A Symbolic Simulator for Hybrid Equations

Pablo Ñañez
Universidad de los Andes and University of Arizona
pa.nanez49@uniandes.edu.co

Nathalie Risso
University of Arizona
nrisso@email.arizona.edu

Ricardo G. Sanfelice[1]
University of Arizona
sricardo@u.arizona.edu

Department of Aerospace and Mechanical Engineering,
University of Arizona, 1130 N. Mountain Ave, AZ 85721.

Keywords: Symbolic simulation, hybrid systems, numerical simulation

Abstract

In this paper, the symbolic simulation of hybrid dynamical systems is studied and an algorithm to compute symbolic solutions for such systems is presented. The tasks to perform such simulations are introduced and an algorithm to symbolically calculate a solution to a hybrid system is presented. The symbolic representation allows the proposed simulator to calculate the actual solution to the system. Benefits and drawbacks of symbolic simulation with respect to the numerical approach are presented. These statements are supported and illustrated in several examples throughout the paper.

1. INTRODUCTION

During the last few decades, modeling and analysis of hybrid systems have been recognized as powerful tools that allow to represent systems with discrete and continuous behavior in a myriad of applications, ranging from air traffic control and thermal reactors to biological systems. Recent advances in the theory of hybrid systems have permitted the analysis of the dynamics of such systems in terms of stability, convergence, and robustness to perturbations [1].

Research targeting the computation of the trajectories of such systems has also been an active area, as the very many tools for simulation of hybrid systems indicate. Such tools include Modelica [2], the Hybrid Chi simulator [3], Shift [4], Charon [5], Hysdel [6], the Hybrid Equations (HyEQ) Toolbox [7], among others. Due to the inherent finite-precision representation of traditional numerical computations, these simulation tools produce approximations of the true system trajectories. Unlike differential equations, the computation of trajectories of hybrid systems presents several challenges, such as the selection of the size of the integration step, event detection and localization to enable transitions in the system, among others [8]. Such limitations make the detection of "failures" or "critical states" of such systems difficult, as the computed trajectory may not actually represent the true system behavior. In fact, finite-precision representation lacks the needed accuracy required to deal with safety relevant hybrid systems or can lead to pathological behaviors, such as those pointed out in [9], where the system's true dynamics are completely lost. To deal with some of the issues mentioned above, some computational tools include interval representation of variables to improve accuracy. Another approach used for hybrid system analysis is to represent their dynamics using specific programming languages. Many such implementations have been developed for this purpose, being those based on linear programming languages the most popular ones. For such implementations, hybrid dynamics are typically represented as constrained sets of equations, for whose it is possible to apply tools for reachable set analysis [10, 11, 12, 13, 14, 15, 16] and sensitivity analysis [17]; see also the software-based optimization methods in [18].

A different approach to overcome the issues due to finite precision consists of including symbolic solvers, such as Mathematica and Matlab's MuPad, that can be used to generate solutions to hybrid systems. While such approach has been deeply explored for the simulation of differential equations, their application to the hybrid setting has been presented in most cases as isolated functions, rather than a hybrid simulation tool. In fact, purely symbolic simulation tools for generic hybrid systems are not available. Symbolic tools for specific types of hybrid systems, such as linear hybrid systems have been presented in [19, 20] though they involve finite-precision computations, hence leading to the previously described accuracy issues.

Motivated by the need to automatically analyze hybrid systems and their solutions, we propose a simulation framework that represents a first step towards a symbolic simulation and verification tool for hybrid systems. The proposed tool consists of a set of scripts that, using Matlab's symbolic engine (MuPad), compute exact symbolic solutions for hybrid system equations, leading to an explicit representation of the trajectories, with no finite-precision limitations, which are more suitable for the analysis of system properties. The scope of systems whose solutions can be simulated with the proposed tool depends, in particular, on whether the symbolic engine can find a closed-form expression for the solutions during flows.

The remainder of this paper is organized as follows, Sec-

[1]Research by R. G. Sanfelice has been partially supported by the National Science Foundation under CAREER Grant no. ECS-1150306 and by the Air Force Office of Scientific Research under YIP Grant no. FA9550-12-1-0366.

tion 2 describes the type of hybrid systems considered here. Section 3 presents as motivational example, a modified version of the bouncing ball system, so as to illustrate the effect of finite-precision arithmetic over hybrid systems solutions. Pseudocode implemented for the proposed symbolic simulator and its characteristics are presented in Section 4, which is followed in Section 5 by the demonstration of the features of the symbolic simulator in two examples.

All the files scripts required to replicate the results of this paper are available at the author's website

http://www.u.arizona.edu/∼sricardo/index.php?n=Main.Software.

2. PRELIMINARIES

Making use of the dynamical properties of hybrid systems modeled as in [21, 22] and the framework of their simulation in [1, 7], we present a simulation scheme that exploits the advantages of the symbolic representation of solutions to these systems. In this paper, a hybrid system $\mathcal{H}$ is given by the *hybrid equations*

$$\mathcal{H}: \; x \in \mathbb{R}^n \begin{cases} \dot{x} &= f(x) & x \in C \\ x^+ &= g(x) & x \in D \end{cases} \tag{1}$$

with the following objects defining its *data*: the set $C \subset \mathbb{R}^n$ called the *flow set*; the function $f : \mathbb{R}^n \to \mathbb{R}^n$ called the *flow map*; the set $D \subset \mathbb{R}^n$ called the *jump set*; the function $g : \mathbb{R}^n \to \mathbb{R}^n$ called the *jump map*. These objects define the *data* of $\mathcal{H}$, which is explicitly denoted as $\mathcal{H} = (f, C, g, D)$.

The description in (1) supports the following natural notion of solution. During flows, a solution to $\mathcal{H}$ satisfies the continuous dynamics

$$\dot{x} = f(x) \quad x \in C \tag{2}$$

over the intervals of flow while, at jumps, it satisfies the discrete dynamics

$$x^+ = g(x) \quad x \in D. \tag{3}$$

To make this notion precise, recall that solutions to continuous-time systems are typically parameterized by $t \in \mathbb{R}_{\geq 0} := [0, \infty)$ (e.g., by ordinary time). On the other hand, solutions to discrete time systems are typically parameterized by $j \in \mathbb{N}$ (e.g., by the number of steps or jumps), where $\mathbb{N} := \{0, 1, 2 \ldots\}$. Combining these two ways of parameterizing time, solutions to $\mathcal{H}$ are parametrized by $(t, j) \in \mathbb{R}_{\geq 0} \times \mathbb{N}$ and are defined on hybrid time domains.

Definition 2.1 (hybrid time domain). *A subset $E \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ is a compact hybrid time domain if*

$$E = \bigcup_{j=0}^{J-1} ([t_j, t_{j+1}], j) \tag{4}$$

for some finite sequence of times $0 = t_0 \leq t_1 \leq \ldots \leq t_J$. It is a hybrid time domain if for all $(T, J) \in E$, $E \cap ([0, T] \times \{0, 1, \ldots, J\})$ is a compact hybrid time domain.

To formally define the notion of solutions to $\mathcal{H}$, first we define hybrid arcs.

Definition 2.2 (hybrid arc) *A function $\phi : E \to \mathbb{R}^n$ is a hybrid arc if E is a hybrid time domain and if, for each $j \in \mathbb{N}$, the function $t \to \phi(t, j)$ is locally absolutely continuous on the interval $I_j := \{t : (t, j) \in E\}$.*

A hybrid arc is a solution to a hybrid system $\mathcal{H}$ as in (1) if it satisfies certain conditions given by the system's data and its hybrid time domain $\operatorname{dom}\phi$, as the following definition states.

Definition 2.3 (solution to a hybrid system) *A hybrid arc $\phi : \operatorname{dom}\phi \to \mathbb{R}^n$ is a solution to the hybrid system $\mathcal{H}$ if $\phi(0,0) \in \bar{C} \cup D$, and:*

(S1) (Flow condition) For all $j \in \mathbb{N}$ and all $t \in int(I_j)$[2],

$$\phi(t, j) \in C, \; \dot{\phi}(t, j) = f(\phi(t, j)), \tag{5}$$

(S2) (Jump condition) For all $(t, j) \in \operatorname{dom}\phi$ such that $(t, j+1) \in \operatorname{dom}\phi$,

$$\phi(t, j) \in D, \; \phi(t, j+1) = g(\phi(t, j)) \tag{6}$$

3. MOTIVATIONAL EXAMPLE

Consider a variation of the widely known bouncing ball example in hybrid systems literature (see, e.g., [22, Example S4]). This hybrid system has state $x \in \mathbb{R}^2$ and data

$$f(x) := \begin{bmatrix} x_2 \\ -\gamma \end{bmatrix}, \qquad C := \left\{ x \in \mathbb{R}^2 \mid x_1 \geq 0 \right\} \tag{7a}$$

$$g(x) := \begin{bmatrix} 0 \\ -\lambda x_2 \end{bmatrix}, \quad D := \left\{ x \in \mathbb{R}^2 \mid x_1 \leq 0, \; x_2 \leq 0 \right\} \tag{7b}$$

where $\gamma > 0$ is the gravity constant and $\lambda \in [0, 2]$ is the restitution coefficient. The state x_1 corresponds to the height of the ball from the floor and x_2 represents the speed of the ball (negative when falling). This system models a ball bouncing on a floor at zero height. Now, consider the case when the restitution coefficient λ is a function of the speed of the ball, namely $\lambda : \mathbb{R} \to [0, 2]$. In particular, we consider the restitution coefficient given by the function $\lambda(x_2) = 1 + \sin(-x_2/\gamma)$.

We select a particular initial condition such that the position x_1 of the system is periodic. For example, fix the initial speed at zero $x_2(0,0) = 0$. Then, we can solve for the initial position $x_1(0,0)$ such that the ball hits the floor at t_1 when $\sin(-x_2(t_1, 0)/\gamma) = 0$. In such case, $\lambda(x_2(t_1, 0)) = 1$, which implies that its energy does not change at impacts. Then, the ball takes $\tau = -x_2(t_1, 0)/\gamma$ seconds to reach its initial height and another τ seconds to hit the floor for the second time at $t_2 = 3\tau$. It is easy to see that the subsequent impacts occur at $(2n+1)\tau$, $n = 0, 1, 2, \ldots$ and the ball bounces periodically

[2] $int()$ denotes the interior of the set.

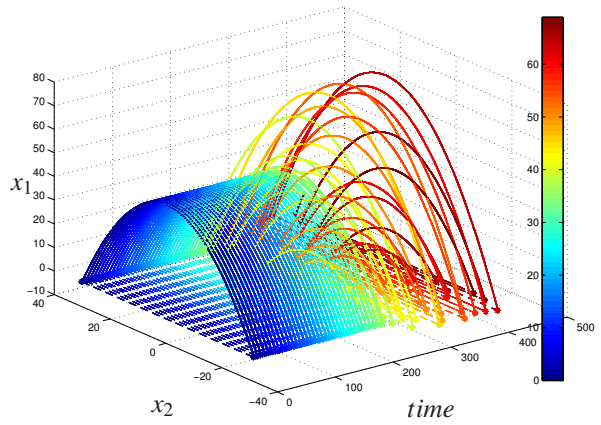


Figure 1. Solution of the system obtained with a numerical simulator. The color scale represents the number of jumps.

indefinitely. For example, for $\gamma = 9.81$, one such initial condition is given by

$$x_1(0,0) = (981\pi^2)/200; \qquad x_2(0,0) = 0$$

To simulate equations (7), which represent the bouncing ball system described previously, we can use several tools such as Modelica [2], Shift [4], Charon [5], Hysdel [6], among others. These simulators perform numerical calculations to solve the differential equations during flows, and evaluate the maps that reset the states at jumps. Particularly, we are going to solve this initial value problem using the Hybrid Equations (HyEQ) Toolbox, which is described in [7]. After coding the hybrid system data into the simulator, we obtain the numerical solution in Figure 1[3].

Due to the chosen initial conditions, the ball should hit zero when $\sin(x_2/\gamma) = 0$, which is when x_2/γ is a multiple of $(2n+1)\pi$. However, since the numerical solution is not exact at the first impact time, which should occur at π seconds, the ball bounces with restitution coefficient close, but not exactly equal to 1. This error is generated by the finite-precision representation of $\sin(x_2/\gamma)$. As a consequence, the subsequent impacts do not occur at exactly $(2n+1)\pi$ seconds. The accumulation of these errors generate the trajectory shown in Figure 1, which differs from the actual solution to the system; namely, the approximated numerical solution does not match the true solution for t (and j) large enough as the induced error accumulates and the bouncing loses its periodic behavior. Notice that, due to the finite-precision representation introduced by every numerical ODE solver (no matter how small the precision/tolerance or the time step selection is), the obtained approximated numerical solution will differ from the true solution of the system for $t + j$ large enough.

[3] In order to solve numerically the differential equations, the Matlab's ODE45 was used with relative error tolerance "RelTol" equal to 1e-26 and max step size "MaxStep" equal to 0.01.

Using the framework presented in the following sections, the solutions to hybrid systems will be represented as symbolic expressions. With this new simulation approach, the impact times will be computed using a symbolic expression of the solution of the differential equations in (1). This symbolic representation does not induce finite-precision perturbations in the computation of the impact time, yielding a simulation that, for the particular case of the bouncing ball system above, maintains the expected periodic property. See Example 5.2 for more details.

4. A SYMBOLIC SIMULATOR FOR HYBRID SYSTEMS

In this section, we describe the algorithm developed to perform symbolic simulation of hybrid systems as in (1). First, notice that the jump and flow sets in (1) may have overlaps. In such case, the hybrid system may have multiple solutions (e.g., a solution that flows and a solution that jumps at the same hybrid time). For simulation purposes, we restrict this feature of the hybrid systems in [21] with a priority rule when C and D overlap. In other words, for points in $C \cap D$, a priority rule determines which solution is going to be simulated (the flow solution or the jump solution). For further details on how the overlaps of C and D can be analyzed, and how different "semantics" can be enforced modifying the data of the hybrid system (1), please refer to [1].

We will assume that the flow and jump sets C and D can be represented by a union of sets C_k and D_k, respectively, where each one of these sets can be described by a collection of finitely many inequalities. Under this assumption, there exist functions $c_k : \mathbb{R}^n \to \mathbb{R}^m$ and $d_k : \mathbb{R}^n \to \mathbb{R}^p$ such that

$$C_k := \{x \in \mathbb{R}^n | c_k(x) \geq 0\}, \quad D_k := \{x \in \mathbb{R}^n | d_k(x) \geq 0\} \quad (8)$$

where $c_k(x) \geq 0$ and $d_k(x) \geq 0$ denote that each component of $c_k(x)$ and $d_k(x)$ is larger or equal than zero. Consequently, given

$$C := \bigcup_{k=1}^{K_C} C_k, \quad D := \bigcup_{k=1}^{K_D} D_k \quad (9)$$

for some K_C and K_D[4], if x^* belongs to the set C then $c_{k^*}(x^*) \geq 0$ for some k^*. Similarly for the jump set D. The sets in (8) are employed in the simulation to determine when the computed solution leaves or reaches the flow/jump sets of the hybrid system. It is assumed that the flow and jump sets can be written as a finite collection of inequalities. Note that the resulting sets C and D in (9) might have overlaps.

To perform symbolic simulation for hybrid systems as in (1), we need to perform the following four tasks:

[4] K_C and K_D values are given by the number of sets C_k and D_k required to describe C and D, respectively.

i) Compute a closed-form expression of the solution to initial value problems for $\dot{x} = f(x)$. In other words, solve symbolically the ODE in (1).

ii) Detect when and where the solution of the system reaches and/or leaves the sets C and/or D. In other words, determine the time interval (which we denote $[t_{j+1}, t_k^C]$) where the closed-form expression of the solution to the flow map belongs to C and the amount of time (which we denote τ_k) to reach the jump set D (namely and the inequalities that describe the flow and jump sets $c_k(x) \geq 0$ and $d_k(x) \geq 0$ from Equations (8), respectively are satisfied).

iii) Compute the value of the solution after jumps given by the evaluation of the jump map g.

iv) Given a priority for jump or flow and the intervals computed in *ii*), select and construct the hybrid time domain and the hybrid arc that define a solution to the hybrid system.

Definition 4.1 (Initial value problem and its solution) *An initial value problem is given by an ODE together with a initial condition: Consider the ODE given by $\dot{x} = f(x)$, where $f : C \to \mathbb{R}^n$ is continuous and $C \subset \mathbb{R}^n$, and the initial condition $(x_0, t_0) \in C \times \mathbb{R}_{\geq 0}$. A solution to this problem is given by a continuously differentiable function $\xi : [t_0, \infty) \to \mathbb{R}^n$ satisfying $\dot{\xi}(t) = f(\xi(t))\ \forall t \geq t_0$ and $\xi(t_0) = x_0$.*

In this paper, we assume the existence of a closed-form solution for all the initial value problems derived from the flow map f and the set C.

Assumption 4.2 (Existence of closed-form solutions during flows) *Consider the initial value problem (IVP) given by the flow map $\dot{x} = f(x)$ and the initial condition $x_0 \in C$. Assume that f is continuously differentiable and a solution $\xi : [0, \infty) \to \mathbb{R}^n$ for the initial value problem in closed-form[5] exists for all $x_0 \in C$.*

In other words, Assumption 4.2 assumes that the automatic symbolic solver returns a solution of the differential equation given by the flow map f and the initial condition x_0, for all $x_0 \in C$.

In Algorithm 1 below, we present the pseudocode proposed to solve the four tasks to perform symbolic simulation. The pseudocode is divided in four procedures. First, the variables are introduced in procedure *a*). Then, the hybrid time domain and hybrid arc are initialized in *b*). In the third procedure, procedure *c*), the jump function is introduced. In procedure *d*), the main simulation loop is presented.

[5]Here, we are going to use the term closed-form derived from the set of well-known functions by the symbolic engine solver Matlab/MuPad.

Example 4.3 (pseudocode example) Consider a hybrid system as in (1), assume that a solution ϕ flows during the interval t_0 to t_1. Also, a jump is performed when $t = t_1$. Assume that $x_0 \in C$ and $x_0 \notin D$ and Assumption 4.2 holds. Following the pseudocode presented in Algorithm 1, it is important to highlight the following computations:

1. Compute the closed-form expression of $\xi : [t_0, \infty) \to \mathbb{R}^n$ for the initial value problem ($\dot{x}(t) = f(x)$ and $x(t_0) = x_0$) such that $\phi(t, 0) = \xi(t),\ t_0 \leq t \in I_0$ **(line 43)**.
2. For each $k \in \{1, \ldots, K_C\}$, compute the largest time interval $[t_0, t_k^C]$ such that $c_k(\xi(t)) \geq 0$, and $t_0 \leq t \leq t_k^C$ **(line 49)**.
3. For each $k \in \{1, \ldots, K_D\}$, compute the smallest time τ_k such that $d_k(\xi(\tau_k)) \geq 0$, and $t_0 \leq \tau_k$ **(line 55)**.
4. Given the jump/flow priority, create a new time interval $[t_0, t_1]$ for the solution domain
   - If jump is prioritized then the new time interval has **(line 60)**
   $$t_1 = \min\left(\max_{k=1,\ldots,K_C}(t_k^C), \min_{k=1,\ldots,K_D}(\tau_k)\right) \tag{10}$$
   - If flow is prioritized then the new time interval has **(line 63)**
   $$t_1 = \max_{k=1,\ldots,K_C}(t_k^C) \tag{11}$$
5. At the end of the last interval a jump might be performed since $\phi(t_1, 0) \in D$. Consequently, add a new interval to the hybrid time domain to represent the jump at $(t_1, 1)$. Also, add the function $\phi(t_1, 1) = g(\phi(t_1, 0))$ to the hybrid arc **(lines 70 and 74)**.

□

The Algorithm 1 is inspired by the structure of the numerical simulator *HyEQsolver* presented in the Hybrid Equations (HyEQ) Toolbox [7]. The Algorithm 1 is implemented in a Matlab script as a function called *HyEQSimbolicSolver* (not part of the Hybrid Equations (HyEQ) Toolbox).

5. EXAMPLES

Example 5.1 (a hybrid system with a jump map with two equilibrium points) Following [9, Example 2], consider the hybrid system with state $x \in \mathbb{R}^2$ given by the hybrid equation with data

$$f(x) := \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \forall x \in C \tag{12}$$

$$g(x) := \begin{bmatrix} x_2 \\ 111 - \left(1130 - \frac{3000}{x_1}\right)\frac{1}{x_2} \end{bmatrix} \quad \forall x \in D \tag{13}$$

Algorithm 1 HyEQ Symbolic Simulator

procedure *a*) VARIABLES DEFINITION. ▷ Simulation data
  $\mathcal{H} := (f, C, g, D)$ ▷ hybrid system data
  rule ▷ priority rule, jump or flow,
  t ▷ time, $t \in \mathbb{R}_{\geq 0}$,
  j ▷ number of jumps,
  $I_j := [t_j, t_{j+1}]$ ▷ time interval,
  t_j ▷ initial time of interval I_j,
  t_{j+1} ▷ final time of interval I_j,
  (I_j, j) ▷ hybrid time domain,
  $\phi : (I_j, j) \to \mathbb{R}^n$ ▷ symbolic function for the interval I_j,
  $\xi : [t_j, \infty) \to \mathbb{R}^n$ ▷ symbolic function,
  x_0 ▷ initial condition,
  $[0, T]$ ▷ simulation time span,
  $[0, J]$ ▷ simulation jump span.
  ▷ The intervals I_j are calculated one interval at the time. The initial time t_j for the interval I_j is equal to the final time for the previous interval I_{j-1} (for $j = 0$, it is the initial time t_0). The final time t_{j+1} for the interval I_j is calculated symbolically given an initial value problem and the inequalities that describe C and D.
end procedure

procedure *b*) HYBRID TIME DOMAIN (I_j, j) AND HYBRID ARC ϕ INITIALIZATION
  $I_j = [0, 0],\ j = 0$ ▷ initial hybrid time domain
  $\phi(I_j, j) = x_0$ ▷ symbolic function (constant)
end procedure

procedure *c*) CHECK FOR JUMPS (and jump function)
  if rule = jump prioritized **then**
    while $j < J$ and $k \neq -1$ **do**
      function JUMP($d_k, \phi(t_{j+1}, j)$)
        k=-1
        for all $d_k,\ k \in \{1, \dots K_D\}$ **do** ▷ for all D_k
          if $d_k(\phi(t_{j+1}, j)) \geq 0$ **then** ▷ $\phi(t_{j+1}, j) \in D$
            $k^* = k$
            break for loop ▷ end for loop
          end if
        end for
        if $d_{k^*}(\phi(t_{j+1}, j)) \geq 0$ **then** ▷ compute a jump
          $\phi(I_{j+1}, j+1) = g(\phi(t_{j+1}, j))$
          $I_{j+1} = [t_{j+1}, t_{j+1}]$ ▷ create a time interval
          $j = j + 1$ ▷ add a jump
        end if
      end function
    end while
  end if
end procedure

procedure *d*) MAIN SIMULATION LOOP
  while $j < J, t_{j+1} < T$ **do**
    $\xi \leftarrow$ Solve IVP $\dot{x} = f(x),\ x(t_{j+1}) = \phi(t_{j+1}, j),\ t \geq t_{j+1}$ ▷ IVP symbolic solver
    for all $c_k,\ k \in \{1, \dots K_C\}$ **do** ▷ for all C_k sets ▷ check if it is possible to flow
      if $c_k(\phi(t_{j+1}, j)) \geq 0$ **then** ▷ if it can flow, ▷ then solve next equations together
        Try: $[t_{j+1}, t_k^C] \leftarrow$ solve $(c_k(\xi(t)) \geq 0,\ t \geq t_{j+1})$ ▷ symbolic solver
      end if
    end for
    for all $d_k,\ k \in \{1, \dots K_D\}$ **do** ▷ for all D_k sets, ▷ compute time to reach D
      Try: $\tau_k \leftarrow$ solve $(d_k(\xi(t)) \geq 0,\ \tau_k \geq t_{j+1})$ ▷ symbolic solver
    end for
    if any flow is possible **then** ▷ $t_{k^*}^C$ exists for some k^*
      if rule = jump prioritized **then**

$$I_{j+1} = \left[t_{j+1}, \min\left(\max_{k=1,\dots,K_C} (t_k^C), \min_{k=1,\dots,K_D} (\tau_k) \right) \right]$$

        ▷ it create a time interval
      else if rule = flow prioritized **then**

$$I_{j+1} = \left[t_{j+1}, \max_{k=1,\dots,K_C} (t_k^C) \right]$$

        ▷ it create a time interval
      end if
      $\phi(I_j, j) = \xi$ ▷ add ξ to the solution ϕ
    end if
    if rule = jump **then** ▷ jump as many times as possible
      while $j < J$ and $k \neq -1$ **do**
        function JUMP($d_k, \phi(t_{j+1}, j)$)
        end function
      end while
    else if rule = flow **then** ▷ jump only once
      function JUMP($d_k, \phi(t_{j+1}, j)$)
      end function
    end if
    if cannot flow or jump from $\phi(t_{j+1}, j)$ **then**
      end while ▷ $\phi(t_{j+1}, j) \notin C \cup D$
    end if
  end while
end procedure

where

$$C := \{x \in \mathbb{R}^2 \mid x_1 \leq a_1,\ x_2 \leq a_2\} \tag{14}$$

$$D := \{x \in \mathbb{R}^2 \mid x_1 \geq a_1\} \cup \{x \in \mathbb{R}^2 \mid x_2 \geq a_2\} \tag{15}$$

for some constants a_1 and a_2, which define the shape of the sets C and D. We can express the set C and D in (12) and (13) as a set of inequalities as in (8). Thus, $K_C = 1,\ K_D = 2$ and

$$c_1(x) := -x + \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{16}$$

$$d_1(x) := \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x - \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad d_2(x) := \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} x - \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{17}$$

Notice that the jump map has an interesting feature. If x_1 and x_2 are equal to 6 (which we denote α^*) the function g maps to the same point in $\mathbb{R}^2$. However, a small perturbation around this point leads to consecutive jumps that get the state away from α^* and steer it to the point $x_1 = 100,\ x_2 = 100$ (which we denote ω^*), which is also an equilibrium point of g.

Even more, there is a sequence of numbers which may be computed with the difference equation given by $\chi(k+1) = g(\chi(k))$, that, when computed using real numbers, approaches α^* (e.g., if the difference equation given by $\chi(k+1) = g(\chi(k))$ is initialized at $\chi_1(0) = 11/2$ and $\chi_2(0) = 61/11$, the obtained sequence of numbers $\chi(k)$ tends to α^* [9, Example 2]). These interesting features are due to the stability properties of the jump map[6].

Now, let us consider two solutions for two sets of parameters a_1, a_2 and two different initial conditions:

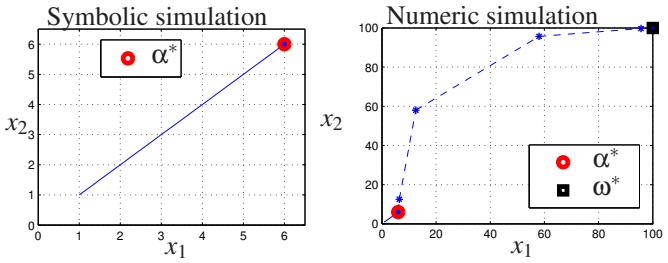
i) Take $a_1 = a_2 = 6$, and the initial conditions given by $x_1(0,0) = 1$ and $x_2(0,0) = 1$.

ii) Take $a_1 = 11/2,\ a_2 = 61/11$, and the initial conditions given by $x_1(0,0) = 11/2$ and $x_2(0,0) = 6/11$.

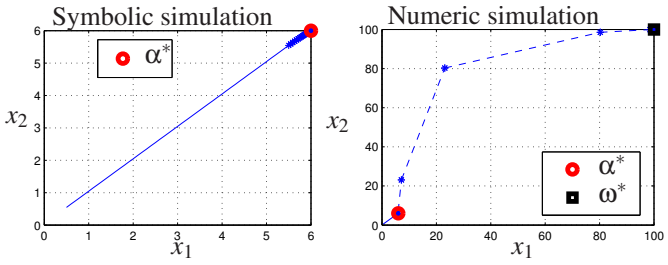
For both sets of parameters and initial conditions, due to the dynamics of the hybrid system, it is expected that the solution flows for 5 seconds. Then, for the i) case, the solution reaches the point α^* and keeps jumping at this point. For the second set of parameters, after flowing for 5 seconds the solution reaches the jump set at $x_1(5,0) = 11/2$ and $x_2(5,0) = 61/11$. Then, the solution jumps towards the point α^*.

With this a priori knowledge of the system's solutions, we solve the hybrid system for both sets of parameters using a numeric hybrid systems simulator [7] and with the symbolic framework presented in this paper. We show the obtained solutions in Figures 2(a) and 2(b). When the solution is computed with the finite precision framework, no matter which precision is used, an approximated value at the end of the flow of the solution is obtained. This small perturbation drives the sequences of jumps given by g far away from α^* and towards ω^* for parameters as in i) and ii).

When computed with the symbolic framework, the end point of the flow part of the solution is given by a symbolic expression. Then, the sequence of jumps given by g converges towards the point α^* for the ii) case and remains in the point α^* for the i) case.



(a) Solution of the system in Example 5.1, case i).



(b) Solution of the system in Example 5.1, case ii).

Figure 2. Simulation results using symbolic and numeric simulators for the system in Example 5.1 □

Example 5.2 (Bouncing ball with speed dependent restitution coefficient.) We now revisit the bouncing ball in Section 3, with state $x \in \mathbb{R}^2$ given by the hybrid system with data in (7a) and (7b). Notice that the flow set C and jump set D can be expressed, according to (9), by a set of inequalities as in (8). Thus, $K_C = K_D = 1$ and

$$c_1(x) := \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x, \qquad d_1(x) := -x \tag{18}$$

Considering restitution coefficient λ as a function of the speed of the ball, as presented in Example 3, and initial condition given by $x_1(0,0) = (981\pi^2)/200$ and $x_2(0,0) = 0$, results obtained with the symbolic simulator framework are presented in Figure 3. Now, since calculations of the ball impacts are performed using the symbolic representation, the restitution coefficient remains unitary leading to the expected periodic bouncing behavior. □

[6] It is possible to show that the equilibrium point α^* is not Lyapunov stable and the equilibrium point ω^* is Lyapunov stable.

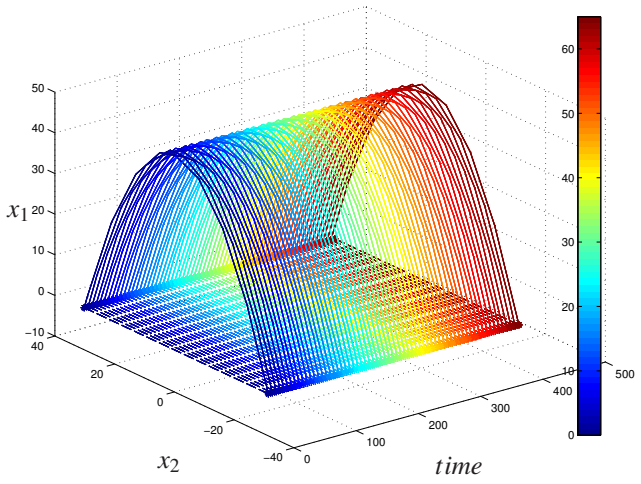


Figure 3. Solution of the bouncing ball system in Example 3 obtained with the symbolic simulator. The color scale represents the number of jumps.

6. DISCUSSIONS

Some of the issues regarding hybrid systems simulation, for cases where numerical precision is critical for accurate system representation, as portrayed in Example 5.1, are solved by the proposed symbolic simulation tool, with no finite-precision perturbations introduced, leading to exact and explicit solutions for the hybrid system. In this way, we extended previous ideas on symbolic simulation under the framework developed in [1, 7].

However, although the symbolic simulator proposed here delivers precise mathematical results, the CPU time required to perform symbolic calculations is still by far larger than the CPU time required by numerical tools. We ran a performance test for each simulator (numeric HyEQsolver and a Matlab script based on algorithm HyEQSymbolicSolver presented in this paper) which consisted of 5000 simulations. For this performance test, we used the previously described bouncing ball example with initial conditions as stated on Example 5.2, but considering now a constant value for λ. The performance test lead to an average execution time of 2.5411 seconds, with a standard deviation of 0.0532 seconds, for the symbolic simulator. Same test performed using the numerical HyEQ simulator reached average simulation time of 0.0351 seconds, with standard deviation of 0.002 for the same system[7].

It should be recognized that there exist some restrictions for the types of systems that can effectively be solved symbolically. In particular, for this work, the symbolic simulation framework employed relies on the symbolic engine solver in Matlab R2013a (MuPad v5.10.0). Consequently, the family of ODE equations that MuPad is capable to solve constraints the family of hybrid systems that can be analyzed with the framework proposed here. According to [23], the symbolic engine is capable to solve the type of ODEs listed in Table 1[8] and some nonlinear equations. While [23] does not provide conditions that the functions f and g need to satisfy, it is expected that symbolic solutions are only possible for a narrow class of such functions.

Table 1. ODE automatically recognized by MuPad

Abel DE	$\frac{dy}{dt} = a_0(t) + a_1(t)y + a_2(t)y^2 + a_3(t)y^3$
Bernoulli DE	$\frac{dy}{dt} + p(t)y = q(t)y^n,\ n \neq 1,\ n \neq 0$
Chini DE	$\frac{dy}{dt} = a_0(t) + a_1(t)y + a(t)y^n$
Clairaut DE	$y = t\frac{dy}{dt} + g\left(\frac{dy}{dt}\right)$
Exact first-order ODE	$\frac{dy}{dt} = f(t,y)$
Exact second-order ODE	$\frac{d^2y}{dt^2} = f\left(t, y, \frac{dy}{dt}\right)$
Linear homogeneous ODE	$\frac{d^ny}{dt^n} + a_1(t)\frac{d^{n-1}y}{dt^{n-1}} + \ldots + a_{n-2}(t)\frac{dy}{dt} + a_n(t)y = 0$
Lagrange DE	$y = tf\left(\frac{dy}{dt}\right) + g\left(\frac{dy}{dt}\right)$
Riccati DE	$\frac{dy}{dt} + p(t)y = q(t)y^2 + r(t)$

7. CONCLUSIONS

In this paper, symbolic simulation for hybrid systems is studied. The tasks to perform such simulations are formulated and an algorithm to calculate a solution to a hybrid system as in (1) is presented. Making use of two examples, the effect of perturbations induced by the finite-precision representation in numeric solvers is shown. For such systems, the effect of this perturbation makes the solution to be far different from the true solution. In two examples, we show that with the symbolic simulation approach proposed in this paper, solutions obtained match the analytical solution. Also, restrictions of this approach are discussed and the benefits and drawbacks compared to the numerical approaches are presented. The simulation framework presented here may be useful towards the development of integrated simulation and verification tools for hybrid systems.

REFERENCES

[1] R. G. Sanfelice and A. R. Teel, "Dynamical properties of hybrid systems simulators," *Automatica*, vol. 46, no. 2, pp. 239–248, 2010.

[2] V. Sanz, A. Urquia, F. E. Cellier, and S. Dormido, "Modeling of hybrid control systems using the DEVS-

[7] Experiments ran on a 64 bit Intel Core $i5-4200$ ($3M$ cache, 2.6 $[Ghz]$) processor, with 8 Gb of RAM, under Windows 8 operative system, using Matlab R2013a.

[8] DE stands for differential equation and ODE for ordinary differential equation.

Lib Modelica library," *Control Engineering Practice*, vol. 20, no. 1, pp. 24–34, 2012.

[3] G. Fábián, D. Van Beek, and J. Rooda, "Integration of the discrete and the continuous behaviour in the hybrid chi simulator," in *1998 European Simulation Multiconference, Manchester*, 1998, pp. 252–257.

[4] M. Antoniotti and A. Gollu, "SHIFT and SMART-AHS: A language for hybrid system engineering modeling and simulation," in *conference on domain specific languages*, 1997, pp. 171–182.

[5] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee, "Modular specification of hybrid systems in charon," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, vol. 1790, pp. 6–19.

[6] F. Torrisi and A. Bemporad, "Hysdel-a tool for generating computational hybrid models for analysis and synthesis problems," *Control Systems Technology, IEEE Transactions on*, vol. 12, no. 2, pp. 235–249, March 2004.

[7] R. G. Sanfelice, D. A. Copp, and P. Nanez, "A toolbox for simulation of hybrid systems in Matlab/Simulink: Hybrid Equations (HyEQ) Toolbox," in *Proceedings of Hybrid Systems: Computation and Control Conference*, 2013, pp. 101–106.

[8] J. Lunze and F. Lamnabhi-Lagarrigue, *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. Cambridge University Press, 2009.

[9] E. Goubault and S. Putot, "Static analysis of finite precision computations," in *Verification, Model Checking, and Abstract Interpretation*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6538, pp. 232–247.

[10] C. Tomlin, I. Mitchell, A. Bayen, and M. Oishi, "Computational techniques for the verification of hybrid systems," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 986–1001, July 2003.

[11] Z. Yang, M. Wu, and W. Lin, "Exact safety verification of interval hybrid systems based on symbolic-numeric computation," *CoRR*, vol. abs/1302.5974, 2013.

[12] R. Alur, A. Kanade, S. Ramesh, and K. C. Shashidhar, "Symbolic analysis for improving simulation coverage of simulink/stateflow models," in *Proceedings of the 8th ACM International Conference on Embedded Software*, 2008, pp. 89–98.

[13] D. Ishii, K. Ueda, H. Hosobe, A. Goldsztejn *et al.*, "Interval-based solving of hybrid constraint systems," in *Proceedings of the 3rd IFAC Conference on Analysis and Design of Hybrid Systems*, 2009, pp. 144–149.

[14] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "HyTech: A model checker for hybrid systems," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1997, vol. 1254, pp. 460–463.

[15] T. Bourke and M. Pouzet, "Zélus: a synchronous language with odes," in *Proceedings of the 16th international conference on Hybrid systems: computation and control*, 2013, pp. 113–118.

[16] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *Computer Aided Verification*. Springer, 2011, pp. 379–395.

[17] A. Donze and O. Maler, "Systematic simulation using sensitivity analysis," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4416, pp. 174–189.

[18] H. Elmqvist, S. E. Mattsson, and M. O. Otter, "Modelica - the new object-oriented modeling language," in *In Proceedings of the 12th European Simulation Multiconference*, 1998, pp. 127–131.

[19] R.-J. Back, C. Seceleanu, and J. Westerholm, "Symbolic simulation of hybrid systems," in *Software Engineering Conference, 2002. Ninth Asia-Pacific*, 2002, pp. 147–155.

[20] T. Hickey and D. Wittenberg, "Rigorous modeling of hybrid systems using interval arithmetic constraints," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 2993, pp. 402–416.

[21] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012.

[22] ——, "Hybrid dynamical systems," *IEEE Control Systems Magazine*, pp. 28–93, 2009.

[23] The MathWorks Inc., *Symbolic Math Toolbox User's Guide*, MathWorks, 3 Apple Hill Drive Natick, MA 01760-2098, September 2013.