eliminate h^2 in eq.(13), we have

$$g_{k+1} = g_k + h^1_{k+1}\frac{\partial g}{\partial x^1}\sum_{j=1}^{m}\beta_j(h^1_{k+1})f^1_{k-j+1} +$$
$$((\gamma-1)+h^1_{k+1})\cdot\frac{\partial g}{\partial x^2}\sum_{j=1}^{m}\beta_j((\gamma-1)+h^1)f^2_{k-j+1}. \quad (17)$$

The algorithm is as follows; assuming $t^1 > t^2$ (otherwise exchange the superscripts below):

```
h = step_select2( $f^1_k,\ldots,f^1_{k-m},f^2_k,\ldots,f^2_{k-m}$,
  $h_{\max},h_{\min},h^1_{err}$, $h^2_{err}$, $\gamma$, $g$) {
  $\tau_k = t^2 - t^1$;
  $Z$ = Roots of eq.17 when $g_{k+1} = (\gamma - 1)g_k$;
  $R = \{r \mid r \in Z,\ Im(r) = 0,\ r \geq 0\}$
  if $r = \emptyset$
        $h^1_{k+1} = \infty$; $h^2_{k+1} = \infty$;
  else
        $h^1_{k+1} = \min(r)$;
        $h^1_{k+1} = (\gamma-1)\tau_k + h^1_{k+1}$;
  end
  $h^1 = \max[h_{\min}, \min(h_{\max}, \min[h^1_{k+1}, h^1_{err}])]$;
  $h^2 = \max[h_{\min}, \min(h_{\max}, \min[h^2_{k+1}, h^2_{err}])]$;
    terminate? }
```

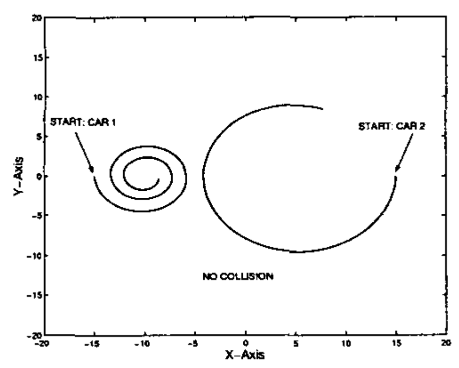
One thing to note about the algorithm is that at each step we choose to advance the agent which lags the most. Since nearly all integration algorithms impose an upper bound on the allowable step size $h_{\max}$ this method of advancing the slowest agent has the effect of ensuring that τ is also bounded. Since our step size selection scheme is based on solving for the roots of an extrapolation polynomial, bounding the extrapolation interval increases the accuracy of the results.

4 Examples

In this section the effectiveness of the algorithm presented in the previous section is illustrated through a set of examples. The basis for these example is illustrated in Figure 1, which depicts two automated vehicles driving in the plane. The dynamics for agent $i = 1, 2$ are

$$\dot{x}^i = v^i\cos(\theta^i),\ \dot{y}^i = v^i\sin(\theta^i),\ \dot{\theta}^i = \omega^i. \quad (18)$$

The v^i and ω^i are functions of x^i, y^i and θ^i and are selected in such a way as to steer the vehicles in spiral trajectories. However Car 1 travels much faster than Car 2. The constraint function is chosen to detect a collision between the



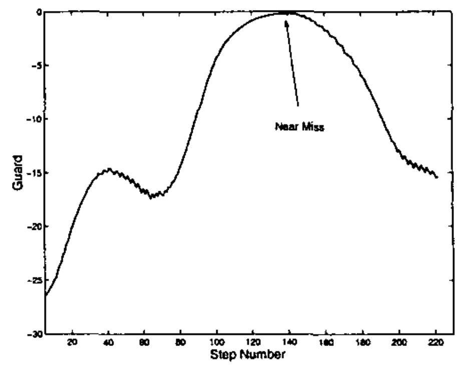
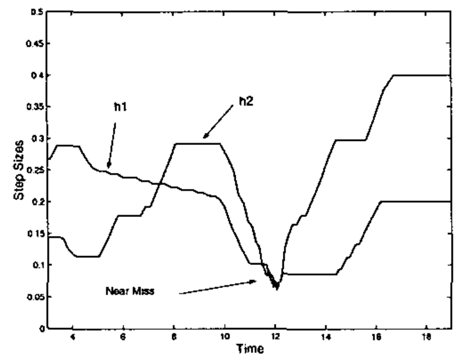


Figure 4: The trajectory of the two cars in the plane from the second example (*left*). D is selected such that the two cars come very close without colliding. The value of the constraint as a function of the step number (*right*). The constraint comes close to zero as the cars barely miss each other.



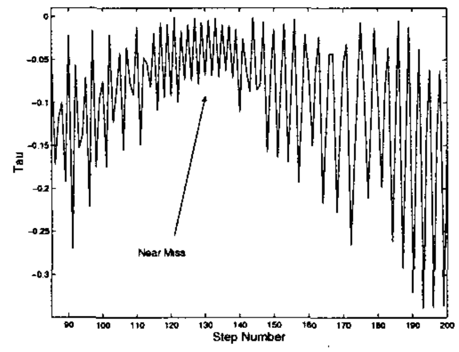


Figure 5: Step sizes for used in the second example(*left*). h^1 and h^2 are selected to slow the simulation down when it seems an event may occur. Once it is apparent that this is not the case the simulation quickly speeds up. The value of τ (*right*), also decreases during this period but soon increases again.

two vehicles

$$g = -\sqrt{(\,x^1 - x^2\,)^2 + (\,y^1 - y^2\,)^2} + R < 0 \qquad (19)$$

where R is the sum of the two vehicle's radii. This example was simulated using the algorithm presented earlier for a variety of scenarios each with a different value for the separation distance (marked D in Figure 1).

In the first example, Fig. 2-3, the value of the separation distance D is small enough that the vehicles eventually collide. The path of the vehicles in the plane is shown in Fig. 2(*l*). Fig. 2(*r*) is a plot of the value of the constraint as a function of the iteration. The constraint peaks first around step 70, when car 1 completes a half of rotation; finally at step number 107 the event detection criterion becomes active and the step sizes are selected in such a way that $g \to 0$ exponentially. Further insight is gained by examining Fig. 3(*l*) which shows the step sizes used for each agent as a function of time. Note how h^1 and h^2 vary independently until $t \approx 9.8$ when the event detection mode is active; whereafter they both begin adjusting to slow down the simulation and synchronize the two local clocks. This is further illustrated in Fig. 3(*r*) which plots the history of the synchronization function τ. Its value rapidly approaches zero to bring the two agent into synchronization.

In the second case, Fig. 4-5, the value of the separation distance D is increased enough that the vehicles do not collide, but they do come very close to one another. The trajectories of the two cars are shown in Fig. 4(*l*). There is a near miss halfway through car 1's second rotation. The history of the constraint in Fig. 4(*r*) shows the value of $g(x)$ comes very close to zero. During this period, the value of τ, Fig. 5(*l*), decreases in preparation for a possible event. Fig. 5(*r*) shows how the step sizes were decreased to slow down the simulation. However once the two cars pass each other and it becomes apparent that no collision will occur, the step sizes quickly increases and the two local clocks are allowed to fall out of synchronization again.

5 Extension to N agents

Extending the simulation methodology to a system of N agents with pairwise inequality constraints is, from a theoretical point of view, straight forward, although there is significant book keeping involved. Given N sets of ODEs and pairwise constraints of the form of eq.(1), for $i = 1, \ldots, N$ and $j > i$. Without loss of generality assume $t_k^i > t_k^j$, $\forall j > i$ then define $\tau_k^{ij} = t_k^j - t_k^i$. In a similar fashion to two agent algorithm eq.(16) and eq.(17) must be solved simultaneously (with i replacing the superscript 1, j replacing 2 and g^{ij} instead of g). This will yield solutions for $h_{k+1}^i(i,j)$, which is the suggested size of the $k+1^{th}$ step for agent i based on constraint ij, along with $h_{k+1}^j(i,j)$. Considering the $N-1$ constraints which depend on x^i will then result in $N-1$ suggested step sizes the smallest of which is appropriate

$$\begin{aligned} h_{k+1}^i &= \min\{h_{k+1}^i(1,i), \ldots, h_{k+1}^i(i-1,i), \\ &\quad h_{k+1}^i(i,i+1), \ldots h_{k+1}^i(i,N)\} \end{aligned} \tag{20}$$

More detailed analysis of the N-agent case is the topic of future work.

6 Conclusion

In this paper a technique is presented for simulating multi-agent hybrid systems whose continuous dynamics are decoupled; however coupling is introduced in the form of constraints (an inequality whose violation signifies the occurrence of a discrete event) which depend on the states of both agents. We introduce a step size selection algorithm, motivated by the control theoretic concept of Input/Output linearization, which allows the two agents to be integrated asynchronously using different step sizes when the state is away from the constraint. As the state approaches the constraint the algorithm is able to bring the two local time clocks into synchronization and slow down the simulation in such a way that the algorithm terminates exactly on the surface of the constraint. The algorithm is guaranteed never to miss or overshoot the constraint, eliminating the need for simulation roll-back. The important question of how the performance of this simulation algorithm compares to that of more traditional simulation methods will be quantified in future work. Further work will also examine how the computational benefits of the algorithm scale as the number of agents gets large.

Acknowledgments

We gratefully acknowledge support from DARPA grant MOBIES F33615-00-C1707, NSF CDS-97-03220 and Hercules Inc. The first author is also partially supported by a DoE GAANN grant.

References

[1] A. Chutinam and B. Krogh, "Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations," in *Hybrid Systems : Computation and Control* (F. Vaandrager and J. H. van Schuppen, eds.), vol. 1569 of *Lecture Notes in Computer Science*, Springer Verlag, 1999.

[2] T. Dang and O. Maler, "Reachability analysis via face lifting," in *Hybrid Systems : Computation and Control* (T. Henzinger and S. Sastry, eds.), vol. 1386 of *Lecture Notes in Computer Science*, pp. 96–109, Berlin: Springer Verlag, 1998.

[3] I. Mitchell and C. Tomlin, "Level set methods for computation in hybrid systems," in *Hybrid Systems : Computation and Control* (N. Lynch and B. H. Krogh, eds.), vol. 1790 of *Lecture Notes in Computer Science*, pp. 310–323, Springer Verlag, 2000.

[4] P.Mosterman, "An overview of hybrid simulation phenomena and their support by simulation packages," in *Hybrid Systems : Computation and Control* (F. Vaandrager and J. H. van Schuppen, eds.), vol. 1569 of *Lecture Notes in Computer Science*, pp. 163–177, Springer Verlag, 1999.

[5] L.F.Shampine, I.Gladwell, and R.W.Brankin, "Reliable solution of special event location problems for ODEs," *ACM transactions on Mathematical Software*, vol. 17, no. 1, pp. 11–25, March 1991.

[6] A. G. A. Deshpande and L. Semenzato, "Shift programming language and runtime system for dynamic networks of hybrid automata," *California PATH*, 1995.

[7] M. Carver, "Efficient integration over discontinuities in ordinary differential equation simulations," *Mathematics and Computers in Simulation*, vol. XX, pp. 190–196, 1978.

[8] C. Gear and O.Osterby, "Solving ordinary differential equations with discontinuities," tech. rep., Dept. of Comput. Sci., University of Illinois, 1981.

[9] T.Park and P.Barton, "State event location in differential-algebraic models," *ACM transactions on modeling and computer simulation*, vol. 6, no. 2, pp. 137–165, 1996.

[10] J. M. Esposito, V. Kumar, and G. J. Pappas, "Accurate event detection for simulating hybrid systems," in *Hybrid Systems : Computation and Control* (M. D. Benedetto and A. Sangiovanni-Vincentelli, eds.), vol. 2034 of *Lecture Notes in Computer Science*, pp. 204–217, Springer Verlag, 2001.

[11] S. C. K.Benan and L. Petzold, *Numerical solutions of initial value problems*. London: North Holland, 1989.

[12] A.Isidori, *Nonlinear Control Systems*. London: Springer, 1995.