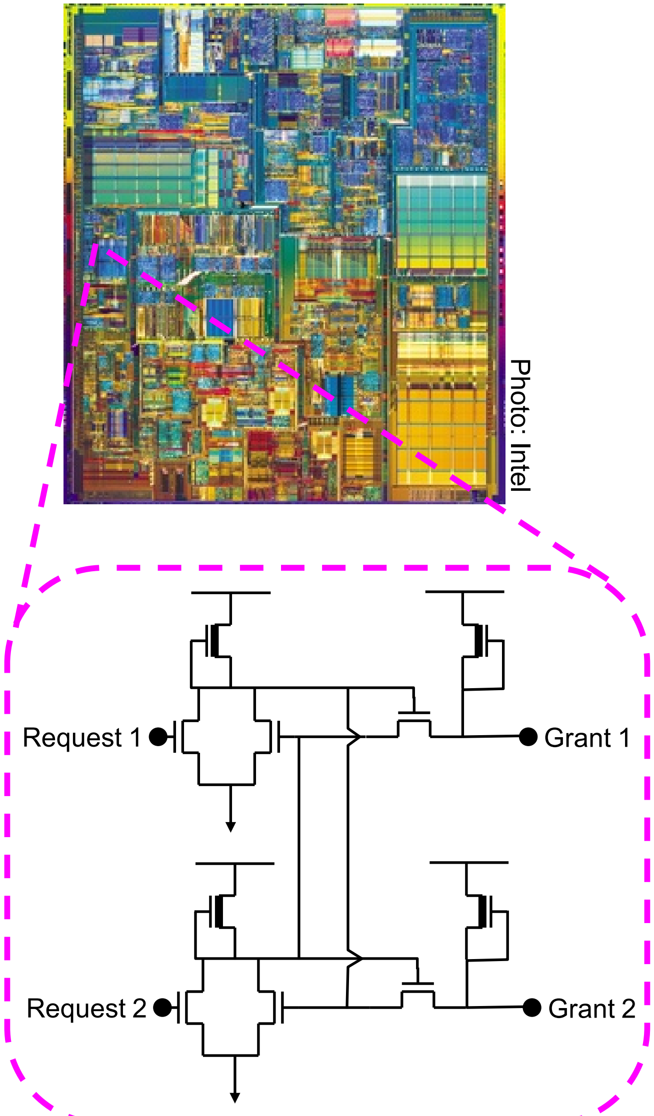
Application of Level Set Methods to Control & Reachability Problems in Continuous & Hybrid Systems

Ian Mitchell

Scientific Computing & Computational Mathematics Program

Stanford

Lots of Complex Systems



asynchronous arbiter

automation interfaces

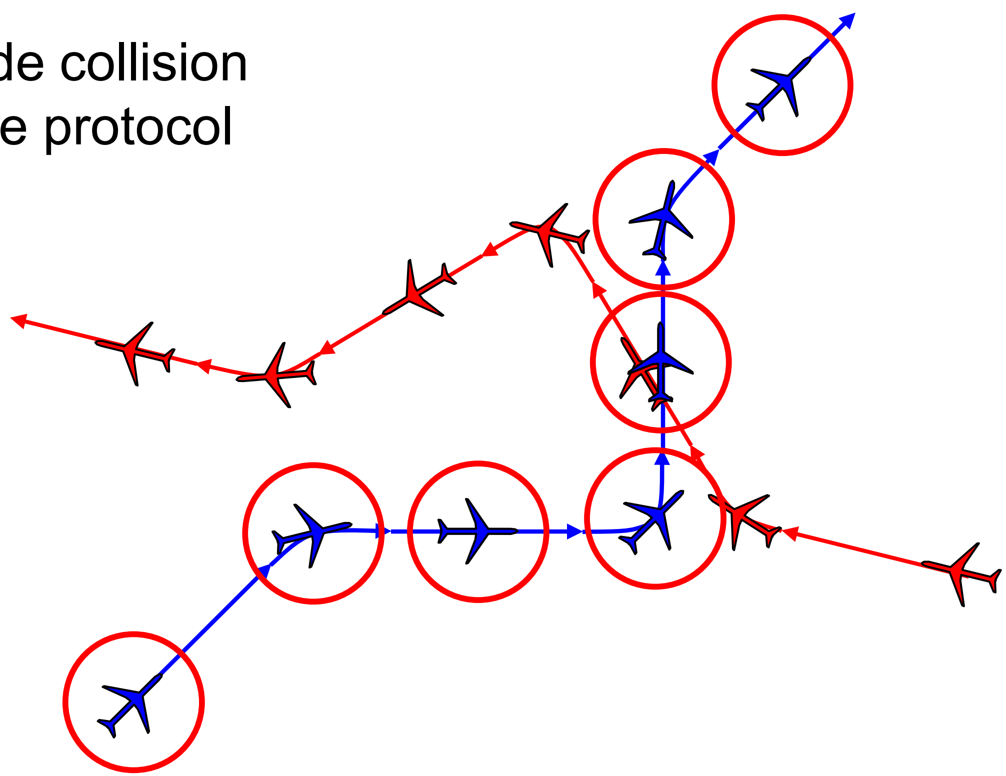
autonomous robots

Photo: ActivMedia

Why Hybrid Systems?

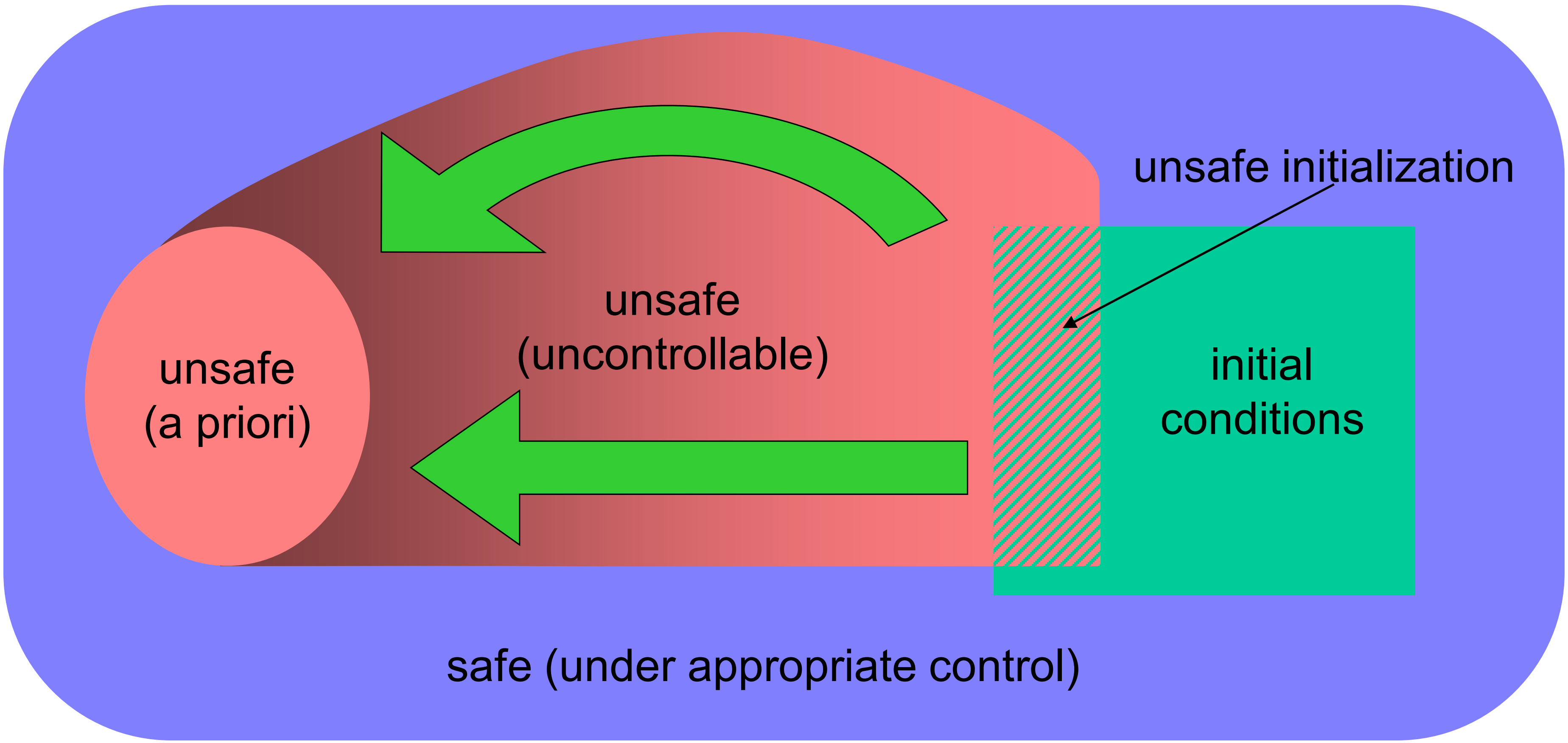
- Computers are increasingly interacting with external world
  - Flexibility of such combinations yields huge design space
  - Design methods and tools targeted (mostly) at either continuous or discrete systems
- Example: aircraft flight control systems

seven mode collision avoidance protocol



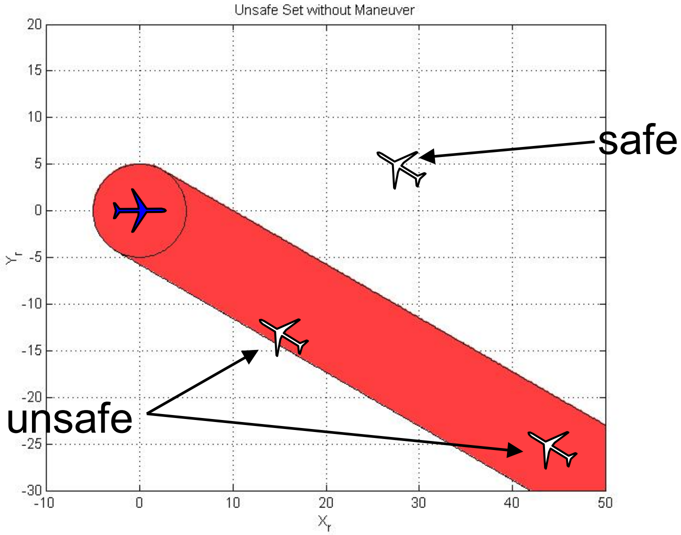
Reachable Sets: What and Why?

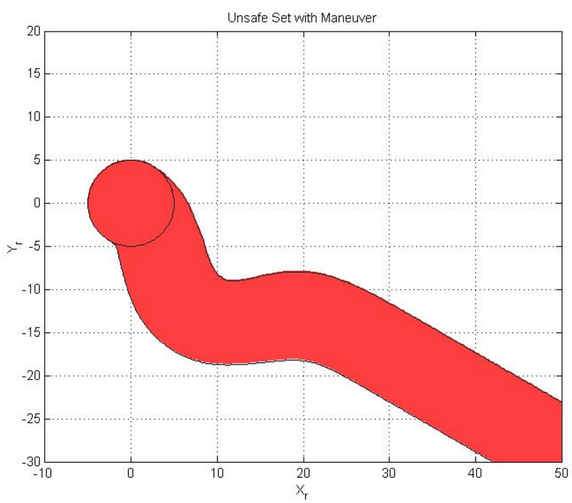
- One application: safety analysis
  - What states are doomed to become unsafe?
  - What states are safe given an appropriate control strategy?



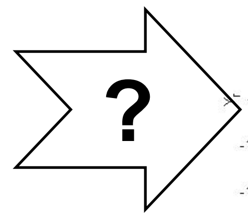
Seven Mode Safety Analysis

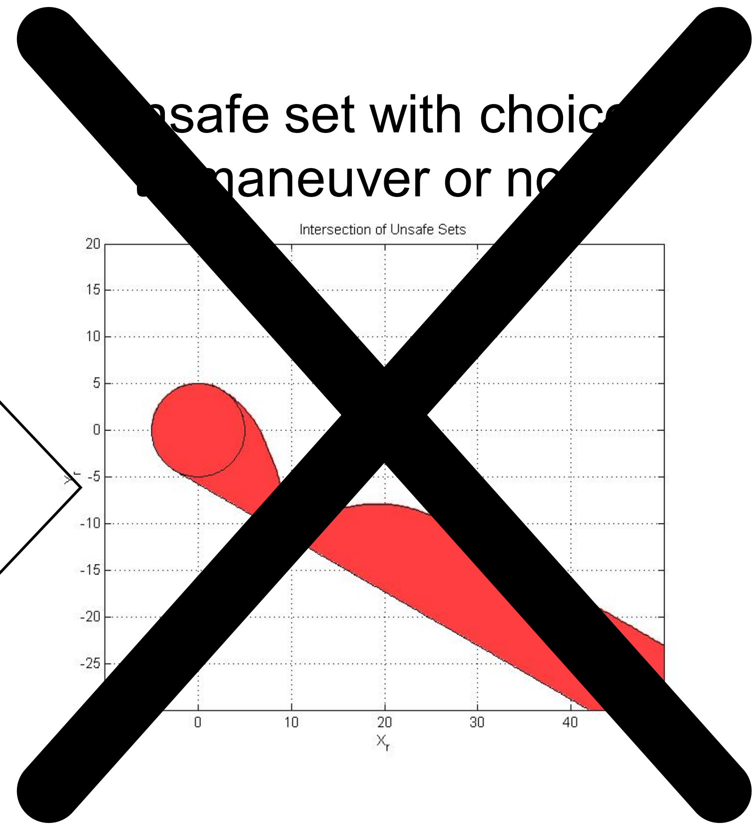
unsafe set without maneuver





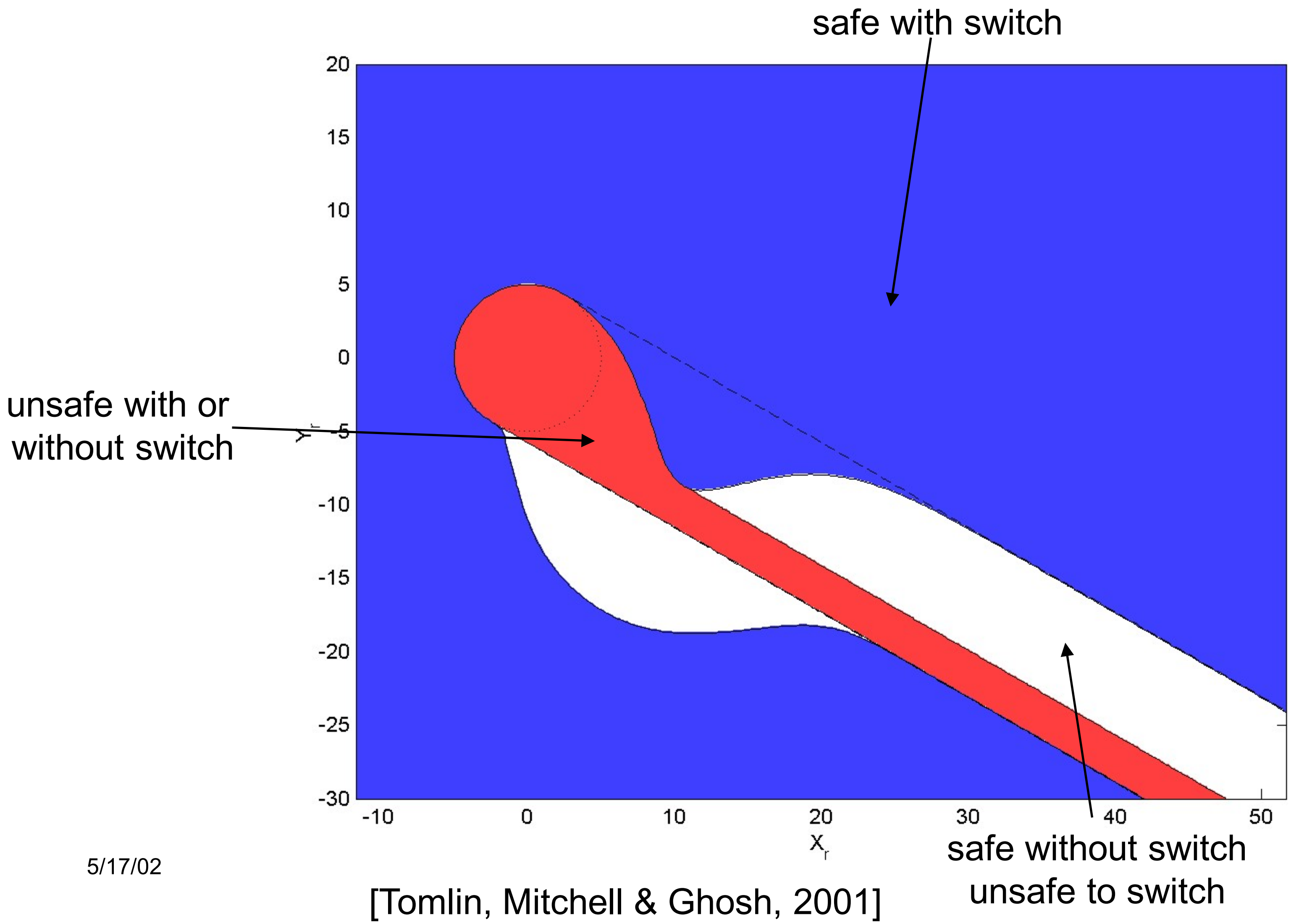
unsafe set with maneuver





Seven Mode Safety Analysis

- Ability to choose maneuver start time further reduces unsafe set



[Tomlin, Mitchell & Ghosh, 2001]

My Contributions

- Proved correctness of Hamilton-Jacobi-Isaacs formulation for continuous backwards reachable sets
- Implemented high resolution level set algorithms to capitalize on this time dependent PDE formulation
- Adapted projection concepts into HJI framework to improve scalability
- Demonstrated accurate computation of reachable sets for nonlinear continuous and hybrid systems
- Applied computed reachable sets to safety verification, control synthesis and discrete abstraction problems in aircraft automation design

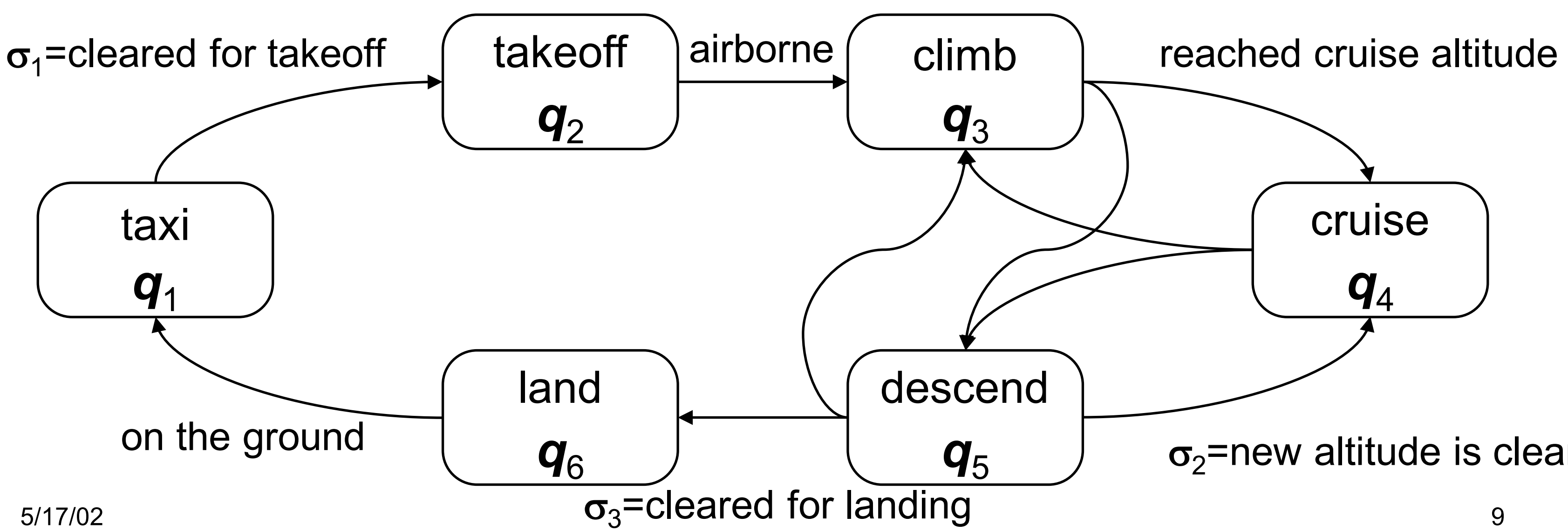
Outline

- The discrete, the continuous and the somewhere in between
  - Hybrid systems
- What are reachable sets?
  - Treating unknown inputs / parameters
- Computing reachable sets for continuous systems
  - A modified Hamilton-Jacobi-Isaacs equation
  - Application: synthesizing a safe control policy
  - Projective overapproximation of reachable sets
- Computing reachable sets for hybrid systems
  - The reach-avoid operator
  - Application: discrete abstraction
  - Application: an aircraft autolander
- Summary

Finite Automata

$$q(t+1) = \delta\big(q(t), \sigma(t)\big)$$

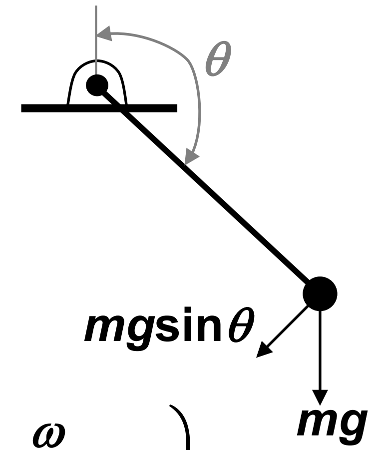
$q_i \in Q$	discrete states
$q(t) : \square \to Q$	discrete trajectrory
$\sigma_i \in \Sigma$	discrete actions
$\sigma(t) : \square \to \Sigma$	action signal
$\delta : Q \times \Sigma \to 2^Q$	transition function



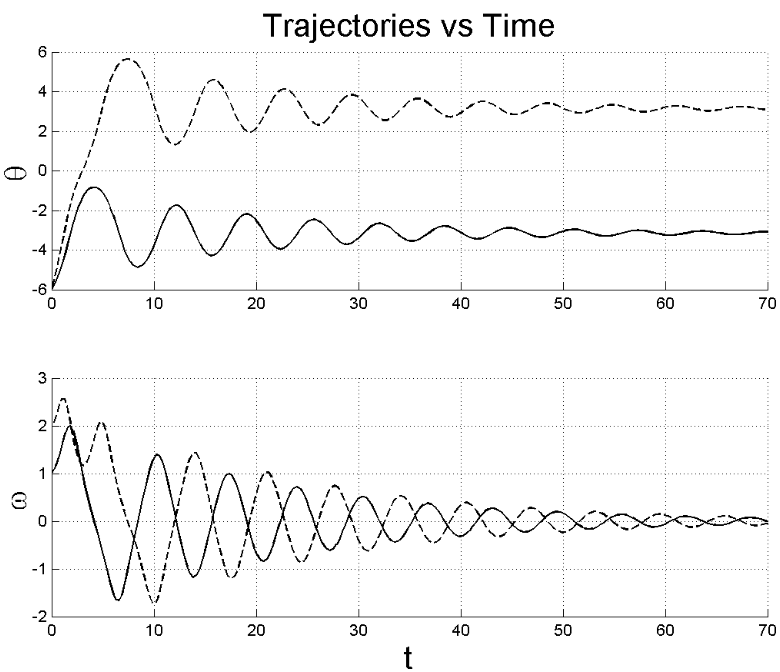
Differential Equations

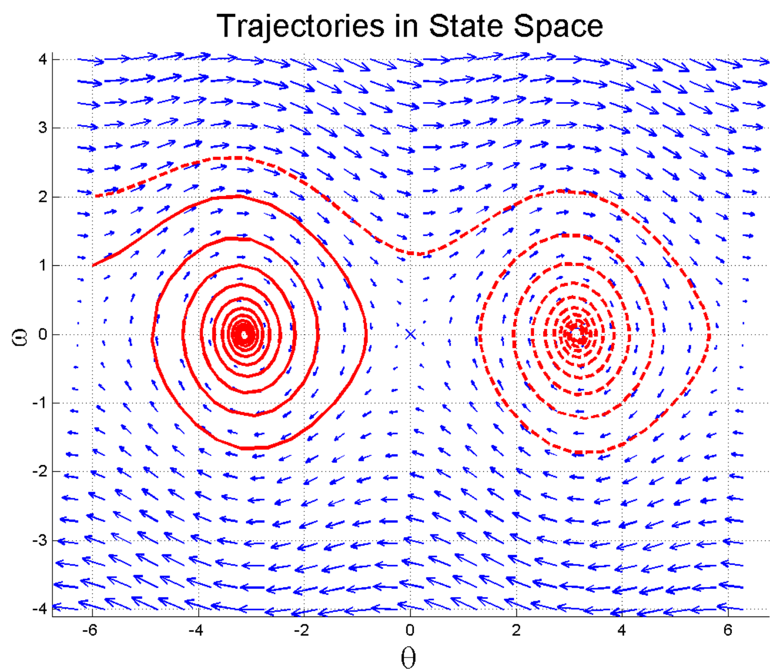
$$\dot{\boldsymbol{x}}(\boldsymbol{t}) = {}^{dx(t)}\!/_{dt} = \boldsymbol{f}\left(\boldsymbol{x}(\boldsymbol{t}), \upsilon(\boldsymbol{t})\right)$$

$\boldsymbol{x} \in \mathbb{R}^{n}$	continuous state
$\boldsymbol{x}(\boldsymbol{t}) : \mathbb{R} \to \mathbb{R}^{n}$	continuous trajectory
$\upsilon \in \Upsilon \subset \mathbb{R}^{n_\upsilon}$	continuous inputs
$\upsilon(\boldsymbol{t}) : \mathbb{R} \to \Upsilon$	input signal
$\boldsymbol{f} : \mathbb{R}^{n} \times \Upsilon \to \mathbb{R}^{n}$	flow field

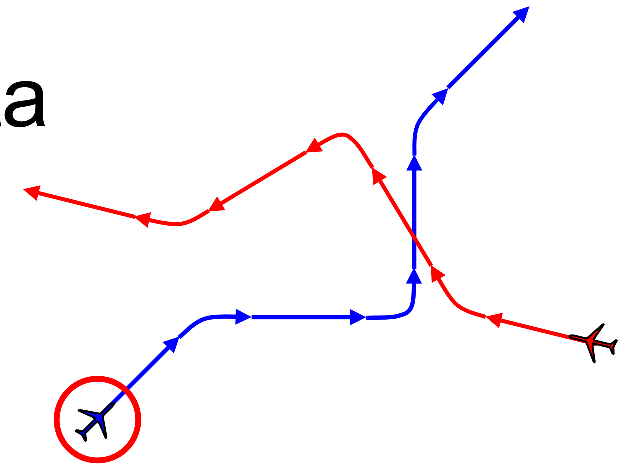


$$\frac{\boldsymbol{d}}{\boldsymbol{dt}}\begin{pmatrix}\theta \\ \omega\end{pmatrix} = \begin{pmatrix}\omega \\ \sin\theta - \mathbf{0.1}\omega\end{pmatrix}$$





Hybrid Automata



- Discrete modes and transitions
- Continuous evolution within each mode

σ_1 = initiate maneuver

straight1: $\dot{\boldsymbol{x}} = \boldsymbol{f}_S(\boldsymbol{x},\upsilon)$, $\boldsymbol{q}_1$ $\xrightarrow{\sigma_1}$ arc1: $\dot{\boldsymbol{x}} = \boldsymbol{f}_A(\boldsymbol{x},\upsilon)$, $\boldsymbol{q}_2$ $\xrightarrow{t = \pi/4}$ straight2: $\dot{\boldsymbol{x}} = \boldsymbol{f}_S(\boldsymbol{x},\upsilon)$, $\boldsymbol{q}_3$ $\xrightarrow{t = T}$ arc2: $\dot{\boldsymbol{x}} = \boldsymbol{f}_A(\boldsymbol{x},\upsilon)$, $\boldsymbol{q}_4$ $\xrightarrow{t = \pi/2}$ straight3: $\dot{\boldsymbol{x}} = \boldsymbol{f}_S(\boldsymbol{x},\upsilon)$, $\boldsymbol{q}_5$ $\xrightarrow{t = T}$ arc3: $\dot{\boldsymbol{x}} = \boldsymbol{f}_A(\boldsymbol{x},\upsilon)$, $\boldsymbol{q}_6$ $\xrightarrow{t = \pi/4}$ straight4: $\dot{\boldsymbol{x}} = \boldsymbol{f}_S(\boldsymbol{x},\upsilon)$, $\boldsymbol{q}_7$

$$\boldsymbol{f}_S\begin{pmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{pmatrix} = \begin{pmatrix}-\boldsymbol{v} + \boldsymbol{v}\cos\psi\\ \boldsymbol{v}\sin\psi\end{pmatrix}$$

dynamics in straight modes

$$\boldsymbol{f}_A\begin{pmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{pmatrix} = \begin{pmatrix}-\boldsymbol{v} + \boldsymbol{v}\cos\psi - \boldsymbol{x}_2\\ \boldsymbol{v}\sin\psi + \boldsymbol{x}_1\end{pmatrix}$$

dynamics in arc modes

Outline

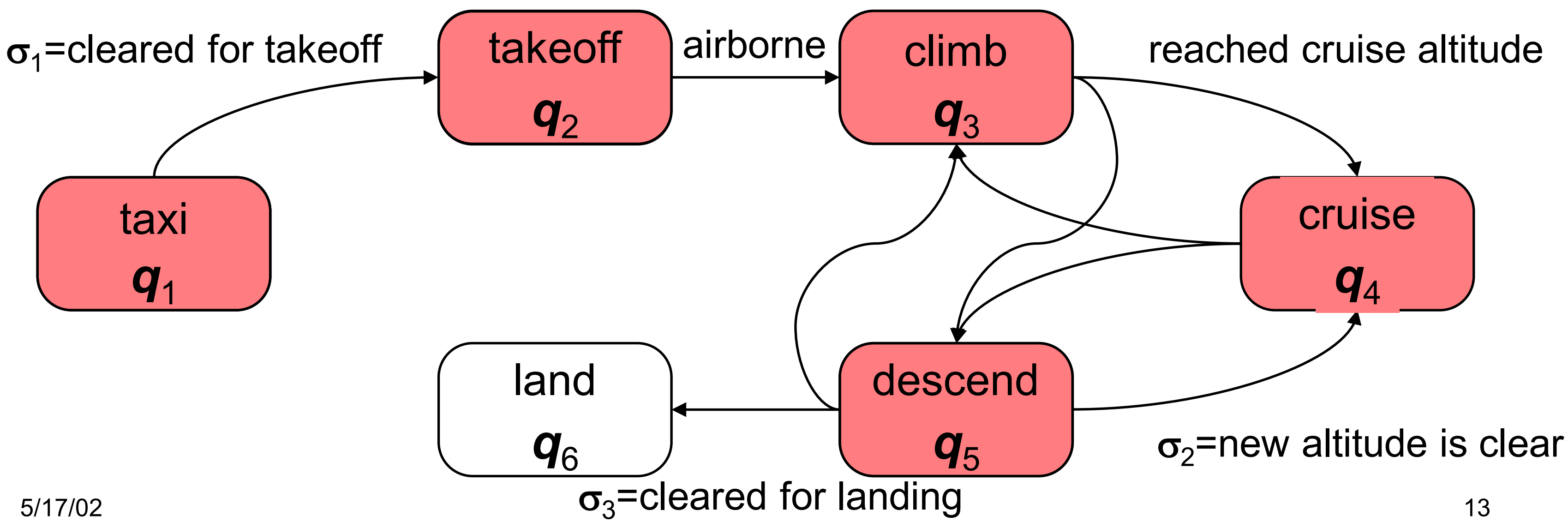
- The discrete, the continuous and the somewhere in between
  - Hybrid systems
- What are reachable sets?
  - Treating unknown inputs / parameters
- Computing reachable sets for continuous systems
  - A modified Hamilton-Jacobi-Isaacs equation
  - Application: synthesizing a safe control policy
  - Projective overapproximation of reachable sets
- Computing reachable sets for hybrid systems
  - The reach-avoid operator
  - Application: discrete abstraction
  - Application: an aircraft autolander
- Summary

Discrete Backward Reachable Sets

- Set of all states from which trajectories can reach some given target state
  - For example: from what states can we reach "cruise"?

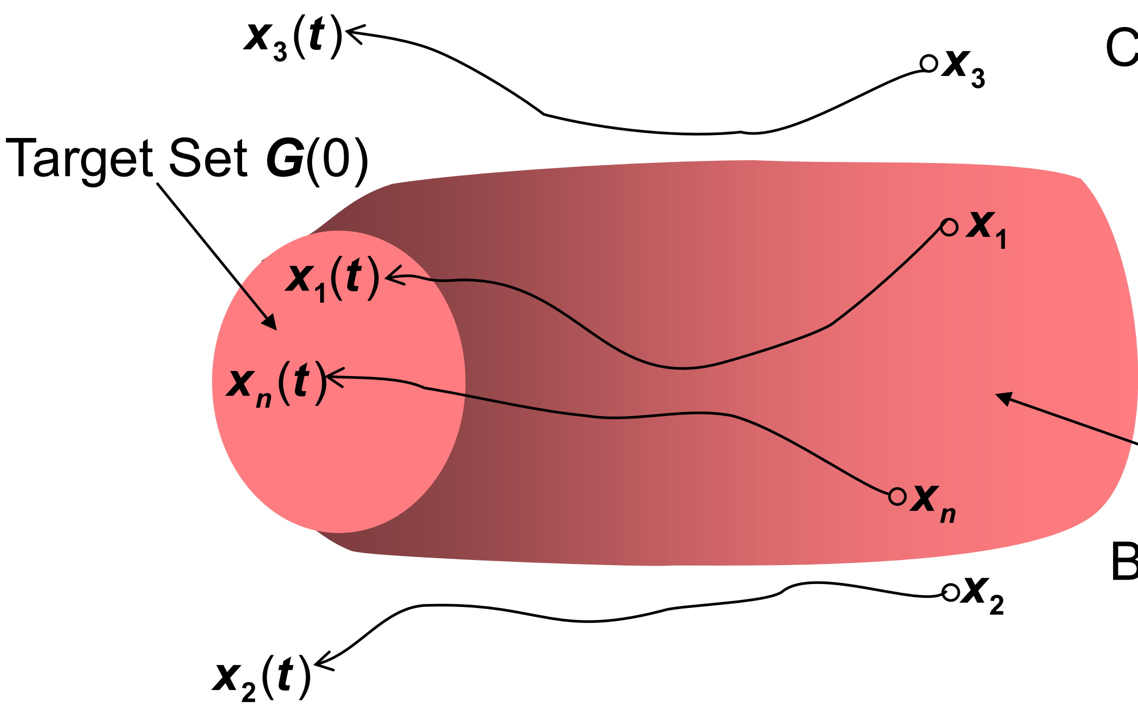
Discrete System Dynamics

$$q(t+1) = \delta\left(q(t), \sigma(t)\right)$$



Continuous Backward Reachable Sets

- Set of all states from which trajectories can reach some given target state
  - For example, what states can reach $\boldsymbol{G}(0)$?



Continuous System Dynamics

$$\dot{x}(t) = f\left(x(t)\right)$$

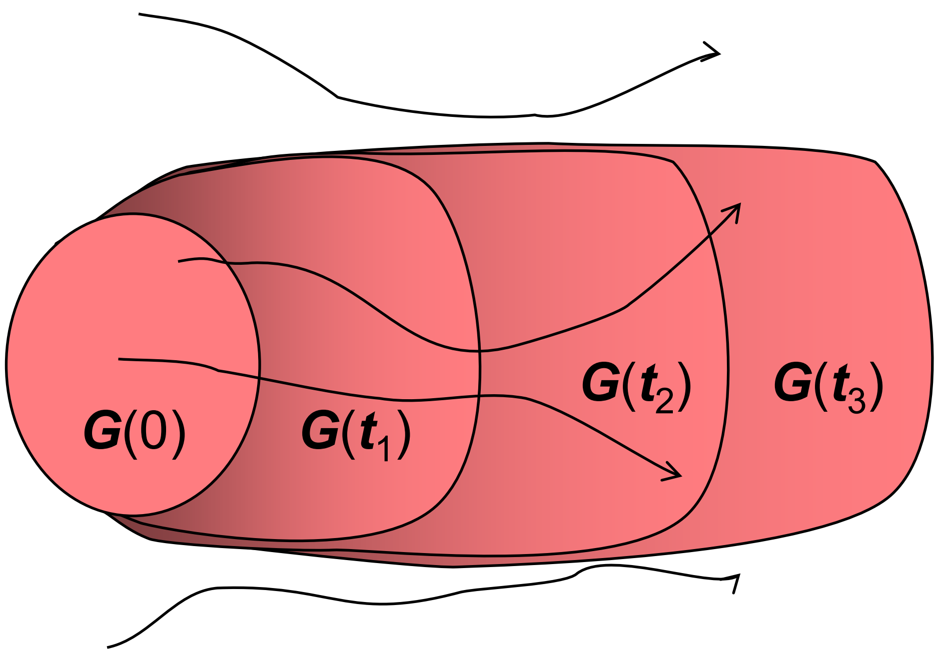
Backward Reachable Set $\boldsymbol{G}(\boldsymbol{t})$

$$x(t) \in G(0)$$

Why "Backward" Reachable Sets?

- To distinguish from forward reachable set
- To compute, run dynamics backwards in time from target set

$$\dot{x}(t) = -f\left(x(t)\right)$$

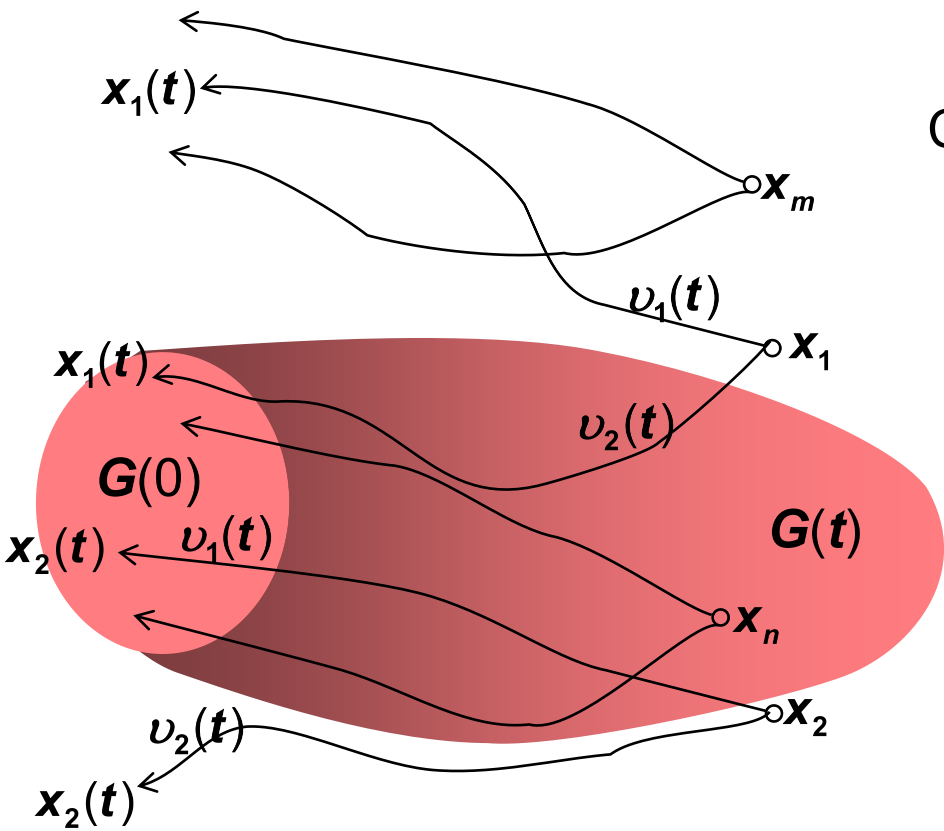


$$0 < t_1 < t_2 < t_3$$

$$G(0) \subseteq G(t_1) \subseteq G(t_2) \subseteq G(t_3)$$

Reachable Sets (controlled input)

- For most of our examples, target set is unsafe
- If we can control the input, choose it to avoid the target set
- Backward reachable set is unsafe no matter what we do



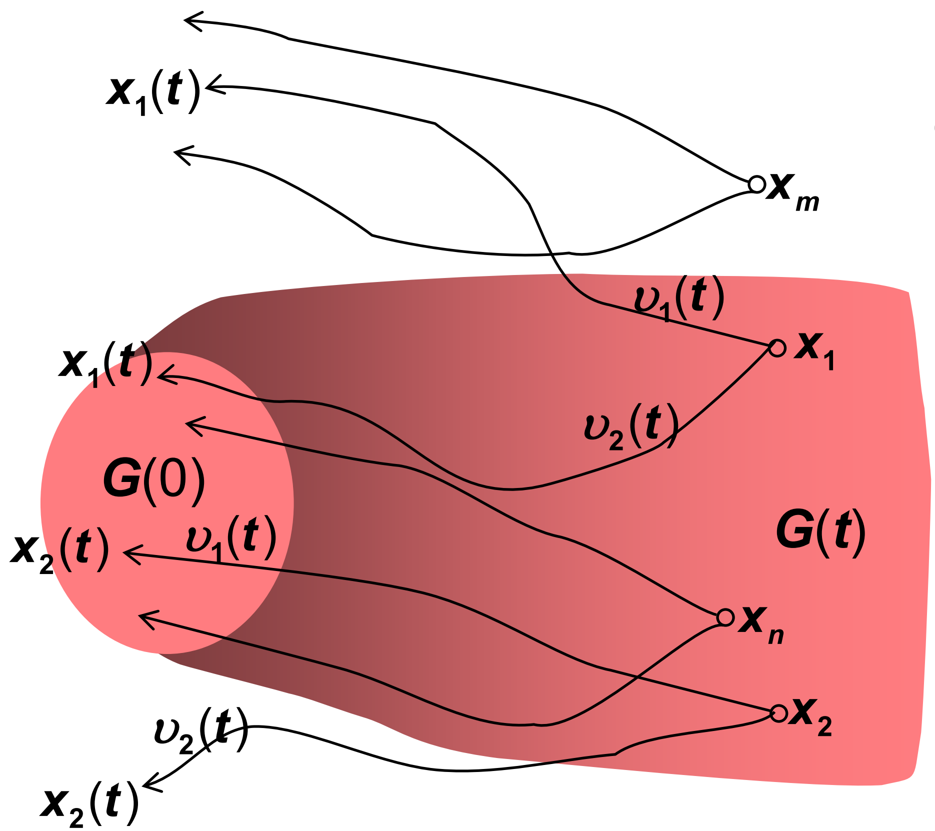
Continuous System Dynamics

$$\dot{x}(t) = f\left(x(t), \upsilon(t)\right)$$

$$\forall \upsilon(\cdot), x(t) \in G(0)$$

Reachable Sets (uncontrolled input)

- Sometimes we have no control over input signal
  - noise, actions of other agents, unknown system parameters
- It is safest to assume the worst case



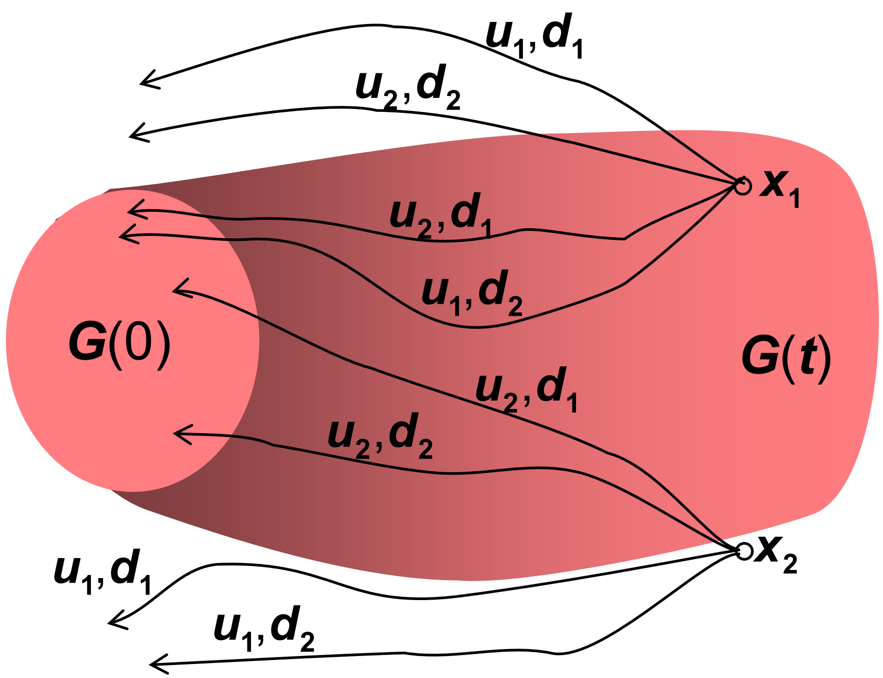
Continuous System Dynamics

$$\dot{x}(t) = f\left(x(t), \upsilon(t)\right)$$

$$\exists \upsilon(\cdot), x(t) \in G(0)$$

Two Competing Inputs

- For some systems there are two classes of inputs $\upsilon = (\boldsymbol{u}, \boldsymbol{d})$
  - Controllable inputs $\boldsymbol{u} \in \boldsymbol{U}$
  - Uncontrollable (disturbance) inputs $\boldsymbol{d} \in \boldsymbol{D}$
- Equivalent to a zero sum differential game formulation
  - If there is an advantage to input ordering, give it to disturbances



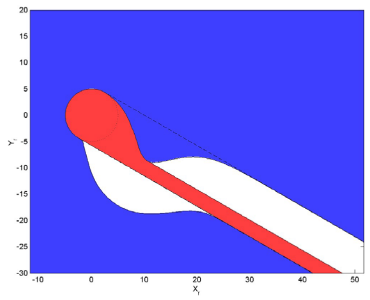
Continuous System Dynamics

$$\dot{x}(t) = f\big(x(t), u(t), d(t)\big)$$

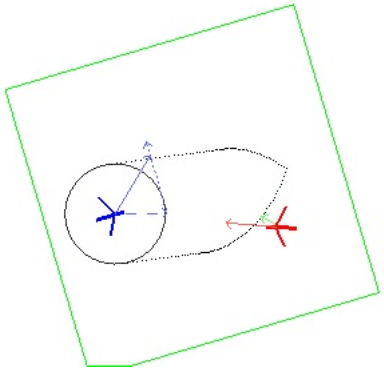
$$\forall u(\cdot), \exists d(\cdot), x(t) \in G(\mathbf{0})$$

Reachability Applications

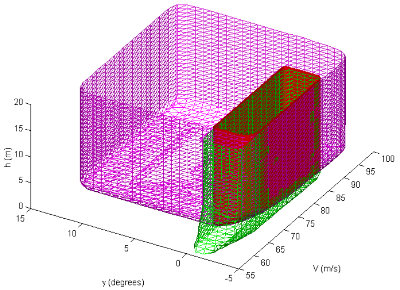
- Safety Verification
  - Seven mode collision avoidance



- Synthesizing safe controllers that can be implemented
  - Collision avoidance filter



- Abstracting continuous behaviors to discrete models
  - Pilot interface for safe TOGA maneuver



Outline

- The discrete, the continuous and the somewhere in between
  – Hybrid systems
- What are reachable sets?
  – Treating unknown inputs / parameters
- Computing reachable sets for continuous systems
  – A modified Hamilton-Jacobi-Isaacs equation
  – Application: synthesizing a safe control policy
  – Projective overapproximation of reachable sets
- Computing reachable sets for hybrid systems
  – The reach-avoid operator
  – Application: discrete abstraction
  – Application: an aircraft autolander
- Summary

Computing Continuous Reachable Sets

- Two key questions:
  - How to represent continuous sets?
  - How to evolve sets according to system dynamics?
- Two philosophies:
  - Forwards reachable set computed by following system trajectories
  - Backwards reachable set computed on a motionless grid
- My method (based on level set algorithms)
  - Backwards reachable set on a fixed grid
  - Implicit surface representation of sets
  - Solve Hamilton-Jacobi equation to evolve sets

Forward Reachability

- Forwards reachable set is computed by following trajectories
- Examples:
  - Timed automata: Uppaal [Larsen, Pettersson…], Kronos [Yovine,…], …
  - Rectangular differential inclusions: Hytech, Hypertech [Henzinger, Ho, Horowitz, Wong-Toi, …]
  - Polyhedra and linear dynamics: Checkmate [Chutinan & Krogh], d/dt [Bournez, Dang, Maler, Pnueli, …], others [Bemporad, Morari, Torrisi, …], [Greenstreet & Mitchell], …
  - Ellipsoids and linear dynamics [Botchkarev, Kurzhanski, Tripakis, Varaiya, …]
  - Discretization (predicate abstraction) on grid [Kurshan & McMillan] or by cylindrical algebraic decomposition [Tiwari & Khanna]
- Advantages: Compact representation of sets, overapproximation guarantees
- Disadvantages: Linear dynamics, reliance on trajectory optimization, restrictive set representation, potentially large error

Backwards Reachability

- Backwards reachable set is computed on a motionless grid
- Examples:
  - Time dependent Hamilton-Jacobi [Sastry, Lygeros, Tomlin, …]
  - Static Hamilton-Jacobi [Bardi, Capuzzo-Dolcetta, Falcone, …]
  - Viability theory [Aubin, Quincampoix, Saint-Pierre, …]
- Advantages: General representation of sets, nonlinear dynamics
- Disadvantages: Exponential growth of representation size with state dimension, direction of error is unknown

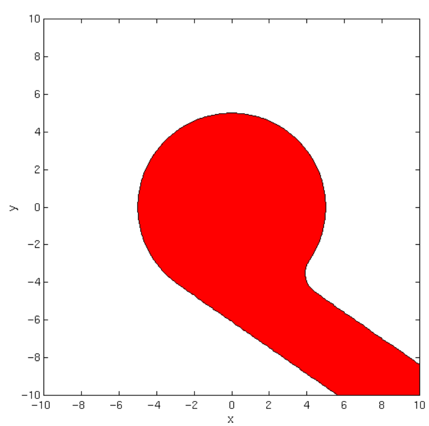
Implicit Surface Representation

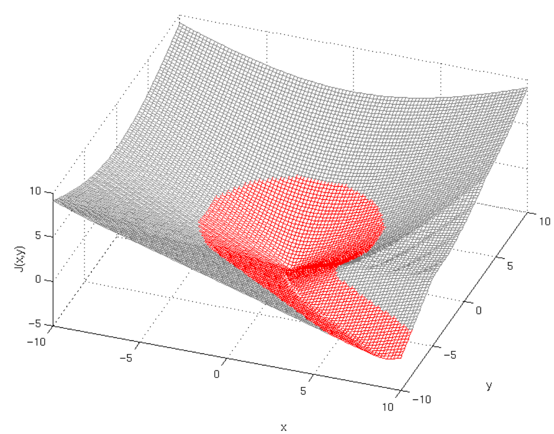
- The reachable set will be the zero sublevel set of a scalar function

$$\boldsymbol{G}(\boldsymbol{t}) = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{J}(\boldsymbol{x},\boldsymbol{t}) \le \boldsymbol{0}\}$$

- Useful fact

$$\left|\boldsymbol{D_x J}(\boldsymbol{x},\boldsymbol{t})\right| = \boldsymbol{1} \quad \Rightarrow \quad \begin{cases} \boldsymbol{D_x J}(\boldsymbol{x},\boldsymbol{t}) \text{ gives direction of nearest point on } \partial \boldsymbol{G}(\boldsymbol{t}) \\ \boldsymbol{J}(\boldsymbol{x},\boldsymbol{t}) \text{ gives distance to nearest point on } \partial \boldsymbol{G}(\boldsymbol{t}) \end{cases}$$





Evolving Reachable Sets

- Modified Hamilton-Jacobi partial differential equation

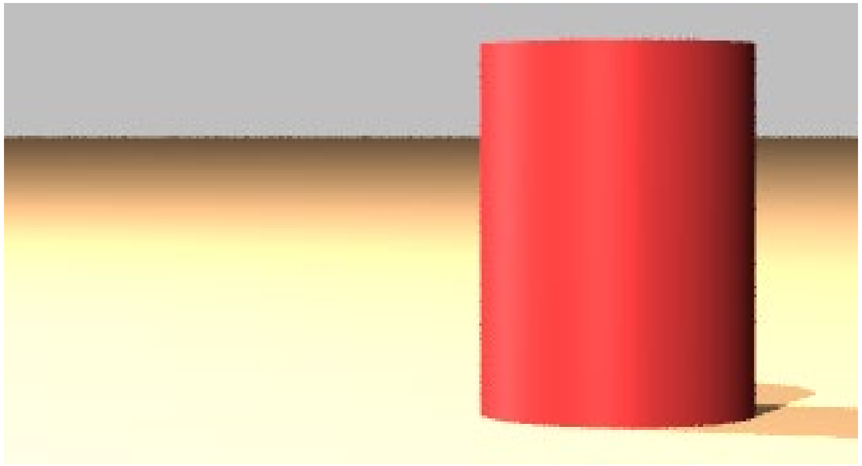
$$D_t J(x,t) + \min\left[0, H\left(x, D_x J(x,t)\right)\right] = 0$$

with Hamiltonian: $H(x,p) = \max_{u \in U} \min_{d \in D} f(x,u,d) \bullet p$

and terminal conditions: $J(x,0) = h(x)$

where $G(0) = \{x \in \mathbb{R}^n \mid h(x) \le 0\}$

and $\dot{x} = f(x,u,d)$



Hamilton-Jacobi Equation

$$D_t J(x,t) + H\big(x, D_x J(x,t)\big) = 0$$

- First order hyperbolic PDE
  - Solution can form kinks (discontinuous derivatives)
  - For the backwards reachable set, find the “viscosity” solution [Crandall, Evans, Lions, …]
- Level set methods
  - Convergent numerical algorithms to compute the viscosity solution [Osher, Sethian, …]
  - Non-oscillatory, high accuracy spatial derivative approximation
  - Stable, consistent numerical Hamiltonian
  - Variation diminishing, high order, explicit time integration

Solving a Differential Game

- Terminal cost differential game for trajectories $\xi_f(\bullet; \boldsymbol{x}, \boldsymbol{t}, \boldsymbol{u}(\bullet), \boldsymbol{d}(\bullet))$

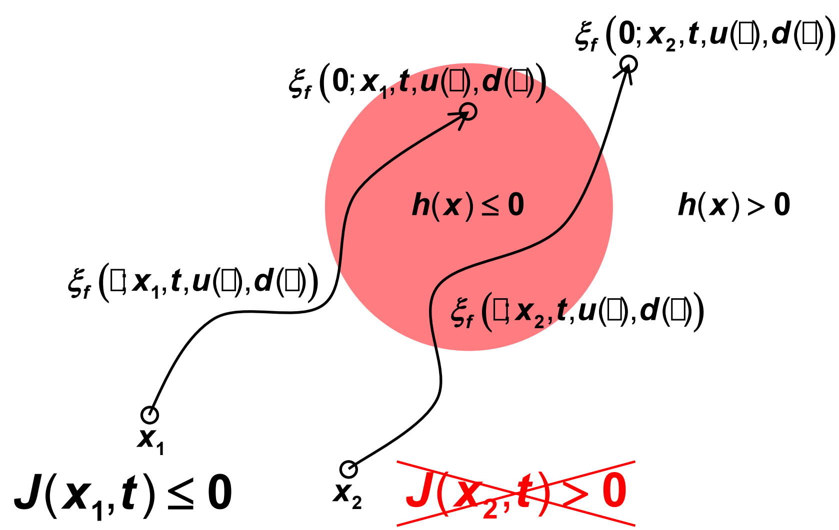
$$J(x,t) = \max_{u(\cdot)} \min_{d(\cdot)} h\left[\xi_f\left(0; x, t, u(\cdot), d(\cdot)\right)\right]$$

$$\text{where } \begin{cases} \xi_f\left(t; x, t, u(\cdot), d(\cdot)\right) = x \\ \dot{\xi}_f\left(s; x, t, u(\cdot), d(\cdot)\right) = f\left(x, u(s), d(s)\right) \\ \quad \text{terminal payoff function } h(x) \end{cases}$$

- Value function solution $\boldsymbol{J}(\boldsymbol{x},\boldsymbol{t})$ given by viscosity solution to basic Hamilton-Jacobi equation
  - [Evans & Souganidis, 1984]

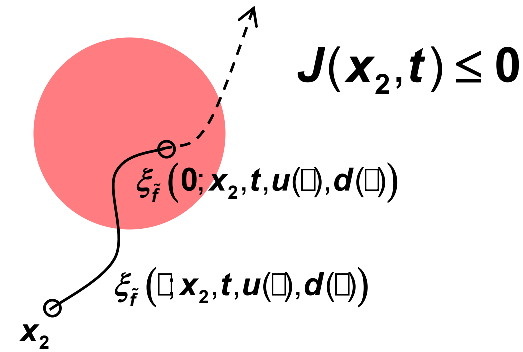
$$D_t J(x,t) + H\left(x, D_x J(x,t)\right) = 0$$

$$\text{where } \begin{cases} H(x,p) = \max_{u \in U} \min_{d \in D} f(x,u,d) \bullet p \\ J(x,0) = h(x) \end{cases}$$



Modification for Optimal Stopping Time

- How to keep trajectories from passing through G(0)?
  - [Mitchell, Bayen & Tomlin 2002]
  - Augment disturbance input



$$\tilde d = \begin{bmatrix} d & \underline{d} \end{bmatrix} \text{ where } \underline{d} : [t, 0] \to [0, 1]$$

$$\tilde f(x, u, \tilde d) = \underline{d}\, f(x, u, d)$$

  - Augmented Hamilton-Jacobi equation solves for reachable set

$$D_t J(x,t) + \tilde H\big(x, D_x J(x,t)\big) = 0 \text{ where } \begin{cases} \tilde H(x,p) = \max\limits_{u \in U} \min\limits_{\tilde d \in \tilde D} \tilde f(x,u,\tilde d) \bullet p \\ J(x,0) = h(x) \end{cases}$$

  - Augmented Hamiltonian is equivalent to modified Hamiltonian

$$\begin{aligned} \tilde H(x,p) &= \max_{u \in U} \min_{\tilde d \in \tilde D} \tilde f(x,u,\tilde d) \bullet p \\ &= \max_{u \in U} \min_{d \in D} \min_{\underline{d} \in [0,1]} \underline{d}\, f(x,u,d) \bullet p \\ &= \min\left[0, \max_{u \in U} \min_{d \in D} f(x,u,d) \bullet p\right] = \min[0, H(x,p)] \end{aligned}$$

Application: Synthesizing Safe Controllers

- By construction, on the boundary of the unsafe set there exists a control to keep trajectories safe
  - Filter potentially unsafe controls to ensure safety

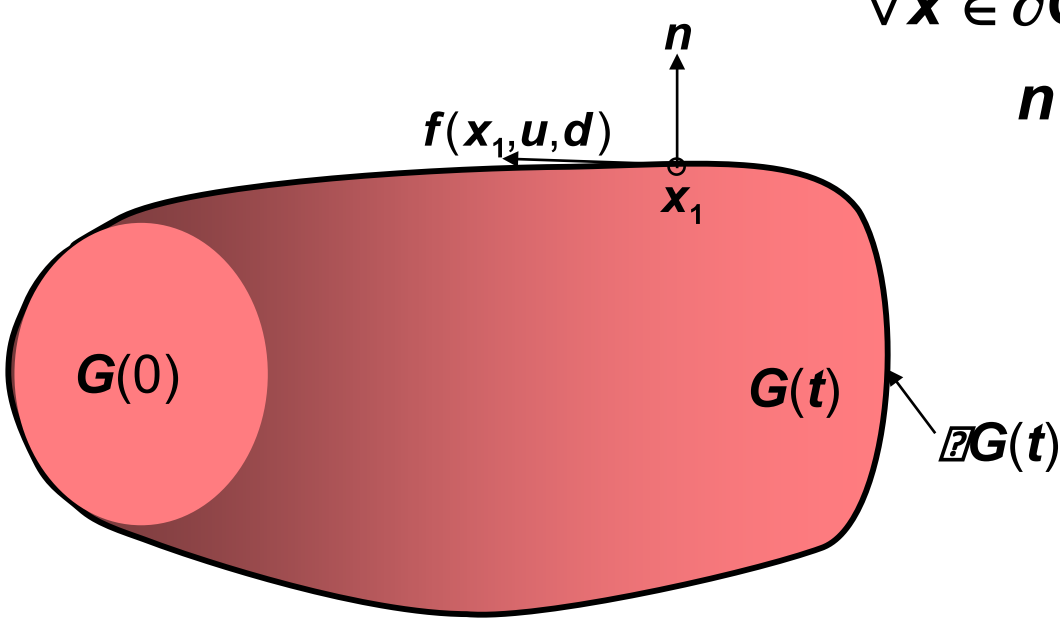
Continuous System Dynamics

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{d})$$

by construction

$$\forall \boldsymbol{x} \in \partial \boldsymbol{G}(\boldsymbol{t}), \exists \boldsymbol{u} \in \boldsymbol{U}, \forall \boldsymbol{d} \in \boldsymbol{D}$$

$$\boldsymbol{n} \cdot \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{d}) \geq \boldsymbol{0}$$

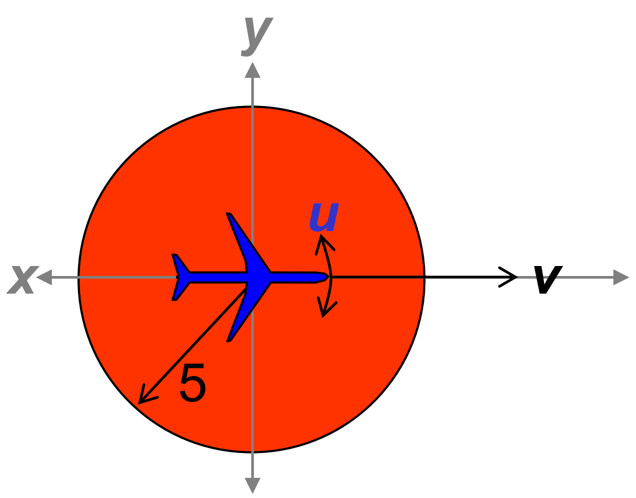


Synthesizing Safe Controllers Example

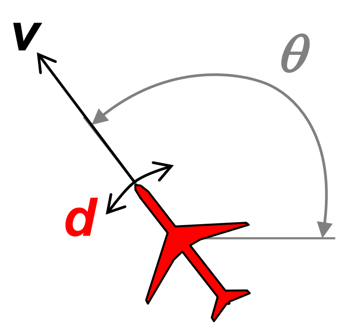
- Filter potentially unsafe input to guarantee safety
- Collision avoidance example
  - Collision occurs if vehicles get within five units of one another
  - Evader chooses turn rate $|u| \leq 1$ to avoid collision
  - Pursuer chooses turn rate $|d| \leq 1$ to cause collision
  - Fixed equal velocity $v = 5$

dynamics (pursuer)

$$\frac{d}{dt}\begin{pmatrix} x_p \\ y_p \\ \theta_p \end{pmatrix} = \begin{pmatrix} v\cos\theta_p \\ v\sin\theta_p \\ d \end{pmatrix}$$



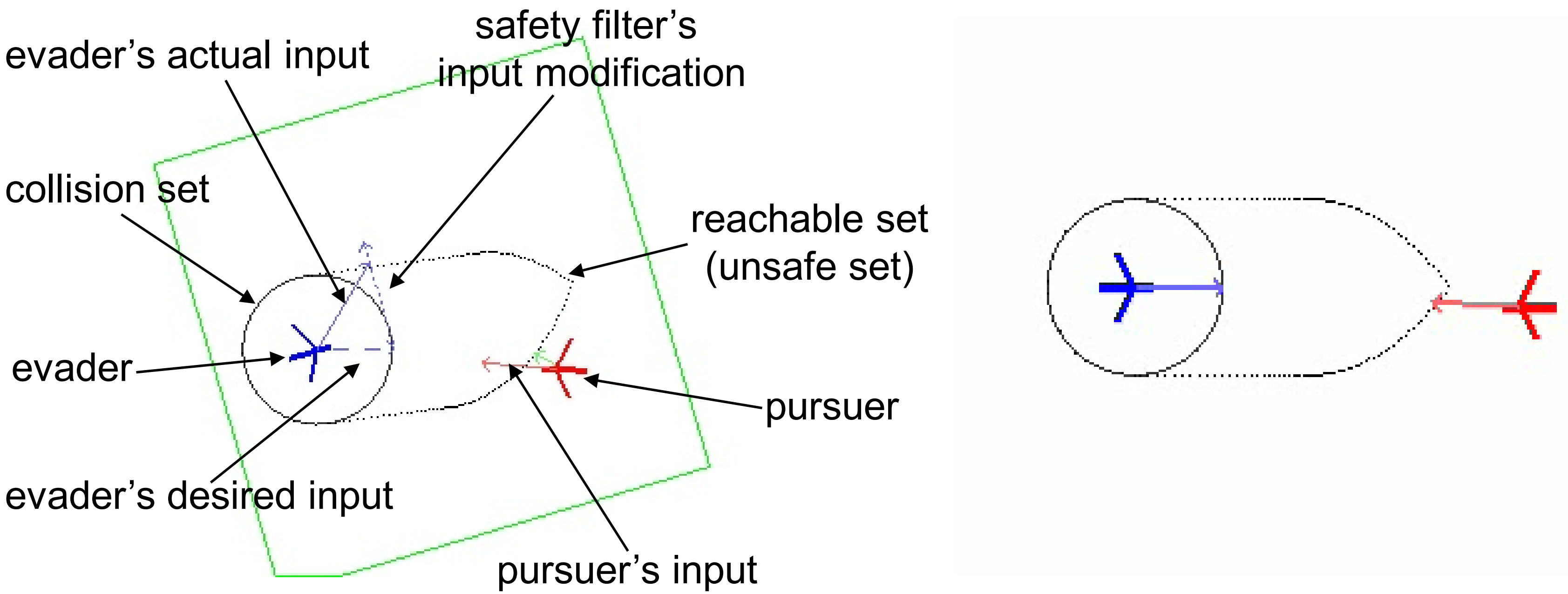
evader aircraft (control)



pursuer aircraft (disturbance)

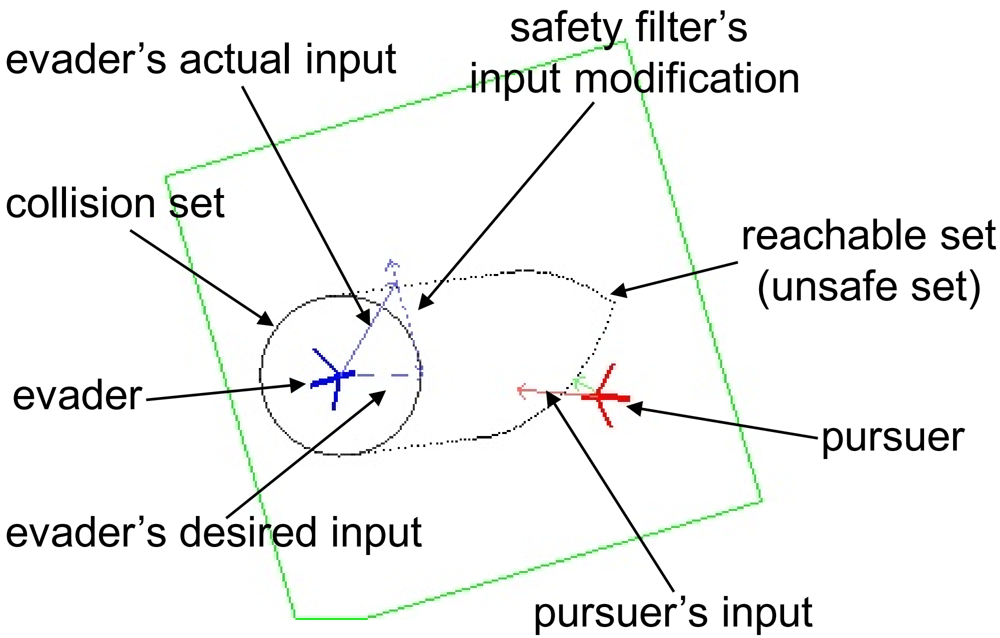
Collision Avoidance Filter 1

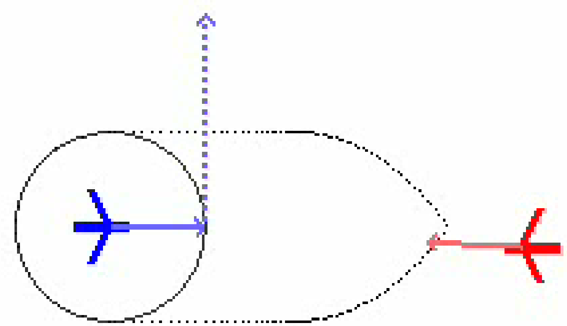
- Simple video game demonstration
  - Pursuer: turn to head toward evader
  - Evader: turn to head east
- No filtering for safety



Collision Avoidance Filter 2

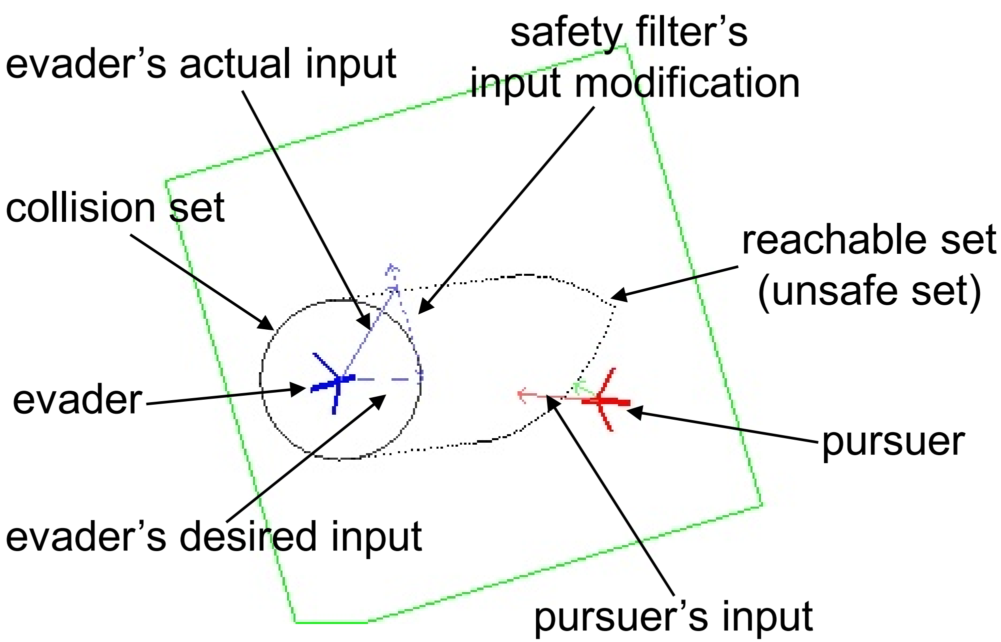
- Simple video game demonstration
  - Pursuer: turn to head toward evader
  - Evader: turn to head east
- Evader's input is filtered to guarantee that pursuer does not enter the reachable set

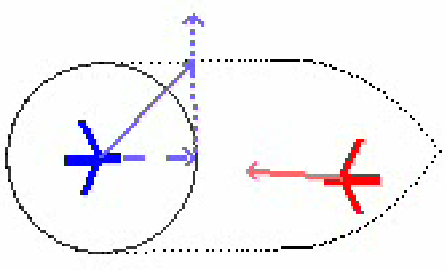




Collision Avoidance Filter 3

- Simple video game demonstration
  - Pursuer: turn to head toward evader
  - Evader: turn to head east
- Evader's input is filtered, but pursuer is already inside reachable set, so collision cannot be avoided

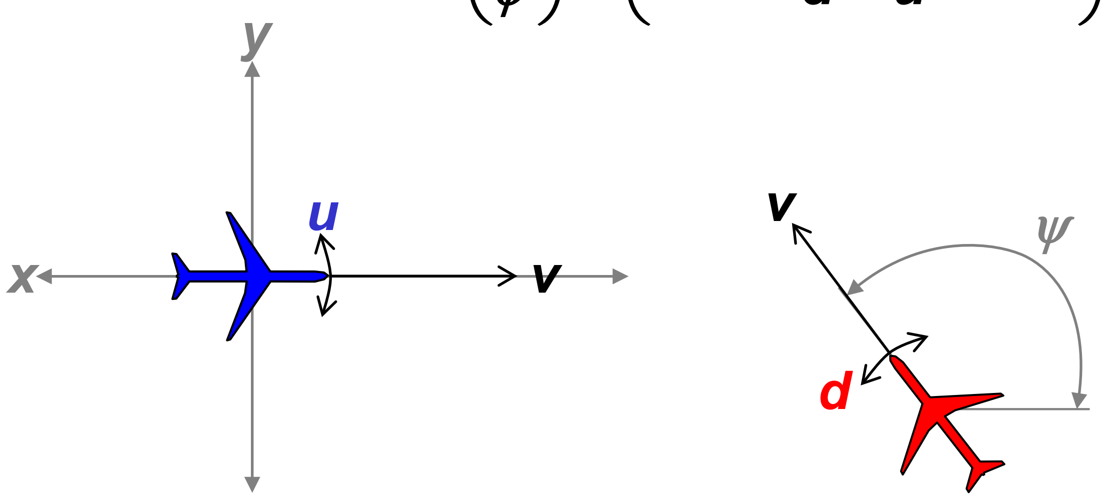




Collision Avoidance Computation

- Work in relative coordinates with evader fixed at origin
  - State variables are now relative planar location (***x***,***y***) and relative heading ψ

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ \psi \end{pmatrix} = \begin{pmatrix} -v + v\cos\psi + uy \\ v\sin\psi - ux \\ d - u \end{pmatrix}$$



evader aircraft (control) pursuer aircraft (disturbance)

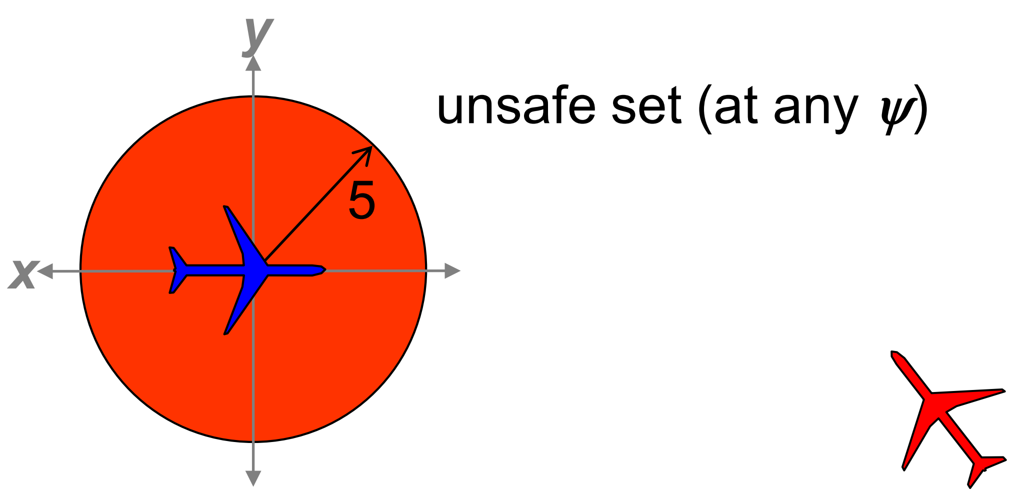
Hamilton-Jacobi Formulation

- Evader tries to maintain five mile separation

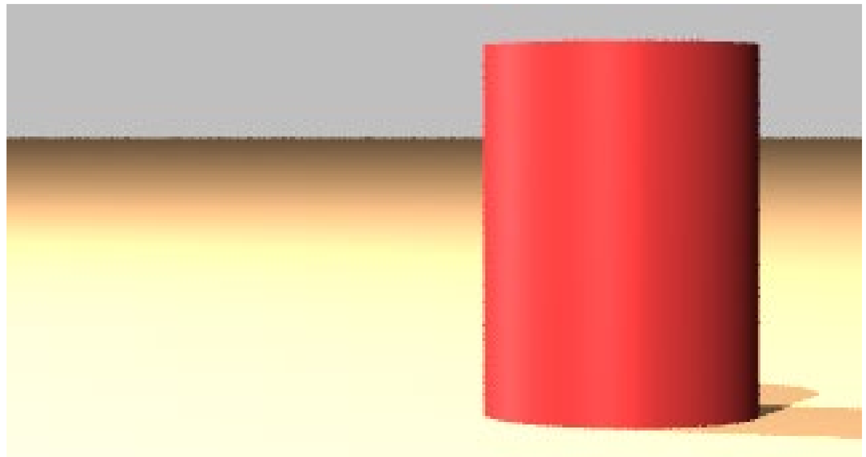
$$\text{let:}\, p = D_x J(x,t), \quad |u| \le 1, \quad |d| \le 1$$

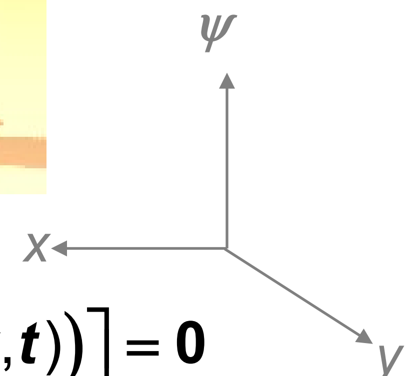
$$J(x, t=0) = \sqrt{x^2 + y^2} - 5$$

$$\begin{aligned} H(x,p) &= p_x v + p_x v\cos\psi + p_y v\sin\psi + u(p_x y - p_y x - p_\psi) - d(p_\psi) \\ &= p_x v + p_x v\cos\psi + p_y v\sin\psi + |\, p_x y - p_y x - p_\psi \,| - |\, p_\psi \,| \end{aligned}$$



Computing Reachable Set



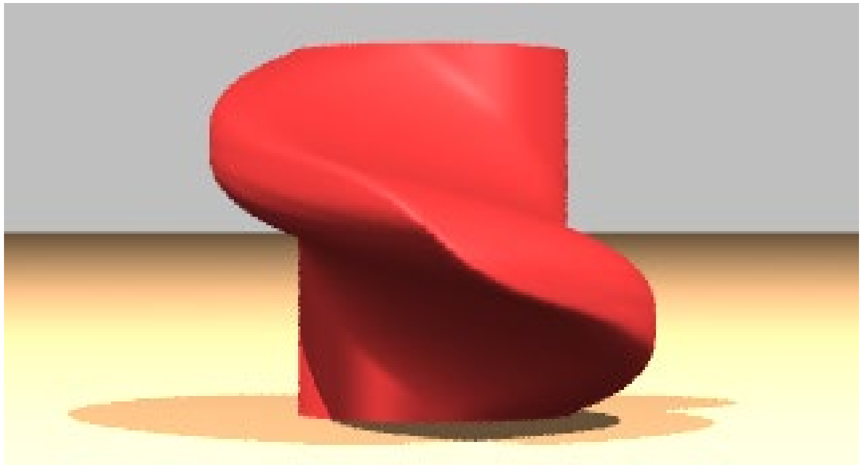


solve: $\boldsymbol{D_t J(x,t)} + \min\left[\boldsymbol{0}, \boldsymbol{H}\left(\boldsymbol{x}, \boldsymbol{D_x J(x,t)}\right)\right] = \boldsymbol{0}$

display: $\boldsymbol{G(t)} = \left\{\boldsymbol{x} \in \mathbb{R}^{n} \mid \boldsymbol{J(x,t)} \leq \boldsymbol{0}\right\}$

rendering software by Prof. Ron Fedkiw

Final Reachable Set



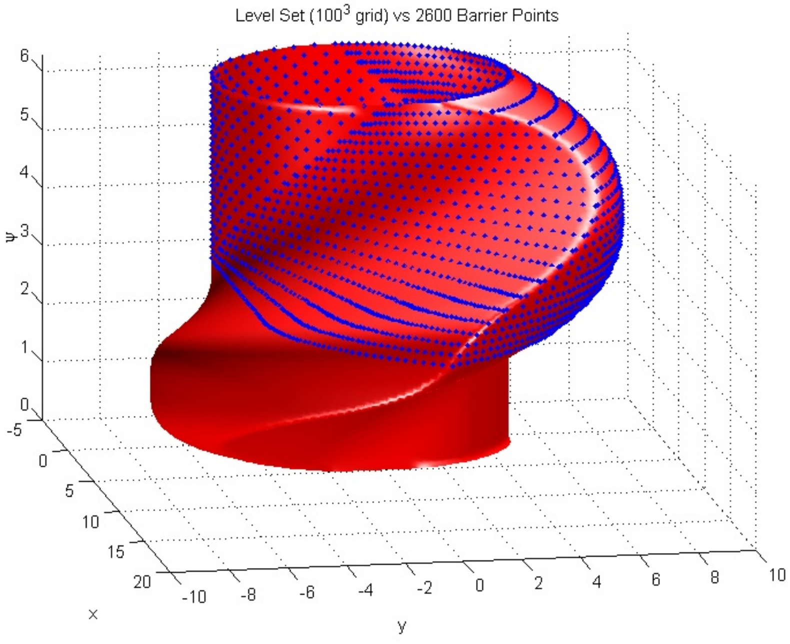
Converged $\boldsymbol{G}(\boldsymbol{t}) = \boldsymbol{G}$

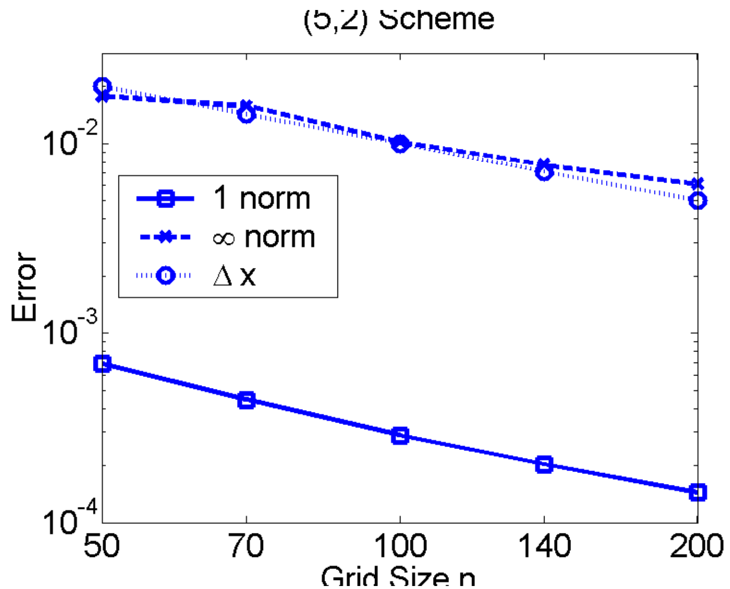
Simulation cost: twenty minutes, four megabytes

rendering software by Prof. Ron Fedkiw

Validating the Numerical Algorithm

- Analytic solution for reachable set can be found [Merz, 1972]
  - Applies only to identical pursuer and evader dynamics
  - Merz's solution placed pursuer at the origin, game is not symmetric
  - Analytic solution can be used to validate numerical solution
  - [Mitchell, 2001]

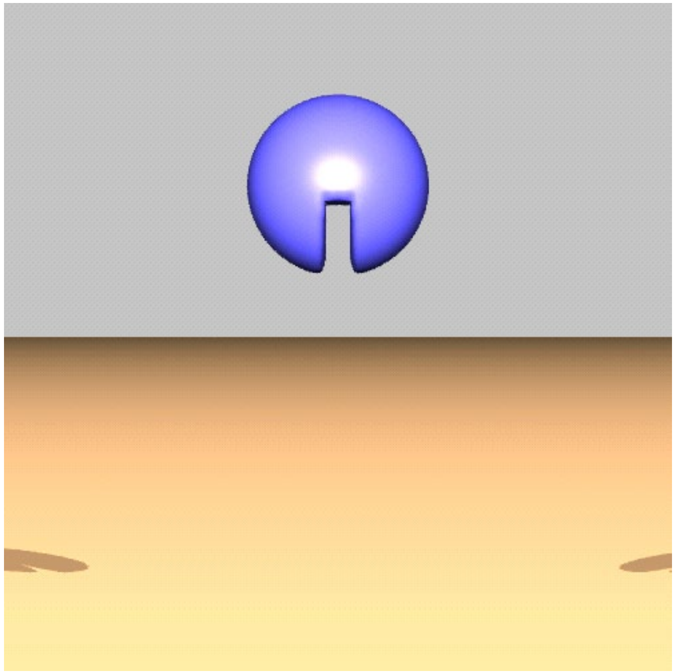




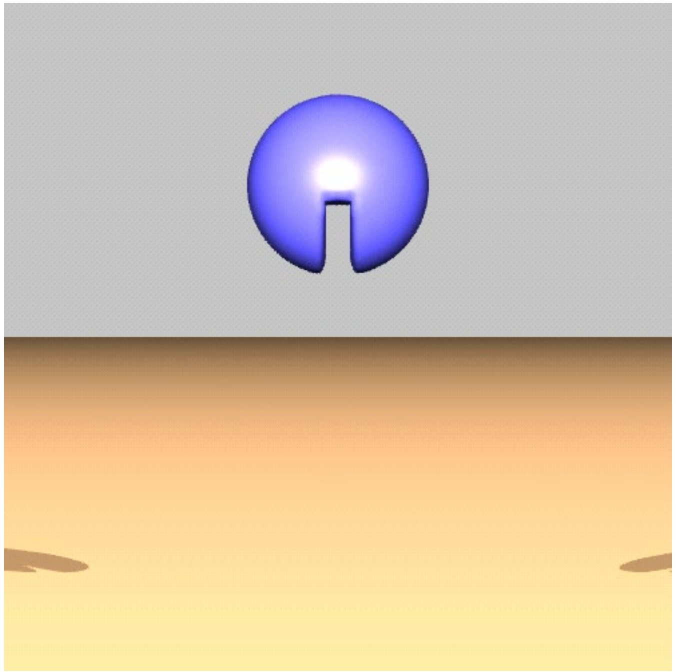
Innovation in Level Set Methods

- Level set methods are prone to area/volume loss
  - Particle level set method is an easy to program technique that significantly improves volume preservation
  - [Enright, Fedkiw, Ferziger & Mitchell, 2002]

Level Set Only



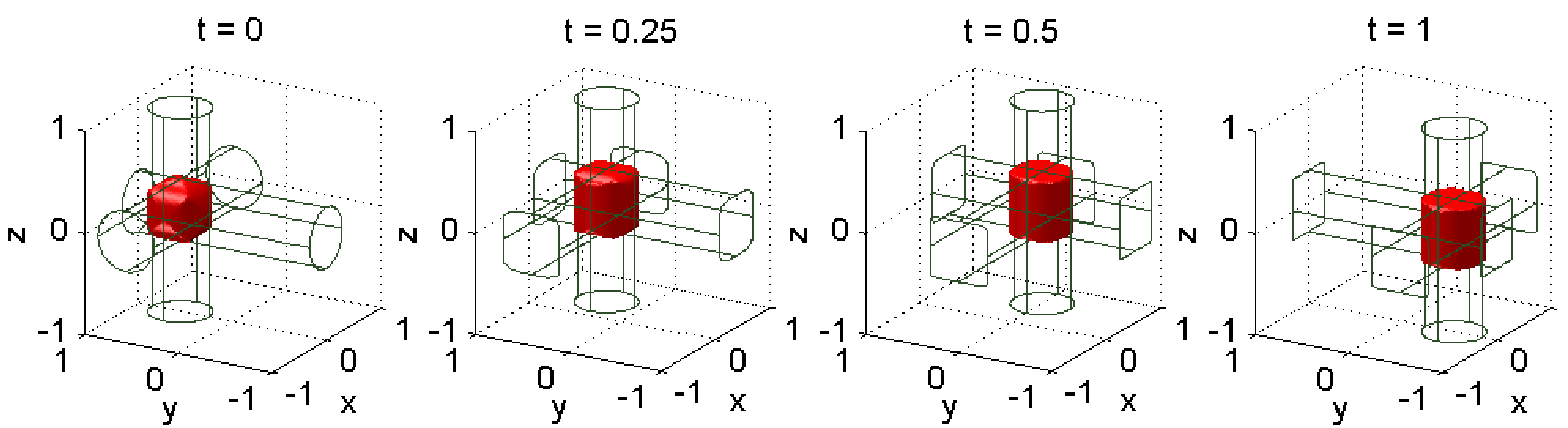
Particle Level Set



Rendering by Sou Cheng Choi

Projective Overapproximation

- Overapproximate reachable set of high dimensional system as the intersection of reachable sets for lower dimensional projections
  - [Mitchell & Tomlin, 2002]
  - Example: rotation of “sphere” about z-axis



Computing with Projections

- Forward and backward reachable sets for finite automata
  - Projecting into overlapping subsets of the variables, computing with BDDs [Govindaraju, Dill, Hu, Horowitz]
- Forward reachable sets for continuous systems
  - Projecting into 2D subspaces, representation by polygons [Greenstreet & Mitchell]
- Level set algorithms for geometric optics
  - Need multiple arrival time (viscosity solution gives first arrival time), so compute in higher dimensions and project down [Osher, Cheng, Kang, Shim & Tsai]

Hamilton-Jacobi in the Projection

- Consider ***x-z*** projection represented by level set $J_{xz}(x,z,t)$
  - Back projection into 3D yields a cylinder $J_{xz}(x,y,z,t)$
- Simple HJ PDE for this cylinder

$$D_t J_{xz}(x,y,z,t) + \sum_{i=1}^{3} p_i f_i(x,y,z) = 0 \qquad \text{where} \begin{cases} p_1 = D_x J_{xz}(x,y,z,t) \\ p_2 = D_y J_{xz}(x,y,z,t) \\ p_3 = D_z J_{xz}(x,y,z,t) \end{cases}$$

  - But for cylinder parallel to y-axis, $p_2 = 0$

$$D_t J_{xz}(x,y,z,t) + p_1 f_1(x,y,z) + p_3 f_3(x,y,z) = 0$$

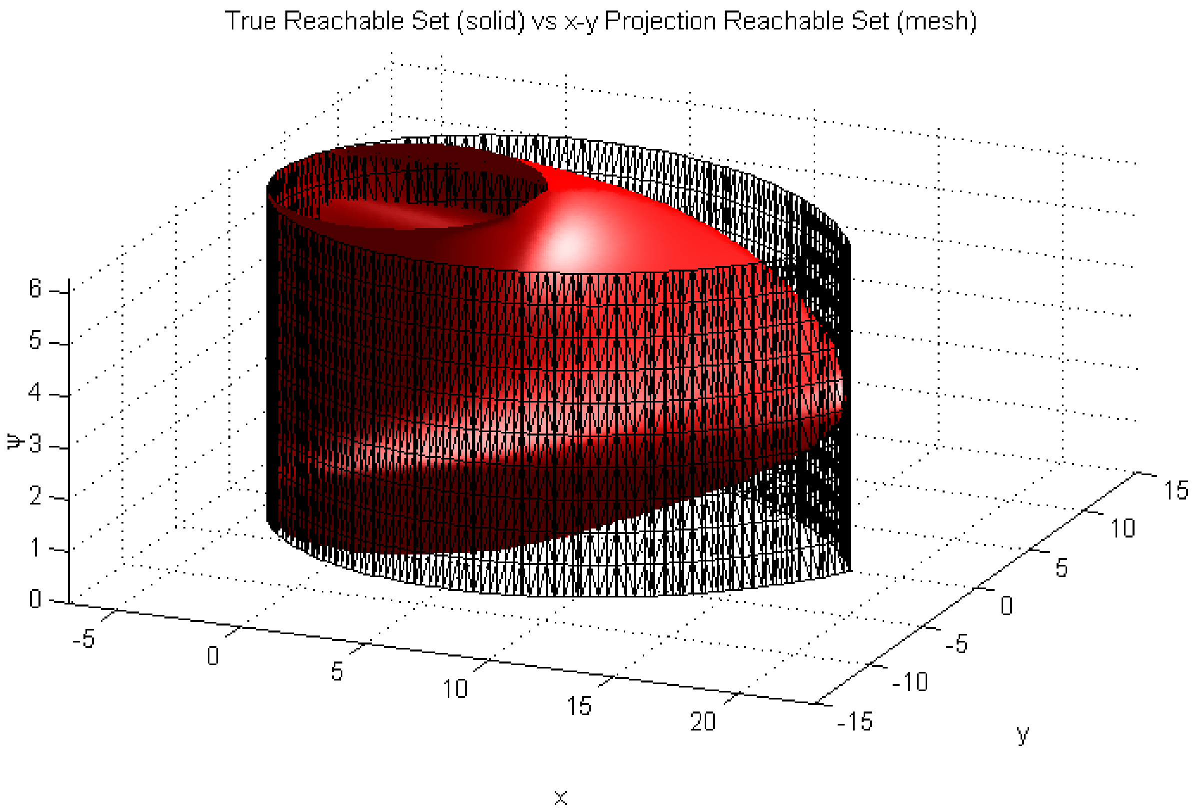
- What value to give free variable y in $f_i(x,y,z)$?
  - Treat it as a disturbance, bounded by the other projections

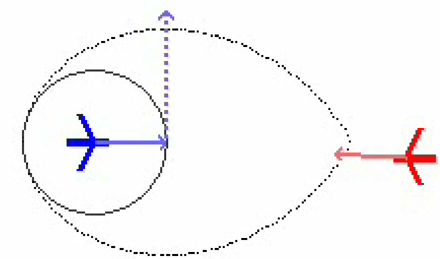
$$D_t J_{xz}(x,y,z,t) + \min_y \left[p_1 f_1(x,y,z) + p_3 f_3(x,y,z) \right] = 0$$

- Hamiltonian no longer depends on y, so computation can be done entirely in ***x-z*** space on $J_{xz}(x,z,t)$

Projective Collision Avoidance

- Work strictly in relative ***x-y*** plane
  - Treat relative heading $\psi \in [\,0, 2\pi\,]$ as a disturbance input
  - Compute time: 40 seconds in 2D vs 20 minutes in 3D
  - Compare overapproximative prism (mesh) to true set (solid)



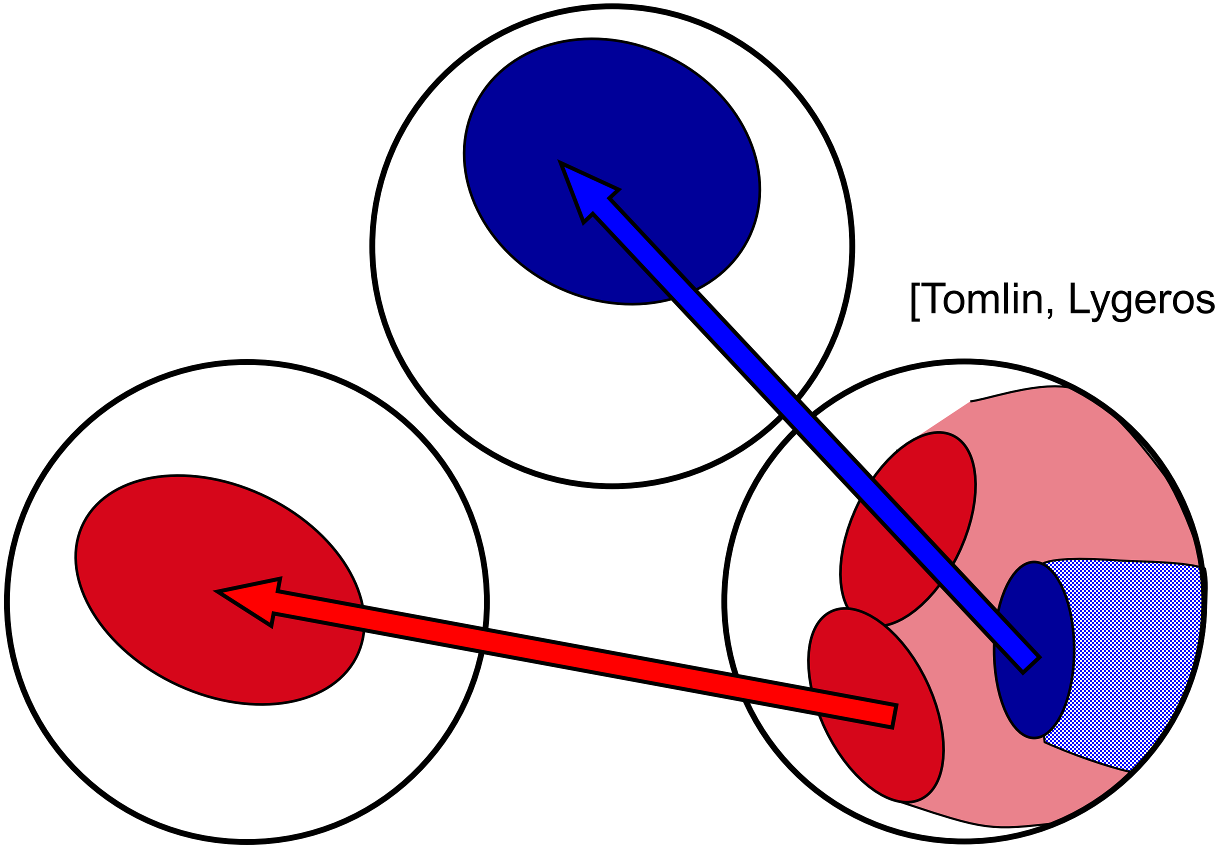


Outline

- The discrete, the continuous and the somewhere in between
  – Hybrid systems
- What are reachable sets?
  – Treating unknown inputs / parameters
- Computing reachable sets for continuous systems
  – A modified Hamilton-Jacobi-Isaacs equation
  – Application: synthesizing a safe control policy
  – Projective overapproximation of reachable sets
- Computing reachable sets for hybrid systems
  – The reach-avoid operator
  – Application: discrete abstraction
  – Application: an aircraft autolander
- Summary

Computing Hybrid Reachable Sets

- Compute continuous reachable set in each mode separately
  - Uncontrollable switches may introduce unsafe sets
  - Controllable switches may introduce safe sets
  - Forced switches introduce boundary conditions



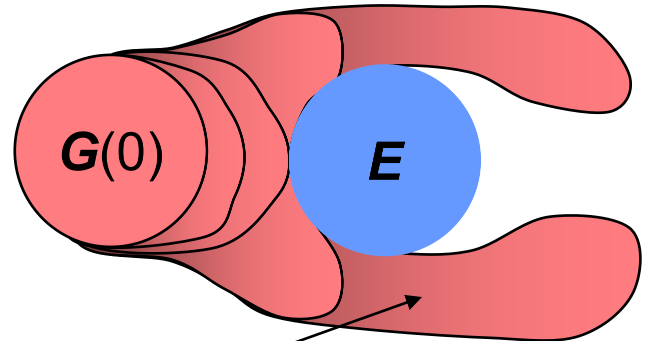
[Tomlin, Lygeros & Sastry, 2000]

Reach-Avoid Operator

- Compute set of states which reaches ***G***(0) without entering ***E***

$$\boldsymbol{G}(\boldsymbol{t}) = \left\{ \boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{J_G}(\boldsymbol{x},\boldsymbol{t}) \le \boldsymbol{0} \right\}$$

$$\boldsymbol{E} = \left\{ \boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{J_E}(\boldsymbol{x}) \le \boldsymbol{0} \right\}$$



Reach-Avoid Set ***G***(***t***)

- Formulated as a constrained Hamilton-Jacobi equation or variational inequality
  - [Mitchell & Tomlin, 2000]

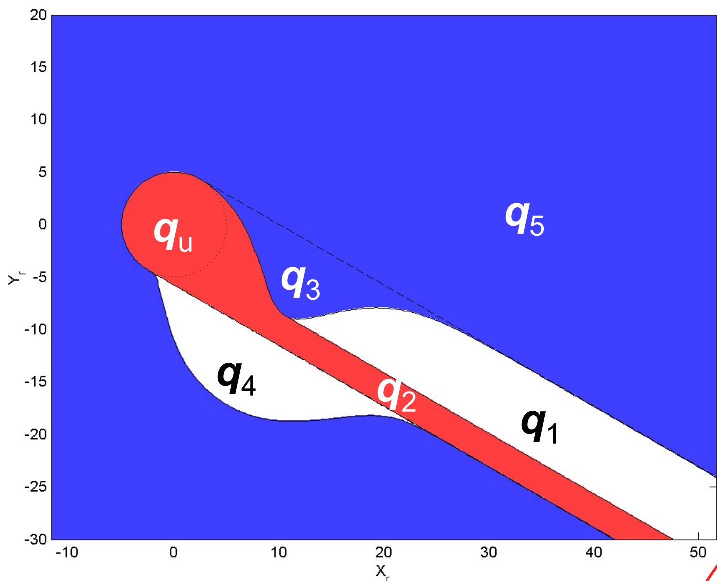
$$\boldsymbol{D_t J_G}(\boldsymbol{x},\boldsymbol{t}) + \min\left[\boldsymbol{0}, \boldsymbol{H}\left(\boldsymbol{x}, \boldsymbol{D_x J_G}(\boldsymbol{x},\boldsymbol{t})\right)\right] = \boldsymbol{0}$$

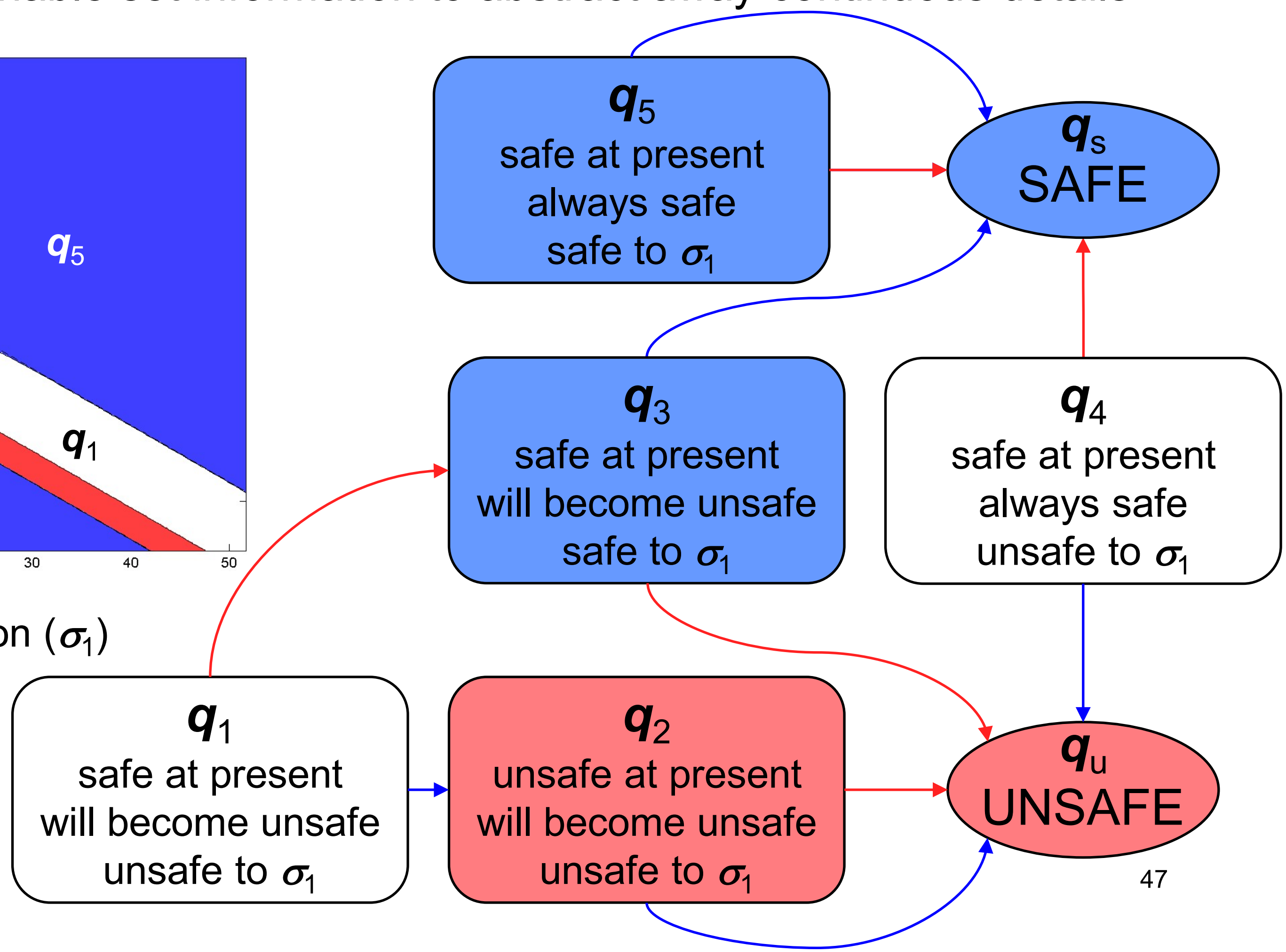
$$\text{subject to: } \boldsymbol{J_G}(\boldsymbol{x},\boldsymbol{t}) \ge \boldsymbol{J_E}(\boldsymbol{x})$$

- Level set can represent often odd shape of reach-avoid sets

Application: Discrete Abstractions

- It can be easier to analyze discrete automata than hybrid automata or continuous systems
  - Use reachable set information to abstract away continuous details





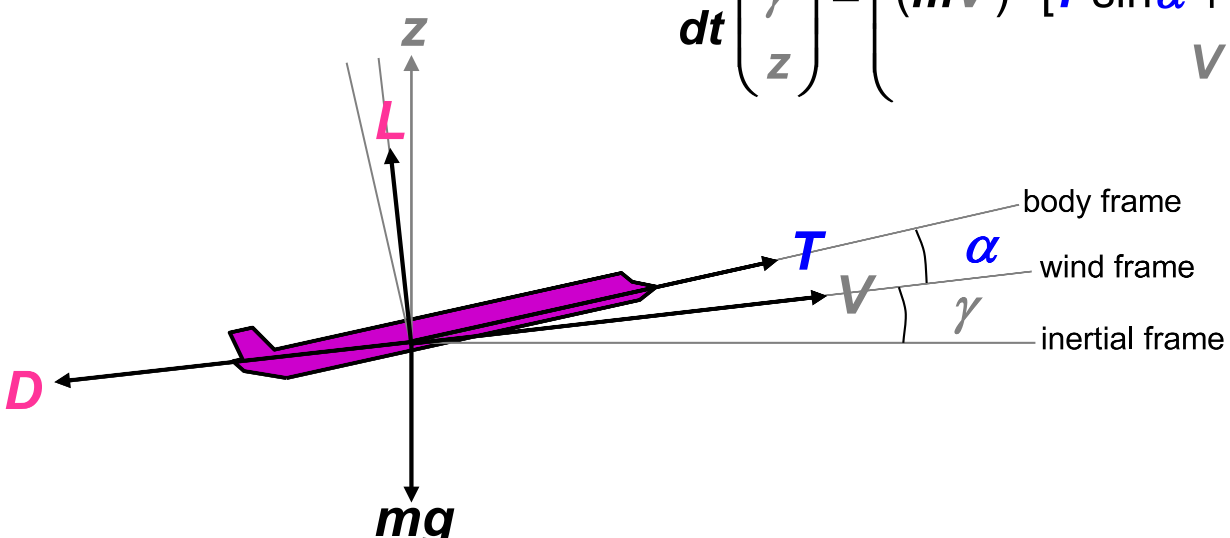
controlled transition (σ_1)

forced transition

Application: Aircraft Autolander

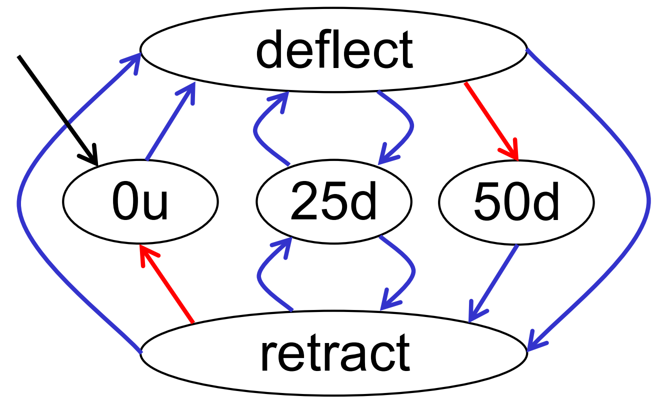
- Airplane must stay within safe flight envelope during landing
  - Bounds on velocity (*V*), flight path angle (γ), height (*z*)
  - Control over engine thrust (*T*), angle of attack (α), flap settings
  - Model flap settings as discrete modes of hybrid automata
  - Terms in continuous dynamics may depend on flap setting
  - [Mitchell, Bayen & Tomlin, 2001]

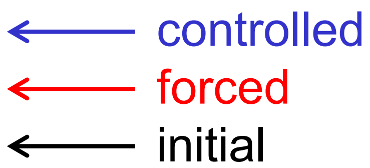
$$\frac{d}{dt}\begin{pmatrix} V \\ \gamma \\ z \end{pmatrix} = \begin{pmatrix} m^{-1}[T\cos\alpha - D(\alpha,V) - mg\sin\gamma] \\ (mV)^{-1}[T\sin\alpha + L(\alpha,V) - mg\cos\gamma] \\ V\sin\gamma \end{pmatrix}$$



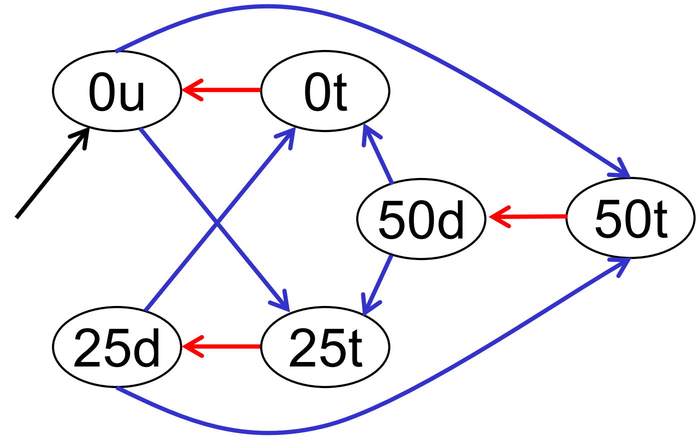
Landing Example: Discrete Model

- Flap dynamics version
  - Pilot can choose one of three flap deflections
  - Thirty seconds for zero to full deflection

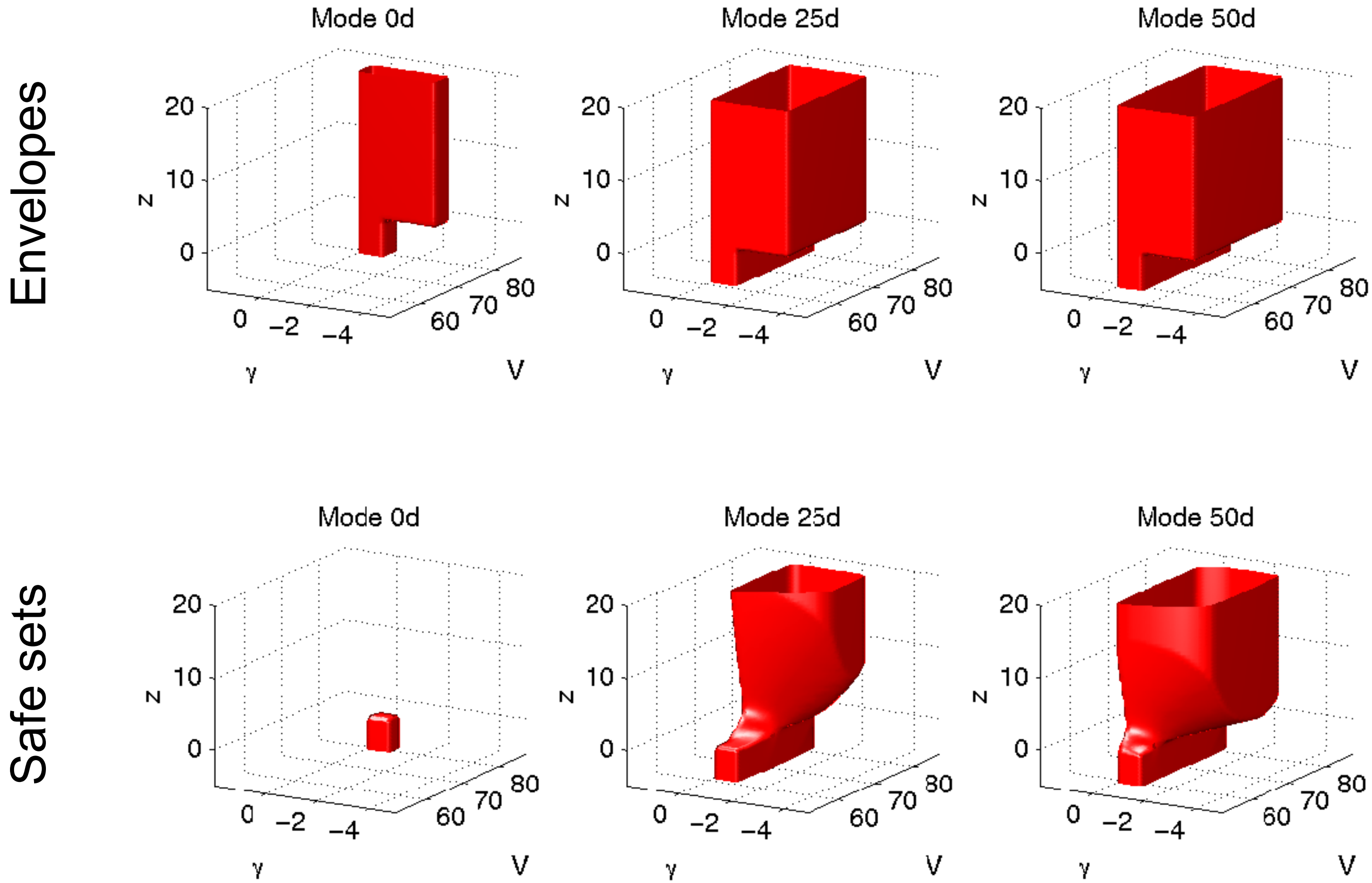




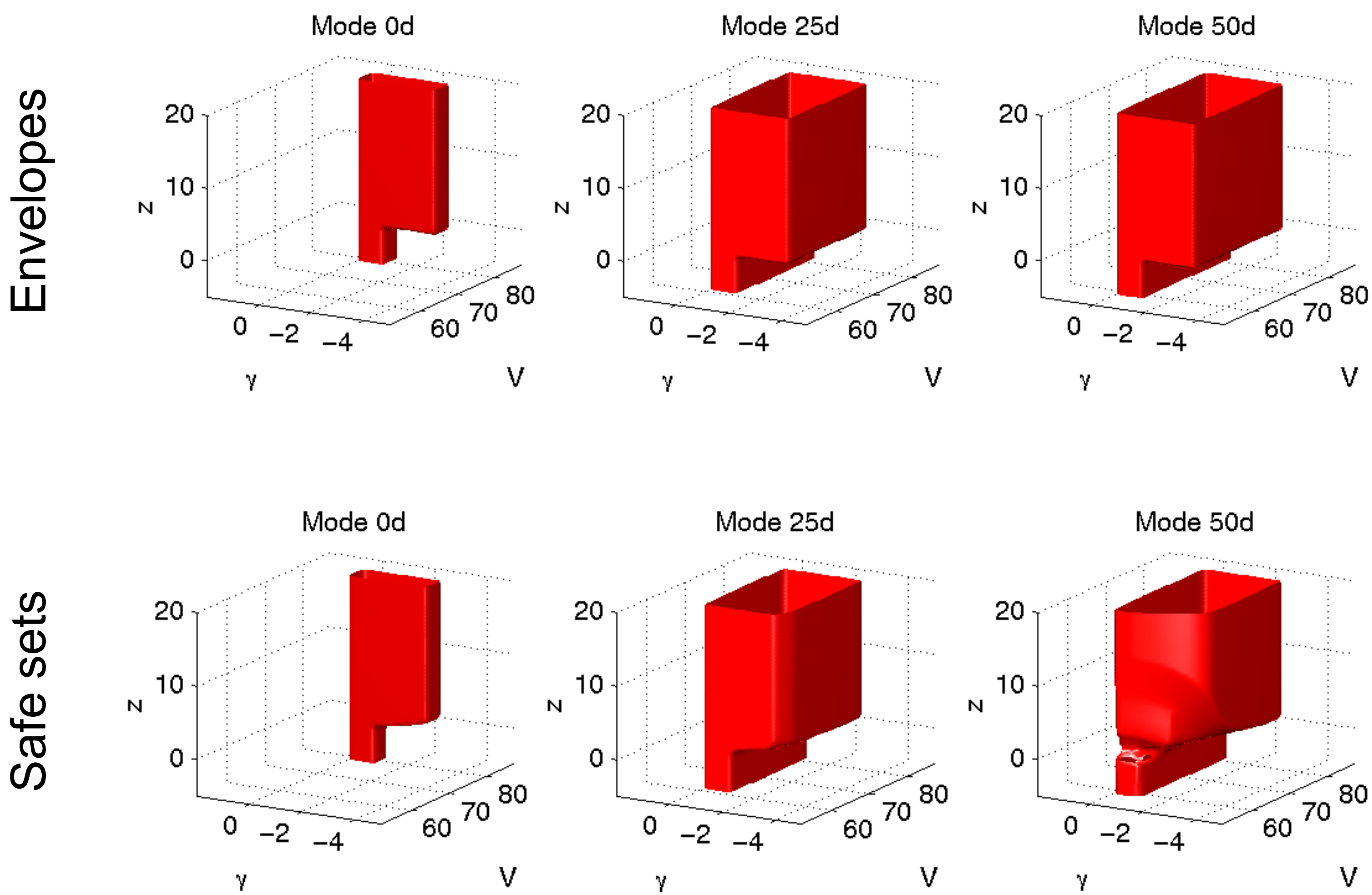
- Implemented version
  - Instant switches between fixed deflections
  - Additional timed modes to remove Zeno behavior



Landing Example: No Mode Switches

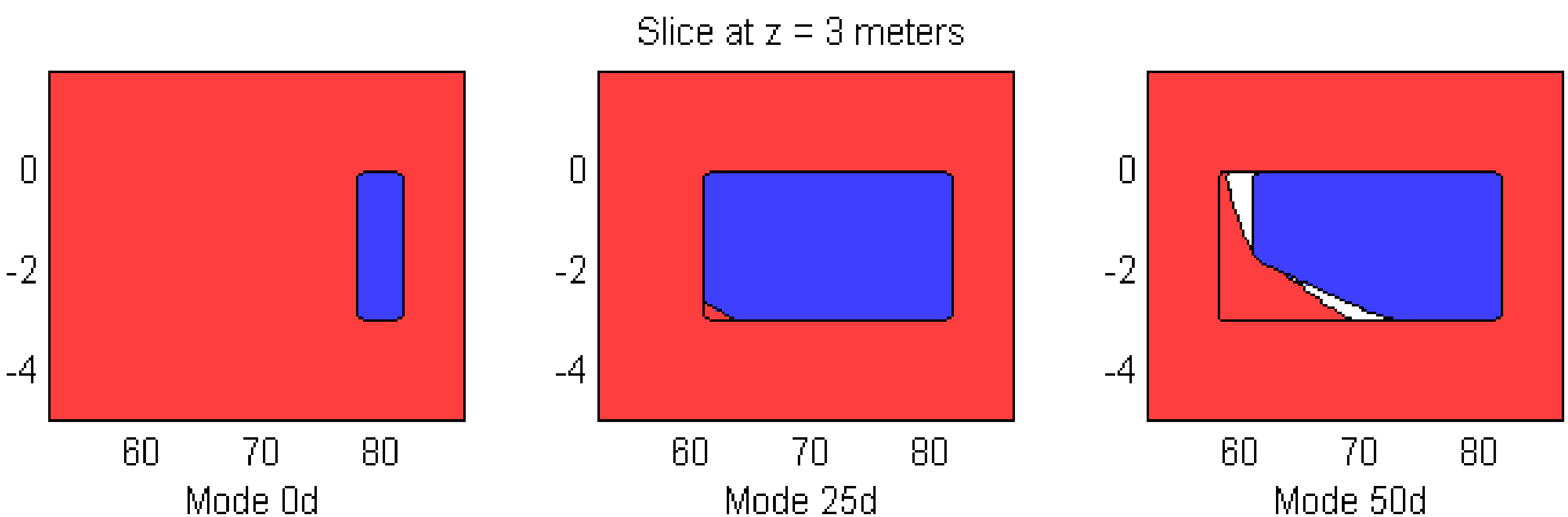


Landing Example: Mode Switches



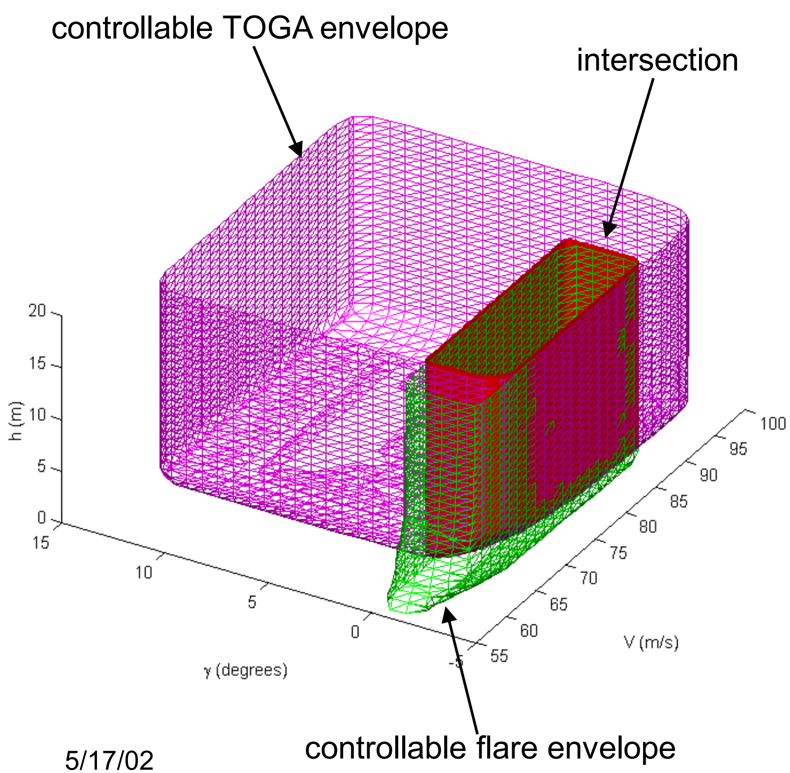
Landing Example: Synthesizing Control

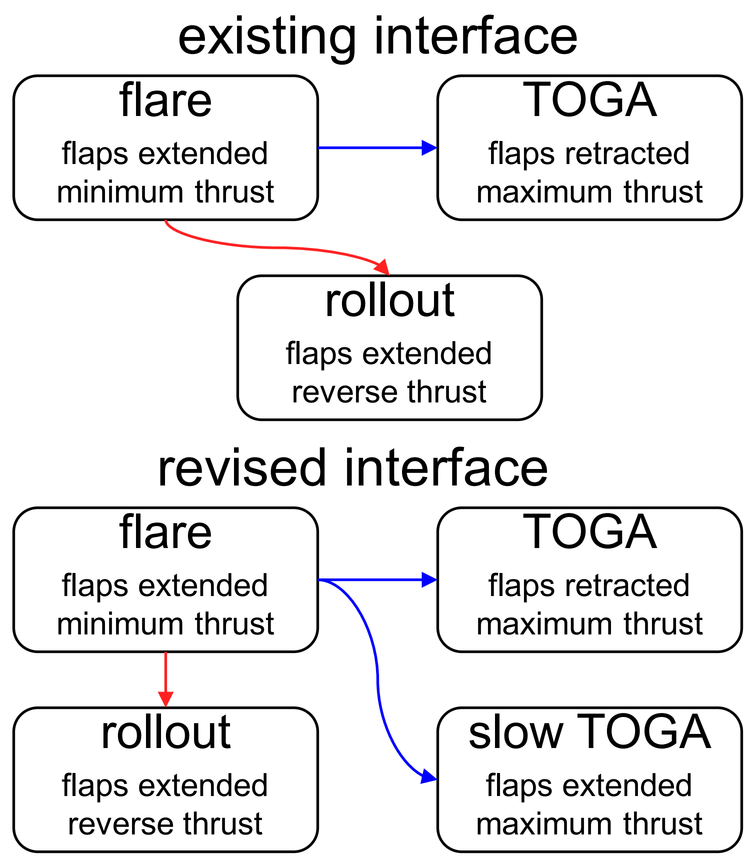
- For states at the boundary of the safe set, results of reach-avoid computation determine
  - What continuous inputs (if any) maintain safety
  - What discrete jumps (if any) are safe to perform
  - Level set values & gradients provide all relevant data



Abstraction Example: Cockpit Display

- Controllable flight envelopes for landing and Take Off / Go Around (TOGA) maneuvers may not be the same
- Pilot's cockpit display may not contain sufficient information to distinguish whether TOGA can be initiated
  - [Oishi, Mitchell, Bayen, Tomlin & Degani, CDC 2002]





Summary

- Hybrid systems can model systems with complex interactions between discrete and continuous components
- Level set methods can accurately compute reachable sets for nonlinear continuous and hybrid systems
  - Continuous and discrete inputs may affect system dynamics
  - Differential game formulation models unknown parameters robustly
  - Projective overapproximation to improve algorithm's scalability
- These reachable sets can be applied to
  - Verify safe behavior
  - Synthesize safe control policies
  - Create discrete abstractions
- Reachability problems motivate innovation in level set methods
  - Particle level set method used for fluid simulation and animation

Future Directions

- Comparing reachable sets computed by different algorithms
  - Create interval or sector bounded linear approximation for landing example and compute control invariant subset of envelope (by LMIs & SDPs) or backwards reachable set (by ellipsoidal methods)
- Fully local level set implementation (time and space)
- Toolkit for computing reachable sets
- Control synthesis and planning
  - Filtering controls for safety: soft walls for aircraft
- Projective overapproximation
  - Characterizing appropriate problems and choices of projections
- Probabilistic models

Acknowledgements

- Kaaren
- Current & Former Supervisors
  - Professors Claire Tomlin, Ronald Fedkiw & Mark Greenstreet
- Committee
  - Professors Stephen Boyd, David Dill & Antony Jameson
- Hybrid Systems Lab
  - Alexandre Bayen, Ronojoy Ghosh, Inseok Hwang, Gokhan Inalhan, Jung Soon Jang, Meeko Oishi, Dusan Stipanovic, Rodney Teo, Sherann Ellsworth
- Physically Based Modeling Group
  - Douglas Enright, Robert Bridson, Frederic Gibou, Eran Guendelman, Neil Molino, Igor Neverov, Duc Nguyen, Joseph Teran
- Scientific Computing & Computational Mathematics Program
- Family & Friends