Proceedings of the American Control Conference
Anchorage, AK May 8-10, 2002

Rigorous Simulation of Hybrid Dynamic Systems with Symbolic and Interval Methods[1]

Nedialko S. Nedialkov and Martin von Mohrenschildt
Department of Computing and Software
McMaster University, Hamilton, Canada
E-mail: {nedialk,mohrens}@mcmaster.ca

Abstract

Dynamic systems with a mix of continuous and discrete components, often called hybrid dynamic systems, frequently arise in engineering applications. Since many of these applications are safety critical, it is important to use reliable methods to simulate hybrid systems.

This paper illustrates two approaches to rigorous simulation of hybrid dynamic systems. In the first approach, we use symbolic methods to compute closed-form solutions, thus avoiding round off and truncation errors. In the second approach, we use interval methods to compute rigorous bounds on the solution of a hybrid system.

1 Introduction

Hybrid dynamic systems (HDSs) are systems with a mix of discrete and continuous components. Depending on the domain of application, HDSs are modeled as hybrid automata [2, 5, 10, 18] or dynamic systems with a discrete control [3, 4, 7]. The continuous components are normally governed by systems of initial value problems (IVPs) of ordinary differential equations (ODEs). When a discrete event occurs, the system describing a continuous component of an HDS normally changes, and the execution of an HDS changes, or switches discreetly. For an overview of HDSs and their applications, see [2].

Hybrid dynamic systems are used, for example, as models of continuous processes controlled by logic controllers or embedded systems. Since HDSs are often used to analyze safety-critical systems, it is important that HDSs are studied with reliable methods. One approach to studying an HDS is by simulating its behavior into the future. For an overview and a comparison of several simulation packages, see [9]. A discussion of the problems arising in simulating HDSs can be found in [14].

A crucial part of a numerical simulation of an HDS is the reliable handling of state events, when the system changes from one discrete mode to another. Reliable handling of state events is associated with accurate and reliable computing of switching points. That is, the times when the system changes between discrete modes.

The current tools for numerical simulation of HDSs compute approximations to the solution of an HDS. Even if such a tool is highly reliable, the user does not normally have a guarantee about how accurate or reliable the computed approximations are. If an HDS models a safety-critical process, such a guarantee may be needed. Furthermore, the solution to a system may not exist, or the solution may not be unique. In this case, the user should be notified. Otherwise, the simulation may continue beyond a point where the system is not well defined, leading to incorrect results and possibly wrong conclusions about the behavior of the system.

This paper illustrates two approaches to rigorous simulation of HDSs: symbolic methods and interval methods. Our focus is on reliable computing of switching points.

Symbolic methods attempt to compute closed-form solutions of continuous components. Then, switching times can be approximated to within a round off error. However, symbolic methods are restricted to problems that have closed-form solutions.

Interval (or validated) numerical methods for IVPs for ODEs have two important advantages over standard numerical methods: if an interval method returns successfully, it (1) ensures that the problem has a unique solution, and (2) produces bounds that contain the true solution. Thus, if a unique solution to a problem does not exist, an interval method detects this situation and can notify the user. Moreover, these methods are not restricted to problems with closed-form solutions. In addition, an interval method does not miss a switching point, while a standard (point) method may integrate over a switching point without noticing it.

[1]This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Section 2 defines the HDS that is the subject of this paper. Section 3 outlines a symbolic approach to computing a solution to an HDS. Section 4 briefly explains how interval methods for IVPs for ODEs work and discusses an interval approach for bounding the solution of an HDS. Numerical experiments are presented in Section 5, and final remarks are given in Section 6.

2 Hybrid Dynamic Systems

In the literature, various definitions of hybrid systems can be found [2, 3, 4, 5, 7, 10, 18]. In this paper, we employ the following definition.

Definition 2.1 (Hybrid Dynamic System) *A hybrid dynamic system is given by a set of ODEs*[1]

$$y'(t) = F(y,u), \tag{1}$$

where $y \in \mathbb{R}^n$ is a state vector and $u \in \mathbb{R}^m$ is a control vector, an initial condition

$$y(t_0) = y_0, \tag{2}$$

and an initial value for the control vector

$$u = s_0. \tag{3}$$

For a fixed u, we assume that $F(y,u)$ is Lipschitz continuous in y in some open $\mathcal{D} \subseteq \mathbb{R}^n$, and $y_0 \in \mathcal{D}$.

The value for u is determined by a controller

$$C_u = (C_{u_1}, C_{u_2}, \dots, C_{u_m})$$

for the HDS. The ith component u_i of u is determined by a piecewise constant function $C_{u_i} : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$,

$$C_{u_i}(u,y) = \begin{cases} c_{i,1} & \textit{if } \chi_{i,1}(y) \\ \vdots & \\ c_{i,p_i} & \textit{if } \chi_{i,p_i}(y) \\ u_i & \textit{otherwise}, \end{cases} \tag{4}$$

where $c_{i,j} \in \mathbb{R}$, $j = 1, 2, \dots, p_i$, and

$$\chi_{i,j} : \mathbb{R}^n \to \{\texttt{true}, \texttt{false}\}$$

are predicates for which

$$\chi_{i,l}(y) \wedge \chi_{i,q}(y) = \texttt{false} \quad \textit{for } l \neq q.$$

The regions in the state space defined by the characteristic predicates $\chi_{i,j}$ should not be of measure zero, which means that, no matter what the value of the state variables, the system remains in any of the modes for at least some time $\delta > 0$.

If, for some state y, $\chi_{i,j}(y) = \texttt{false}$ for all $j = 1, 2, \dots, p_i$, then the value of u_i remains unchanged. In this case, the behavior of the system depends on its history. That is, we have *hysteresis* behavior.

We refer to a particular value s_i for u as the *mode* of an HDS. Since each component of the control vector u can take only a finite number of values, an HDS has a finite number of modes. We denote the set of these modes by S.

The *run* of an HDS is a finite or infinite sequence

$$r = \big(\tau_0, \phi_0(t)\big),\ \big(\tau_1, \phi_1(t)\big),\ \big(\tau_2, \phi_2(t)\big), \dots,$$

where $\tau_0 = t_0$, and τ_i for $i \geq 1$ are switching times, or switching points. The functions $\phi_i(t)$, $i \geq 0$, satisfy

$$\phi_i'(t) = F\big(\phi_i(t), s_i\big) \quad \text{for} \quad t \in [\tau_i, \tau_{i+1}), \quad \text{and}$$
$$\phi_i(\tau_i) = \begin{cases} \phi_0(\tau_0) = y(t_0) & \text{if } i = 0 \\ \phi_{i-1}(\tau_i) & \text{if } i \geq 1, \end{cases}$$

where $s_{i-1} \neq s_i$, $i \geq 1$.

For $i \geq 1$, if

$$\tau_i = \min_{t > \tau_{i-1}} C_u\big(s_{i-1}, \phi_{i-1}(t)\big) = s_i$$

exists, then the system switches from mode s_{i-1} to s_i. If this minimum does not exist, the system stays in the mode s_{i-1} forever. The sequence of modes $s_0, s_1, s_2, \dots$, is called the *trace* of an HDS.

The *solution* to an HDS corresponding to a run r [7] is a piecewise continuous function $y : \mathbb{R} \to \mathbb{R}^n$ such that

$$y(t) = \begin{cases} \phi_0(t) & \text{if } \tau_0 \leq t < \tau_1 \\ \phi_1(t) & \text{if } \tau_1 \leq t < \tau_2 \\ \vdots & \end{cases}$$

In this paper, we consider simulating HDSs in finite time. We do not consider discrete changes (resetting) of the state variables. However, the methods described here can be extended to handle this case.

3 Symbolic Simulation

In this section, we assume that an explicit solution can be obtained for each of the IVPs that arise.

To perform a symbolic simulation of an HDS, we solve two problems: (1) compute explicit solutions of IVPs

[1]For expositional convenience, we consider only autonomous systems. This is not a restriction of consequence, since a nonautonomous system of ODEs can be converted into an autonomous one.

and (2) determine switching points by solving symbolically sets of equations.

Consider

$$\phi_0'(t) = F(\phi_0, s_0), \quad \phi_0(\tau_0) = y_0.$$

First, we compute an explicit form for $\phi_0(t)$. Then, we determine, by solving the set of equations

$$\{\, \chi_{i,j}(\phi_0(t)) = \texttt{true} \mid i = 1, \ldots, m,\ j = 1, \ldots, p_i \,\},$$

the smallest $\tau_1 > t_0$ such that at least one of the predicates in this set is true.

If $\tau_1 > t_0$ can be computed, the system changes to a new mode $s_1 = C_u(s_0, \phi_0(\tau_1))$. Otherwise, we cannot continue our simulation beyond τ_1. If we have determined τ_1 and s_1, we apply the same steps to the problem

$$\phi_1'(t) = F(\phi_1, s_1), \quad \phi_1(\tau_1) = \phi_0(\tau_1),$$

and so on.

Part of the numerical results in this paper are produced using this approach implemented in Maple; for a more detailed discussion, see [23].

4 Simulation with Interval Methods

If we use an approximate (point) numerical method with event location facilities, for example a Runge-Kutta [6] or a multi-step method [22], to simulate an HDS, we compute approximations to its solution, assuming that it exists. Although such approximations are generally reliable, the user does not have a guarantee about the accuracy of the computed results. Moreover, if the system is not well defined in the sense that it does not have a solution or has more then one solution, an approximate method may not be able to notify the user and may produce misleading results.

In contrast to point numerical methods for IVPs, interval methods verify first that a unique solution to the problem exists and then produce bounds that contain the mathematically correct result. Such bounds can be used to prove properties of a system or to give the user an indication about the reliability of the computed results. Furthermore, if an HDS is not well defined at some point, an interval method stops with a message that the integration cannot continue.

We introduce in §4.1 some of the notation that we use later. In §4.2, we give a brief overview of how interval methods for IVPs for ODEs work. Since these methods have been discussed in detail in several works, for example in [11, 15, 20], we do not present details here and refer the reader to those works. In §4.3, we describe how we enclose a switching point and the solution of an HDS to the right of a switching point. In §4.4, we summarize the main features of the VNODE (Validated Numerical ODE) solver [15], which we have used to produce some of the numerical results in this paper.

4.1 Notation

An interval $[a] = [\underline{a}, \bar{a}]$ is the set of real numbers

$$[a] = [\underline{a}, \bar{a}] = \{\, x \mid \underline{a} \le x \le \bar{a}, \quad \underline{a}, \bar{a} \in \mathbb{R} \,\}.$$

If $[a]$ and $[b]$ are intervals and $\circ \in \{+, -, *, /\}$, then the interval-arithmetic operations [13] are defined by

$$[a] \circ [b] = \{\, x \circ y \mid x \in [a],\ y \in [b] \,\}, \tag{5}$$

where $0 \notin [b]$ when $\circ = /$.

One of the strengths of interval arithmetic when implemented on a computer is in computing rigorous enclosures of real operations by including rounding errors in the computed bounds. Such errors can be easily included by rounding the real intervals in (5) outwards. In this paper, we do not discuss properties of interval arithmetic and refer the reader to [1] and [13].

An interval vector is a vector with each component an interval. Let $\chi : \mathbb{R}^n \to \{\texttt{true}, \texttt{false}\}$ be a predicate. We extend this predicate to interval vectors as follows:

$$\chi([a]) = \begin{cases} \texttt{true} & \text{if } \chi(a) = \texttt{true}, \ \forall a \in [a] \\ \texttt{false} & \text{if } \chi(a) = \texttt{false}, \ \forall a \in [a] \\ \texttt{undetermined} & \text{otherwise.} \end{cases}$$

Let $s_i \in S$. Then, for an interval vector $[a]$, if none of the predicates associated with C_u in (4) is undetermined, $C_u(s_i, [a]) = C_u(s_i, a)$ for all $a \in [a]$. That is, we can compute a mode for all $a \in [a]$.

If at least one of these predicates is undetermined, then we cannot determine a mode from s_i and $[a]$. In this case, we set $U = C_u(s_i, [a])$, where $U \in \mathbb{R}^n$ and $U \notin S$. This U is not part of the definition of an HDS. We shall use U in our method for detecting switching points and enclosing the solution past a switching point.

4.2 Interval Methods for IVPs

Consider the set of autonomous IVPs

$$y'(t) = f(y) \tag{6}$$
$$y(t_0) \in [y_0], \tag{7}$$

where $t \in [t_0, t_m]$ for some $t_m > t_0$. Here t_0 and $t_m \in \mathbb{R}$, $f \in C^{k-1}(\mathcal{D})$, $k \ge 2$ (k is the order of the truncation error of the method), $\mathcal{D} \subseteq \mathbb{R}^n$ is open, $f : \mathcal{D} \to \mathbb{R}^n$, and $[y_0] \subseteq \mathcal{D}$. The condition (7) permits the initial value $y(t_0)$ to be in an interval, rather than specifying a particular value. We assume that the representation of f contains a finite number of constants, variables, elementary operations, and standard functions. Since we assume $f \in C^{k-1}(\mathcal{D})$, we exclude functions that contain, for example, branches, abs, or min. For expositional convenience, we consider only autonomous systems. This is not a restriction of consequence, since a nonautonomous system of ODEs can be converted into an autonomous one.

We consider a grid $t_0 < t_1 < \cdots < t_m$ and denote the stepsize from t_{j-1} to t_j by $h_{j-1} = t_j - t_{j-1}$. We denote the solution of (6) with an initial condition $y(t_{j-1}) = y_{j-1}$ at t_{j-1} by $y(t; t_{j-1}, y_{j-1})$. For an interval, or an interval vector $[y_{j-1}]$, let

$$y(t; t_{j-1}, [y_{j-1}]) = \{\, y(t; t_{j-1}, y_{j-1}) \mid y_{j-1} \in [y_{j-1}] \,\}.$$

Validated methods compute interval vectors $[y_j]$ that contain the solution of (6–7) at t_j, $j = 1, 2, \ldots, m$. That is,

$$y(t_j; t_0, [y_0]) \subseteq [y_j], \quad \text{for } j = 1, 2, \ldots, m.$$

Usually, these are one-step methods, where each step consists of two phases, Algorithm I and Algorithm II [15, 16].

Algorithm I computes a stepsize h_{j-1} and an a priori enclosure $[\tilde{y}_j]$ such that (6) with $y(t_{j-1}) = y_{j-1}$ has a unique solution $y(t; t_{j-1}, y_{j-1})$ that satisfies

$$y(t; t_{j-1}, y_{j-1}) \in [\tilde{y}_j]$$

for all $t \in [t_{j-1}, t_j]$ and all $y_{j-1} \in [y_{j-1}]$.

Algorithm II uses $[\tilde{y}_j]$ to bound the truncation error of the method and computes a tight enclosure $[y_j] \subseteq [\tilde{y}_j]$ on the solution at t_j, such that

$$y(t_j; t_0, [y_0]) \subseteq [y_j].$$

The algorithm to validate the existence of a unique solution typically uses the Picard-Lindelöf operator and the Banach fixed-point theorem [12, 17]. The computation of a tight enclosure is usually based on Taylor series plus a remainder term, the mean-value theorem, and various interval transformations [12, 13, 15]. Recently, the Taylor series approach was generalized to the interval Hermite-Obreschkoff method [15], which is shown to be superior to interval Taylor series methods.

4.3 Enclosing the Solution of an HDS

Our goal is to compute bounds on the solution of an HDS. In this section, we describe how to compute such bounds when an HDS is in the initial mode, and if the system switches to mode s_1, how to locate and enclose the time when it switches. Then, we are interested in enclosing the solution of the HDS after it has switched to mode s_1. Once we have bounds on the solution when the system is in s_1, we can apply the same approach to enclose the solution as the system evolves.

Since we use interval methods, we assume that, for any $s_i \in S$, $F(y, s_i) \in C^{k-1}(\mathcal{D})$. We also assume that the initial condition of the HDS is given by an interval $[y_0] \subseteq \mathcal{D}$. Thus, we can have a set of solutions to (1–3), where each solution corresponds to some $y_0 \in [y_0]$.

Detecting and Enclosing Switching Points
Consider the problem

$$y'(t) = F(y, s_0), \quad y(t_0) = y_0 \in [y_0]. \qquad (8)$$

Suppose that we have computed at $t_{j-1} > t_0$ an enclosure $[y_{j-1}]$ of the solution of (8) such that

$$y(t_{j-1}; t_0, [y_0]) \subseteq [y_{j-1}]$$

and $C_u(s_0, [y_{j-1}]) = s_0$.

Let $[\tilde{y}_j]$ be such that $y(t; t_{j-1}, y_{j-1}) \in [\tilde{y}_j]$ for all $y_{j-1} \in [y_{j-1}]$ and all $t \in [t_{j-1}]$. If $C_u(s_0, [\tilde{y}_j]) = s_0$, then

$$C_u(s_0, y(t; t_{j-1}, y_{j-1})) = s_0$$

for all $y_{j-1} \in [y_{j-1}]$ and all $t \in [t_{j-1}, t_j]$. Thus, we can guarantee that, for any $y_0 \in [y_0]$, the HDS given in (1–3) does not have a switching point in $[t_{j-1}, t_j]$.

However, if $C_u(s_0, [\tilde{y}_j]) = U$, then there may be a switching point in $[t_{j-1}, t_j]$ for some $y_0 \in [y_0]$. In this case, we try to determine

1. as small a stepsize $0 < h'_{j-1} \leq h_{j-1}$ as possible, such that $C_u(s_0, [y'_j]) = s_1$, where

$$y(t'_j; t_0, [y_0]) \subseteq [y'_j], \quad t'_j = t_{j-1} + h'_{j-1},$$

or

2. as large a stepsize $0 < h'_{j-1} < h_{j-1}$ as possible, such that $C_u(s_0, [\tilde{y}'_j]) = s_0$, where

$$y(t; t_{j-1}, y_{j-1}) \in [\tilde{y}'_j],$$

for all $y_{j-1} \in [y_{j-1}]$ and all $t \in [t_{j-1}, t'_j]$, $t'_j = t_{j-1} + h'_{j-1}$.

Consider first 1. If $C_u(s_0, [y'_j]) = s_1$, then for any $y_0 \in [y_0]$, there exists a switching point $\tau_1 \in [t_{j-1}, t'_j]$. The interval $[t_{j-1}, t'_j]$ encloses the switching points for all $y_0 \in [y_0]$. We also guarantee that

$$y(t; t_0, [y_0]) \subseteq y(t; t_{j-1}, [y_{j-1}]) \subseteq [\tilde{y}'_j],$$

for all $t \in [t_{j-1}, t'_j]$.

If we have reached a prescribed value for the minimum stepsize allowed, and $C_u(s_0, [y'_j]) = U$ or $C_u(s_0, [y'_j]) = s_0$ and $C_u(s_0, [\tilde{y}'_j]) = U$, then we do not continue the solution beyond t'_j. The reason is that we do not know with certainty if, for any $y_0 \in [y_0]$, there is a switching point in $[t_{j-1}, t'_j]$.

In 2., if $C_u(s_0, [\tilde{y}'_j]) = s_0$, we continue the integration with $F(y, s_0)$ and $[y'_j]$ at t'_j. In this case, we can guarantee that, for any $y_0 \in [y_0]$, there is no switching point in $[t_{j-1}, t'_j]$.

Enclosing the Solution Past Switching Points
We assume that $C_u(s_0, [y_{j-1}]) = s_0$ and $C_u(s_0, [y'_j]) = s_1$. We use the two-step approach described in [19] to compute $[y_j]$ such that the solution to

$$y'(t) = F(y, s_1), \quad y(\tau_1) \in [\tilde{y}'_j],$$

for all $\tau_1 \in [t_{j-1}, t'_j]$, is contained in $[y_j]$ at t'_j. Thus, we "extend" the solution to the right of a switching point with $F(y, s_1)$.

If $C_u(s_0, [y_j]) = s_1$, we continue the integration with

$$y'(t) = F(y, s_1), \quad y(t'_j) \in [y_j].$$

Otherwise, we do not continue the integration beyond t'_j and notify the user that our solver cannot continue. One reason for $C_u(s_0, [y_j]) = s_1$ not to hold is that a (classical) unique solution may not exist beyond a switching point. In §5.3, we illustrate this situation.

4.4 The VNODE Solver

The VNODE package is an object-oriented C++ package for computing bounds that contain the true solution of an IVP for an ODE. Currently, the methods in VNODE are based on Taylor series [12, 15] and the Hermite-Obreschkoff scheme [15]. On each integration step, VNODE carries out the two-algorithm approach outlined in §4.2. The user can supply a constant value for the stepsize, or specify a variable stepsize control, where the stepsize is controlled such that the local error per unit step is within a user-specified tolerance [15].

The numerical results in the next section are produced with a high-order method for validating existence and uniqueness of the solution [17] of order 17, an interval Hermite-Obreschkoff method of the same order, variable stepsize control, and the event location scheme described in §4.3.

5 Numerical Experiments

We present three examples. For each of them, we use Maple to compute symbolically switching points and then VNODE to compute numerically bounds on the switching points. In the last example, we consider a problem for which a classical solution does not exist beyond a switching point.

5.1 Water-Level Control

The water-level controller is often used as an example in the literature on hybrid systems [5, 7, 10, 23]. Here, we consider the following simplified model.

We have a reservoir that has an in-flow valve and an out-flow valve. The out-flow valve is always open, and the in-flow valve can be open, closed, opening, or closing. We assume that initially the water level is between $\underline{w}$ and $\bar{w}$, where $\underline{w}$ and $\bar{w}$ are constants, $\underline{w} < \bar{w}$, and the in-flow valve is open.

Our water-level controller is described by

$$y'_1 = y_2, \quad y'_2 = 0.5u, \tag{9}$$

where y_1 is the water level, y_2 is the flow rate, and

$$C_u(u, y) = \begin{cases} -1 & \text{if } y_1 \geq \bar{w} \\ 1 & \text{if } y_1 \leq \underline{w} \\ 0 & \text{otherwise.} \end{cases}$$

For our simulation, we have chosen $\underline{w} = 3$, $\bar{w} = 7$, $y(0) = (5, 1)^T$, and $s_0 = 0$.

We integrated this water-level controller for $t \in [0, 35]$ using Maple and VNODE. In the first column of Table 1, we show the exact values for the first five switching points. We have computed these values symbolically with Maple. In the second column, we show the enclosures of these points as computed by VNODE. The notation 3.9999999999999[9, 11] in the first line in this table specifies the interval

Switching Points	
Exact	Enclosure
2	1.9999999999999[9, 11]
6	5.999999999999[96, 111]
10	9.999999999999[68, 126]
14	13.99999999999[911, 1140]
18	17.99999999999[558, 1469]

Table 1: The first five switching points of the water-level controller.

$$\begin{aligned} &1.9999999999999[9, 11] \\ &\quad = [1.99999999999999, 2.00000000000001], \end{aligned}$$

which contains the mathematically correct value 2. The rest of the enclosures in this table can be interpreted in a similar manner.

In Figure 1, we plot the bounds produced by VNODE for $y_1(t)$ versus the time t. The dashed lines denote the a priori bounds on the solution in each of the integration intervals. The solid lines connect the tight bounds of the solution between integration points. When plotted, these bounds cannot be distinguished since their widths are very small. For example, at $t = 35$,

$$\begin{aligned} y_1(t) &\in 7.74999999[831040, 1193196] \\ &= [7.74999999831040, 7.75000000193196]. \end{aligned}$$

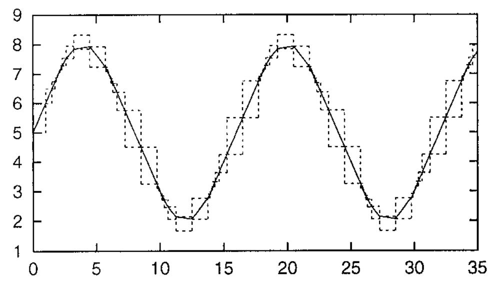

Figure 1: Plot of the bounds on the water level, $y_1(t)$, versus t.

On each step, the stepsize controller in our solver accepts a stepsize such that a tolerance requirement is satisfied. However, when the solution is close to a switching point, the stepsize is reduced such that the true value for the switching point is enclosed as accurately as possible. This reduction of the stepsize around a switching point can be seen from Figure 1, where the a priori bounds are tighter and tighter, as the solution is closer and closer to a switching point.

These a priori bounds can be used to guarantee that the true solution is within certain bounds. For example, our method guarantees that, for all $t \in [0, 35]$,

$$y_1(t) \in [1.67187499978418, 8.32812500001357].$$

As can be seen from Figure 1, if we take smaller stepsizes when $y_1(t)$ is close to 2 and 8, the a priori bounds on $y_1(t)$ can be made tighter. With an upper limit of 0.1 on the stepsize, we obtained that

$$y_1(t) \in [1.99749999965518, 8.00250000002154],$$

for all $t \in [0, 35]$. Therefore, the water level is within these bounds for all $t \in [0, 35]$.

5.2 A Model of a Chemical Reactor

We consider a simplified model of a chemical reactor [21]. The state variables are the fluid level in the reactor, y_1, and the temperature in the reactor, y_2. The mode is determined by the vector

$$u = (u_\text{i}, u_\text{d}, u_\text{b}, u_\text{h}, u_\text{c}, u_\text{r})^T,$$

where each component encodes a control signal:

Signal	Interpretation
u_i	inflow valve signal
u_d	draining valve signal
u_b	blender signal
u_h	heater signal
u_c	cooler signal
u_r	reactor signal

Each of these signals can be 0 or 1. For more details, see [21].

The dynamics of this reactor is specified by

$$y' = A(u)y + b(u), \tag{10}$$

where

$$A(u) = \begin{pmatrix} -a_\text{h}u_\text{d} & 0 \\ 0 & -(a_{\text{T}_1}(1-u_\text{b}) + a_{\text{T}_2}u_\text{b}) \end{pmatrix} \tag{11}$$

and

$$b(u) = \begin{pmatrix} b_\text{h}u_\text{i} \\ b_\text{heat}u_\text{h} + b_\text{cool}u_\text{c} + b_\text{reac}u_\text{r} \end{pmatrix}. \tag{12}$$

We use the following constants in (11–12): $a_\text{h} = 1.13 \times 10^{-3}$, $a_{\text{T}_1} = 0.15 \times 10^{-3}$, $a_{\text{T}_2} = 0.22 \times 10^{-3}$, $b_\text{h} = 9.938 \times 10^{-3}$, $b_\text{heat} = 29.43 \times 10^{-3}$, $b_\text{cool} = -44.15 \times 10^{-3}$, $b_\text{rec} = 44.15 \times 10^{-3}$.

The initial condition is $y(0) = (0,0)^T$, and the initial mode is $s_0 = (1,0,0,1,0,0)^T$. The values for the components of the control vector u are determined as follows:

- u_i is set to 0 when $25y_1 + y_2 = 300$ and is set to 1 when $25y_1 + y_2 = 250$;
- $u_\text{d} = 0$ when $y_2 < 50$ and $u_\text{d} = 1$ otherwise;
- $u_\text{b} = 0$ when $y_1 < 3$ and $u_\text{b} = 1$ otherwise;
- $u_\text{h} = 1$ when $y_2 < 50$ and $u_\text{h} = 0$ otherwise;
- u_c is set to 0 when $y_2 = 110$ and is set to 1 when $y_2 = 130$; and
- $u_\text{r} = 0$ when $y_2 < 50$ and $u_\text{r} = 1$ otherwise.

We integrated the above problem for $t \in [0, 1200]$. There are 29 switching points in this interval. Starting from the initial mode, the execution of this system goes through six other modes.

Here are floating-point approximations to the symbolically computed (with Maple) switching points:

	Switching Points
1	301.87160394445562487
2	1092.1766084799351739
3	2124.1454957215599882
⋮	⋮
28	11464.510543006407988
29	11714.526388901017066

In the second column of Table 2, we have shown the enclosures of these switching points. In the third column, we have included the corresponding widths of these enclosures. Each of these widths can be interpreted as the error, or uncertainty, in enclosing a switching point.

The values for the switching times computed by Maple are contained in the corresponding intervals computed

	Switching Points	
	Enclosure	Error
1	301.87160394445[47, 70]	2×10^{-12}
2	1092.1766084799[28, 47]	2×10^{-11}
3	2124.1454957215[20, 93]	8×10^{-11}
⋮	⋮	⋮
28	11464.[48486940166, 56299440167]	8×10^{-2}
29	11714.[49460713296, 65085713297]	2×10^{-1}

Table 2: Enclosures on the switching points of the chemical reactor problem for $t \in [0, 1200]$.

by VNODE. From Table 2, the widths of the enclosing intervals grow as the integration proceeds. One reason for this growth is that, in an interval enclosing a switching point, we enclose the solution in two modes.

Although it is generally difficult to obtain tight bounds for the switching times for large values of t with our approach, we guarantee that a switching point is not missed, and we produce bounds on both the switching points and the solution.

In Figure 2, we plot the temperature, y_2, versus the fluid level, y_1. The "corners" in this plot correspond to the switches in the system.

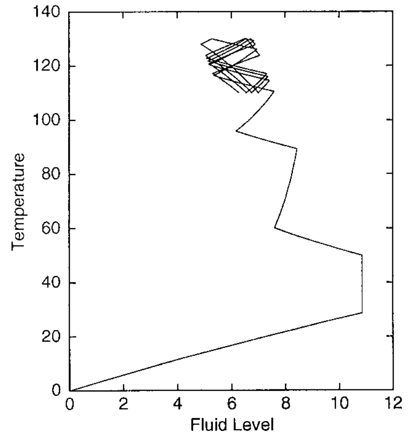

Figure 2: Phase portrait of the chemical reactor model.

5.3 A Problem without a Classical Solution

The problem

$$
\begin{aligned}
y_1' &= y_2 \\
y_2' &= -0.2\,y_2 - y_1 + 2\cos(\pi t) - \begin{cases} 4 & \text{if } y_2 > 0 \\ -4 & \text{if } y_2 < 0 \end{cases} \qquad (13) \\
y(0) &= (3, 4)^T
\end{aligned}
$$

is considered in [8] as an example for which a classical solution does not exist beyond a point $t \approx 2.0352$. In [8], 0.5628 and 2.0352 are given as approximations for the switching points.

With Maple, we computed 0.56280532524534910455 as an approximation to the first switching point. However, Maple could not compute an approximation to the second switching point, since this package could not determine a real root of $y_2(t) = 0$.

With VNODE, we computed the following enclosures on the switching points:

$$0.5628053252453[4, 7] \text{ and } 2.035200434043[25, 55].$$

After the second switching point, our solver could not continue since it could not determine an enclosure $[y_{2,j}]$ for y_2 such that $y_{2,j} > 0$, for all $y_{2,j} \in [y_{2,j}]$.

In Figure 3, we have plotted the bounds on $y_2(t)$ versus t for $t \in [0, 2.03520043404325]$.

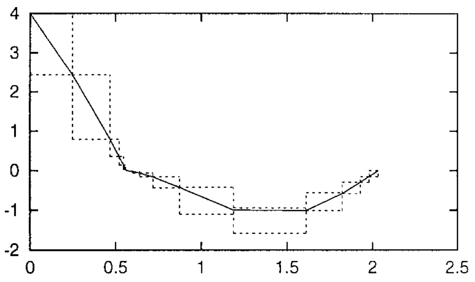

Figure 3: Plot of the bounds on $y_2(t)$ of (13) versus t.

A standard (point) solver may continue the integration past the second switching point. In this situation, the user may not have information that a solution to the problem does not exist. When simulating systems for which the reliability of the results is important, such information may be desirable.

6 Concluding Remarks

Symbolic methods are suitable for studying HDSs when closed-form solutions of continuous components can be computed.

Interval methods ensure that a unique solution to an HDS exists and compute bounds that contain the true solution. Hence, these methods guarantee that a switching point is not missed and that the system does not enter a "wrong" mode. Furthermore, if a unique solution does not exist, an interval method notifies the user.

Since our method encloses roundoff and truncation errors on each step and propagates such errors from previous steps, these errors may accumulate over long integration intervals. As a result, the computed bounds may become too wide. In this case, our method normally stops with a message that the integration cannot proceed. Thus, the proposed interval approach performs well when the computed bounds on the solution of an HDS remain sufficiently small.

Acknowledgments
The authors thank Dr. George Corliss for reading a late draft of this paper. His subtle comments helped to improve the presentation of the material.

References

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Academic Press, New York, 1983.

[2] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. In *Proceedings of IEEE*, volume 88, pages 971–984, 2000.

[3] M. Branicky, V. Borkar, and S. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Trans. Automat. Contr.*, 43:31–45, 1998.

[4] F. Clarke, Y. Ledyaev, and P. Stern. *Nonsmooth Analysis and Control Theory.* Springer-Verlag, New York, 1998.

[5] D. Kapur and R. K. Shyamasundar. Synthesizing controllers for hybrid systems. In O. Maler, editor, *Hybrid and Real-Time Systems*, volume 1201 of *Lecture Notes in Computer Science*, pages 361–375. Springer-Verlag, 1997.

[6] W. H. Enright, K. R. Jackson, S. P. Norsett, and P. G. Thomsen. Effective solution of discontinuous IVPs using a Runge-Kutta formula pair with interpolants. *Applied Mathematics and Computation*, 27:313–335, 1988.

[7] A. Gollu and P. Varaiya. Hybrid dynamical systems. In *Proc. 28th IEEE Conf. Decision Contr.*, pages 2708–2712, 1989. Tampa, Florida.

[8] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer-Verlag, 2nd revised edition, 1991.

[9] S. Kowalewski, O. Stursberg, M. Fritz, H. Graf, I. Hoffmann, J. Preusig, M. Remelhe, S. Simon, and H. Treseler. A case study in tool-aided analysis of discreetly controlled continuous systems: The two tanks problem. In *Hybrid Systems*, pages 163–185, 1997.

[10] G. Laerriere, G. J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 137–151. Springer-Verlag, 1999.

[11] R. J. Lohner. Enclosing the solutions of ordinary initial and boundary value problems. In E. W. Kaucher, U. W. Kulisch, and C. Ullrich, editors, *Computer Arithmetic: Scientific Computation and Programming Languages*, pages 255–286. Wiley-Teubner Series in Computer Science, Stuttgart, 1987.

[12] R. J. Lohner. *Einschließung der Lösung gewöhnlicher Anfangs– und Randwertaufgaben und Anwendungen.* PhD thesis, Universität Karlsruhe, 1988.

[13] R. E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, N.J., 1966.

[14] P. Mosterman. An overview of hybrid simulation phenomena and their support by simulation packages. In *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 165–177. Springer-Verlag, 1999.

[15] N. S. Nedialkov. *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, M5S 3G4, February 1999.

[16] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999.

[17] N. S. Nedialkov, K. R. Jackson, and J. D. Pryce. An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE. *Reliable Computing*, 7(6):1–17, 2001.

[18] X. Nicollin, J. Sifakis, and S. Yovine. Compiling real-time specifications into extended automata. *IEEE Transactions on Software Engineering*, 18(9):794–804, September 1992.

[19] R. Rihm. Enclosing solutions with switching points in ordinary differential equations. In L. Atanassova and J. Herzberger, editors, *Computer Arithmetic and Enclosure Methods*, pages 419–425. North-Holland, Amsterdam, 1992.

[20] R. Rihm. Interval methods for initial value problems in ODEs. In J. Herzberger, editor, *Topics in Validated Computations: Proceedings of the IMACS-GAMM International Workshop on Validated Computations, University of Oldenburg*, Elsevier Studies in Computational Mathematics, pages 173–207. Elsevier, Amsterdam, New York, 1994.

[21] J. Roll. Invariance of approximating automata for piecewise linear systems with uncertainties. In *Hybrid Systems: Computation and Control*, volume 1790 of Lecture Notes in Computer Science, pages 396–406. Springer-Verlag, 2000.

[22] L. F. Shampine and S. Thompson. Event location for ordinary differential equations. *Comput. Math. Appl.*, 39(5–6):43–54, 2000.

[23] M. v. Mohrenschildt. Symbolic verification of hybrid systems: An algebraic approach. *European Journal of Control*, 7(5):541–556, 2001.