Computing Polyhedral Approximations to Flow Pipes for Dynamic Systems

Alongkrit Chutinan and Bruce H. Krogh
Department of Electrical and Computer Engineering
Carnegie Mellon University
ac4c@andrew.cmu.edu/krogh@ece.cmu.edu

Abstract

This paper presents a new approach to approximating the flows of continuous time dynamic systems from sets of initial conditions. The reachable set, or flow pipe, over a time interval $[0, t_f]$ is approximated by the union of a sequence of convex polyhedra. Each polyhedron contains a segment of the flow pipe. Properties of the approximation technique are discussed and illustrated with examples for linear and nonlinear systems, including the application of flow pipe approximation to verification of a simple hybrid system.

1 Introduction

This paper presents a method to compute outer boundaries for the set of all trajectories of a dynamic system starting from a given set of initial states. Following Zhao, we call the reachable set over an interval of time $[0, t_f]$ a *flow pipe*. Computing flow pipes constitutes the fundamental step towards computing the mappings between switching surfaces in hybrid systems [4], [6], [7], [11]. In contrast to methods that conservatively approximate the reachable set by discretizing the state space, the resulting approximation error does not grow with time or distance from the initial set. The computation method is presented for non-linear and linear systems and its application to hybrid system verification is illustrated with an example.

For certain classes of hybrid systems, the flow pipe can be computed exactly [1],[3]. For more complex systems, several methods for approximating reachable sets have been proposed. Most of these methods rely on the discretization of the continuous state space and use the information on the vector field to connect each cell and its neighbors [6],[7],[10],[11]. Although these methods can give arbitrarily accurate approximations by increasing the resolution of the discretization, the number of cells explodes rapidly with the dimension of the state space. Furthermore, if the approximation is to be conservative, the accuracy resulting from such techniques is good only locally.

Zhao [12] introduced a method to approximate flow pipes by first computing an approximation to the stability region of an attractor. The method generates points on the boundary of the stability region through simulations and performs the Delaunay triangulation [9] so that the union of the simplices resulting from the triangulation approximates the region. The vector field determines the trajectory flows between the simplices in the triangulation. This method is attractive because a non-convex region is represented well by the triangulation. It has the same problem as the discretization approach, however, in that the approximation error grows with the distance from the initial set. To improve the accuracy, interior points must be supplied and the region must be re-triangulated.

2 Problem Statement

In this paper we consider autonomous dynamical systems with state equation $\dot{x}(t) = f(x(t))$. We assume that vector field f is *Lipschitz* so that for every initial state x_0 there is a unique solution $x(t, x_0)$ to the state equation. The set of reachable states at time t from a set of initial states X_0 is defined as

$$\mathcal{R}_t(X_0) = \{x_f \mid x_f = x(t, x_0), \exists x_0 \in X_0\} \qquad (1)$$

The set of reachable states, or flow pipe, from X_0 in the time interval $[t_1, t_2]$ is defined as

$$\mathcal{R}_{[t_1,t_2]}(X_0) = \bigcup_{t \in [t_1,t_2]} \mathcal{R}_t(X_0)$$

We now state the main objective of this paper:

Given a set X_0 of initial states which is a polytope, and a final time t_f, compute a polyhedral approximation $\hat{\mathcal{R}}_{[0,t_f]}(X_0)$ to the flow pipe $\mathcal{R}_{[0,t_f]}(X_0)$ such that

$$\mathcal{R}_{[0,t_f]}(X_0) \subseteq \hat{\mathcal{R}}_{[0,t_f]}(X_0).$$

Our method represents the set $\hat{\mathcal{R}}_{[0,t_f]}(X_0)$ by a sequence of convex polytopes.

We introduce the following notation and definitions. Let $POLY(C,d)$ denote the convex polytope defined by the pair $(C,d) \in R^{m \times n} \times R^m$ according to

$$POLY(C,d) = \{x \mid Cx \leq d\}.$$

Each row $c_i^T, i = 1, \ldots, m$ of C is the normal vector to the i^{th} face of the polytope. A polytope P has a finite number of vertices which are points in P that cannot be written as a strict convex combination of any other two points in P. We let $V(P)$ denote the set of vertices of P.

Given a finite set of points, Γ, the *convex hull* of Γ is the smallest convex set (a polytope) that contains Γ. We let $CH(\Gamma)$ denote the convex hull of Γ. Methods for generating a convex hull from a set of points can be found in [9].

3 Basic Computational Procedure

To approximate a non-convex flow pipe effectively, we divide the flow pipe into segments, each of which corresponds to a time interval. The k^{th} flow pipe segment corresponding the time interval $[t_{k-1}, t_k]$ is the set $\mathcal{R}_{[t_{k-1},t_k]}(X_0)$. We first describe the approximation of each flow pipe segment by a convex polytope.

3.1 Approximating a Flow Pipe Segment

Since our objective is to compute an outer approximation to the entire flow pipe, the polytope approximating the k^{th} segment of the flow pipe should contain that particular segment. If the approximating polytope corresponds to a matrix-vector pair (C,d), then we want

$$\mathcal{R}_{[t_{k-1},t_k]}(X_0) \subseteq POLY(C,d). \tag{2}$$

Suppose that C is given. To obtain minimal approximation error for fixed C, we compute d to solve the following optimization problem:

$$\begin{aligned} \min_{d} \quad & volume[POLY(C,d)] \\ \text{s.t.} \quad & \mathcal{R}_{[t_{k-1},t_k]}(X_0) \subseteq POLY(C,d) \end{aligned} \tag{3}$$

We denote the minimal set $POLY(C,d^*)$ which is the solution to (3) by $S_C^{min}(\mathcal{R}_{[t_{k-1},t_k]}(X_0))$. The components of d^* solving (3) can be found by solving the following constrained optimization problems for $i = 1, \ldots, m$

$$\begin{aligned} \max_{x} \quad & c_i^T x \\ \text{s.t.} \quad & x \in \mathcal{R}_{[t_{k-1},t_k]}(X_0) \end{aligned} \tag{4}$$

From the definitions introduced in Section 2, we can rewrite (4) as

$$\begin{aligned} \max_{x_0,t} \quad & c_i^T x(t,x_0) \\ \text{s.t.} \quad & x_0 \in X_0 \\ & t \in [t_{k-1}, t_k] \end{aligned} \tag{5}$$

Let (x_0^*, t^*) be a solution to (5), then $x(t^*, x_0^*)$ is a solution to (4). Then the solution to (3) is given by $d_i^* = c_i^T x(t^*, x_0^*)$, for $i = 1, \ldots, m$. To show this, first note that d^* is feasible, i.e. $\mathcal{R}_{[t_{k-1},t_k]}(X_0) \subseteq POLY(C,d^*)$. Now consider an alternative solution $d \neq d^*$. For any i such that $d_i \neq d_i^*$, if $d_i < d_i^*$, then there exists a state $x \in \mathcal{R}_{[t_{k-1},t_k]}(X_0)$ such that $c_i^T x > d_i$, namely the state x^* such that $c_i^T x^* = d_i^*$. This implies that $\mathcal{R}_{[t_{k-1},t_k]}(X_0) \not\subseteq POLY(C,d)$ and d is not feasible. If $d_i > d_i^*$, assume that $d_j \geq d_j^*$ for all $j \neq i$ so that d is a feasible solution. Then, $POLY(C,d^*) \subseteq POLY(C,d)$. Thus, no other solution produces a smaller feasible set than $POLY(C,d^*)$.

By embedding simulation inside the objective function computation routine, one can use a numerical package such as MATLAB to solve (5) numerically. First, $x(t,x_0)$ can be computed for the given t and x_0 through simulation. The result can then be multiplied with c_i^T to obtain the objective function value. Note (5) is not a convex problem. We discuss this issue in Section 6.

3.2 Choosing a Set of Direction Vectors

To compute the minimal set containing a flow pipe segment, a set of normal vectors, that is, the rows of the matrix C, must be provided. This section describes a heuristic to produce a set of normal vectors for each flow pipe segment.

To gain some geometrical information about the flow pipe segment, we begin by taking sample points at times t_{k-1} and t_k from the trajectories emanating from the vertices of X_0. In particular, we compute the sets $V_{t_{k-1}}(X_0)$ and $V_{t_k}(X_0)$ where

$$V_t(X_0) = \{x(t,v) \mid v \in V(X_0)\}.$$

Each point in the above set can be obtained through simulations, one starting at each vertex of X_0. We then use these points to form a convex hull which serves as an initial approximation to the flow pipe segment $\mathcal{R}_{[t_{k-1},t_k]}(X_0)$, denoted by

$$\Phi_{[t_{k-1},t_k]}(X_0) = CH(V_{t_{k-1}}(X_0) \cup V_{t_k}(X_0)).$$

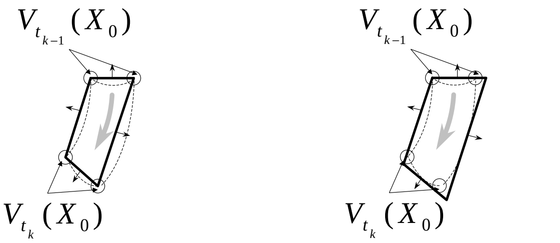
This step is illustrated for a 2-D case in Figure 1a. The dashed lines indicate the boundary of the flow pipe segment. The initial set X_0 is a line segment (not shown) so that there are two vertices in X_0. We simulate these vertices to find $V_{t_{k-1}}(X_0)$ and $V_{t_k}(X_0)$. The convex hull $\Phi_{[t_{k-1},t_k]}(X_0)$, drawn with thick lines, is then constructed from these points. As illustrated in Figure 1a, $\Phi_{[t_{k-1},t_k]}(X_0)$ may not contain the flow pipe segment $\mathcal{R}_{[t_{k-1},t_k]}(X_0)$. Let (C_Φ, d_Φ) be the matrix-vector pair defining $\Phi_{[t_{k-1},t_k]}(X_0)$, i.e.

$$\Phi_{[t_{k-1},t_k]}(X_0) = POLY(C_\Phi, d_\Phi).$$

Note that our convention for the polytope representation by linear inequality constraints dictates that the normal vector on each face of the polytope points outward. We use the normal vectors to the faces of this convex hull as a set of direction vectors to compute the minimal convex set containing the flow pipe segment. Formally, our approximation for the k^{th} flow pipe segment is

$$\hat{\mathcal{R}}_{[t_{k-1},t_k]}(X_0) = S^{min}_{C_\Phi}(\mathcal{R}_{[t_{k-1},t_k]}(X_0))$$

The result is illustrated in Figure 1b.



(a) Initial directions from convex hull

(b) Minimal containing set computed

Figure 1:

3.3 The Complete Flow Pipe

Let the time interval $[0, t_f]$ be divided into N time steps $[0, t_1], [t_1, t_2], \ldots, [t_{N-1}, t_f]$. The complete flow pipe approximation from $t = 0$ to $t = t_f$ is the union of all N flow pipe segments

$$\hat{\mathcal{R}}_{[0,t_f]}(X_0) = \bigcup_{k=1,\ldots,N} \hat{\mathcal{R}}_{[t_{k-1},t_k]}(X_0)$$

As an illustration, Figure 2 shows the resulting approximation for the *Van der Pol* equation

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -0.2({x_1}^2 - 1)x_2 - x_1 \end{aligned}$$

The flow pipe is computed from the initial set $X_0 = \{0.8 \leq x_1 \leq 1, x_2 = 0\}$ from $t = 0$ to $t = 10$. It consists of 20 segments. The time steps are uniform with $\Delta t_k = t_k - t_{k-1} = 0.5$. The boundary of the flow pipe is indicated by the dotted line and is obtained by simulating the two vertices of X_0.

4 Linear Time Invariant Systems

Improvements can be made to the computational procedure discussed in the previous section for linear time invariant systems, $\dot{x}(t) = Ax(t)$, where A is an $n \times n$ constant matrix. Since we now have the analytical solution

$$x(t, x_0) = e^{At}x_0, \tag{6}$$

we no longer need to embed simulation into the computation of the objective function in (5). The optimization problem (5) corresponding to the i^{th} normal vector

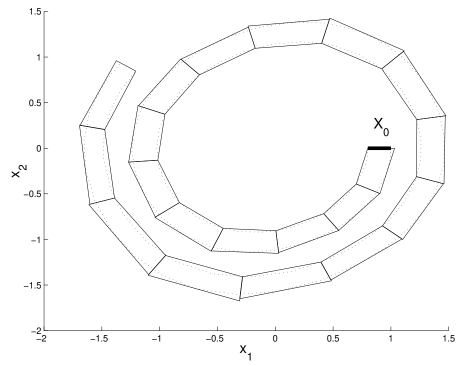


Figure 2: Flow pipe for the Van der Pol system

in C with (6) substituted for the simulation becomes

$$\begin{aligned} \max_{x_0,t} \quad & c_i^T e^{At} x_0 \\ \text{s.t.} \quad & x_0 \in X_0 \\ & t \in [t_{k-1}, t_k] \end{aligned} \tag{7}$$

An additional computational savings is obtained from the observation

$$\hat{\mathcal{R}}_{[t,t+\Delta t]}(X_0) = e^{At}\hat{\mathcal{R}}_{[0,\Delta t]}(X_0). \tag{8}$$

To obtain (8), we introduce the following notation. Given a set X and a transformation matrix T, we write the set Y obtained by applying T to each element in X as

$$Y = TX = \{y \mid y = Tx, \text{ for some } x \in X\}.$$

Clearly, a polytope defined by a set of vertices can be transformed by applying T to each vertex. On the other hand, a polytope described by a set of inequality constraints can be transformed by applying the result of the following lemma to each inequality constraint in the set.

Lemma 1 Given a set $X = \{x \mid c^T x \leq d\}$ and an invertible linear transformation T, the set $Y = TX$ can be written as $Y = \{y \mid \hat{c}^T y \leq d\}$ where $\hat{c} = (T^{-1})^T c$.

Pf. $y \in Y \iff x = T^{-1}y \in X \iff c^T(T^{-1})y \leq d$.

To prove (8), we compare the procedure for computing the flow pipe segments $\hat{\mathcal{R}}_{[0,\Delta t]}(X_0)$ and $\hat{\mathcal{R}}_{[t,t+\Delta t]}(X_0)$. For the time intervals $[0, \Delta t]$ and $[t, t+\Delta t]$, we construct the convex hulls

$$\begin{aligned} & \Phi_{[0,\Delta t]}(X_0) && = CH(V_0(X_0) \cup V_{\Delta t}(X_0)) \\ \text{and} \quad & \Phi_{[t,t+\Delta t]}(X_0) && = CH(V_t(X_0) \cup V_{t+\Delta t}(X_0)), \end{aligned}$$

respectively. By substituting (6) into the definition of $V_t(X_0)$, we see that

$$V_t(X_0) = e^{At}V_0(X_0).$$

Consequently,

$$V_{t+\Delta t}(X_0) = e^{At}e^{A\Delta t}V_0(X_0) = e^{At}V_{\Delta t}(X_0).$$

It follows that $\Phi_{[t,t+\Delta t]}(X_0) = e^{At}\Phi_{[0,\Delta t]}(X_0)$. Thus, if $\Phi_{[0,\Delta t]}(X_0) = POLY(C,d)$ then we have by Lemma 1 that $\Phi_{[t,t+\Delta t]}(X_0) = POLY(\hat{C},d)$ with $\hat{C} = Ce^{-At}$. For the time interval $[t, t+\Delta t]$, we use $\hat{C}$ and $\mathcal{R}_{[t,t+\Delta t]}(X_0)$ to solve (3).

By first rewriting (7) for $\hat{c}_i^T$ and time interval $[t, t+\Delta t]$ with $\hat{\tau}$ as the time variable and then substituting $\tau = \hat{\tau} - t$ and $\hat{c}_i^T = c_i^T e^{-At}$, we obtain

$$\begin{aligned} \max_{x_0,\tau} \quad & c_i^T e^{A\tau} x_0 \\ \text{s.t.} \quad & x_0 \in X_0 \\ & \tau \in [0, \Delta t] \end{aligned} \tag{9}$$

The above problem is exactly the problem corresponding to the i^{th} normal vector in C when (3) is being solved using C and $\mathcal{R}_{[0,\Delta t]}(X_0)$. Thus, we conclude that the solution d^* to (3) is the same for both flow pipe segments, implying that $\hat{\mathcal{R}}_{[0,\Delta t]}(X_0) = POLY(C, d^*)$ and $\hat{\mathcal{R}}_{[t,t+\Delta t]}(X_0) = POLY(\hat{C}, d^*)$. By Lemma 1, we have that $\hat{\mathcal{R}}_{[t,t+\Delta t]}(X_0) = e^{At}\hat{\mathcal{R}}_{[0,\Delta t]}(X_0)$.

The above result implies that our computational procedure for each flow pipe segment depends only on the size of the time step Δt. To compute the segment between time t_1 and t_2 with $\Delta t = t_2 - t_1$, we simply apply our computational procedure for the time interval $[0, \Delta t]$ and then transform the resulting polytope to the starting time of the segment by e^{At_1}. This suggests that the efficiency of the flow pipe computation can be improved by caching the resulting polytopes (before the transformation) for different Δt's. To compute each flow pipe segment, we first check whether there is already a result for the Δt for that segment, if so we retrieve the polytope and transform it to the starting time of the segment. If not, we compute the segment from scratch and store the data for future use.

An example of a flow pipe computed for a linear system is shown in Figure 3. The system matrix is

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -2 & -2 \end{bmatrix}$$

and the initial set X_0 defined by the vertices

$$v_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} v_2 = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} v_3 = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}, \text{ and } v_4 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}.$$

The total number of segments is 50 and the time step is uniform with $\Delta t_k = 0.1$. We only need to compute the polytope for the first segment. The other polytopes are obtained by transforming the first segment by $e^{A\Delta t_k}$ 49 times consecutively.

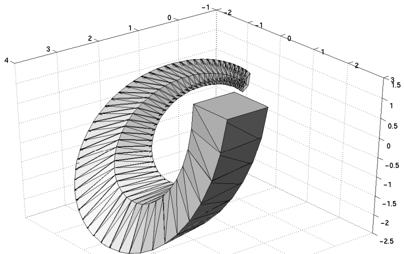


Figure 3: Flow pipe computed for a 3-D linear system

5 Application to Hybrid Systems

In this section, we illustrate the application of our computational method to hybrid system verification. The example hybrid system is represented by the *hybrid automaton* [5] in Figure 4.

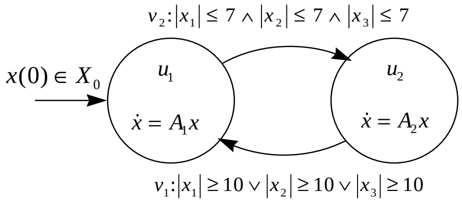


Figure 4: Example Hybrid System

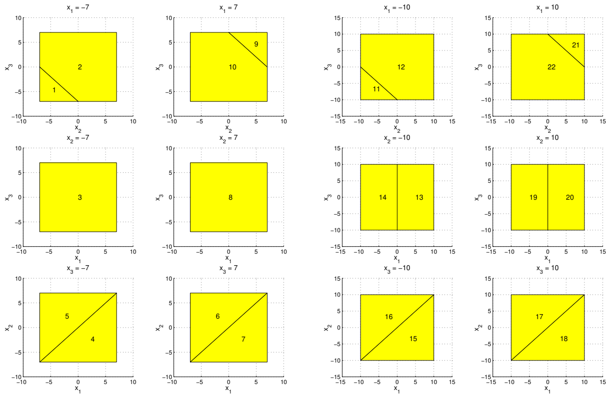
The continuous dynamics in each location is characterized by the LTI system $\dot{x} = A_i x$, $i = 1, 2$, where

$$A_1 = \begin{bmatrix} -1 & 1 & 1 \\ -1 & 0 & 0 \\ -1 & 1 & 0 \end{bmatrix} \text{ and } A_2 = \begin{bmatrix} -1 & 1 & 1 \\ 0.5 & -1 & 0 \\ -1 & 1 & 0 \end{bmatrix}.$$

Both system matrices are stable and it is assumed that every transition is taken as soon as its guard is enabled. For this hybrid system, our objective is to verify that there is only a finite number of switchings between the two locations starting from the initial location u_1 and any initial set of continuous states $X_0 \subseteq R^3$.

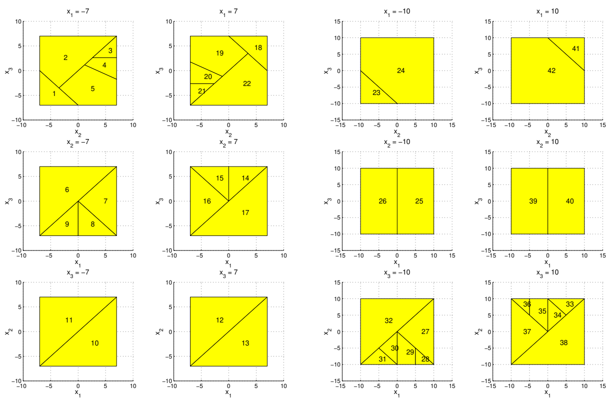
We perform the verification by constructing a series of *approximating automata* [2] which are conservative approximations to the hybrid system. Each automaton is constructed from a finite partition of the switching surfaces (the 3-D cubes $||x||_\infty = 7$ and $||x||_\infty = 10$). Each state corresponds to a member of the partition. A state transition is defined if the flow pipe computed from the source state set intersects the target state set. A transition to a special *equilibrium state* is also defined if the flow pipe converges to the equilibrium $x = 0$ without going through any switching surfaces completely. The matrices A_1 and A_2 are identified with states associated with the surfaces $||x||_\infty = 10$ and $||x||_\infty = 7$, respectively.

The initial partition, shown in Figure 5, is obtained by dividing each face of the cube according to the direction of the vector field (in or out of the parent cube). The automaton obtained from this partition violates the specification as it contains cycles, indicating the possibility of infinite switching. The automaton is then refined by splitting every patch that is a part of a cycle. The transitions are recomputed for the new automaton. After three refinements, we have the partition shown in Figure 6. For this partition, the approximating automaton contains no cycles and therefore satisfies the specification. Since the approximating automaton is a conservative approximation, we conclude that only finite switching can occur for any trajectory starting from the switching surfaces. Since A_1 is stable, the trajectory starting from any initial state $x(0)$ must reach either the equilibrium or the surface $||x||_\infty = 7$ where the hybrid system switches to u_2. Either way the hybrid system satisfies the specification.



(a) $||x||_\infty = 7$ (b) $||x||_\infty = 10$

Figure 5: Initial partition



(a) $||x||_\infty = 7$ (b) $||x||_\infty = 10$

Figure 6: Partition after 3 refinements

6 Discussion

The approximation obtained from our computation procedure is an outer approximation only if the optimization software gives us the global solution to (5) or (9). This result is not necessarily guaranteed as there may be multiple local maxima. Our heuristic for solving (5) is to choose the time step that is small enough so that the objective function is concave or convex, and to use multiple seeds. Future research in applying various global optimization techniques to (5) is needed.

For linear systems, as one might expect, we can do better. We rewrite (4) for the first segment as

$$\max_{t \in [0, \Delta t]} \quad \hat{g}_i(t) \tag{10}$$

where

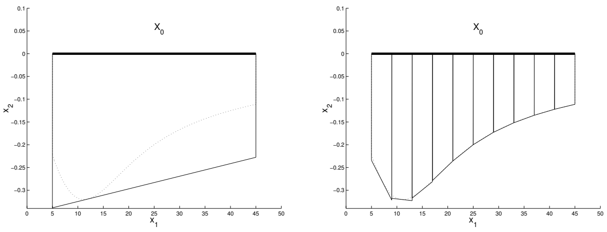
$$\hat{g}_i(t) = \max_{x \in \mathcal{R}_t(X_0)} \quad c_i^T x \tag{11}$$

Since $\mathcal{R}_t(X_0) = e^{At} X_0$, (11) is a linear program. Therefore, the global solution, $\hat{g}_i(t)$, can always be found. $\mathcal{R}_t(X_0)$ can be computed by transforming the inequality constraints for X_0 by e^{At} as discussed in Section 4. Thus, it only remains to find the global solution to (10). Again, if we use the small enough Δt, $\hat{g}_i(t)$ will be either concave or convex. Either way, the global solution is guaranteed to be found by using two seeds $t = 0$ and $t = \Delta t$ (the global maximum of a convex function over a polytope P occurs at some vertex of P [8]).

In the linear system case, one can see that the accuracy of the approximation can be improved by simply using smaller time steps. This is because the reachable set at any time $\mathcal{R}_t(X_0)$ is a polytope which is a linear transformation of X_0. The flow pipe is simply the union of all these polytopes over the time interval $[0, t_f]$. Each segment beginning at time t_k approaches $\mathcal{R}_{t_k}(X_0)$ as Δt_k gets smaller and smaller.

For non-linear systems, reducing Δt may not be enough to guarantee the approximation converges to the flow pipe as the following example illustrates. Consider the Van der Pol system discussed previously. Let the initial set $X_0 = \{5 \le x_1 \le 45, x_2 = 0\}$, a single segment flow pipe is computed as shown in Figure 7a. The dotted line marks the boundary of the flow pipe from $t = 0$ to $t = 0.05$. As we can see from the figure, the reachable set at time $t = 0.05$ is not a straight line connecting the trajectories emanating from the end points of X_0. With the nonlinearity in $\mathcal{R}_{0.05}(X_0)$, the last segment will not be able to bound $\mathcal{R}_{0.05}(X_0)$ arbitrarily closely. At best, it will approach the convex hull of $\mathcal{R}_{0.05}(X_0)$. To improve the approximation, we need to break X_0 into smaller sets and compute the flow pipe from each set separately as shown in Figure 7b. In the figure, each flow pipe has only one segment from $t = 0$ to $t = 0.05$.

It is worth noting that the approximation accuracy for the flow pipe segment for each time interval does not



(a) No partition (b) 10-subset partition

Figure 7: Improving accuracy by partitioning X_0

depend on the segments computed for previous or future time intervals. In fact, the segments can be computed in any order. In other methods, conservative approximation of the reachable sets at later time points depends on the approximation for the previous time points which causes the approximation error to grow with time.

On the other hand, the simulations embedded in the optimization routine for nonlinear systems must always start from $t = 0$. Thus, it will take longer and longer to compute the segments for later time intervals. However, the fact that the computation for each segment is independent facilitates parallelization of the computations, which may help alleviate the computation time problem.

References

[1] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. on Software Engineering*, 22(3):181–201, Mar 1996.

[2] A. Chutinan and B.H. Krogh. Computing approximating automata for a class of hybrid systems. *submitted to Mathematical Modeling of Systems: Special Issue on Discrete Event Models of Continuous Systems*, 1997.

[3] A. Chutinan and B.H. Krogh. Computing approximating automata for a class of linear hybrid systems. In *Hybrid Systems V*, Lecture Notes in Computer Science. Springer-Verlag, 1998.

[4] J.E.R. Cury, B.H. Krogh, and T. Niinomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Trans. on Automatic Control*, to appear in the Special Issue on Hybrid Systems, 1997. a preliminary version of this paper also appeared in *Proc. 34th IEEE CDC, pp. 1461-1466*.

[5] T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996. Invited tutorial.

[6] T.L. Johnson and M.E. Kaliski. Finite state control of continuous-state processes: Progress & prospects. In *Proc.* 29^{th} *IEEE Conference on Decision and Control*, pages 1910–1915, Dec 1992.

[7] E.K. Kornoushenko. Finite-automation approximation to the behavior of continuous plants. In *Automation and Remote Control*, number 12, pages 2068–2074, 1975.

[8] P.M. Pardalos and J.B. Rosen. *Constrained Global Optimization: Algorithms and Applications*, volume 268 of *Lecture Notes in Computer Science*. Springer-Verlag, 1987.

[9] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[10] A. Puri, P. Varaiya, and V. Borkar. ϵ-approximation of differential inclusions. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III: Verification and Control*, pages 362–376. Springer-Verlag, 1996.

[11] D.G. Wimpey. *Finite-State Control of Discrete-Time Coninuous Processes: An Automata Motivated Approach*. PhD thesis, Massachusetts Institute of Technology, June 1982.

[12] Feng Zhao. *Automatic Analysis and Synthesis of Controllers for Dynamical Systems Based on Phase-Space Knowledge*. PhD thesis, MIT Artificial Intelligence Laboratory, 1992.