# Computational Tools for the Verification of Hybrid Systems*

Claire J. Tomlin and Stephen P. Boyd
Ian Mitchell, Alexandre Bayen, Mikael Johansson, and Lin Xiao
Stanford University, Stanford, CA 94305
{tomlin,boyd,imitchel,bayen,mikaelj,lxiao}@stanford.edu

**Editors' Summary**

The hybrid systems framework provides an appealing means for verifying the safety of dynamical systems. The authors address safety verification as a reachability problem: Given an unsafe subset of the system state space and an initial state, is the former reachable from the latter? If it is reachable despite any controllable actions that we can take then the system is provably unsafe.

The continuous-time nonlinear dynamics of the system need to be considered in assessing reachability. The authors' formulation requires the solution of a Hamilton-Jacobi partial differential equation, and a grid-based numerical solution approach based on level set methods is used for this purpose. Simulation examples are presented for three flight management applications: two-aircraft collision avoidance, the related problem of conflict resolution, and ensuring safety during final landing approach.

The exact reachability computation is prey to the curse of dimensionality: its computational complexity is exponential with respect to the continuous dimension. The authors also present an alternative approach, which is based on over-approximating the reachable set of states with a polyhedron. This is also computationally intractable since the propagation of the system's dynamics will result in a potentially unlimited number of constraints (faces of the polyhedron), but the authors have developed a novel technique for identifying and pruning redundant and irrelevant constraints. This technique promises to be computationally feasible for very high dimensional problems. The basis of the approach is the computation of the maximum volume ellipsoid contained in a polyhedron, a computation that can be formulated as a convex optimization problem (for which global and efficient algorithms are available).

## 1 Introduction

For about the past ten years, researchers in the traditionally distinct fields of control theory and computer science verification have proposed models, and verification and controller synthesis techniques for complex, safety critical systems. The area of *hybrid systems theory* studies systems which involve the interaction of discrete event and continuous time dynamics, and provides an effective modeling

*Research supported by the DARPA Software Enabled Control (SEC) Program administered by AFRL under contract F33615-99-C-3014.

framework for complex continuous systems with large numbers of operating modes. Examples include continuous systems controlled by a discrete logic such as the autopilot modes for controlling an aircraft, systems of many interacting processes such as air or ground transportation systems in which discrete dynamics are used to model the coordination protocols among processes, or continuous systems which have a phased operation, such as biological cell growth and division.

The current and potential impact of hybrid systems lies in the confluence of computational methods from control theory and from formal methods in computer science verification. In the examples mentioned above, the system dynamics are complex enough that traditional analysis and control methods based solely on differential equations are not computationally feasible; analysis based solely on discrete event dynamics ignores critical system behavior. Our interest lies in developing computational tools for analyzing and controlling the behavior of hybrid systems; our eventual goal is to develop a real-time tool to provide online verification that a hybrid system satisfies its specified behavior. This goal addresses one of the central themes of *software-enabled control*, in which we seek a systematic methodology for the design, validation, and implementation of control software.

In our work to date, the system specification that we are most interested in is that of system safety, which asks the question: Is a potentially unsafe configuration of the system reachable from an initial configuration? More importantly for control theory, given a set of desired configurations, it is crucial to be able to design the hybrid system control inputs to achieve these configurations. Previous work in this area had focused on hybrid systems with very simple continuous dynamic equations (such as clocks, or linear decoupled systems). Our research has three components: the problem formulation for computing *exact reachable sets* of hybrid systems and the design of a software tool to perform such calculations; the development of an algorithm for computing overapproximations of reachable sets which works efficiently in high dimensions; the design of a real-time aircraft testbed for these algorithms. We will focus on the first component in this paper, with references to our work in ellipsoidal overapproximations and the implementation on the aircraft testbed.

## 2 Hybrid System Model

This section presents our hybrid system model: the main difference between our model and leading models in the literature (see for example the work of Alur and Henzinger [1] on linear hybrid automata) is that we include accurate, nonlinear models of the hybrid system continuous dynamics. The model and algorithm have been developed jointly with Lygeros and Sastry, full details are presented in [2].

A hybrid automaton is a finite state machine with discrete states $\{q_1, q_2, \ldots, q_n\}$, in which each discrete state has associated continuous dynamics $\dot{x} = f(q_i, x, \nu)$, with $x \in \Re^n$, and continuous inputs and disturbances $\nu = (u, d)$, where $u \in U$ and $d \in D$.

**Definition 1 (Hybrid Automaton)** *A hybrid automaton $H$ is a collection*

$$H = (Q, X, \Sigma, V, \mathrm{Init}, f, \mathrm{Inv}, R)$$

*where*

- $Q \cup X$ *is the set of state variables, with* $Q$ *a finite set of discrete states, and* $X = \Re^n$*;*
- $\Sigma = \Sigma_1 \cup \Sigma_2$ *is a finite collection of discrete input variables, where* $\Sigma_1$ *is the set of discrete control inputs, and* $\Sigma_2$ *is the set of discrete disturbance inputs;*
- $V = U \cup D$ *is the set of continuous input variables, where* $U$ *is the set of continuous control inputs, and* $D$ *is the set of continuous disturbance or uncontrollable inputs; we denote the spaces of input (disturbance) trajectories as the sets of piecewise continuous functions* $\mathcal{U}$ *(*$\mathcal{D}$ *respectively), which take values in* $U$ *(*$D$ *respectively);*
- $\mathrm{Init} \subseteq Q \times X$ *is a set of initial states;*
- $f : Q \times X \times V \to X$ *is a vector field describing the evolution of* $x$ *for each* $q \in Q$*;* $f$ *is assumed to be globally Lipschitz in* $X$ *(for fixed* $q \in Q$*) and continuous in* $V$*;*
- $\mathrm{Inv} \subseteq Q \times X \times \Sigma \times V$ *is called an invariant, and defines combinations of states and inputs for which continuous evolution is allowed;*
- $R : Q \times X \times \Sigma \times V \to 2^{Q \times X}$ *is a reset relation, which encodes the discrete transitions of the hybrid automaton.*

We refer to $(q, x) \in Q \times X$ as the *state* of $H$ and to $((\sigma_1, \sigma_2), (u, d)) \in \Sigma \times V$ as the *input* of $H$. The control actions $(\sigma_1, u)$ model those inputs over which the designer has control, and the disturbance actions $(\sigma_2, d)$ model inputs over which the designer has no control, such as uncertainties in the actions or behaviors of the other aircraft in the system. We assume that the designer has complete knowledge of the *bounds* on these disturbance actions $(\Sigma_2, D)$.

Associated to the hybrid automaton $H$ is a *specification* which describes the condition one would like the system to satisfy. Our work has been motivated by verification and synthesis for safety critical applications, and as such we have been primarily interested in safety specifications. These specifications are encoded as subsets of the state space of the hybrid system: the *unsafe set* $\mathsf{G}_0 \subseteq \mathbf{Q} \times \mathbf{X}$ is that subset in which the system is unsafe.

## 3 Exact Reach Set Computation using Level Sets

In this section, we describe the development of a general purpose tool for exact reachable set computation—the core of which is a new variant of a "local level set" algorithm that efficiently computes an accurate

representation of the reachable set boundary. 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 [3, 4], the three mode example of the previous section, as well as an example of a six mode commercial aircraft auto-lander [5], which exhibits nondeterminism and cycles in its discrete behavior.

Our motivation for this component of the research 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 the convex overapproximations of the next section) 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.

### 3.1 Reachability for Hybrid Systems

In this section we summarize the general framework for handling the interaction between discrete and continuous dynamics (following [2]).

Fundamentally, reachability analysis in discrete, continuous or hybrid systems seeks to partition states into two categories: those that can reach a given target set, and those that cannot. We will label these two sets of states $\mathsf{G}$ 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 $\xi_{\mathsf{G}}$ that maximize the size of $\mathsf{G}$ and values for inputs $\xi_{\mathsf{E}}$ that minimize the size of $\mathsf{G}$ (and hence maximize the size of $\mathsf{E}$). Any nondeterminism in the transition relation is also utilized to consistently maximize or minimize $\mathsf{G}$, depending on the goal of the reachability computation. For hybrid automata, the discrete inputs $\sigma$ and continuous inputs $\nu$ 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 $\mathsf{G}$. For example, for our hybrid system model with control actions $(\sigma_1, u)$ and disturbance actions $(\sigma_2, d)$, we would have that $\xi_{\mathsf{G}} = (\sigma_{\mathsf{G}}, \nu_{\mathsf{G}}) = (\sigma_2, d)$, and $\xi_{\mathsf{E}} = (\sigma_{\mathsf{E}}, \nu_{\mathsf{E}}) = (\sigma_1, u)$.

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 $\mathsf{G}_i$ and $\mathsf{E}_i$ at iteration $i = 0, -1, -2, \ldots$, where a negative index is used to indicate that the algorithm is initialized

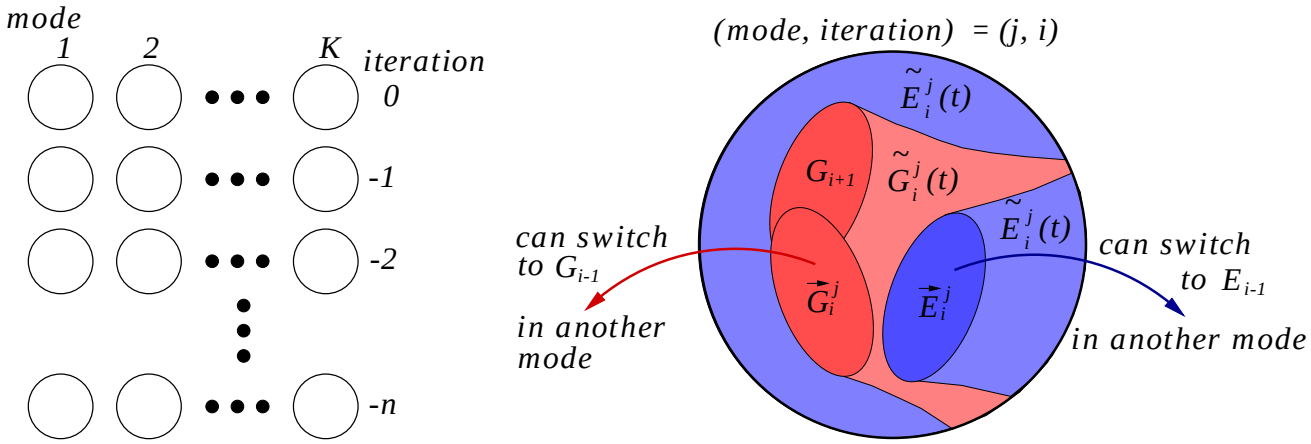


Figure 1: Iterative Reachability Algorithm: Showing detail of iteration for discrete mode $j$ at iteration $i$.

with a "target set" and run backward to compute those states which can reach the target set. The inner iteration runs a separate continuous reachability problem in each of the discrete modes $j = 1, 2, \ldots K$ to compute the estimates $\mathsf{G}_i^j$ and $\mathsf{E}_i^j$. 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 $\mathsf{G}_{i-1}$ (typically a state in another mode) can be forced to occur through the application of a discrete input $\sigma_{\mathsf{G}}$; 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 $\mathsf{E}_{i-1}$ can be forced to occur through the application of a discrete input $\sigma_{\mathsf{E}}$; 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 $\nu_{\mathsf{E}}$ there exists $\nu_{\mathsf{G}}$ 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 $\nu_{\mathsf{E}}$ that for all $\nu_{\mathsf{G}}$ 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 $\mathsf{G}_0^j$ 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 [2, 4], was motivated by the work of [6, 7] for reachability computation and controller synthesis on timed automata, and that of [8] for controller synthesis on linear hybrid automata. In that development the reachability problem's objective was to determine $\mathsf{E}$—the largest controllable invariant subset of the state space—by computing the set of states $\mathsf{G}$ 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 $\xi_{\mathsf{E}}$ and disturbance inputs in $\xi_{\mathsf{G}}$. For safety, any model nondeterminism would be used to maximize the unsafe set $\mathsf{G}$.

### 3.2 Continuous Reachability using 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.

For our exact 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, [3])

$$\begin{aligned} -\frac{\partial J(x,t)}{\partial t} &= \min\{0, \max_{\nu_{\min}} \min_{\nu_{\max}} f(x,\nu_{\min},\nu_{\max})^T \nabla J(x,t)\}, \\ &= \min\{0, H(x, \nabla J(x,t))\} \end{aligned} \tag{1}$$

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 (1), and assume that the modeler can compute the appropriate optimization over inputs in (1) if given $x$ and $\nabla J(x,t)$.

The HJ PDE (1) 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 [9]. For most problems of interest, finding the analytic viscosity solution is not possible (see [10] for cases in which it is possible), and so we seek a numerical solution. We approximate the solution of (1) 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 $\nabla J$, 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 $\nabla J$, depending on which neighboring grid points' values are used in the finite difference calculation. We label the vector of left approximations $\nabla J^-$, the vector of right approximations $\nabla J^+$, and will see below that $\nabla J^-$, $\nabla J^+$ 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 and a weighted, essentially non-oscillatory fifth order accurate approximation for high fidelity. We have chosen to use the well studied Lax-Friedrichs numerical Hamiltonian approximation $\hat{H}$ [11]

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

where $H(x,\nabla J)$ is given by (1) and the term containing the vector coefficient $\alpha$ is a high order numerical dissipation added to damp out spurious oscillations in the solution. 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 (2). General ODE solvers, such as Runge-Kutta (RK) schemes, can then be applied.

The Hamilton-Jacobi equation (1) 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" [12] or "narrowbanding" [13]. 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 $\Delta x^{-1}$) and the dimension $d$, the total number of nodes is $\mathcal{O}(n^d)$; with the restriction on the timestep necessary for numerical stability, the total computational cost is $\mathcal{O}(n^{d+1})$. With local level sets, we reduce computational costs back down to $\mathcal{O}(n^d)$.

### 3.3 Examples

In this section we present three examples: a validation of our numerical implementation on a single mode, three dimensional aircraft collision avoidance example (see [4, 3] for details), and two multimodal examples. Once a method of determining continuous reachability is available, the discrete iteration of the algorithm described in Section 3.1 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 (the three mode example below). 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— we present an example representing the landing of a civilian airliner.

#### Numerical Validation of Aircraft Collision Avoidance

This 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 $\psi_r$ 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 $\psi_r$ 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 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

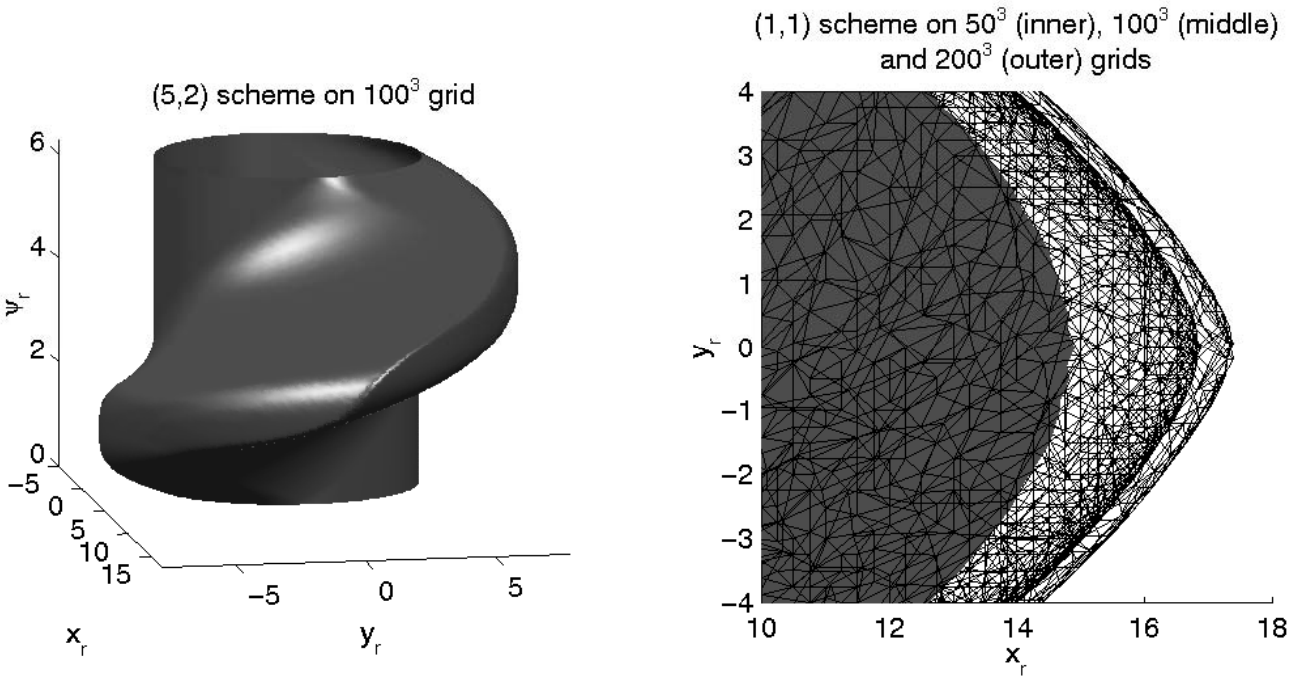


Figure 2: Reachable Set for Aircraft Collision Avoidance Example

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.

Results are visualized[1] 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.

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 (1) 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.

Two conclusions can be drawn from Figures 2 and 3. First, for this example, low order schemes are not at all competitive in terms of accuracy with the (5,2) scheme: our (5,2) implementation can produce more accurate results in about fifteen minutes using only the coarsest grid. Second, the pointwise

[1]Figure 2 and Figure 8 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.

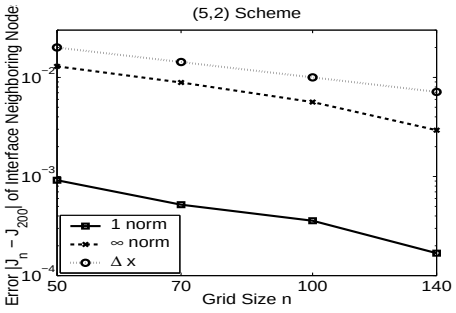


Figure 3: Convergence of (5,2) Scheme to Finest Grid Solution ($J_n$ is the solution $J(x,t)$ on a grid size of $n$)

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.

### Three Mode Conflict Resolution

Consider the following conflict resolution problem between two aircraft ($i \in \{1,2\}$), which is presented in [14]. Let $(x_r, y_r, \psi_r) \in \Re^2 \times [-\pi, \pi)$ represent the relative position and orientation of aircraft 2 with respect to aircraft 1. In terms of the absolute positions and orientations of the two aircraft, $(x_i, y_i, \psi_i)$ for $i = 1, 2$, it may be verified that $x_r = \cos\psi_1(x_2 - x_1) + \sin\psi_1(y_2 - y_1), y_r = -\sin\psi_1(x_2 - x_1) + \cos\psi_1(y_2 - y_1), \psi_r = \psi_2 - \psi_1$, and thus

$$\begin{aligned} \dot{x}_r &= -v_1 + v_2\cos\psi_r + \omega_1 y_r \\ \dot{y}_r &= v_2\sin\psi_r - \omega_1 x_r \\ \dot{\psi}_r &= \omega_2 - \omega_1 \end{aligned} \tag{3}$$

where $v_i$ is the velocity along the body axis of aircraft $i$, and $\omega_i$ is the rate of change of heading. A conflict occurs when a pair of aircraft incur a lateral spacing of less than 5 nautical miles; for safety of the maneuver, the relative position $(x_r, y_r)$ must remain outside of the interior of the 5 mile radius disk centered at the origin:

$$\{(x_r, y_r, \psi_r) : x_r^2 + y_r^2 \leq 5^2\} \tag{4}$$

Consider the maneuver illustrated in Figure 4, with the protocol of the maneuver defined as follows: the aircraft are nominally in *Mode 1*, in which the aircraft fly straight and level with given constant velocities $v_i$ and constant heading $\psi_i$; at a certain relative separation distance each aircraft turns to its right, follows an arc of a circle in *Mode 2*, in which the velocity is held at a prescribed constant value $v_i$, until it intersects its original trajectory, then turns to its right and returns to its desired trajectory.

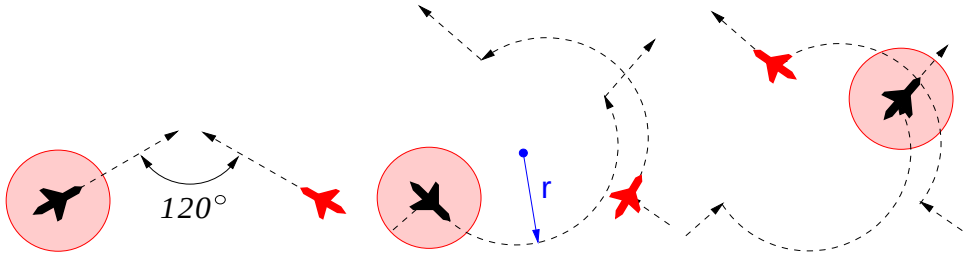


Figure 4: Two aircraft in two modes of operation: in the first and third figures, the aircraft follow a constant heading and velocity (in *Mode 1*); in the second figure, the aircraft follow a half circle (in *Mode 2*).

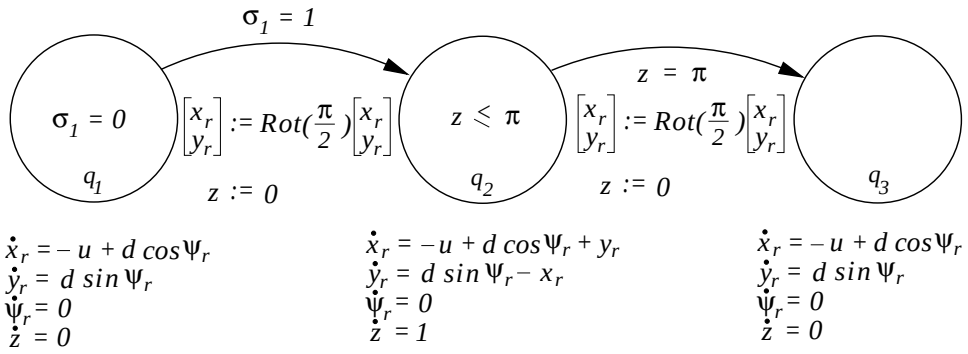


Figure 5: In $q_1$ both aircraft follow a straight course, in $q_2$ a half circle, and in $q_3$ both aircraft return to a straight course.

The model allows instantaneous changes in heading which is an obvious abstraction of the true aircraft dynamics and is treated here for clarity; a model with heading capture dynamics is presented in [14]. In each mode, the continuous dynamics may be expressed in terms of the relative motion of the two aircraft (3): in *Mode 1*, $\omega_i = 0$ for $i = 1, 2$ and in *Mode 2*, $\omega_i = 1$ for $i = 1, 2$. We assume that both aircraft switch modes simultaneously, so that the relative orientation $\psi_r$ is constant. This assumption simply allows us to display the state space in two dimensions, making the results easier to present. The problem statement is therefore: generate the relative distance between aircraft at which the aircraft may switch safely from *Mode 1* to *Mode 2*, and a turning radius $r$ in *Mode 2*, to ensure that the 5 nautical mile separation is maintained.

The dynamics of the maneuver can be encoded by the hybrid automaton of Figure 5, where $q_1$ corresponds to *Mode 1* before the maneuver, $q_2$ corresponds to *Mode 2* "avoid mode", and $q_3$ corresponds to *Mode 1* after the avoid maneuver has been completed. There is one discrete control input $\sigma_1$, such that the switch from $\sigma_1 = 0$ to $\sigma_1 = 1$ triggers the transition from $q_1$ to $q_2$. The transition from $q_2$ to $q_3$ is required to take place after the aircraft have completed a half circle: note that with $\omega_i = 1$, for $i = 1, 2$, it takes $\pi$ time units to complete a half circle. The continuous state space is augmented with a

timer $z \in \Re^+$ to force this transition. Let $x = (x_r, y_r, \psi_r, z)^T$. At each transition, both aircraft change heading instantaneously by $\pi/2$ radians; we represent this with the standard rotation matrix $Rot(\frac{\pi}{2})$. As discussed in the previous section, we assume that $v_1$ is controllable, and $v_2$ is the disturbance input with known bounds. Safety is defined in terms of the unsafe set:

$$\mathsf{G}_0 = \{q_1, q_2, q_3\} \times \{x \in X \ : \ x_r^2 + y_r^2 \leq 5^2\} \tag{5}$$

which encodes the fact that the two aircraft should never come within 5 nautical miles of each other.

The solution of the three mode conflict resolution example is presented in Figure 6. Each of the twelve graphs in Figure 6 is a plot in the $(x_r, y_r)$ axes, with each of the four rows representing an iteration of the algorithm, and the three columns corresponding to discrete states $q_1, q_2$, and $q_3$. The computation is initialized with the set $\mathsf{G}_0 = \{q_1, q_2, q_3\} \times \{\mathbf{x} \in \Re^3 : x_r^2 + y_r^2 \leq 5^2\}$, which is shown as the dark shaded disks. A fixed point is reached after three iterations. The dark shaded region in the fourth row, first column of Figure 6 is the most interesting: it is a plot of the unsafe set of states in $q_1$. As long as the relative position of aircraft 2 with respect to aircraft 1 is not inside this region, then there exists a control policy such that the conflict is resolved. If the relative position is inside the unshaded (white) region, then switching instantaneously will lead to a conflict; the aircraft must remain in $q_1$ until the relative state enters the light shaded region, then either switch to $q_2$, or remain in $q_1$.

### Aircraft landing example

The autopilots of modern jets are highly automated systems which assist the pilot in constructing and flying four-dimensional trajectories, as well as altering these trajectories on line in response to air traffic control directives. The autopilot typically controls the throttle input and the vertical and lateral trajectories of the aircraft to automatically perform such functions as: acquiring a specified altitude and then leveling (ALT ACQ), holding a specified altitude (ALT HLD), acquiring a specified vertical climb or descend rate (V/S), automatic vertical or lateral navigation between specified way points (VNAV, LNAV), or holding a specified throttle value (THR HLD). The combination of these throttle-vertical-lateral modes is referred to as the *flight mode* of the aircraft. A typical autopilot has several hundred flight modes – it is interesting to note that these flight modes were designed to automate the way pilots fly aircraft manually: by controlling the lateral and vertical states of the aircraft to set points for fixed periods of time, pilots simplify the complex task of flying an aircraft. Those autopilot functions which are specific to aircraft landing are among the most safety critical, as it is extremely difficult, if not impossible, for the pilot to take over control of the aircraft when the aircraft is close to the ground. Thus, the need for automation designs which guarantee safe operation of the aircraft has become paramount. Testing and simulation may overlook trajectories to unsafe states. "Automation surprises" have been extensively studied [15] *after* the unsafe situation occurs, and "band-aids" are added to the design to ensure the same problem does not occur again. We believe that the ability to

compute accurate reachable sets inside the aerodynamic flight envelope under flight mode switching, may be used to help prevent the occurrence of automation surprises.

A simple point mass model for aircraft vertical navigation is used, which accounts for lift $L$, drag $D$, thrust $T$, and gravity $mg$ (see [4] 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 $\alpha$, where aircraft pitch $\theta = \gamma + \alpha$. The equations of motion can be expressed as follows:

$$\frac{d}{dt}\begin{bmatrix} V \\ \gamma \\ x \\ z \end{bmatrix} = \begin{bmatrix} \frac{1}{m}[T\cos\alpha - D(\alpha, V) - mg\sin\gamma] \\ \frac{1}{mV}[T\sin\alpha + L(\alpha, V) - mg\cos\gamma] \\ V\cos\gamma \\ V\sin\gamma \end{bmatrix} \tag{6}$$

The functions $L(\alpha, V)$ and $D(\alpha, V)$ are modelled based on empirical data [16] and Prandtl's lifting line theory [17]:

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

where $\rho$ 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 [16]. The value of $\alpha$ at which the vehicle stalls decreases with increasing flap deflection: $\alpha^{\max}_{0°} = 16°$, $\alpha^{\max}_{25°} = 13°$, $\alpha^{\max}_{50°} = 11°$; slat deflection adds $7°$ to the $\alpha^{\max}$ in each mode. The left side of Figure 7 gives a graphical summary of the possible configurations. The drag coefficient is computed from the lift coefficient as [17] $C_D(\alpha) = 0.041 + 0.045 C_L^2(\alpha)$ and includes flap deflection, slat extension and gear deployment corrections. So for a DC9-30 landing at sea level and for all $\alpha \in [-5°, \alpha^{\max}_\delta]$, the lift and drag terms in (6) are given by

$$L(\alpha, V) = 68.6\ (h_\delta + 4.2\alpha)V^2 \qquad D(\alpha, V) = (2.81 + 3.09\ (h_\delta + 4.2\alpha)^2)V^2 \tag{7}$$

**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. 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 Delay, or *transition* modes $0t$, $25t$ and $50t$, which use the envelopes and flight dynamics of the regular modes $0u$, $25d$ and $50d$ and include a timer to model the actual delay in flap change (the discrete automaton is shown on the right side of Figure 7). 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 $t_{\text{delay}}$ time units and the system is nonzeno. For the executions shown below, $t_{\text{delay}} = 0.5$ seconds.

**Landing:** The aircraft enters its final stage of landing close to 50 feet above ground level ([16, 18]). Restrictions on the flight path angle, aircraft velocity and touchdown (TD) speed are used to determine the initial safe set $\mathsf{E}_0$:

$$\left\{\begin{array}{ll} z \le 0 & \text{landing or has landed} \\ V > V_\delta^{\text{stall}} & \text{faster than stall speed} \\ V < V^{\max} & \text{slower than limit speed} \\ V \sin\gamma \ge \dot{z}_0 & \text{limited TD speed} \\ \gamma \le 0 & \text{monotonic descent} \end{array}\right. \cup \left\{\begin{array}{ll} z > 0 & \text{aircraft in the air} \\ V > V_\delta^{\text{stall}} & \text{faster than stall speed} \\ V < V^{\max} & \text{slower than limit speed} \\ \gamma > -3^\circ & \text{limited descent flight path} \\ \gamma \le 0 & \text{monotonic descent} \end{array}\right. \tag{8}$$

We again draw on numerical values for a DC9-30 [16]: 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 $\dot{z}_0 = 0.9144$ 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 $\alpha \in [0^\circ,\ 10^\circ]$.

The interior of the surface shown in the first row of Figure 8 represents $\mathsf{E}_0$ for each mode. The second row of the figure shows the safe envelope $\mathsf{E}$ when there is no mode switching. Portions of $\mathsf{E}_0$ are excluded from $\mathsf{E}$ 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 ($z \le 0$) are controllable in this mode. The third row shows how $\mathsf{E}$ 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 $\mathsf{E}$ 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 $\mathsf{G}$ and the dark grey are safe $\mathsf{E}$. The figure shows that modes $0u$ and $25d$ are safe only because there exists a discrete switch to a safe state in another mode.

## 4 Overapproximations of Reachable Sets

In the reachable set computation of the previous section, computational complexity introduced by increasing the dimension of the continuous state is a limiting factor: the technique suffers from expo-

nential growth with respect to the continuous dimension. To challenge this curse of dimensionality, we are investigating techniques for computing overapproximations to the reachable set of states.

In [19], we devise and implement a method based on projecting the reachable set of a high dimensional system into a collection of lower dimensional subspaces where computational costs are much lower. We formulate a method to evolve the lower dimensional reachable sets such that they are each an overapproximation of the full reachable set, and thus their intersection will also be an overapproximation of the reachable set: the method uses a set of lower dimensional Hamilton-Jacobi PDEs for which, in any projection, the set of disturbance inputs is augmented with the unmodeled dimensions. For the three dimensional aircraft collision avoidance problem of the previous section, the result of performing the computation in the $(x_r, y_r)$ space, and then back-projecting this set to form a cylindrical overapproximation of the reachable set in three dimensions, is computed in less than a minute.

A second technique is to overapproximate nonlinear dynamics by linear dynamics with bounded disturbance, and to overapproximate reachable sets by polyhedra. Propagation of sets is then simply described by the propagation, through linear dynamics, of linear inequalities describing the polyhedral set. With this technique however, comes the problem of the potentially unlimited number of inequalities needed to describe the polyhedral estimate. This problem has been addressed in the literature, yet previous solutions fall short in several ways: exact pruning of polytopes [20] requires complex data structures to be maintained and the number of "active" hyperplanes could still be unlimited; the use of simple polyhedral shapes for bounding the polyhedral set [21] is itself computationally intensive; the ellipsoidal overapproximation to the reachable set [22, 23, 24, 25], while an elegant and efficient representation of the set, is generally too conservative an overapproximation.

We have developed a technique which combines the efficiency of the ellipsoidal data structure with the tightness of polyhedral overapproximation. The heart of our technique is a novel pruning procedure based on the Maximum Volume Ellipsoid (MVE) contained in the polyhedron. The basic idea of the method is to rank the constraints by their distance (measured in the norm induced by the ellipsoid) to the center of the MVE, and delete the ones that appear to be least relevant. Conceptually, constraints that are far away from the center are less relevant than those that are close. This approach will allow us to perform guaranteed pruning (where only provably redundant constraints are deleted).

For example, consider a discrete-time linear system (which represents a linear overapproximation of the nonlinear systems of the previous section):

$$x(t+1) = Ax(t) + Bw(t)$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $w(t) \in \mathbb{R}^d$ is a bounded disturbance, and $A$ and $B$ are matrices with consistent dimensions. We assume that $w(t) \in \mathcal{W}(t)$ for some bounded convex set $\mathcal{W}(t)$. Assume that $x(t)$ is known to lie in a polyhedron $\mathcal{P}(t)$, where $\mathcal{P} \subseteq \mathbb{R}^n$ is the intersection of a finite number of

half-spaces,

$$\mathcal{P} = \bigcap_{i=1}^{m} \mathcal{H}_i$$

Using the parameterization of half-spaces introduced above, we can represent $\mathcal{P}$ in the form

$$\mathcal{P} = \{x \mid f_i^T x \leq g_i,\ i = 1, \ldots, m\} \equiv \{x \mid Fx \preceq g\} \tag{9}$$

where $\preceq$ denotes componentwise inequality. Now, an ellipsoid is defined as the image of a unit ball under an affine transformation

$$\mathcal{E} \quad = \quad \{Pu + q \mid \|u\| \leq 1\} \tag{10}$$

We assume that $A$ is invertible, which is always the case if the dynamics is discretized from some continuous-time system, and we propagate the set $\mathcal{P}(t)$ through the dynamic matrix $A$:

$$\begin{aligned} A\mathcal{P}(t) &= \{x \mid f_i(t)^T A^{-1} x \leq g_i(t),\ i = 1, \ldots, m(t)\} \\ &= \{x \mid F(t)A^{-1}x \leq g(t)\}. \end{aligned}$$

To account for the bounded disturbance, we need to find the polytope $A\mathcal{P}(t) + B\mathcal{W}(t)$. An easily computed outer approximation of it is given by

$$\begin{aligned} \widetilde{\mathcal{P}}_{t+1} = \{z \mid f_i(t)^T A^{-1} z \leq g_i(t) + \|f_i(t)^T A^{-1} B\|_1,\ i = 1, \ldots, m(t)\} \\ = \quad \{z \mid \widetilde{F}(t)z \leq \widetilde{g}(t)\}. \end{aligned}$$

As $\mathcal{P}$ evolves through hybrid dynamics, its representation grows in dimension. To limit the computational requirements for storing and operating on $\mathcal{P}$, it is therefore necessary to occasionally simplify the description of $\mathcal{P}$ by dropping, or discarding, constraints. We will call this procedure polyhedral pruning.

**The maximum volume inscribed ellipsoid (MVE)**

We are interested in finding the maximum volume ellipsoid that is contained in a polyhedron. Let $\mathcal{E}$ be an ellipsoid of the form (10), and $\mathcal{P}$ be a polyhedron parameterized as in (9). Then, $\mathcal{E} \subset \mathcal{P}$ if and only if

$$\sup_{x \in \mathcal{E}} f_i^T x \leq g_i, \quad i = 1, \ldots, m$$

Using the parameterization of the ellipsoid, this condition can be expressed as

$$\sup_{\|u\|_2 \leq 1} f_i^T P u + f_i^T q \leq g_i, \quad i = 1, \ldots, m$$

Maximization of the left hand side expression gives the equivalent containment condition

$$\|P f_i\|_2 + f_i^T q \leq g_i, \quad i = 1, \ldots, m$$

Hence, we can find the MVE inside a polyhedron by solving the convex optimization problem

$$\begin{array}{ll} \text{maximize} & \log \det P \\ \text{subject to} & P = P^T \succeq 0 \\ & \|Pf_i\|_2 + f_i^T q \leq g_i, \quad i = 1, \ldots, m \end{array}$$

This problem follows in the general class of maxdet problems, which can be solved globally and efficiently using the algorithms described in [26]. Moreover, specialized algorithms for computing the MVE have been proposed in [27, 28]. The MVE gives an optimal (in terms of volume) ellipsoidal approximation of the polyhedron. Clearly, the MVE also provides an inner bound on the polyhedron, since $\mathcal{E}_{\text{mve}} \subseteq \mathcal{P}$. More surprisingly, perhaps, is that the MVE also provides an *outer* bound of the polyhedron: the ellipsoid obtained by scaling the MVE a factor $n$ around its center is guaranteed to contain the polyhedron:

$$\mathcal{E}_{\text{mve}} \subseteq \mathcal{P} \subseteq q_{\text{mve}} + n(\mathcal{E}_{\text{mve}} - q_{\text{mve}})$$

This is a dual result to the celebrated Löwner-John ellipsoid [29, 30] (which treats the minimum volume ellipsoid that contains a convex set). Thus, the MVE gives a good approximation of a polyhedron, and provides simultaneous inner and outer bounds.

**Polyhedral pruning using the MVE**

The MVE also gives a natural measure for ranking the relevance of the constraints that define $\mathcal{P}$. For a halfspace $\mathcal{H}_i = \{x \mid f_i^T x \leq g_i\}$, we define the *relevance ranking* $\eta_i$ as the largest factor by which the ellipsoid can be enlarged and yet still lie within the halfspace:

$$\eta_i = \max\{\alpha \mid q_{\text{mve}} + \alpha(\mathcal{E}_{\text{mve}} - q_{\text{mve}}) \subseteq \mathcal{H}_i\}$$

It is easy to verify that the relevance ranking is given by

$$\eta_i = (g_i - f_i^T q)/\|Pf_i\|_2$$

This is, in fact, nothing more than the minimum distance from the halfspace to the center of the ellipsoid in the $P^2$ metric. The relevance ranking has the following properties:

1. $\eta_i \geq 1$ for all $i$
2. $\eta_i = 1 \Rightarrow f_i^T x \leq g_i$ active
3. $\eta_i \geq n \Rightarrow f_i^T x \leq g_i$ redundant

We define a constraint to be *redundant* if the polyhedron $\mathcal{P}$ does not change when the constraint is removed. We will say that the polyhedron is *minimal* if it does not contain any redundant constraints,

and we will say that a constraint is *active* if $f_i^T x^\star = g_i$ for some $x^\star \in \mathcal{P}$. A constraint is (strongly) redundant if and only it is inactive for all $x \in \mathcal{P}$. The last property allows us to discard redundant constraints. There is a gray zone $\eta_i \in (1, n)$ for which the relevance ranking neither proves that a constraint is active nor that it is redundant. In practice, guaranteed pruning can often be too conservative in that it fails to discard many redundant constraints. We therefore propose to use the ranking in a heuristic pruning procedure: given $\mathcal{P} = \{x \mid f_i^T x \leq g_i \quad i = 1, \ldots, m\}$, we compute $\mathcal{E}_{\text{mve}}$, sort the inequalities by their relevance ranking, $\eta_i$, and keep the $k$ most relevant (the ones with smallest $\eta_i$). This yields an approximate simplified polyhedron $\hat{\mathcal{P}}$, with $\mathcal{P} \subseteq \hat{\mathcal{P}}$.

We believe this algorithm to be an effective method for computing fast polyhedral overapproximations of reachable sets, in very high dimensions.

## 5 Summary

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. In this arena, we have made real progress on problems in a few dimensions, where we have close to exact solutions. We show that the computation may be made faster as we loosen the numerical approximation, however, even with such approximation, these methods will not scale beyond five or six continuous dimensions.

In response to this problem, we are developing methods that gracefully extend to problems with a much higher dimension of the continuous state (these methods have been shown to work well in dimension up to 100). While these are approximate methods, they are, however, non-heuristic: when they determine that a reachable set does not intersect some other set, the result is certain.

Finally, in order to test our reachability tools, we are designing and building a system of three Unmanned Aerial Vehicle (UAVs), with on-board sensing, navigation, and control systems enabling automatic flight. The Stanford DragonFly UAV Project currently consists of two 10-foot wingspan, fixed wing, unmanned aircraft, each equipped with GPS, inertial sensors, and on-board computers running QNX Neutrino.

## References

[1] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," in *Hybrid Systems* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), LNCS, pp. 366–392, New York: Springer Verlag, 1993.

[2] C. Tomlin, J. Lygeros, and S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 949–970, July 2000.

[3] 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.

[4] C. J. Tomlin, *Hybrid Control of Air Traffic Management Systems*. PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1998.

[5] I. Mitchell, A. Bayen, and C. J. Tomlin, "Validating a Hamilton-Jacobi approximation to hybrid system reachable sets," in *Hybrid Systems: Computation and Control* (M. D. D. Benedetto and A. Sangiovanni-Vincentelli, eds.), LNCS 2034, pp. 418–432, Springer Verlag, 2001.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] A. M. Bayen and C. J. Tomlin, "A construction procedure using characteristics for viscosity solutions of the hamilton-jacobi equation," in *Proceedings of the IEEE Conference on Decision and Control*, (Orlando, FL), 2001.

[11] 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.

[12] 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.

[13] J. A. Sethian, *Level Set Methods and Fast Marching Methods*. New York: Cambridge University Press, 1999.

[14] C. J. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 2, pp. 110–120, 2001. June.

[15] A. Degani, *Modeling Human-Machine Systems: On Modes, Error, and Patterns of Interaction*. PhD thesis, Department of Industrial and Systems Engineering, Georgia Institute of Technology, 1996.

[16] I. M. Kroo, *Aircraft Design: Synthesis and Analysis*. Stanford, California: Desktop Aeronautics Inc., 1999.

[17] J. Anderson, *Fundamentals of Aerodynamics*. New York: McGraw Hill Inc., 1991.

[18] United States Federal Aviation Administration, *Federal Aviation Regulations*, 1990. Section 25.125 (landing).

[19] I. Mitchell and C. J. Tomlin, "Overapproximating reachable sets by hamilton-jacobi projections," *Journal of Scientific Computing*, May 2002. Submitted.

[20] N. A. Bruinsma and M. Steinbuch, "A fast algorithm to compute the $\mathbf{H}_\infty$-norm of a transfer function matrix," *Systems and Control Letters*, vol. 14, no. 5, pp. 287–293, 1990.

[21] M. Milanese and G. Belforte, "Estimation theory and uncertainty intervals evaluation in presence of unknown but bounded errors – linear families of models and estimators," *IEEE Transactions on Automatic Control*, vol. 27, no. 2, pp. 408–414, 1982.

[22] F. C. Schweppe, "Recursive state estimation: Unknown but bounded errors and system inputs," *IEEE Transactions on Automatic Control*, vol. AC-13, no. 1, pp. 22–28, 1968.

[23] H. S. Witsenhausen, "Sets of possible states of linear systems given perturbed observations," *IEEE Transactions on Automatic Control*, vol. 13, pp. 556–558, 1968.

[24] G. M. Bakan and V. V. Volosov, "State estimation by the ellipsoid method from continuous-time linear observations," *Cybernetics and Systems Analysis*, vol. 31, no. 4, pp. 571–582, 1995.

[25] N. Shishido and C. J. Tomlin, "Ellipsoidal approximations of reachable sets for linear games," in *Proceedings of the IEEE Conference on Decision and Control*, (Sydney, Australia), 2000.

[26] S.-P. Wu and S. Boyd, SDPSOL: *A Parser/Solver for Semidefinite Programming and Determinant Maximization Problems with Matrix Structure. User's Guide, Version Beta.* Stanford University, June 1996.

[27] Y. Nesterov and A. Nemirovsky, *Interior-point polynomial methods in convex programming*, vol. 13 of *Studies in Applied Mathematics*. Philadelphia, PA: SIAM, 1994.

[28] Y. Zhang, "An interior-point algorithm for the maximum-volume ellipsoid problem," Technical Report TR98-15, Rice University, Houston, Texas, 1998.

[29] F. John, "Extremum problems with inequalities as subsidiary conditions," in *Fritz John, Collected Papers* (J. Moser, ed.), pp. 543–560, Birkhauser, Boston, Massachussetts, 1985.

[30] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. Algorithms and Combinatorics, New York: Springer-Verlag, 1988.

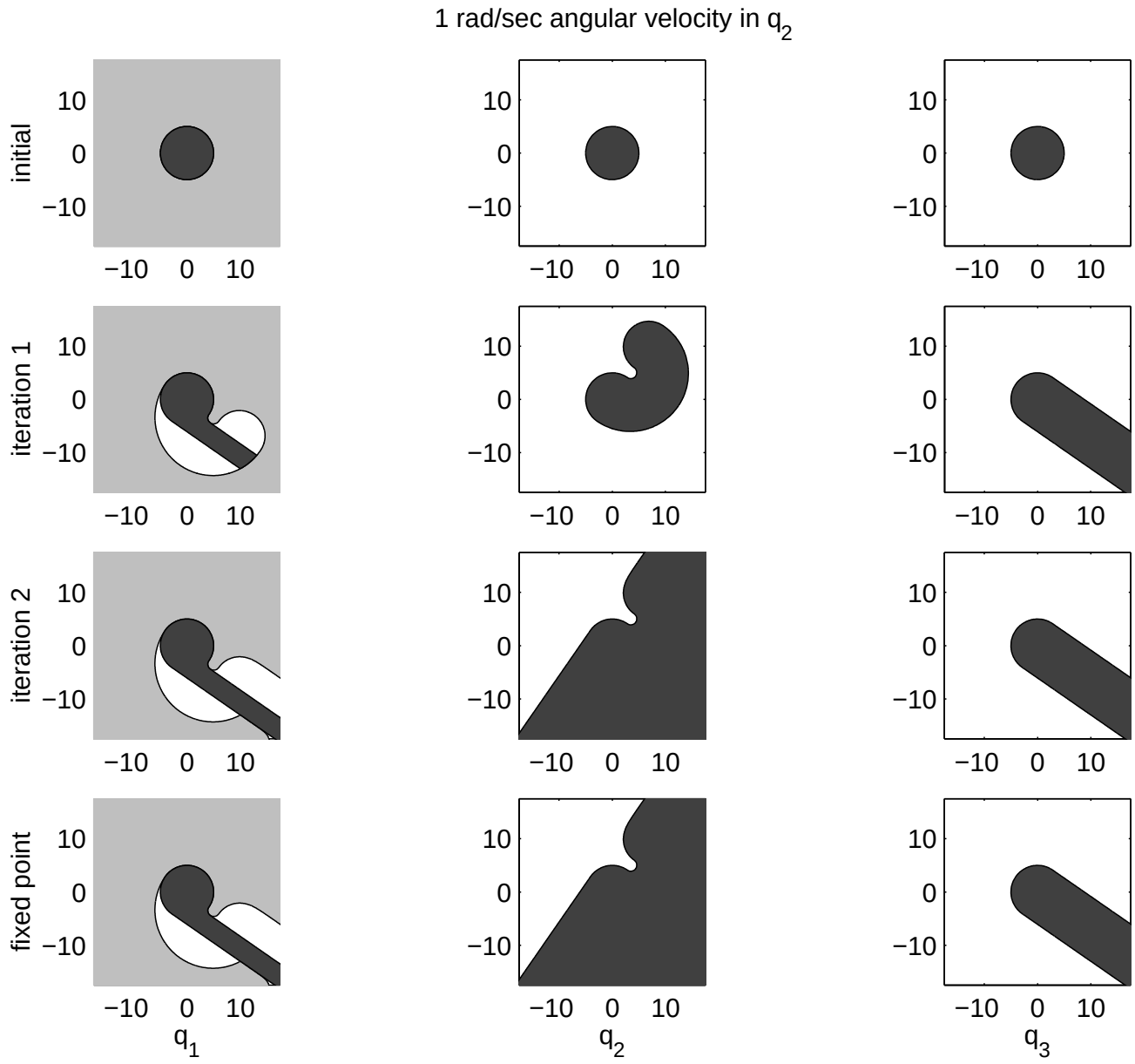


Figure 6: The rows indicate the iterations of the algorithm, the columns indicate the discrete state. Each figure displays a projection of part of the unsafe set onto the $(x_r, y_r)$ axes. A fixed point is reached after 3 iterations. Angular velocity $\omega_i = 1$ in $q_2$.

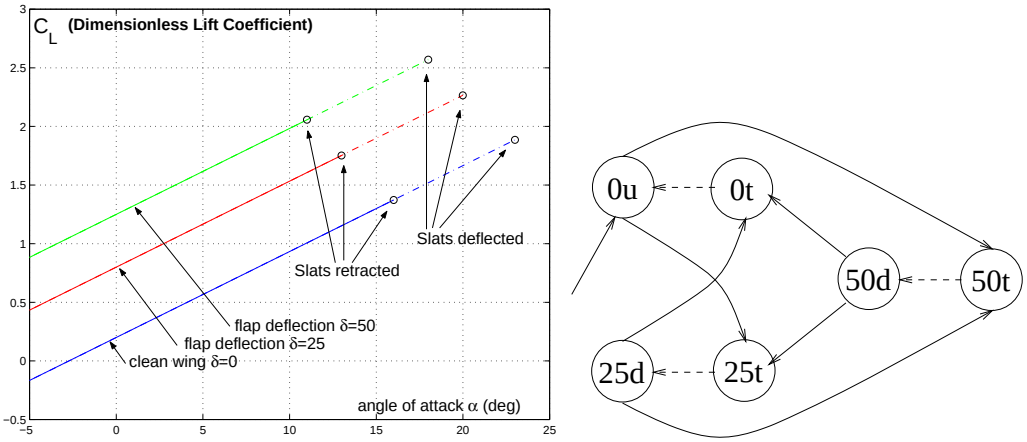


Figure 7: Left: lift coefficient $C_L(\alpha)$ model for the DC9-30 [16]. Circles located at $(\alpha_\delta^{\max}, C_L(\alpha_\delta^{\max}))$ indicate the stall angle and the corresponding lift coefficient in each mode. Right: Discrete transition graph of slat and flap settings. Solid lines are controlled switches ($\sigma_{\mathsf{E}}$ in this version of the reachability problem) and dashed lines are uncontrolled switches ($\sigma_{\mathsf{G}}$).