Validating a Hamilton-Jacobi Approximation to Hybrid System Reachable Sets[*]

Ian Mitchell[1][**], Alexandre M. Bayen[2], and Claire J. Tomlin[2]

[1] Scientific Computing and Computational Mathematics Program,
Gates 2B, Stanford University, Stanford, CA, 94305, USA
mitchell@sccm.stanford.edu
[2] Department of Aeronautics and Astronautics,
Durand 250, Stanford University, Stanford, CA, 94305, USA
{bayen,tomlin}@stanford.edu

Abstract. We develop a general framework for solving the hybrid system reachability problem, and indicate how several published techniques fit into this framework. The key unresolved need of any hybrid system reachability algorithm is the computation of continuous reachable sets; consequently, we present new results on techniques for calculating numerical approximations of such sets evolving under general nonlinear dynamics with inputs. Our tool is based on a local level set procedure for boundary propagation in continuous state space, and has been implemented using numerical schemes of varying orders of accuracy. We demonstrate the numerical convergence of these schemes to the viscosity solution of the Hamilton-Jacobi equation, which was shown in earlier work to be the exact representation of the boundary of the reachable set. We then describe and solve a new benchmark example in nonlinear hybrid systems: an auto-lander for a commercial aircraft in which the switching logic and continuous control laws are designed to maximize the safe operating region across the hybrid state space.

1 Introduction

The focus of this paper is the development and numerical validation of a computational tool to perform as exact as possible reachability computation and controller synthesis for nonlinear hybrid systems. As such, we draw on our previous work in which we characterized the boundary of the reachable set of a hybrid system as the zero level set of the viscosity solution of a particular Hamilton-Jacobi equation [1], and in which we showed that it was feasible to compute this zero level set using so-called "level set methods" [2]. The current paper reflects our progress in the development of a general purpose tool for this reachable

[*] Research supported by DARPA under the Software Enabled Control Program (AFRL contract F33615-99-C-3014), by a Frederick E. Terman Faculty Award, and by a graduate fellowship provided by the Délégation Générale pour l'Armement in France.

[**] corresponding author

M.D. Di Benedetto, A. Sangiovanni-Vincentelli (Eds.): HSCC 2001, LNCS 2034, pp. 418–432, 2001.
© Springer-Verlag Berlin Heidelberg 2001

set computation—the core of which is a new variant of a “local level set” algorithm that more efficiently computes a more accurate representation of the reachable set boundary. In addition, we demonstrate the numerical convergence of our computation by analyzing the results as the continuous state space grid is made finer, a standard method of validation for scientific computing codes. In this way, we show that high accuracy can be achieved at the cost of increased computational time and space. We illustrate our tool on a single mode aircraft conflict resolution example [2,3], as well as on a new benchmark example of a six mode commercial aircraft auto-lander, which exhibits nondeterminism and cycles in its discrete behavior.

Our motivation for this project stems from the belief that for many applications of hybrid systems, it is important to be able to accurately represent the reachable set. We have dealt primarily in the safety verification of avionic systems, where accurate representation of the safe region of operation translates into the ability to operate the system closer to the boundaries of that region, at a higher performance level than previously allowed. For very high dimensional state spaces, additional logic (such as projection operators) or new techniques (such as convex overapproximations) will be needed; however, our results in this paper show that it is feasible to do exacting computation for hybrid systems with nonlinear continuous dynamics in three continuous state dimensions and six discrete modes, and we believe it will be feasible to extend this up to five continuous dimensions and large numbers of discrete modes.

2 Reachability for Hybrid Systems

Assuming that tools for discrete and continuous reachability are available—we postpone to subsequent sections the problems of creating such tools—computing reachable sets for hybrid systems requires keeping track of the interplay between these discrete and continuous tools. In this section we summarize the general framework for handling this interaction (following [1]), and we show how various hybrid system reachability algorithms described in the literature fit into this framework.

Fundamentally, reachability analysis in discrete, continuous or hybrid systems seeks to partition states into two categories: those that are reachable from the initial conditions, and those that are not. We will label these two sets of states G and $\mathsf{E} = \mathsf{G}^c$ respectively.

Any inputs to the hybrid automata are assumed to lie in bounded sets and to have the goal of locally maximizing or minimizing the reachable set: at each iteration, the reachability algorithm chooses values for inputs ξ_{G} that maximize the size of G and values for inputs ξ_{E} that minimize the size of G (and hence maximize the size of E). Any nondeterminism in the transition relation is also utilized to consistently maximize or minimize G, depending on the goal of the reachability computation. For hybrid automata, the discrete inputs σ and continuous inputs ν can be assigned to the two categories $\xi_{\mathsf{G}} = (\sigma_{\mathsf{G}}, \nu_{\mathsf{G}})$ and $\xi_{\mathsf{E}} = (\sigma_{\mathsf{E}}, \nu_{\mathsf{E}})$ according to whether they seek to maximize or minimize G.

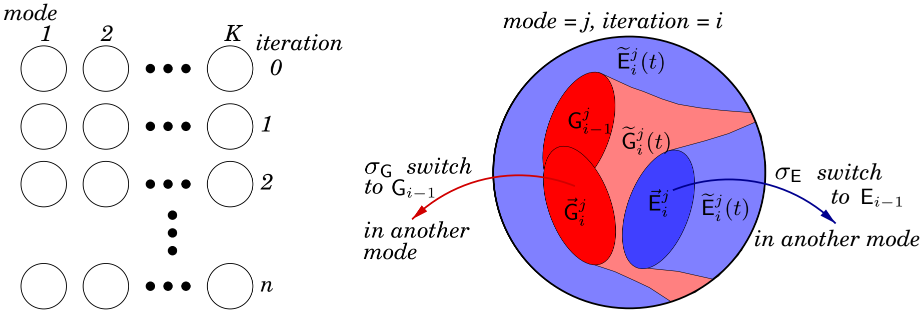
mode
1
2
K
iteration
0
1
2
n
mode = j, iteration = i
$\widetilde{\mathsf{E}}_i^j(t)$
G_{i-1}^j
$\widetilde{\mathsf{G}}_i^j(t)$
σ_G switch to G_{i-1}
in another mode
$\vec{\mathsf{G}}_i^j$
$\vec{\mathsf{E}}_i^j$
$\widetilde{\mathsf{E}}_i^j(t)$
σ_E switch to E_{i-1}
in another mode

Fig. 1. Iterative Reachability Algorithm: Showing detail of iteration for discrete mode k at iteration i.

The reachability computation follows an iterative, two stage algorithm shown graphically in Figure 1. The outer iteration computes reachability over the discrete switches, producing iterates G_i and E_i at iteration $i = 1, 2, \ldots$. The inner iteration runs a separate continuous reachability problem in each of the discrete modes $j = 1, 2, \ldots K$ to compute the estimates G_i^j and E_i^j. We define the "switch" sets

- $\vec{\mathsf{G}}_i^j$ contains all states in mode j from which a discrete transition to a state in G_{i-1} (typically a state in another mode) can be forced to occur through the application of a discrete input σ_G; these states will be defined by the invariant of mode j and the guards of the transitions from mode j.
- $\vec{\mathsf{E}}_i^j$ contains all states from which a discrete transition to a state in E_{i-1} can be forced to occur through the application of a discrete input σ_E; these states are also defined by the invariant of mode j and the guards of transitions from mode j.

Then the goal of the continuous reachability tool is to identify the "flow" sets

- $\widetilde{\mathsf{G}}_i^j(t)$ contains states from which for all ν_E there exists ν_G that will force the resulting trajectory to flow into $\mathsf{G}_{i-1}^j \cup \vec{\mathsf{G}}_i^j$ within time t.
- $\widetilde{\mathsf{E}}_i^j(t)$ contains states from which there exists ν_E that for all ν_G will force the resulting trajectories to flow into $\vec{\mathsf{E}}_i^j$ within time t or to stay outside of $\mathsf{G}_{i-1}^j \cup \vec{\mathsf{G}}_i^j$ for at least time t.

Note that in some problems the order of the existential and universal quantifiers in the definition above must be reversed. Given these sets,

$$\mathsf{G}_i^j = \lim_{t\to\infty} \widetilde{\mathsf{G}}_i^j(t), \qquad \mathsf{G}_i = \bigcup_{j=1}^{K} \mathsf{G}_i^j, \qquad \mathsf{G} = \lim_{i\to\infty} \mathsf{G}_i,$$

$$\mathsf{E}_i^j = \lim_{t\to\infty} \widetilde{\mathsf{E}}_i^j(t), \qquad \mathsf{E}_i = \bigcup_{j=1}^{K} \mathsf{E}_i^j, \qquad \mathsf{E} = \lim_{i\to\infty} \mathsf{E}_i,$$

where G_0^j is the set of initial conditions of the reachability problem and $\mathsf{E}_0^j = (\mathsf{G}_0^j)^c$. Simple modifications of this algorithm suffice to solve finite time reachability problems.

The procedure described above, developed in [1,3], was motivated by the work of [4,5] for reachability computation and controller synthesis on timed automata, and that of [6] for controller synthesis on linear hybrid automata. In that development the reachability problem's objective was to determine E—the largest controllable invariant subset of the state space—by computing the set of states G which were reachable in backwards time from the set of predefined unsafe states. In terms of the definitions above, control inputs from this problem lie in ξ_{E} and disturbance inputs in ξ_{G}. For safety, any model nondeterminism would be used to maximize the unsafe set G.

Other hybrid system reachability algorithms fall within this framework; the differences lie in their discrete and continuous reachability solvers and the types of initial conditions, inputs, invariants and guards that they admit. Most are described as running forwards in time from a set of safe initial conditions, in which case G is computed as the smallest controllable invariant set. For example, in [7,8] reachability is run with ξ_{G} as the controlled inputs and ξ_{E} as the disturbance inputs with the resulting safe set as G. The *CheckMate* tool [9] deals with threshold event-driven hybrid systems—meaning that switches are both enabled and forced only at hyperplanes in the continuous state space—so there is no equivalent to σ_{E} and thus $\vec{\mathsf{E}}_i^j = \emptyset$. Because *VeriSHIFT*'s algorithm [10] is designed for bounded time, decidability can be proven for certain hybrid automata. If we are willing to forgo decidability then its extension to infinite time is straightforward and produces a reachability procedure similar in expressive capacity to *CheckMate*, albeit for different continuous representations.

3 Continuous Reachability with Level Sets

While practical algorithms for computing discrete reachability over many thousands of states have been designed and implemented, determination of continuous reachability for even low dimensional systems is still an open problem. The continuous portion of a hybrid reachability problem requires methods of performing four key operations on sets: unions, intersections, tests of equality, and evolution according to the discrete mode's continuous flow field. The choice of representation for sets dictates the complexity and accuracy of these operations; consequently, continuous reachability algorithms can be classified according to how they represent sets.

Polygonal representations have proven the most popular. The tool **d/dt** [7, 11] tracks the motion of convex polyhedra under linear flow, collecting the nonconvex union of this result into "orthogonal polyhedra" [12]. The developers of *CheckMate* describe optimization based methods of tracking convex polyhedra under general flows, including specializations for the affine case [13,14]. Projectagons [15] is the term used to describe the idea of storing nonconvex high dimensional polyhedra as the intersection of two dimensional projections, which are evolved under affine overapproximations of general flows using linear programming. *VeriSHIFT* [10] uses ellipsoidal representation of reach sets for linear flows with linear input; it implements techniques developed in [16].

3.1 The Hamilton-Jacobi Partial Differential Equation

For our representation scheme, we characterize the set being tracked implicitly by defining a “level set function” $J(x,t)$ throughout the continuous state space which is negative inside the set, zero on its boundary, and positive outside, and which encodes the initial data in $J(x,0)$. The intersection of two such sets is simply the maximum of their level set functions at each point in state space, and the union is the minimum; a variety of easily implemented equality tests are possible. Evolution of a level set under a nonlinear flow field is governed by the Hamilton-Jacobi (HJ) partial differential equation (PDE) (see, for example, [2])

$$\begin{aligned} -\frac{\partial J(x,t)}{\partial t} &= \max_{\nu_{\min}} \min_{\nu_{\max}} f(x, \nu_{\min}, \nu_{\max})^T \nabla J(x,t), & (1) \\ &= H(x, \nabla J(x,t)). & (2) \end{aligned}$$

where $\nu_{\min}$ are those continuous inputs trying to minimize the size of the set being tracked, and $\nu_{\max}$ are those inputs trying to maximize its size. The order of the optimization must be chosen appropriately for the situation. The implicit representation has a number of advantages when compared with the explicit representations that other researchers are pursuing, including a conceptually simple representation of very general sets and a size which is independent of the complexity of the set (although it grows exponentially with dimension). In addition, a set of sophisticated numerical techniques to accurately solve PDEs may be drawn upon for computation. In the remainder of this section, we focus on the representation (2), and assume that the modeler can compute the appropriate optimization over inputs in (1) if given x and $\nabla J(x,t)$.

3.2 Solving the Hamilton-Jacobi PDE

The HJ PDE (2) is well known to have complex behavior. Even with smooth initial data $J(x,0)$ and continuous Hamiltonian $H(x, \nabla J)$, the solution $J(x,t)$ can develop discontinuous derivatives in finite time; consequently, classical infinite time solutions to the PDE are generally not possible. In the quest for a unique weak solution Crandall and Lions introduced the concept of the viscosity solution [17], which has since been shown to be the appropriate weak solution for Hamilton-Jacobi-Bellman type control problems such as (1) (see, for example, [18]). For most problems of interest, finding the analytic viscosity solution is not possible, and so we seek a numerical solution.

Floating point arithmetic and the truncation required by finite series expansions conspire to ensure that any numerical approximation of the solution of a differential equation will contain errors. The algorithms presented in [7]–[16] seek guaranteed overapproximations (and in some cases, underapproximations) of the system's reachable sets. Numerical methods for solving PDEs, on the other hand, have traditionally aimed for convergent approximations: those approximations that will become exact as some parameter of the method—the grid spacing Δx, for example—goes to zero. While guaranteed overapproximation has its pros and cons for use in reachability applications, we have decided to focus first on convergent approximations of (2) in order to take advantage of existing schemes and numerical analyses [19,20,21,22,23]. We can develop confidence in a convergent approximation's accuracy by successive refinement of Δx.

If we are willing to pursue convergent numerical approximations of (2), a reasonable question is whether it would be simpler and as reliable to solve for the optimal trajectories starting from points on the boundary of the initial set, and thereby approximate the boundary of the reachable set. This technique, however, is equivalent to solving the PDE by the characteristic method, and the characteristics of the Hamilton-Jacobi equation are known to collide and/or separate [18], which would make for an incorrectly represented reachable set.[1]

Returning to methods of solving (2) numerically, the state space over which we compute reachability is topologically simple, and so we approximate the solution of (2) on a Cartesian grid of nodes. Three terms in the equation must be approximated at each node, based on the values of the level set function at that node and its neighbors: the gradient ∇J, the Hamiltonian H, and the time derivative $\frac{\partial J(x,t)}{\partial t}$. We discuss each of these separately.

In each dimension at each grid point there exist both left and right approximations of the gradient ∇J, depending on which neighboring grid points' values are used in the finite difference calculation. We label the vector of left approximations ∇J^-, the vector of right approximations ∇J^+, and will see below that ∇J^-, ∇J^+ or some combination of the two will be used to compute the numerical Hamiltonian $\hat{H}$. The accuracy of a derivative approximation is measured in terms of the order of its local truncation error; an order p method has error $\|\nabla J - \nabla J^{\pm}\| = \mathcal{O}(\Delta x^p)$. At the current time, we have implemented the basic first order accurate approximation for speed [21] and a weighted, essentially non-oscillatory fifth order accurate approximation for high fidelity [20,22]. "Non-oscillatory" in this context indicates that near discontinuities in the level set derivative, a scheme may revert to lower order accuracy so as to avoid introducing spurious numerical oscillations into the solution. Technically, therefore, all schemes are globally first order accurate, but in practice the higher order accuracy in the smooth parts of the solution produces better global results. This property is sometimes called "high resolution" to distinguish it from true high order accuracy.

We have chosen to use the well studied Lax-Friedrichs numerical Hamiltonian approximation $\hat{H}$ [20,24]

$$\hat{H}(x, \nabla J^-, \nabla J^+) = H(x, \tfrac{\nabla J^- + \nabla J^+}{2}) - \tfrac{1}{2}\alpha^T(\nabla J^+ - \nabla J^-), \qquad (3)$$

[1] For example, it turns out that much of the helical bulge of the reach set computed in Section 3.4 lies on a collection of optimal trajectories fanning out from a single point on the boundary of the problem's initial conditions.

where $H(x, \nabla J)$ is given by (2) and the term containing the vector coefficient α is a high order numerical dissipation added to damp out spurious oscillations in the solution. Upwinded numerical Hamiltonians were considered; but although they do not require the artificial dissipation of Lax-Friedrichs, they cannot easily deal with the ∇J dependent flow appearing in (2).

The time derivative of the PDE is handled by the method of lines: the value of the level set function J at each node is treated as an ODE $\frac{dJ}{dt} = \hat{H}$, with $\hat{H}$ given by (3). General ODE solvers, such as Runge-Kutta (RK) schemes, can then be applied. The explicit nature of these techniques, however, limits the size of the timestep to some flow speed dependent multiple of the grid spacing—typically a small fraction—called the Courant-Friedrichs-Lewy (CFL) number. Standard RK iterations lead to very small CFL values and can introduce spurious oscillations into a numerical Hamilton-Jacobi solution; therefore, we use total variation diminishing (TVD) versions of Runge-Kutta (see, for example, [19,23]). We have currently implemented TVD RK schemes which are first and second order accurate in time. Due to CFL restrictions the timestep is usually much smaller than the grid spacing, so it is possible to use lower order accuracy in time than in space without noticeable loss of solution quality.

3.3 Localizing Computation

The Hamilton-Jacobi equation (2) describes the evolution of the level set function over all of space. But we are only interested in its zero level set; thus, we can restrict our computational updates to nodes near the boundary between positive and negative $J(x, t)$—an idea variously called “local level sets” [25] or “narrowbanding” [21]. We have implemented a new variant of this method in our code.

Because the boundary is of one dimension less than the state space, considerable savings are available for two and three dimensional problems. If the number of nodes in each dimension is n (proportional to Δx^{-1}) and the dimension d, the total number of nodes is $\mathcal{O}(n^d)$; the CFL restriction on timestep means that total computational cost is $\mathcal{O}(n^{d+1})$. With local level sets, we reduce computational costs back down to $\mathcal{O}(n^d)$.

3.4 Numerical Validation of Aircraft Collision Avoidance

The numerical schemes mentioned above for solving the Hamilton-Jacobi equation are complicated; therefore, it is not surprising that theoretical proofs of convergence to the viscosity solution are available for only the very simplest low order accuracy methods [24]. High resolution methods have instead been subjected to “numerical validation”: comparison to known analytic solutions and lower order accurate approximations of an extensive collection of examples for a broad range of grid sizes [20], from which can be drawn encouraging conclusions regarding their accuracy.

In this section we present a similar validation of our implementation on the single mode, three dimensional aircraft collision avoidance example (see [3,2] for

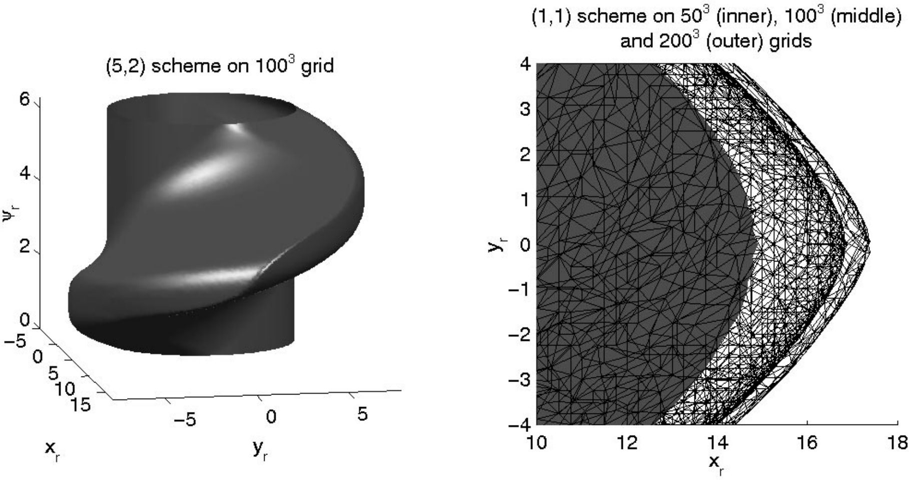

Fig. 2. Reachable Set for Aircraft Collision Avoidance Example

details). The example features a control aircraft trying to avoid collision with a disturbance aircraft, where both aircraft have fixed and equal altitude, speed and turning radius—they may only choose which direction they will turn:

$$\dot{x}_r = -v_u + v_d \cos\psi_r + u y_r, \qquad \dot{y}_r = v_d \sin\psi_r - u x_r, \qquad \dot{\psi}_r = d - u,$$

where $v_u = v_d = 5$ are the aircraft speeds, x_r and y_r are the relative planar location of the aircraft and ψ_r is their relative heading. The inputs $|u| \leq 1$ and $|d| \leq 1$ are the control's and disturbance's respective turn rates. The initial unsafe set $J(x, 0)$ is the interior of the radius five cylinder centered on the ψ_r axis. Choosing optimal inputs according to (1) with $\nu_{\mathsf{G}} = \nu_{\max} = d$ and $\nu_{\mathsf{E}} = \nu_{\min} = u$, we get the optimal Hamiltonian:

$$H(x, p) = -p_1 v_u + p_1 v_d \cos\psi_r + p_2 v_d \sin\psi_r + |p_1 y_r - p_2 x_r - p_3| - |p_3|.$$

Using our new C++ implementation, grid sizes corresponding to 50, 70, 100, 140, and 200 nodes in each dimension were tried with a low order accurate scheme (first order space and time, hereafter referred to as the "(1,1)" scheme) and a high resolution scheme (fifth order space and second order time, hereafter the "(5,2)" scheme). On the eight million node finest grid—only around 10% of which is being actively updated on any one timestep by the local level set algorithm—execution time for the (5,2) scheme was about eighteen hours on a Sun UltraSparc II with lots of memory. Reducing the grid size in half results in the expected eightfold savings in memory and time; hence, the coarsest grid takes only fifteen minutes with the (5,2) scheme.

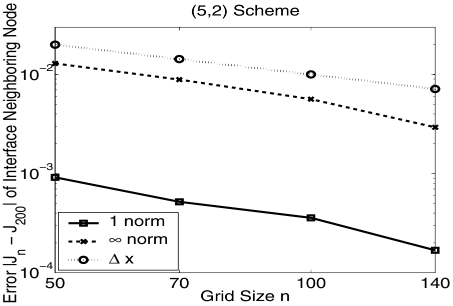

Fig. 3. Convergence of (5,2) Scheme to Finest Grid Solution (J_n is the solution $J(x,t)$ on a grid size of n)

Results are visualized[2] by the zero level isosurface of the unsafe reachable set G, shown in Figure 2. On the left is a head-on view of the (5,2) solution. On the right is a zoomed overhead view of the point of the bulge computed by the (1,1) scheme for several grid sizes. The fact that the solutions grow closer together as the grid is refined provides visual evidence of convergence.

The solutions produced by the (5,2) scheme are visually identical for all grids, and to show quantitative convergence as the grid is refined we require a suitable error metric. Comparing the value of $J(x,t)$ over the entire domain is inappropriate, since our algorithms assume that we seek only an accurate computation of its zero level set. Instead, we consider just the nodes neighboring the zero level set—those nodes which have at least one adjacent node whose J value is of opposite sign. We compare solutions on the four coarser grids to the solution on the finest grid, using linear interpolation on the finest grid if necessary. Figure 3 demonstrates that the scheme is converging to the finest grid's solution of (2) at approximately a linear rate in both average error and pointwise maximum error. We cannot expect to show a higher order convergence rate because of the linear interpolation used to evaluate the error and, as explained in Section 3.2, the scheme is truly high order accurate only in smooth portions of the solution.

Two conclusions can be drawn from Figures 2 and 3. First, low order schemes are not at all competitive in terms of accuracy with the (5,2) scheme. Thus, while our previously reported best results [2] took only an hour to run in Matlab, because they used a (slightly different) first order scheme, our new (5,2) implementation can produce more accurate results in about fifteen minutes using only the coarsest grid. Second, the pointwise maximum error of the (5,2) scheme is always less than the grid spacing, so if a $50^{-1} = 2\%$ error is tolerable for this application, only this fastest, coarsest grid need ever be run.

[2] Figure 2 and Figure 6 visualize some level set surfaces as triangular meshes; these are not the meshes on which the Hamilton-Jacobi PDE was solved, but rather an artifact of three dimensional Matlab visualization techniques.

4 Aircraft Landing Example

Once a method of determining continuous reachability is available, the discrete iteration of the algorithm described in Section 2 is relatively straightforward. In fact, for discrete transition graphs with no cycles it is possible to order the continuous reachability problems such that no discrete iteration is required (e.g. the three mode example presented in [2]). In order to examine the complications induced by discrete cycles—such as how to avoid zenoness, in what order to execute the continuous reachability problems, and how to determine which switches are active—a new example has been developed, which exhibits those difficulties and has real life applications: the landing of a civilian airliner.

Physical model: A simple point mass model for aircraft vertical navigation is used, which accounts for lift L, drag D, thrust T, and gravity mg (see [3] and references therein). State variables are aircraft height z, horizontal position x, velocity $V = \sqrt{\dot{x}^2 + \dot{z}^2}$ and flight path angle $\gamma = \tan^{-1}(\frac{\dot{z}}{\dot{x}})$. Inputs are thrust T and angle of attack α, where aircraft pitch $\theta = \gamma + \alpha$ (see the left side of Figure 4). The equations of motion can be expressed as follows:

$$\frac{d}{dt}\begin{bmatrix} V \\ \gamma \\ x \\ z \end{bmatrix} = \begin{bmatrix} \frac{1}{m}[T\cos\alpha - D(\alpha, V) - mg\sin\gamma] \\ \frac{1}{mV}[T\sin\alpha + L(\alpha, V) - mg\cos\gamma] \\ V\cos\gamma \\ V\sin\gamma \end{bmatrix} \tag{4}$$

The functions $L(\alpha, V)$ and $D(\alpha, V)$ are modelled based on empirical data [26] and Prandtl's lifting line theory [27]:

$$L(\alpha, V) = \tfrac{1}{2}\rho S V^2 C_L(\alpha), \qquad D(\alpha, V) = \tfrac{1}{2}\rho S V^2 C_D(\alpha),$$

where ρ is the density of air, S is wing area, and $C_L(\alpha)$ and $C_D(\alpha)$ are the dimensionless lift and drag coefficients.

In determining $C_L(\alpha)$ we will follow standard auto-lander design and assume that the aircraft switches between three fixed flap deflections $\delta = 0°$, $\delta = 25°$ and $\delta = 50°$ (with slats either extended or retracted), thus constituting a hybrid system with different nonlinear dynamics in each mode. This model is representative of current aircraft technology; for example, in Airbus cockpits the pilot uses a lever to select among four predefined flap deflection settings. We assume a linear form for the lift coefficient $C_L(\alpha) = h_\delta + 4.2\alpha$, where parameters $h_{0°} = 0.2$, $h_{25°} = 0.8$ and $h_{50°} = 1.2$ are determined from experimental data for a DC9-30 [26]. The value of α at which the vehicle stalls decreases with increasing flap deflection: $\alpha_{0°}^{\max} = 16°$, $\alpha_{25°}^{\max} = 13°$, $\alpha_{50°}^{\max} = 11°$; slat deflection adds 7° to the $\alpha^{\max}$ in each mode. The right side of Figure 4 gives a graphical summary of the possible configurations. The drag coefficient is computed from the lift coefficient as [27] $C_D(\alpha) = 0.041 + 0.045 C_L^2(\alpha)$ and includes flap deflection, slat extension and gear deployment corrections. So for a DC9-30 landing at sea level and for all $\alpha \in [-5°, \alpha_\delta^{\max}]$, the lift and drag terms in (4) are given by

$$L(\alpha, V) = 68.6\ (h_\delta + 4.2\alpha)V^2 \qquad D(\alpha, V) = (2.81 + 3.09\ (h_\delta + 4.2\alpha)^2)V^2 \tag{5}$$

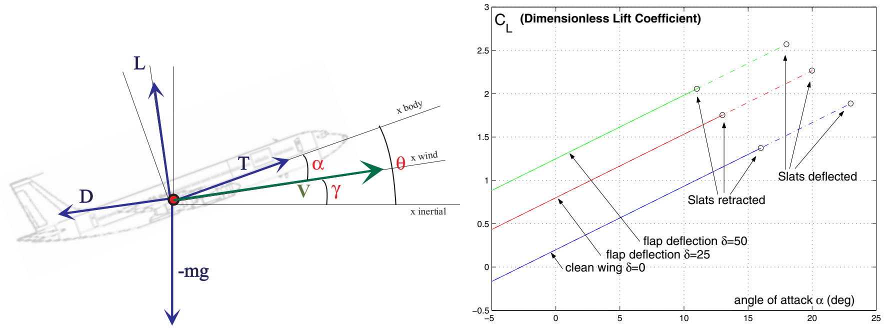

Fig. 4. Left: Force diagram for the point mass approximation of the aircraft. Right: lift coefficient $C_L(\alpha)$ model for the DC9-30 [26]. Circles located at $(\alpha_\delta^{\max}, C_L(\alpha_\delta^{\max}))$ indicate the stall angle and the corresponding lift coefficient in each mode.

Flap deflection dynamics model: In reality, the decision to move from one deflection setting to another can occur at any time, but approximately 10 seconds are required for a 25° degree change in flap deflection. A five state model of this situation is shown on the left side of Figure 5, where the system is in state R if the flaps are retracting and state D if the flaps are deflecting. The system is zeno because instantaneous switches are allowed between any modes.

Current implementation: For our preliminary implementation, we have chosen to ignore the continuous dynamics associated with discrete mode switching, allowing the flaps and slats to move instantly to their commanded positions. However, if such instantaneous controlled switches were always enabled then the system would be zeno; therefore, we introduce transition modes $0t$, $25t$ and $50t$, which use the envelopes and flight dynamics of the regular modes $0u$, $25d$ and $50d$ (the discrete automaton is shown on the right side of Figure 5). A regular mode may make a controlled switch to a transition mode, so flight dynamics can be changed instantly. Transition modes have only a timed switch at $t = t_{\text{delay}}$, so controlled switches will be separated by at least t_{delay} time units and the system is nonzeno. For the executions shown below, $t_{\text{delay}} = 0.5$ seconds.

Landing: Extensive descriptions of the final stage of landing, when aircraft height is below 50 feet, exist (see, for example, [26,28]). Restrictions on the flight path angle, aircraft velocity and touchdown (TD) speed are used to determine the initial safe set E_0:

$$\begin{cases} z \le 0 & \text{landing or has landed} \\ V > V_\delta^{\text{stall}} & \text{faster than stall speed} \\ V < V^{\max} & \text{slower than limit speed} \\ V \sin\gamma \ge \dot{z}_0 & \text{limited TD speed} \\ \gamma \le 0 & \text{monotonic descent} \end{cases} \cup \begin{cases} z > 0 & \text{aircraft in the air} \\ V > V_\delta^{\text{stall}} & \text{faster than stall speed} \\ V < V^{\max} & \text{slower than limit speed} \\ \gamma > -3^\circ & \text{limited descent flight path} \\ \gamma \le 0 & \text{monotonic descent} \end{cases} \qquad (6)$$

We again draw on numerical values for a DC9-30 [26]: stall speeds $V_{0u}^{\text{stall}} = 78$ m/s, $V_{25d}^{\text{stall}} = 61$ m/s, $V_{50d}^{\text{stall}} = 58$ m/s, maximal touchdown speed $\dot{z}_0 = 0.9144$

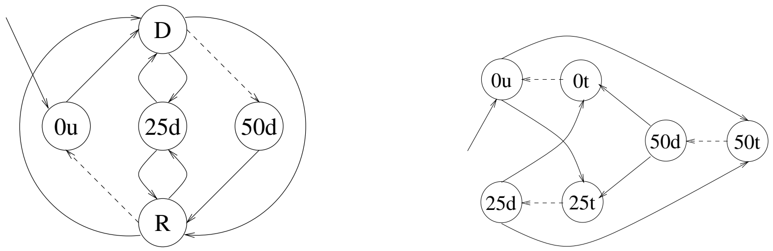

Fig. 5. Discrete transition graph of slat and flap settings. The left graph shows the model with flap deflection dynamics and the right graph shows the currently implemented model. Solid lines are controlled switches (σ_{E} in this version of the reachability problem) and dashed lines are uncontrolled switches (σ_{G}).

m/s, and maximal velocity $V^{\max} = 83$ m/s. For passenger comfort, the aircraft's input range is restricted to $T \in [0 \text{ kN},\ 160 \text{ kN}]$ and $\alpha \in [0°,\ 10°]$.

The interior of the surface shown in the first row of Figure 6 represents E_0 for each mode. The second row of the figure shows the safe envelope E when there is no mode switching. Portions of E_0 are excluded from E for two reasons. States near $z = 0$ correspond to low altitudes and are too close to the ground at steep flight path angles to allow control inputs time to prevent the plane from crashing. States close to the stall velocity correspond to low speeds where there is insufficient lift and the flight path angle becomes steeper than that allowed by the flight envelope. This latter condition holds throughout the very narrow range of speeds allowed in mode $0u$, with the result that only post-touchdown states ($z \leq 0$) are controllable in this mode. The third row shows how E can be increased if switches are permitted (for example, mode $0u$ becomes completely controllable). Mode $50d$ is the best to be in for landing and there is no difference in E with or without switching enabled. The fourth row shows slices of the set in the third row, taken at $z = 3$ meters. The light grey regions are unsafe G and the dark grey are safe E. The figure shows that modes $0u$ and $25d$ are safe only because there exists a discrete switch to a safe state in another mode.

We have presented and numerically validated a tool for determining accurate approximations of reachable sets for hybrid systems with nonlinear continuous dynamics and adversarial continuous and discrete inputs. By developing convergent approximations of such complex systems, we will be better able to synthesize aggressive but safe controllers. As an example, the six mode auto-lander shows that for envelope protection purposes the safest control decisions are to switch directly to full flap deflection, but to maintain airspeed until touchdown. With the summary data from the reachability analysis, such decisions can be made based on local state information; without it the auto-lander may not detect that low speeds—while still within the flight envelope—lead inevitably to unsafe flight path angles.

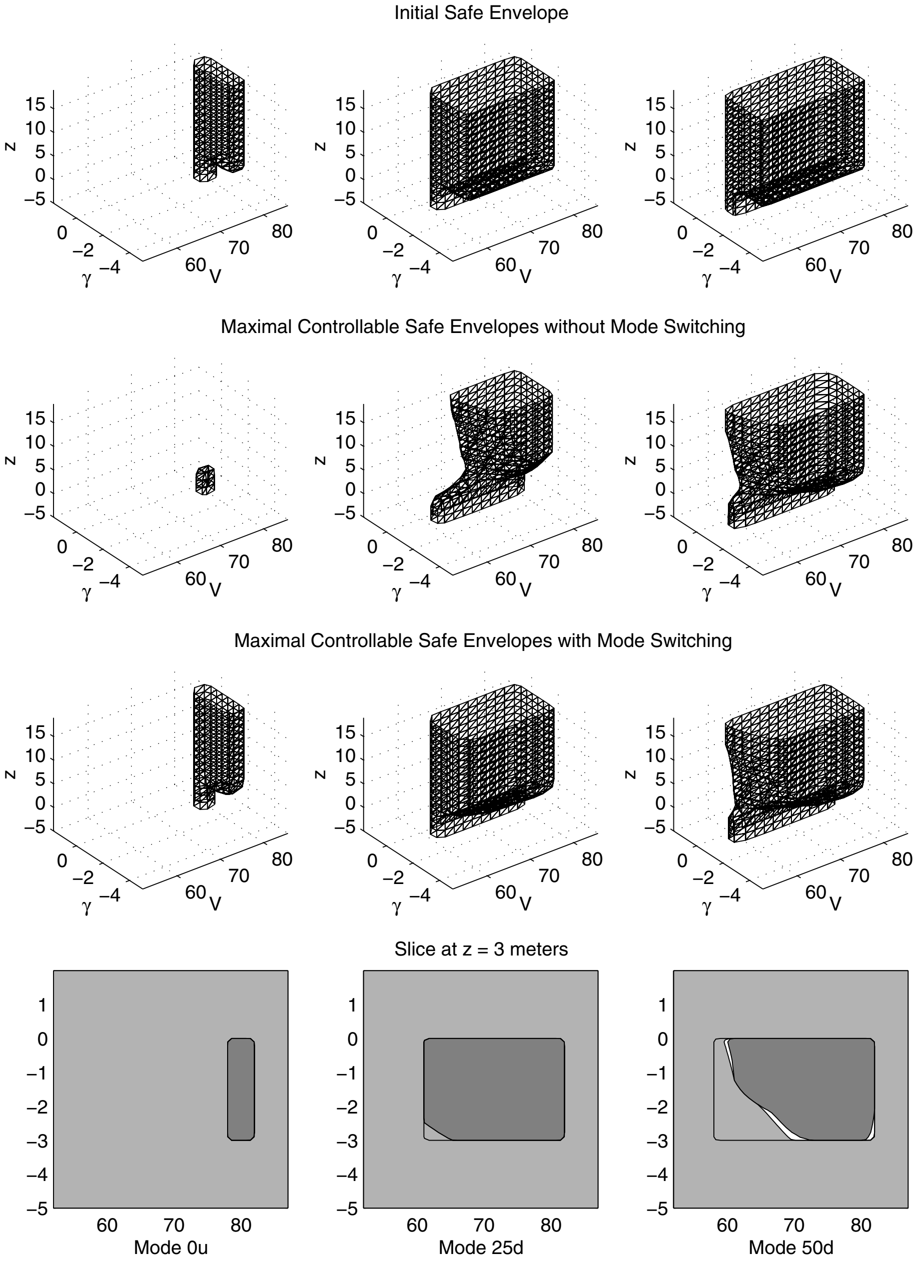

Fig. 6. Maximally controllable safe envelopes for the multimode landing example. From left to right the columns represent modes $0u$, $25d$ and $50d$.

Our current work includes further validation of our numeric algorithm, extending our implementation to four continuous dimensions in order to capture the full landing example dynamics, projections to capture higher dimensional dynamics, schemes for overapproximating the solution of the HJ PDE, automation of the discrete algorithm, and parallel implementations.

Acknowledgements. We would like to thank Professors Ron Fedkiw and Stanley Osher for extensive discussions about the details of numerical schemes for solving the Hamilton-Jacobi PDE. Professor Fedkiw is also responsible for some key ideas in our new variant for localizing the level set computation. In addition, we would like to thank Professor Ilan Kroo for discussions about flight dynamics and for his help in the design of the auto-lander example.

References

1. C. Tomlin, J. Lygeros, and S. Sastry, “Controller design for hybrid systems,” *Proceedings of the IEEE*, vol. 88, no. 7, July 2000.
2. I. Mitchell and C. Tomlin, “Level set methods for computation in hybrid systems,” in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), LNCS 1790, pp. 310–323, Springer Verlag, 2000.
3. C. J. Tomlin, *Hybrid Control of Air Traffic Management Systems.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1998.
4. O. Maler, A. Pnueli, and J. Sifakis, “On the synthesis of discrete controllers for timed systems,” in *STACS 95: Theoretical Aspects of Computer Science* (E. W. Mayr and C. Puech, eds.), no. 900 in LNCS, pp. 229–242, Munich: Springer Verlag, 1995.
5. E. Asarin, O. Maler, and A. Pnueli, “Symbolic controller synthesis for discrete and timed systems,” in *Proceedings of Hybrid Systems II, Volume 999 of LNCS* (P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, eds.), Cambridge: Springer Verlag, 1995.
6. H. Wong-Toi, “The synthesis of controllers for linear hybrid automata,” in *Proceedings of the IEEE Conference on Decision and Control*, (San Diego, CA), 1997.
7. T. Dang, *Vérification et synthèse des systèmes hybrides.* PhD thesis, Institut National Polytechnique de Grenoble (Verimag), 2000.
8. E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli, “Effective synthesis of switching controllers for linear systems,” *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1011–1025, July 2000.
9. B. Silva and B. H. Krogh, “Formal verification of hybrid systems using *CheckMate*: A case study,” in *Proceedings of the American Control Conference*, (Chicago, IL), pp. 1679–1683, 2000.
10. O. Botchkarev and S. Tripakis, “Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations,” in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), LNCS 1790, pp. 73–88, Springer Verlag, 2000.
11. E. Asarin, O. Bournez, T. Dang, and O. Maler, “Approximate reachability analysis of piecewise-linear dynamical systems,” in *Hybrid Systems: Computation and Control* (N. Lynch and B. Krogh, eds.), no. 1790 in LNCS, pp. 21–31, Springer Verlag, 2000.

12. O. Bournez, O. Maler, and A. Pnueli, "Orthogonal polyhedra: Representation and computation," in *Hybrid Systems: Computation and Control* (F. Vaandrager and J. van Schuppen, eds.), no. 1569 in LNCS, pp. 46–60, Springer Verlag, 1999.
13. A. Chutinan and B. H. Krogh, "Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations," in *Hybrid Systems: Computation and Control* (F. Vaandrager and J. H. van Schuppen, eds.), no. 1569 in LNCS, pp. 76–90, New York: Springer Verlag, 1999.
14. A. Chutinan and B. H. Krogh, "Approximating quotient transition systems for hybrid systems," in *Proceedings of the American Control Conference*, (Chicago, IL), pp. 1689–1693, 2000.
15. M. Greenstreet and I. Mitchell, "Reachability analysis using polygonal projections," in *Hybrid Systems: Computation and Control* (F. Vaandrager and J. van Schuppen, eds.), no. 1569 in LNCS, pp. 103–116, Springer Verlag, 1999.
16. A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), LNCS 1790, pp. 202–214, Springer Verlag, 2000.
17. M. G. Crandall, L. C. Evans, and P.-L. Lions, "Some properties of viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 282, no. 2, pp. 487–502, 1984.
18. L. Evans, *Partial Differential Equations.* Providence, Rhode Island: American Mathematical Society, 1998.
19. C.-W. Shu and S. Osher, "Efficient implementation of essentially non-oscillatory shock-capturing schemes," *Journal of Computational Physics*, vol. 77, pp. 439–471, 1988.
20. S. Osher and C.-W. Shu, "High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations," *SIAM Journal on Numerical Analysis*, vol. 28, no. 4, pp. 907–922, 1991.
21. J. A. Sethian, *Level Set Methods and Fast Marching Methods.* New York: Cambridge University Press, 1999.
22. R. Fedkiw, T. Aslam, B. Merriman, and S. Osher, "A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method)," *Journal of Computational Physics*, vol. 152, pp. 457–492, 1999.
23. M. Kang, R. Fedkiw, and X.-D. Liu, "A boundary condition capturing method for multiphase incompressible flow," *Journal of Computational Physics*, 2000. Submitted.
24. M. G. Crandall and P.-L. Lions, "Two approximations of solutions of Hamilton-Jacobi equations," *Mathematics of Computation*, vol. 43, no. 167, pp. 1–19, 1984.
25. D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, "A PDE based fast local level set method," *Journal of Computational Physics*, vol. 165, pp. 410–438, 1999.
26. I. M. Kroo, *Aircraft Design: Synthesis and Analysis.* Stanford, California: Desktop Aeronautics Inc., 1999.
27. J. Anderson, *Fundamentals of Aerodynamics.* New York: McGraw Hill Inc., 1991.
28. United States Federal Aviation Administration, *Federal Aviation Regulations*, 1990. Section 25.125 (landing).