Approximate Reachability Analysis of Piecewise-Linear Dynamical Systems*

Eugene Asarin[1], Olivier Bournez[2], Thao Dang[1], and Oded Maler[1]

[1] Verimag, Centre Equation, 2, av. de Vignate, 38610 Gières, France
{asarin,tdang,maler}@imag.fr
[2] Loria, Campus Scientifique, BP 239, 54506 Vandoeuvre les Nancy, France
Olivier.Bournez@loria.fr

Abstract. In this paper we describe an experimental system called $\boxed{\mathbf{d/dt}}$ for approximating reachable states for hybrid systems whose continuous dynamics is defined by linear differential equations. We use an approximation algorithm whose accumulation of errors during the continuous evolution is much smaller than in previously-used methods. The $\boxed{\mathbf{d/dt}}$ system can, so far, treat non-trivial continuous systems, hybrid systems, convex differential inclusions and controller synthesis problems.

1 Introduction

The problem of calculating reachable states for continuous and hybrid systems has emerged as one of the major problems in hybrid systems research [G96,GM98,DM98,KV97,V98,GM99,CK99,PSK99,HHMW99]. It constitutes a prerequisite for exporting algorithmic verification methodology outside discrete systems or hybrid systems with piecewise-trivial dynamics. For computer scientists it poses new challenges in treating continuous functions and their approximations and in applying computational geometry techniques to problems in higher dimensional spaces. For control theorists and engineers the problem suggests a fresh way of looking at systems with under-specified inputs and increases their awareness to some practical computational aspects of controller design.

In this paper we describe an experimental system called $\boxed{\mathbf{d/dt}}$ which can approximate reachable states for hybrid systems whose continuous dynamics is defined by linear differential equations. The performance is much better than the more general method of "face-lifting" we have used in the past [DM98].

The rest of the paper is organized as follows. In section 2 we define the problem of calculating reachable states and suggest a general procedure which solves it iteratively. The basic computation step of the procedure cannot be performed exactly and in section 3 we describe an over-approximation scheme for linear systems, having the advantage of not propagating errors from one step to another.

* This work was partially supported by the European Community Esprit-LTR Project 26270 VHS (Verification of Hybrid systems), the French-Israeli collaboration project 970MAEFUT5 (Hybrid Models of Industrial Plants) and the Russian Foundation for Basic Research under grant 97-01-00692.

Extensions of the algorithm to deal with hybrid systems, controller synthesis and continuous disturbances are described in section 4 along with several examples.

2 The Basic Problem

Let $T = \mathbb{R}_+$ be a *time domain*, let X be a bounded subset of $\mathbb{R}^n$ and consider a continuous dynamical system $\mathcal{A}$ over X defined by the equation $\dot{\mathbf{x}} = f(\mathbf{x})$. We use the notation $\mathbf{x} \xrightarrow{t} \mathbf{x}'$ to indicate that the solution α of the equation with $\mathbf{x}$ as an initial condition satisfies $\alpha[t] = \mathbf{x}'$. In words we say that $\mathbf{x}'$ is reachable from $\mathbf{x}$ in time t.

Definition 1 (Successors). *Let $\mathcal{A}$ be a dynamical system defined by $\dot{\mathbf{x}} = f(\mathbf{x})$. The successor operator $\delta : 2^X \to 2^X$ is defined for a subset F of X and an interval $I \subseteq T$ as:*

$$\delta_I(F) = \{\mathbf{x}' : \exists \mathbf{x} \in F \; \exists t \in I \; \mathbf{x} \xrightarrow{t} \mathbf{x}'\}$$

We use the notation δ_r for $\delta_{[r,r]}$ (states reachable after exactly r time), δ for $\delta_{[0,\infty)}$ (all states reachable after any non-negative amount of time) and $\delta_I(\mathbf{x})$ for $\delta_I(\{\mathbf{x}\})$. Note that δ has the semi-group property, i.e. $\delta_{I_2}(\delta_{I_1}(F)) = \delta_{I_1 \oplus I_2}(F)$ where $\oplus$ is the Minkowski sum, and in particular $\delta_{[0,r_2]}(\delta_{[0,r_1]}(F)) = \delta_{[0,r_1+r_2]}(F)$. In certain cases when the differential equation admits a closed-form solution, one may characterize $\delta(F)$ symbolically by a formula and then try to obtain a closed-form solution by quantifier elimination. However, this works in rather exceptional cases (see for example [CV95,PLY99]). Instead we propose a numerical algorithm which works by discretizing time into multiples of a fixed time step r. The abstract algorithm for calculating $\delta(F)$ is the following:

Algorithm 1 (Exact Calculation of $\delta(F)$)

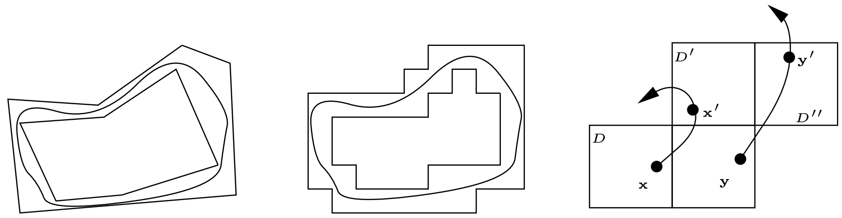
$P^0 := F$
repeat
  $P^{k+1} := P^k \cup \delta_{[0,r]}(P^k)$
until $P^{k+1} = P^k$

In order for a function to be computable by a discrete device its domain and range need an effective representation as well as an effective and terminating procedure which takes the representation of any element of the domain and transforms it to a representation of its image by the function. For example, functions over the integers can be computed by applying well-known algorithms for addition and multiplication to unary, binary or decimal representations of numbers. The mathematical real numbers pose a special problem in this respect, a problem which we do not address here but assume to be solved for all practical purposes. Our main concern here is to compute functions over *subsets of X*. From this perspective Algorithm 1, when applied *exactly* suffers from the following two problems:

1. The exact calculation of $\delta_{[0,r]}$ is not more feasible than the calculation of the whole δ.
2. Even if $\delta_{[0,r]}$ was computable, the algorithm usually does not terminate after a finite number of steps.

To overcome these problems we resort to approximate calculation of $\delta_{[0,r]}$ and δ. In order to be effective, i.e. to do any computation at all, we can replace 2^X by a countable and effectively enumerable subset C whose union gives X, e.g. the set of all polyhedra with rational vertices. Elements of 2^X not in C are thus either under- or over-approximated (see Figure 1-(a)). The type of approximation which is used depends on the problem to be solved. If we want to characterize all the possible behaviors starting from a given initial set, an over-approximation is used. If we want to characterize the set of states from which a property can be satisfied, under-approximation is preferred.

An effective approximation of Algorithm 1 can thus be implemented by replacing all the operations (Boolean operations, equivalence testing and calculation of $\delta_{[0,r]}$) by their approximated versions.[1] If the approximate algorithm terminates, the result is an over-approximation of $\delta(F)$.



(a) (b) (c)

Fig. 1. (a) A set F and over- and under-approximated by polyhedra. (b) The same set approximated by orthogonal polyhedra. (c) Accumulation of errors in naïve approximate computation.

The termination of the procedure, however, cannot be guaranteed since there are infinitely many polyhedral sets. Moreover, the implementation is very complicated because the sets P^k can be very complex non-convex polyhedra for which there is no useful canonical form and the test $P^{k+1} = P^k$ is very expensive. Hence we restrict further the class of sets to be what we call *griddy polyhedra*, i.e. 2^B where B is the set of all closed unit hypercubes with integer leftmost corners. Using this finite class of sets guarantees convergence of Algorithm 2 (provided we restrict our analysis to bounded domains) and allows

[1] Note that if the class C is closed under Boolean operations, only $\delta_{[0,r]}(F)$ needs to be approximated. This holds for arbitrary polyhedral sets but not for convex polyhedra or ellipsoids.

us to benefit from a relatively-efficient canonical representation for both convex and non-convex sets [BMP99], supported by an experimental software package. The price, however, for using orthogonal polyhedra is that the quality of the approximation they provide in terms of Haussdorf distance per vertex is poorer than that of arbitrary polyhedra (zero-order vs. first-order in the approximation jargon) but such a compromise seems unavoidable.

A naïve approximate version of Algorithm 1 is guaranteed to converge to a superset of $\delta(F)$ after finitely many steps. However, the distance between the result and $\delta(F)$ might be too big for the result to be useful. The reason is that over-approximation errors accumulate dramatically as illustrated in Figure 1-(c) where we try to calculate successors of the set D. Since $\mathbf{x}'$ is reachable from $\mathbf{x}$ we must include the whole box D' in the set of successors. This box contains points such as $\mathbf{y}$ not really reachable from D, which bring in the next iteration new points, such as $\mathbf{y}'$, and we end up adding boxes such as D'' which are not reachable from D at all. This over-approximation error can propagate fast and the result might cover the whole space unless some hardly-formalizable hacking is used [DM98,GM99]. Similar phenomena are exhibited, for example, in abstract interpretation of programs over the integers [CC92] where over-approximation is called *widening*. This is why there is not much hope in finding finite quotients of continuous systems, except for special cases such as timed automata [AD94].

Here we need to find the right compromise between the desire to converge and the accumulation of errors. We propose a method, specialized for linear systems of the form $\dot{\mathbf{x}} = A\mathbf{x}$ which achieves this trade-off. The basic idea here is to separate the accumulation and storage of states reachable in one step (and those must contain an approximation error) from the computation of states reachable in the next step (see also [GM99]). The main attraction of this method compared to traditional ways to treat linear systems is in its adaptability to hybrid systems and to systems with under-specified input.

3 The Approximate Method for Linear Systems

Let $conv(\{\mathbf{x}_1, \ldots, \mathbf{x}_m\})$ be the convex hull of a set of points, i.e. $\{\mathbf{x} : \mathbf{x} = \lambda_1\mathbf{x}_1 + \cdots, \lambda_m\mathbf{x}_m\}$ for non-negative λ_i whose sum is 1. For linear systems we have $\delta_t(\mathbf{x}) = e^{At}\mathbf{x}$ and the matrix exponential, as a linear operator, preserves convexity:

$$\delta_t(conv(\{\mathbf{x}_1, \ldots, \mathbf{x}_m\})) = conv(\{\delta_t(\mathbf{x}_1), \ldots, \delta_t(\mathbf{x}_m)\}).$$

This means that for a convex set $F = conv(\mathbf{V})$ where $\mathbf{V} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, and for every t, the states reachable from F can be determined by the states reachable from $\mathbf{V}$ (see Figure 2-(a)). We exploit this property to approximate $\delta_{[0,r]}(conv(\mathbf{V}))$ based on the set of points $\mathbf{V} \cup \delta_r(\mathbf{V})$ where $\delta_r(\mathbf{V})$ is computed from $\mathbf{V}$ by a finite number of matrix exponentiations or numerical integration steps. Our approximation scheme consists of three steps:

1. Compute $G = conv(\mathbf{V}, \delta_r(\mathbf{V}))$ (see Figure 2-(b)). This set is an approximation of $\delta_{[0,r]}(conv(\mathbf{V}))$ but neither an over-approximation nor under-

approximation. The convex-hull algorithm provides us with information concerning the orientation of the faces which is used in the next step.[2]
2. Push the faces of G outward to obtain a bloated convex polyhedron G' which is guaranteed to contain the required set (Figure 2-(c)). The amount of pushing is determined by the time step r and the matrix A (see the analysis in the appendix). Pushing inward will result in an under-approximation.
3. Over-approximate G' by a griddy polyhedron $\delta'_{[0,r]}(F)$ (Figure 2-(d)).

The approximate algorithm for calculating $\delta(F)$ for $F = conv(\mathbf{V})$ is defined below:

Algorithm 2 (Approximate Calculation of $\delta(F)$ for Linear Systems)

$P^0 := F$; $\mathbf{V}^0 := \mathbf{V}$; k:=0;
repeat
  $k \quad := k + 1$;
  $\mathbf{V}^k := \delta_r(\mathbf{V}^{k-1})$;
  $G^k := conv(\mathbf{V}^{k-1} \cup \mathbf{V}^k)$;
  $G^k := bloat(G^k)$;
  $G^k := griddy(G^k)$;
  $P^k := P^{k-1} \cup G^k$
until $P^k = P^{k-1}$

There are two types of errors accumulated in the process of calculating P^k: from the actual set to its bloated convex hull and from there to the griddy polyhedron. However these errors do not propagate to the next step which computes P^{k+1} based on $\mathbf{V}^k \cup \mathbf{V}^{k+1}$ and not on P^k (Figure 2-(e)). Recall that our orthogonal polyhedra package [BMP99] maintains $\delta'_{[0,2r]}(F)$ as a *single* canonical object and *not* as a union of convex polyhedra or ellipsoids (Figure 2-(f)). The algorithm can be fine-tuned by changing the time step r and the size of the hypercubes.

Result 1 (Computation of Reachable States for Linear Systems) *There exists an implemented algorithm for over-approximating the reachable sets of systems defined by linear differential equations.*

The reason this result is not a theorem is due to the following facts:

1. There is always a trivial over-approximation of any subset F of X, namely X itself.
2. The smallest polyhedral or griddy set which contains $\delta(F)$ is as impossible to compute as $\delta(F)$.
3. Like in many other numerical problems, the best upper-bounds which can be easily proved on the approximation error are much larger than what happens in practice.

[2] We use the convex-hull algorithm supplied with the LEDA library [MV99].

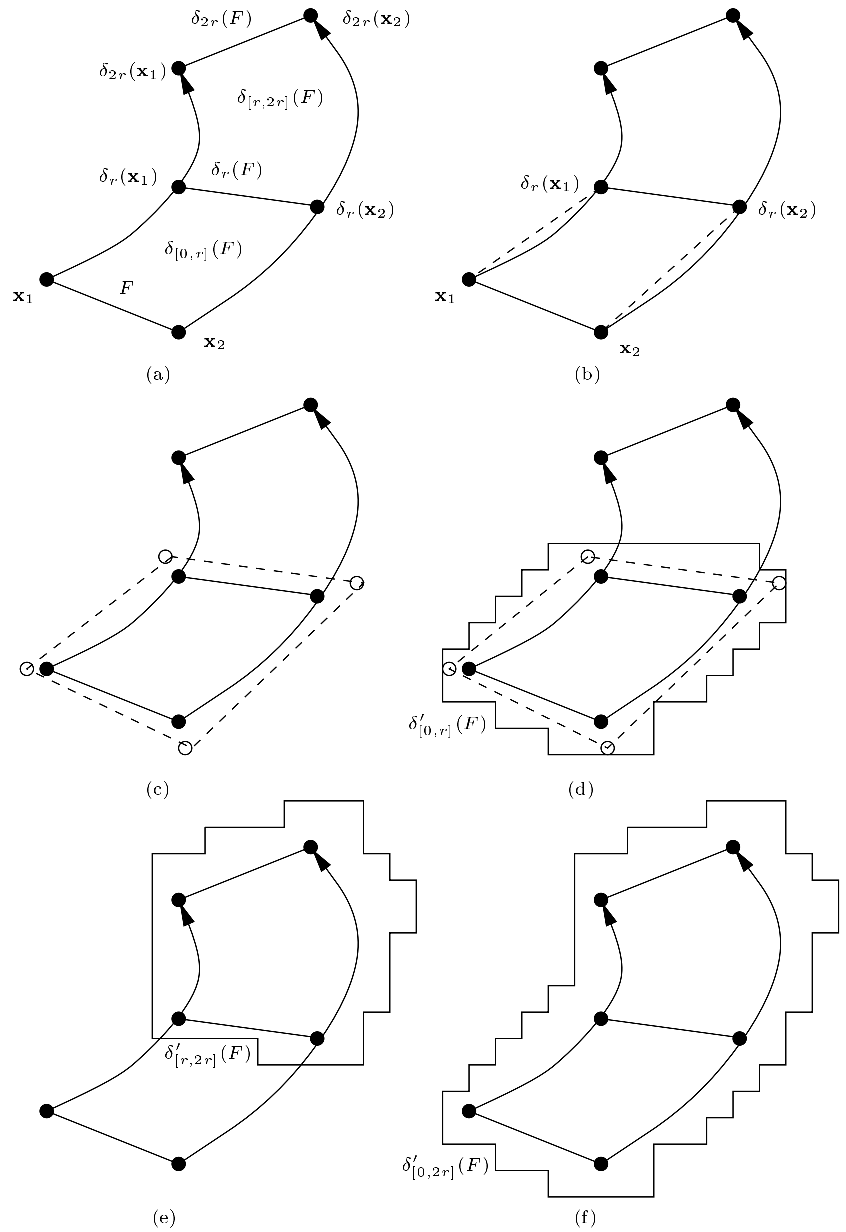


Fig. 2. (a) A set $F = conv(\{\mathbf{x}_1, \mathbf{x}_2\})$ and its exact successors for time intervals $[0, r]$ and $[r, 2r]$. (b) Approximating $\delta_{[0,r]}(F)$ by convex hull. (c) Bloating the convex polyhedron to obtain a polyhedral over-approximation. (d) Rectangulating the polyhedron into $\delta'_{[0,r]}(F)$. (e) Repeating the same procedure in the next time step to obtain $\delta'_{[r,2r]}(F)$. (f) The accumulated states $\delta'_{[0,2r]}(F) = \delta'_{[0,r]}(F) \cup \delta'_{[r,2r]}(F)$.

So let us be content with the fact that the method gives reasonable approximation in rather short time. So far we were able to calculate rather easily the reachable states of non-trivial systems with up to 6 dimensions (in fact, the measure of complexity for such problems depends on the dimensionality, the coupling of the variables and the granularity of the discretization). Figure 3 shows the states reachable from

$$F = [0.025, 0.05] \times [0.1, 0.15] \times [0.05, 0.1]$$

by the 3-dimensional system defined by

$$A = \begin{pmatrix} -1.0 & -4.0 & 0.0 \\ 4.0 & -1.0 & 0.0 \\ 0.0 & 0.0 & 0.5 \end{pmatrix}$$

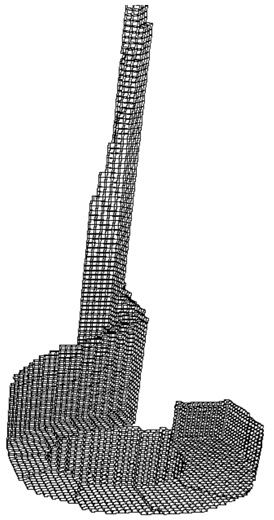


Fig. 3. Calculating reachable states for a 3-dimensional system.

4 Extensions and Applications

4.1 Piecewise-Linear Systems

For purely continuous linear systems there are classical methods, more efficient than ours, for solving certain problems such as stability or controller synthesis. However the main advantage of our approach is manifested in the analysis and controller synthesis for linear hybrid automata which may switch between several "modes" and hence define piecewise-linear dynamical systems. We demonstrate the adaptation of our method to such systems informally using the hybrid automaton of Figure 4, which consists of two continuous variables, and two discrete states. In each discrete state the continuous variables evolve according to the corresponding linear dynamics and when some switching conditions (transition guards) are satisfied, the system moves from one state to another.

Starting from an initial set (q_0, F) the reachable states are calculated as follows: we apply our procedure to F with the A_0 dynamics and calculate forward $\delta^0(F)$. Then we calculate the intersection of the result with the guard to obtain a set F', move to state q_1 with F' as the set of initial states, calculate $\delta^1(F')$ and so on and so forth. This method is similar to the one used in tools such as KRONOS [DOTY96] for timed automata and HyTech [HHW97] for hybrid automata with constant derivatives [ACH+95].

The main technical difficulty in applying our vertex-based approximation technique to such systems is that not all trajectories departing from the vertices reach a transition guard simultaneously (some may not reach it at all). Hence we have to calculate $\delta(F)$ and intersect it with the guard to obtain the new initial set. Unfortunately, this set is already an over-approximation and, moreover, it might have many vertices and the reduction of their number might require further approximation. The bottom line is that we can avoid propagation of over-approximation errors during the continuous evolution but not while doing transitions.

An example run of $\boxed{\mathbf{d/dt}}$ on the hybrid automaton of Figure 4 where

$$A_0 = \begin{pmatrix} -2.0 & -3.0 \\ 3.0 & -2.0 \end{pmatrix} \quad \text{and} \quad A_1 = \begin{pmatrix} 0.0 & -0.6 \\ 3.0 & 0.0 \end{pmatrix}$$

and the initial set is $F = \{q_0\} \times [0.3, 0.6] \times [-0.2, 0.2]$, appears in Figure 5. Initially the successors by A_0 (a "center" dynamics) are calculated until they all intersect the guard $x_1 \leq -0.15$ (a). Then dynamics A_1 is applied, shrinking the set until intersection with the guard $x_1 \geq -0.02$ (b). From this guard the dynamics A_0 induces a "ring" of states which stay in q_0 forever (c).

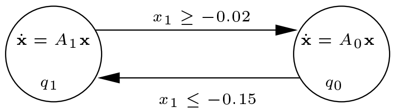
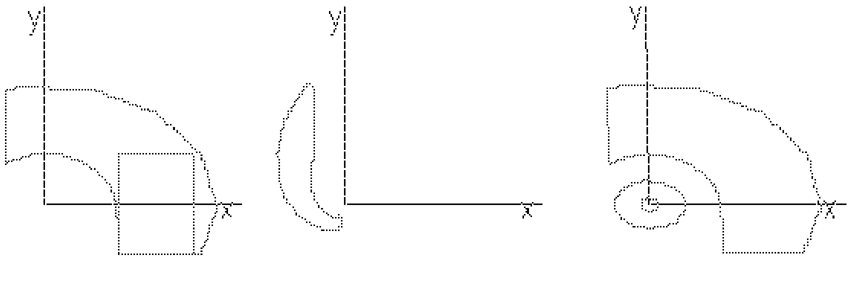


Fig. 4. A hybrid automaton.

4.2 Under-Approximation, Backward Reachability and Control

The δ operator is a basic ingredient in forward reachability analysis. Other verification and synthesis problems require different variants of this operator.



q_0 $\quad$ q_1 $\quad$ q_0

(a) (b) (c)

Fig. 5. The 3 stages in the calculation of $\delta(F)$ of the hybrid automaton of Figure 4.

The reader might have guessed that calculating under-approximations is done by a slight variation of the algorithm, i.e. pushing the faces of the polyhedron *inside* and finding an orthogonal under-approximation. Backward reachability, that is, finding all the points from which a set F is reachable can be performed by computing δ for the reversed system $\dot{\mathbf{x}} = -A\mathbf{x}$.

For the purpose of controller synthesis for hybrid systems [ABDPM00] we need an under-approximation of the "F *Until* G" operator, which returns the points from which you can stay within the set F either forever or until you reach a set G (which is typically the guard of a transition to another state). A similar operator is needed for analyzing hybrid systems with invariants. Consider $F = [-0.1, 0.1] \times [-0.030.1]$, $G = [0.02, 0.06] \times [-0.05, -0.02]$ and a dynamics

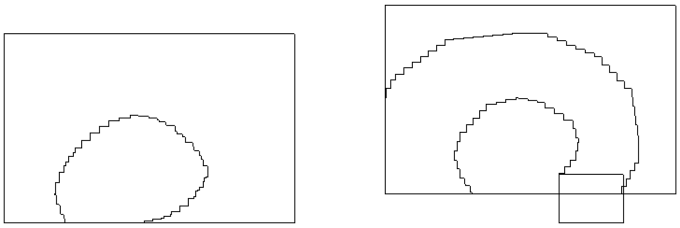
$$A = \begin{pmatrix} -0.5 & 4.0 \\ -3.0 & -0.5 \end{pmatrix}$$

The two parts of F *Until* G, as calculated by $\boxed{\mathbf{d/dt}}$ appear in Figure 6.

4.3 Continuous Disturbances

Consider systems of the form $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$ where $\mathbf{u}$ ranges inside a convex set U. It has been suggested in [V98] to use the maximum principle from optimal control to find $\delta_r(F)$ of a convex set $F = conv(\mathbf{V})$ under all possible input signals. We have implemented this procedure and incorporated it into our system. We have tested it on a 4-dimensional example adapted from example 4.5.1 of [KV97], pp. 279-285, where ellipsoids are used instead of polyhedra. The system is defined by:

$$A = \begin{pmatrix} 0.0 & 1.0 & 0.0 & 0.0 \\ -8.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & -4.0 & 0.0 \end{pmatrix} \qquad B = 1$$



(a) (b)

Fig. 6. The *Until* operator: (a) The states which can stay in F forever. (b) The states which can stay in F until reaching G.

$$F = [0.02.0] \times [-1.01.0] \times [0.0, 2.0] \times [-1.0, 1.0]$$

$$U = [-0.5, 0.5] \times [-0.005, 0.005] \times [-0.5, 0.5] \times [-0.005, 0.005]$$

In Figure 7 one can see the evolution of the projection on dimensions 3 and 4 over time, similar to the results in [KV97]. Further work on these technique might suggest effective methods for approximate strategies for differential games.

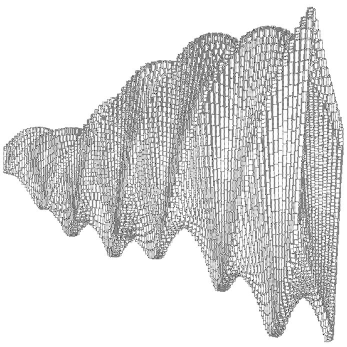


Fig. 7. The evolution of a 4-dimensional convex differential inclusion over time (projected on dimensions 3 and 4).

5 Discussion

In this work we have advanced the state-of-the-art in computer-aided reachability analysis for continuous and hybrid systems. We have implemented the tool $\boxed{\mathbf{d/dt}}$ and tested it over reproducible non-trivial examples. We are currently investigating various improvements and studying the trade-offs between accuracy and computational efficiency. We hope that such techniques and tools will be used in the future by control engineers.

References

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine, The Algorithmic Analysis of Hybrid Systems, *Theoretical Computer Science* 138, 3–34, 1995.

[AD94] R. Alur and D.L. Dill, A Theory of Timed Automata, *Theoretical Computer Science* 126, 183–235, 1994.

[ABDPM00] E. Asarin, O. Bournez, T. Dang, A. Pnueli and O. Maler, Effective Synthesis of Switching Controllers for Linear Systems, submitted for publication, 2000.

[BMP99] O. Bournez, O. Maler and A. Pnueli, Orthogonal Polyhedra: Representation and Computation, in [VS99], 46-60.

[CV95] K. Cerans and J. Viksna, Deciding Reachability of Planar Multi-polynomial systems, in R. Alur, T.A. Henzinger and E.D. Sontag (Eds.), *Hybrid Systems III*, 389-400, LNCS 1066, Springer, 1996.

[CK99] A. Chutinan and B.H. Krogh, Verification of Polyhedral Invariant Hybrid Automata Using Polygonal Flow Pipe Approximations, in [VS99] 76-90.

[CC92] P. Cousot and R. Cousot, Abstract Interpretation and Application to Logic Programs, *Journal of Logic Programming*, 103-179, 1992.

[DM98] T. Dang, O. Maler, Reachability Analysis via Face Lifting, in T.A. Henzinger and S. Sastry (Eds), *Hybrid Systems: Computation and Control*, LNCS 1386, 96-109, Springer, 1998.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine, The Tool KRONOS, in R. Alur, T.A. Henzinger and E. Sontag (Eds.), *Hybrid Systems III*, LNCS 1066, 208-219, Springer, 1996.

[G96] M.R. Greenstreet, Verifying Safety Properties of Differential Equations, in R. Alur and T.A. Henzinger (Eds.), *Proc. CAV'96*, LNCS 1102, 277-287, 1996.

[GM98] M.R. Greenstreet and I. Mitchell, Integrating Projections, in T.A. Henzinger and S. Sastry (Eds), *Hybrid Systems: Computation and Control*, LNCS 1386, 159-174, Springer, 1998.

[GM99] M.R. Greenstreet and I. Mitchell, Reachability Analysis Using Polygonal Projections, in [VS99] 76-90.

[HHMW99] T.A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi, Beyond HyTech: Hybrid System Analysis Using Interval Numerical Methods, AAAI Spring Symposium on Hybrid Systems, Stanford University, 1999.

[HHW97] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi, HyTech: A Model Checker for Hybrid Systems, *Software Tools for Technology Transfer* 1, 110-122, 1997.

[KV97] A. Kurzhanski ans I. Valyi, *Ellipsoidal Calculus for Estimation and Control*, Birkhauser, 1997.

[MV99] K. Mehlhorn and St. Näher, The LEDA Platform of Combinatorial and Geometric Computing, Cambridge University Press, 1999.

[PLY99] G. Pappas, G. Lafferriere and S. Yovine, A New Class of Decidable Hybrid Systems, in [VS99] 29-31.

[PSK99] J. Preussig, O. Stursberg and S. Kowalewski, Reachability Analysis of a Class of Switched Continuous Systems by Integrating Rectangular Approximation and Rectangular Analysis, in [VS99] 208-222.

[VS99] F. Vaandrager and J. van Schuppen (Eds.), *Hybrid Systems: Computation and Control*, LNCS 1569, Springer, 1999.

[V98] P. Varaiya, Reach Set Computation using Optimal Control, *Proc. KIT Workshop*, Verimag, Grenoble, 1998.

Appendix: Conservative Approximation

As we have already mentioned when describing the approximate method for linear systems, the set $G = conv(\mathbf{V}, \delta_r(\mathbf{V}))$ is not an over-approximation of $\delta_{[0,r]}(conv(\mathbf{V}))$ and should be replaced by its ϵ-neighborhood (or something bigger) in order to become such an over-approximation. Here we calculate the ϵ that should be used.

Consider an arbitrary point $p_0 \in conv(\mathbf{V})$ and a trajectory p_t starting from this point. We have $p_r = e^{rA}p_0$. This point belongs to $\delta_r(\mathbf{V})$ and hence to G. By convexity so does all the line segment $[p_0, p_r]$. Let us estimate now the distance between points of the true trajectory p_t for $t \in [0, r]$ and this line segment. In fact p_t may be approximated by linear interpolation between p_0 and p_r. The result of this interpolation is

$$\hat{p}_t = p_0 + \frac{t}{r}(p_r - p_0), \quad 0 \leq t \leq r$$

and by construction it belongs to the segment $[p_0, p_r]$. The error of this interpolation can be written as follows:

$$\epsilon(p_0, t) = ||\hat{p}_t - p_t|| = ||p_0 + \frac{t}{r}(e^{rA} - I)p_0 - e^{tA}p_0||.$$

Since

$$e^{tA} = I + At + \frac{1}{2}A^2t^2 + \sum_{i=3}^{\infty}\frac{1}{n!}A^it^i$$

and $0 \leq t \leq r$ we find after obvious simplifications the bound of the error:

$$\epsilon(p_0, t) \leq \epsilon = M\frac{1}{8}||A||^2r^2 + M\sum_{i=3}^{\infty}\frac{1}{n!}||A||^ir^i,$$

where M is a constant bounding the norm $||p_0||$.

Hence, for every r, one can find a $\epsilon = O(r^2)$ such that all the points reachable from $conv(\mathbf{V})$ in time r are in ϵ-neighborhood of G. In order to over-approximate the set we just replace G by its ϵ-neighborhood.