From *Proc. 34th IEEE Conference on Decision and Control,* New Orleans, LA, December 1995.

Modeling and Simulation of Hybrid Systems

James H. Taylor & Dawit Kebede
Department of Electrical Engineering
University of New Brunswick
Fredericton, NB CANADA E3B 5A3
Internet: *jtaylor@unb.ca*

Abstract

Previous research in the area of modeling and simulation of hybrid systems led to the development of a general hybrid systems modeling language (HSML), part of which was implemented in a preliminary way by extending MATLAB. This paper describes the second phase of algorithmic implementation of the HSML ideas and language constructs for dealing with state-event handling and model structural changes in continuous-time system components. Specifically, the standard MATLAB model framework and integration algorithms are extended to support these phenomena. An example is presented to show the efficacy of these extensions within the MATLAB environment.

1 Introduction

The HSML language [1, 2] was designed to support a broad definition of a hybrid system, which we may express informally as being an arbitrary interconnection of components that are arbitrary instances of continuous-time, discrete-time and logic-based systems. Requirements for HSML focused particularly on rigorous characterization and execution of "events", both discrete- and continuous-time, that cause discontinuous changes in system trajectories and/or the model structure itself. In this respect, there is much commonality between the HSML project and recent developments by Cellier *et al.* in the area of object-oriented modeling [3]; for a detailed view of state-event handling see especially [4, 5].

This paper outlines the completion of the first phase in implementing a subset of the HSML concept in a working modeling and simulation environment, MATLAB [6]. Preliminary steps are documented more completely in [7]. Here we again focus narrowly on the issues surrounding state-event handling in continuous-time components (CTCs) and provide an illustrative example to demonstrate the efficacy of the approach.

Here we consider CTCs that may be represented as[1]:

$$\begin{aligned} \dot{x_c} &= f_c(x_c, u_c, u_d, m, t) \\ y_c &= h_c(x_c, u_c, u_d, m, t) \end{aligned}$$

where x_c is the state vector, y_c is the output vector, u_c and u_d are numeric input signals (continuous- and discrete-time, respectively), m is comprised of a finite alphabet of numeric or symbolic input variables that characterizes the "mode" of the model, and t is the time; in general u_c, u_d and m are vectors. There are implicit "zero-order holds" operating on the elements of u_d and m, i.e., these inputs remain constant between those times when they change instantaneously. Of particular importance to the present exposition, the **mode** input m is included to provide means of controlling the model's structure and coordinating its behavior with the numerical integration process in state-event handling, as described below.

State events are characterized by *zero-crossings,*

$$S(x_c, m, t) = 0 \tag{1}$$

where S is a general expression involving the state, time and perhaps the mode of the CTC model. An arbitrary state change in the CTC model can be classified as a `negative-going` event (i.e., one in which S becomes negative), an `on-event` situation (S remains equal to zero for a time interval), or a `positive-going` event. Note that this framework provides support for models that undergo structural changes (e.g., changes in the definition or number of state variables) [9]; e.g., in the case of mechanical subsystems engaging, the number of states decreases. Finally, we include provision for instantaneous reset of the model state variables at an event, according to

$$x_c^+ = x_c(t_e^+) = r(x_c(t_e^-), m, t_e^-) \tag{2}$$

[1] The specific class of CTC that can be modeled depends on the simulator's integration methods; MATLAB cannot handle differential algebraic equations (DAEs), so we restrict ourselves to ordinary differential equations and simplify the variable types in comparison with [1, 2, 7].

where r is also an arbitrary expression and t_e is the event time. This feature is useful in resetting velocities after engagement to conserve momentum, for example.

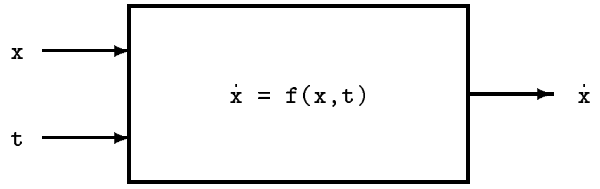
Given the above problem definition, the correct handling of state events is as follows:

1. The model should not be allowed to switch during a numerical integration step. Integrate as usual as long as the variable S does not change sign; each integration point is treated as a "trial" point until the sign condition is checked; if no sign change has occurred, the point becomes "accepted".
2. When a sign change is detected, the trial point is discarded and an iterative procedure is initiated (within the simulator) to find the time step h^* such that S has just passed zero, e.g., for a positive-going event $S \in (0, \epsilon)$. The model does not switch during this procedure.
3. The integrator produces an accepted point on the switching curve (Eqn. 1) and then signals the model to switch (e.g., by changing `mode` from -1 to 0 or $+1$ depending on the nature of the event).
4. States are reset, if needed, and normal integration proceeds from that point.

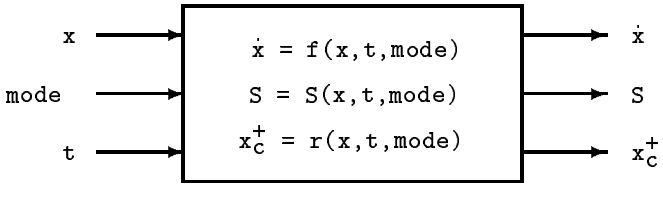
This requires coordination between the model and simulation package, as illustrated below using MATLAB extensions.

2 Extended Model Schema

One significant extension needed in MATLAB for modeling and simulating state events in CTCs is in the input/output structure of the model. The existing and extended schema are depicted in Fig. 1:



(a) Standard MATLAB model schema



(b) Extended MATLAB model schema

Figure 1: MATLAB model input/output structure

The additional outputs are the **state-event signal** S (Eqn. 1) and the **state-reset vector** x_c^+ (Eqn. 2); the new input `mode` allows the numerical integration routine to request that the model switch according to the state event just detected. Again, note that S and m (mode) may be vectors, to support multiple state events and switching boundaries.

3 Extended Integration Schema

A second significant extension must be made in the MATLAB numerical integration algorithms: neither those in MATLAB, i.e., `ode23` and `ode45`, nor those in SIMULINK, i.e., `gear`, `rk23` and `rk45`, can handle state events in the desired fashion. There are three features needed to permit the MATLAB integration routines to deal with state events:

1. the numerical integrator must coordinate with the extended model to set the initial value of `mode`,
2. it must continuously test for the occurrence of the event by watching for zero crossings in the `flag` variable(s), and
3. the routine should permit state variables to be reset at a state event, in a rigorously prescribed manner.

MATLAB code for such an integrator is omitted, to meet space limitations.

4 Example Application

The example used in testing the above integration approach is an extension of that in [7]. We have two uncoupled systems of the form:

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -\text{sign}(x_1) \end{aligned} \tag{3}$$

and similarly for x_3, x_4, with state events defined by

$$S_1 = x_1 \; ; \quad S_2 = x_3 \tag{4}$$

and a reset definition akin to the bounce of a ball with coefficient of restitution 0.8,

$$x_c(t_e^+) = \text{col}[x_1(t_e^-) \; 0.8 \times x_2(t_e^-) \; x_3(t_e^-) \; 0.8 \times x_4(t_e^-)] \tag{5}$$

Figure 2 depicts the results of running a 6.36396-second simulation with initial condition $x(0) =$ `[ 0.25 ; 0 ; 0 ; 0.8 ]` using an implementation of the above scheme with trapezoidal integration and state-event handling that is much more powerful than that reported in [7]. This artificial but challenging test case has the property that the time between switching in subsystem (x_1, x_2) becomes zero at 6.363961 seconds, so integrating correctly to $t_f = 6.36396$ is noteworthy. Note that we cannot compare integration times with standard MATLAB integrators in this example as we did in [7], because they cannot handle this problem at all.

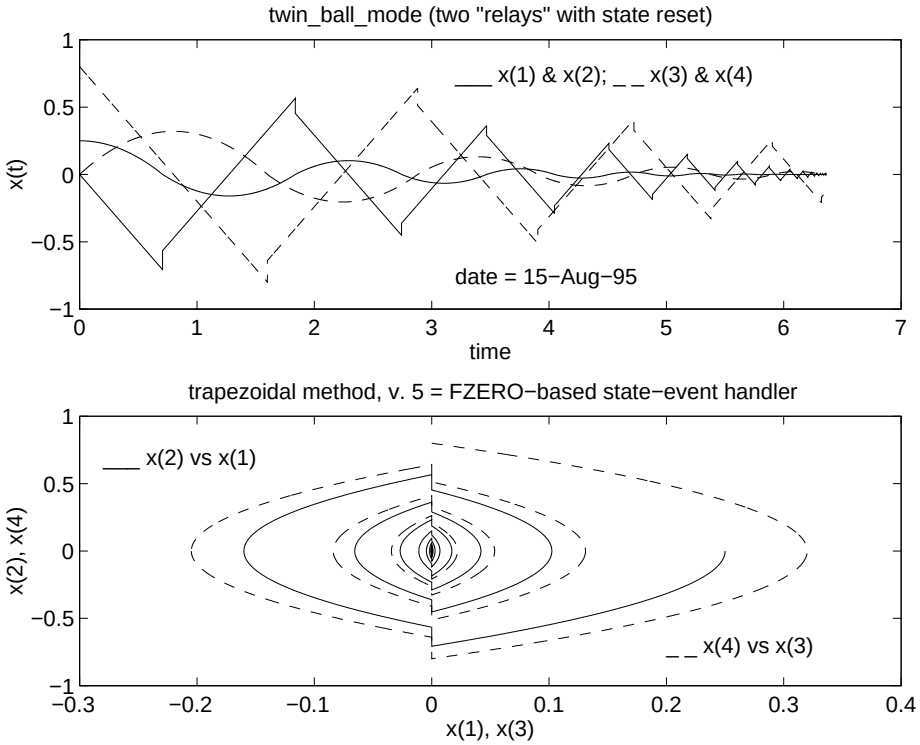


Figure 2: Numerical Integration Example

5 Conclusion

The methodology presented above provides a simple but compelling demonstration of HSML in general and of the the concept of state-event handling and its value for modeling and simulating switching systems in particular. The algorithms have been extended from [7] in several ways, i.e., they have been inserted into more sophisticated models and integration routines (like ode45), and they now support vector modes and switching functions for defining multiple state events in one model, including simultaneous ones. In addition, provision for state resetting at state events has been implemented.

References

[1] Taylor, J. H. "Toward a Modeling Language Standard for Hybrid Dynamical Systems", *Proc. 32nd* IEEE *Conference on Decision and Control,* San Antonio, TX, December 1993.

[2] Taylor, J. H. "A Modeling Language for Hybrid Systems", *Proc.* IEEE/IFAC *Symposium on Computer-Aided Control System Design*, Tucson, AZ, March 1994.

[3] Elmqvist, H., Cellier, F. E. and Otter, M., "Object-Oriented Modeling of Power-Electronic Circuits Using Dymola", *Proc. CISS'94* (First Joint Conference of International Simulation Societies), Zurich, Switzerland, August 1994.

[4] Cellier, F. E., Elmqvist, H., Otter, M. and Taylor, J. H., "Guidelines for Modeling and Simulation of Hybrid Systems", *Proc.* IFAC *World Congress*, Sydney Australia, 18–23 July 1993.

[5] Cellier, F. E., Otter, M. and Elmqvist, H., "Bond Graph Modeling of Variable Structure Systems", *Proc. ICBGM'95* (Second International Conference on Bond Graph Modeling and Simulation), Las Vegas, Nevada, January 1995.

[6] MATLAB *User's Guide*, The MathWorks, Inc., Natick, MA 01760.

[7] Taylor, J. H. "Rigorous Handling of State Events in MATLAB", *Proc. 4th (*IEEE *Conference on Control Applications,* Albany, NY, September 1995.

[8] SIMULINK *User's Guide*, The MathWorks, Inc., Natick, MA 01760.

[9] Taylor, J. H. *A Rigorous Modeling and Simulation Package for Hybrid Systems*, US National Science Foundation SBIR Report, Award No. III-9361232, Odyssey Research Associates, Inc., June 1994 (available only from the author).