Lecture 9

Computer Simulation of Hybrid Systems

Karl Henrik Johansson

This lecture discusses simulation of hybrid systems. Information about the simulation software OmSim is available on the homepage for the course. Before we discuss simulation of systems with mixed continuous and discrete dynamics, we give a brief overview of simulation including a short historical perspective.

Simulations are sometimes necessary due to lack of analytical tools. This was already pointed out by Vanevar Bush (1890–1974):

> *"Formal mathematics is frequently inadequate for numerous problems pressing for solution, and in the absence of radically new mathematics, a mechanical solution offers the most promising and powerful attack wherever a solution in graphical form is adequate for the purpose. This is usually the case in engineering problems."*

Bush invented the differential analyzer with which he could solve differential equations $\dot{x} = f(x)$ by mechanical integrators. Today *mechanical solution* should probably be replaced by the more appropriate *numerical solution* (or maybe *digital solution*).

One may make a distinction between general purpose simulation softwares and application-oriented simulation softwares. The former include softwares such as ACSL, Dymola, OmSim, SHIFT, Simnon, Simulink, and SystemBuild, which can be used regardless of domain. A great advantage of such tools is that it is easy to incorporate models from different areas, such as from electrical and mechanical systems. Application-oriented simulation softwares include ADAMS (mechanical systems), EMTDC (power systems), HVACSIM$^+$ (heat & vent), SpeedUp (chemical systems), SPICE (electrical circuits). They are efficient and easy to use for their specific application domain. However, it is difficult to combine models from different domains.

This lecture is restricted to general purpose simulation software. A historical summary is presented in the following table:

1930	Mechanical simulation	Bush
1947	Analog electronic simulation	Raggazzini et al.
1955	Analog computer-based simulation	DYNASAR (GE), CSMP (IBM)
1967	CSSL modeling standard	ACSL
1980	Block diagrams and GUI	Simulink, SystemBuild
1990	HS, OO, DAE	ABACUSS, Dymola, gPROMS, OmSim, SHIFT, StateFlow
1997	Standards?	Modelica

Simulation of Continuous Dynamical Systems

Consider the ordinary differential equation

$$\dot{x}(t) = F\big(x(t)\big), \qquad x(0) = x_0. \tag{1}$$

There are several methods to integrate this equation on a computer. The simplest one is the Euler method, which gives the following discretization

$$x_{k+1} = x_k + h_k F(x_k), \tag{2}$$

where the subscripts denotes the integration step and h_k the step size. The discretization gives an error $e_k = x(t_k) - x_k$ that depends on the step size, the round-off errors in computations, and possible initial error. If we neglect the latter two, we usually can make the accuracy arbitrarily good by choosing the step size small enough.

Proposition 1. *Consider the Euler method (2) with initial state x_0 and step size $h_k = t/k$. If F is globally Lipschitz continuous, then $x_k \to x(t)$ uniformly as $k \to \infty$.*

It is straightforward to show that the error of the Euler method behaves like h as $h \to 0$. therefore it is called a first-order method. There are rth-order methods for any $r > 1$, which hence behaves like h^r as $h \to 0$. This higher-order methods in general give a smaller error for the same size of integration step.

Probably the most popular numerical integration method is a fourth-order method and called the Runge-Kutta method. For example, the default integration routine in MATLAB, ode45, is a fourth-order Runge-Kutta. The discrete update for the classical Runge-Kutta method is given by

$$x_{k+1} = x_k + \frac{h_k}{6}(a_0 + 2a_1 + 2a_2 + a_3), \tag{3}$$

where

$$\begin{aligned} a_0 &= F(x_k), & a_1 &= F(x_k + ha_0/2) \\ a_2 &= F(x_k + ha_1/2), & a_3 &= F(x_k + ha_2). \end{aligned}$$

Note the similarities of the formula (3) with Simpson's rule

$$\int_{t_k}^{t_{k+1}} g(t)\,dt \approx \frac{h}{6}\big(g(t_k) + 4g(t_k + h/2) + g(t_{k+1})\big).$$

They agree if we consider $x \equiv t$.

Variable step size is needed in order to get efficient simulation. If (an estimate of) the error $e_k = x(t_k) - x_k$ is smaller than a certain value, the step size h_k is increased. If e_k is large, h_k is instead decreased. In general, this scheme leads to that when the solution of the differential equation does not change very much fewer steps are taken, while if there are abrupt changes in the solution many steps taken. The errors are compared to pre-specified values for the simulation parameters *absolute error* $\|e_k\|$ and *relative error* $\|e_k\|/\|x_k\|$. Default values of these are typically 10^{-6} and 10^{-3}, respectively. The latter hence saying that we accept an error of 0.1%.

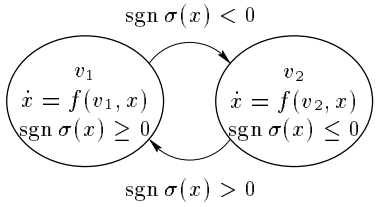

Fig. 1. Hybrid automaton to illustrate hybrid simulation.

Simulation of Hybrid Systems

A numerical simulation of a hybrid system gives a single execution accepted by the corresponding hybrid automaton. Therefore it is natural to consider only deterministic hybrid automata. Furthermore, we assume in general that the hybrid automaton also is non-blocking and non-Zeno. Sometimes Zeno hybrid automata can be dealt with in a constructive way by extending the Zeno execution beyond the Zeno time. This will be further discussed in next lecture.

Simulation of hybrid systems are based on simulation routines for continuous dynamical systems. As long as no discrete transitions take place, a conventional integration routine for continuous systems is employed. When the invariant is violated, a discrete mode change is done and a new continuous integration is initialized. The hybrid simulation can thus be divided into

1. Continuous simulation,
2. Discrete simulation.

To illustrate the idea, we consider the simple hybrid automaton in Figure 1, where $\sigma : \mathbb{R}^n \to \mathbb{R}$ is a smooth function. The diagrams in Figure 2 shows how an integration step is performed. The simulation algorithm can be summarized as follows, where we assume that current time is $t = t_k$ and that $v(t_k) = v_1$.

1. Solve $\dot{x} = f(v_1, x)$ from t_k to t_{k+1}
2. If $\sigma(x(t))$ crossed zero in (t_k, t_{k+1}) then
   Event location $t^* = \min\{t \in (t_n, t_{n+1}) : \sigma(x(t)) = 0\}$
   Event handling $v(t_{k+1}) := v_2$ and $t_{k+1} := t^*$

Consider the simple hybrid automaton in Figure 3. Figure 4 shows a comparison of conventional simulation and the hybrid simulation algorithm just described. The circles denote integration step. Even if the discrete transition in this example is quite nice, we notice that there is an accumulation of integration steps close to the jump for the conventional simulation routine. The situation is actually worse the illustrated by the figure, because all those steps that have been disregarded are not shown in the diagram. Hence, for the same accuracy, conventional simulation is usually much slower than hybrid simulation in simulating hybrid systems. Note that the hybrid simulation is a proper generalization of conventional simulation, in the sense that if there is no discrete transitions the simulation works with full efficiency.

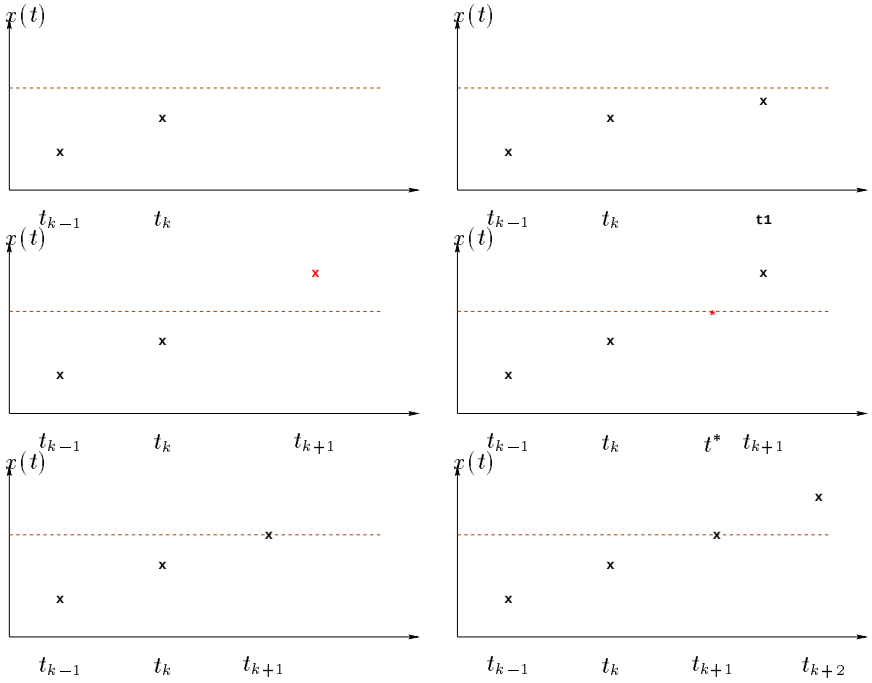

Fig. 2. Simulation of hybrid system step-by-step. The dashed line represents the boundary of the invariant, i.e., $\{\sigma(x) = 0\}$. Conventional continuous simulation is used until the invariant is violated. Then, a discrete simulation step is taken, where the discrete jump is located in time and the discrete state is updated. The continuous simulation is initialized at the jump instant. This instant is found by solving an algebraic equation.

Also other methods for simulation of hybrid systems have been suggested. One such method is the time-stepping method, which is basically a method with fixed step length. Of course, using a fixed step length when simulating hybrid systems has the same disadvantages as in the continuous case, for example, many integration steps even if the solution is behaves well.

Background

The historical reflections are based on [2]. Numerical integration of ordinary differential equations are discussed in [3]. Hybrid simulation is for instance discussed in [1].

References

1. Mats Andersson. *Object-Oriented Modeling and Simulation of Hybrid Systems.* PhD thesis, Department of Automatic Control, Lund University, Sweden, December 1994.

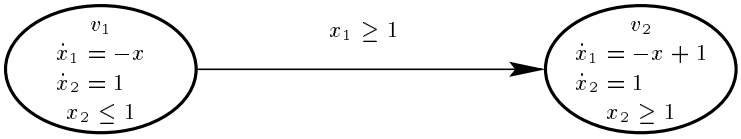

Fig. 3. Hybrid automaton with Init $= \{(v_1, 1)\}$ to illustrate conventional and hybrid simulations.

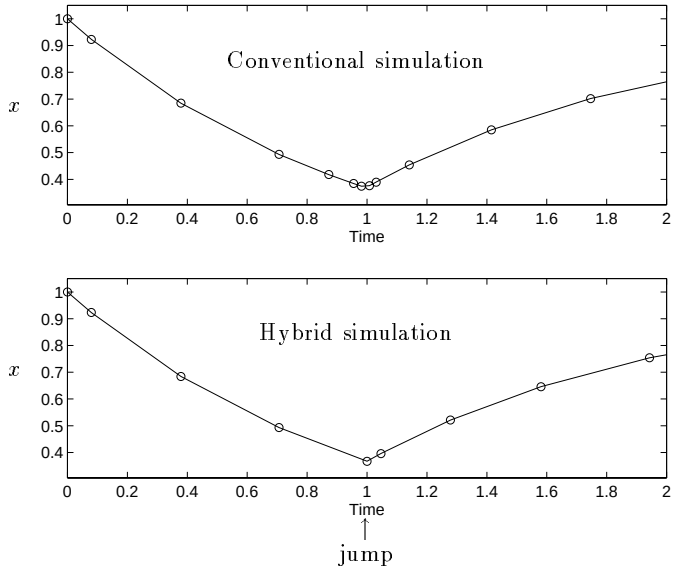

Fig. 4. Comparison of hybrid simulation and continuous simulation for the hybrid automaton in Figure 3.

2. K. J. Astrom, H. Elmqvist, and S. E. Mattsson. Evolution of continuous-time modeling and simulation. In *12th European Simulation Multiconference*, Manchester, UK, 1998.
3. C. W. Gear. *Numerical initial value problems in ordinary differential equations.* Prentice Hall, Englewood Cliffs, NJ, 1971.