Using Hybrid-System Verification Tools in the Design of Simplex-Based Systems

Scott D. Stoller

2014 Annual Safe and Secure Systems and Software Symposium (S5)

Simplex Architecture

- Simplex Architecture allows use of complex, advanced controllers in cyberphysical systems, thereby increasing performance, while maintaining reliability and certifiability.

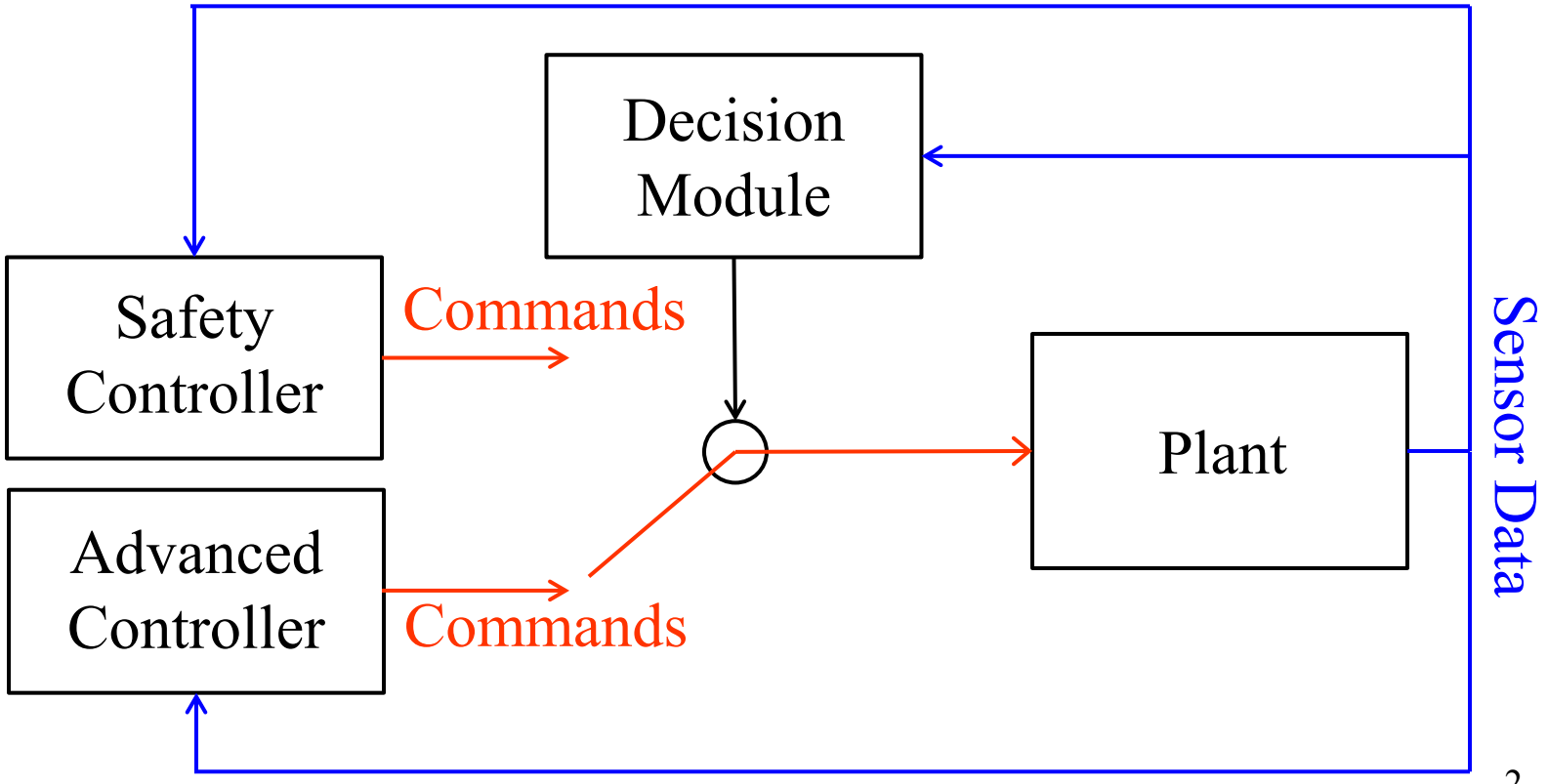


Figure adapted from [Bak et al., ICCPS 2011]

Simplex Architecture

- When system is in danger of violating a safety requirement, decision module switches control from advanced controller to safety controller. Decision module makes the switching decision periodically, with period τ.

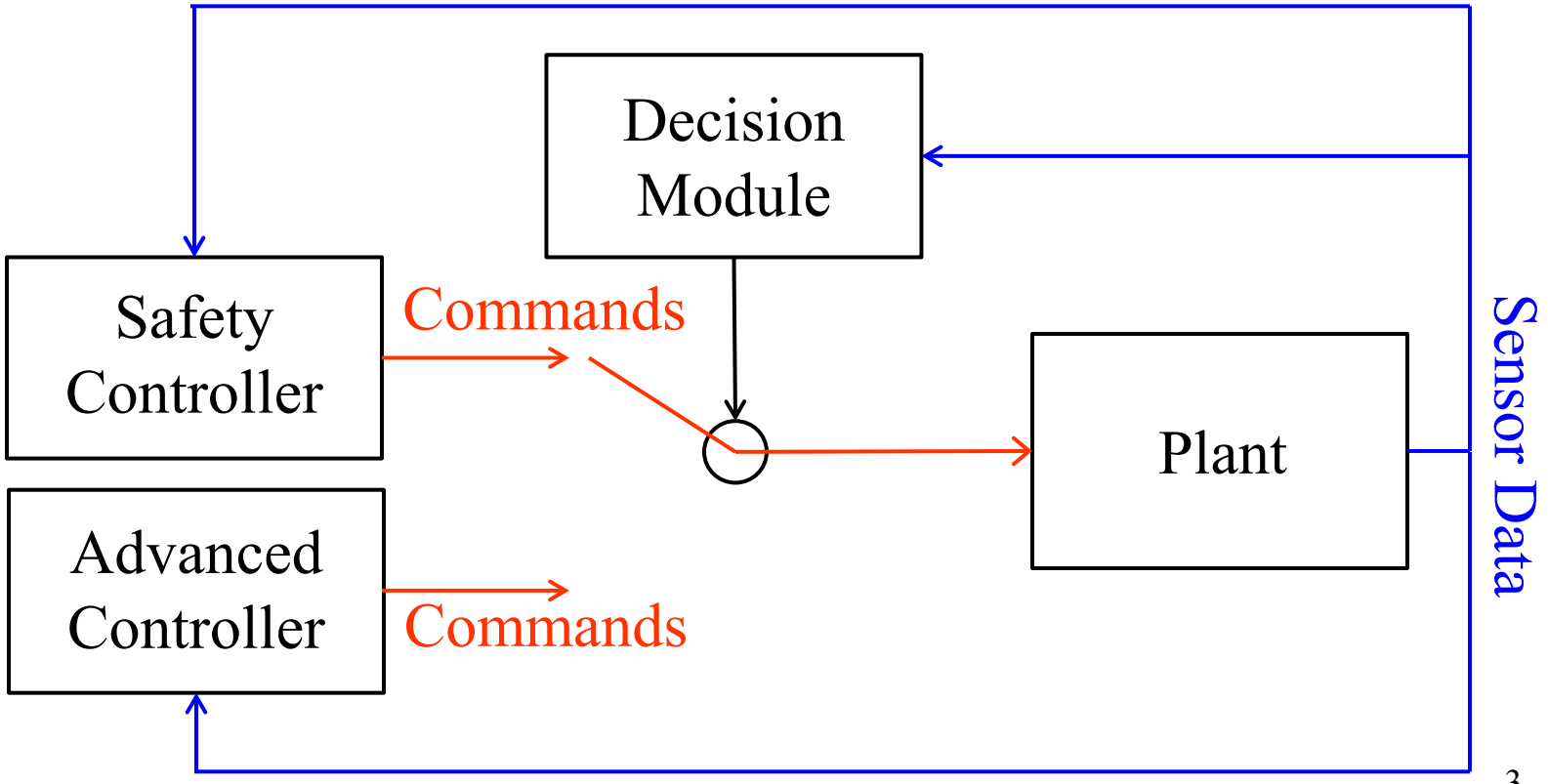
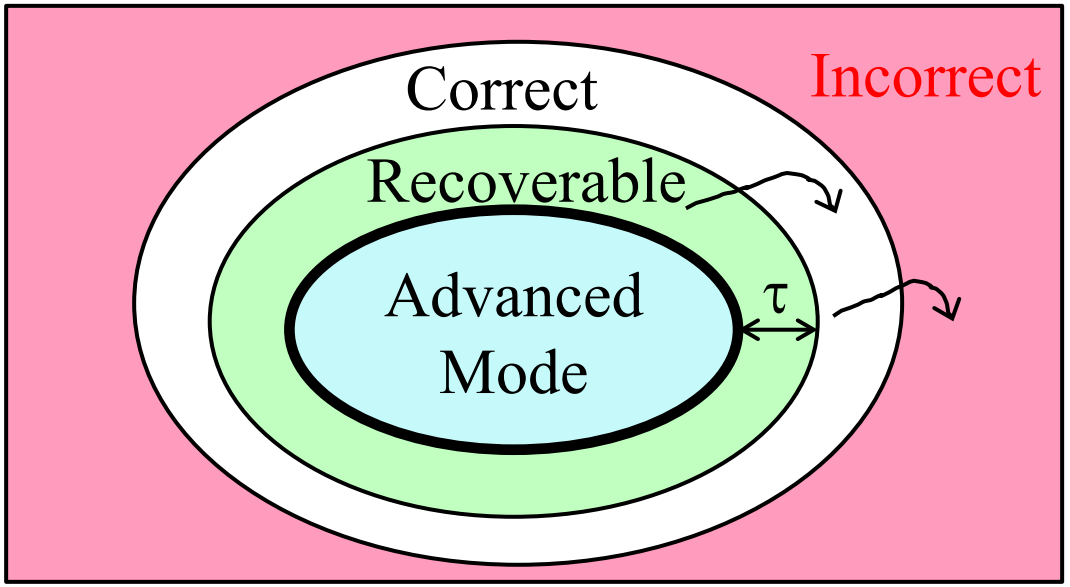


Figure adapted from [Bak et al., ICCPS 2011]

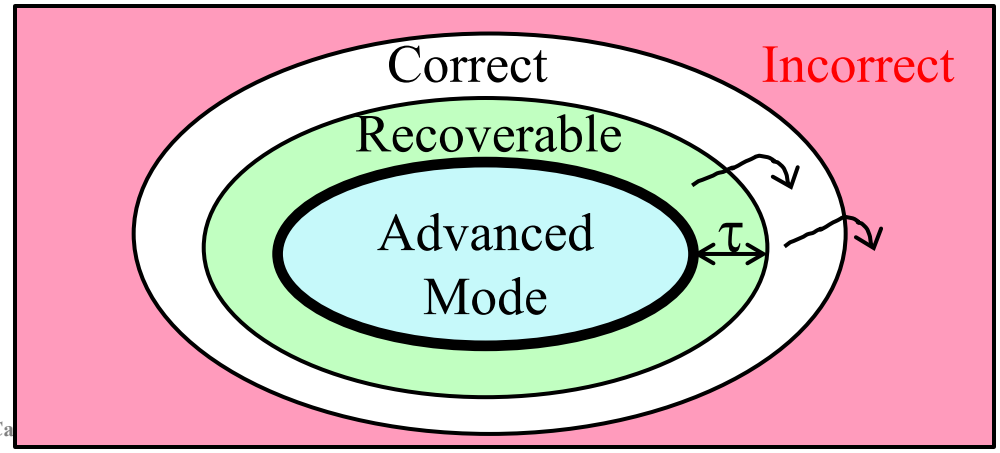
Switching Condition

- Recoverable states (green and blue): If safety controller takes over now, it can ensure that behavior remains correct.
- Continue using advanced controller (blue states) if state will remain recoverable until next decision point, i.e., for time τ.
- Switching condition: if state is outside blue region, switch to safety controller.
- Behavior is correct even if advanced controller has bugs.



How to Compute the Switching Condition?

- Computing the switching condition is a central problem in the design of Simplex-based systems. Bak et al. [2010, 2011] give methods that work for non-linear systems.
- Start with incorrect states, find all states backward-reachable from them (i.e., with time running backward) by system with safety controller. This gives unrecoverable states (white).
- Start with unrecoverable states, find all states that can reach them in time τ by system with advanced controller. These states (green) might become unrecoverable before next decision point. The remaining states are the blue ones.



Computing Reachability for Hybrid Systems

- Some verification tools can compute conservative approximation (i.e., over-approximation) of reachable states for non-deterministic non-linear hybrid automata.
- Hybrid automaton: finite state machine to specify discrete behavior, plus differential equations/inequalities and invariants to specify continuous behavior.
- Non-deterministic: derivative is not uniquely determined.
  - Example: $0 \leq \dot{x} \leq 1$
- Non-linear: derivatives involve polynomials, trigonometry
  - Example: $\dot{V} = \frac{T - C_D V^2}{\mathrm{M}} - \mathrm{g}\ \sin\gamma$
- Approach: Model plant and controllers as hybrid automata, then apply these tools to compute the switching condition.

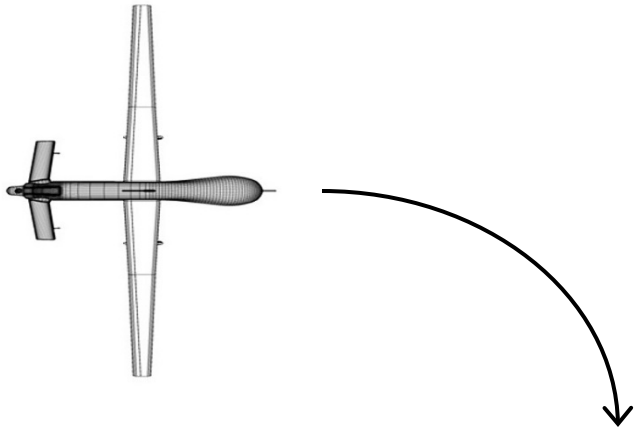
Challenges

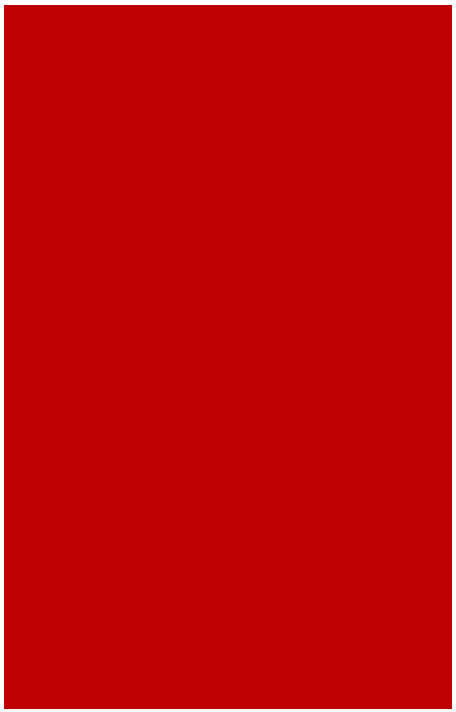
- Scalability: dimensionality, time horizon, …
  - All of the tools rely on discretization of the state space.
    - Divide the state space into regions (typically boxes).
    - Determine whether each region is reachable from given initial states.
  - Number of regions grows rapidly with high dimensionality
- Accuracy of conservative approximation
- Tuning of the many parameters of the tool
- Limitations of modeling language: no data structures, …

- Evaluate the feasibility of this approach using a benchmark.

Benchmark

- Compute switching condition that ensures a UAV avoids keep-out zones.
- Start simple:
  - One keep-out zone, directly ahead of the UAV.
  - Avoid it by turning at maximum bank angle.
  - No other obstacles.





Benchmark: UAV Model

- 9 dimensions
  - Thrust T, Lift L, Bank Angle ϕ These are control inputs.
  - Velocity V, Flight Path Angle γ, Heading Angle ψ
  - Position Coordinates X, Y, H
- "Standard" equations of motion.

Sample Equations:

$$\dot{V} = \frac{T - C_D V^2}{M} - \mathrm{g}\ \sin\gamma$$

$$\dot{\gamma} = \frac{\mathrm{L}\cos\phi}{MV} - g\frac{\cos\gamma}{V}$$

$$\dot{\psi} = \frac{L\ \sin\phi}{M\ V\ \cos\gamma}$$

Benchmark: Controller Models

- Model of Advanced Controller
  - Uncertified, so make no assumptions about it.
  - Completely non-deterministic model: control inputs may change at maximum possible rate, determined by physical limits of UAV or limits set by a safety system
- Model of Safety Controller
  - Turn at maximum bank angle to avoid keep-out zone ahead.
  - Feedback loops for thrust and lift to maintain level flight.
  - Example: $\dot{T} = p_T(k_1 x_T + k_2 \dot{x_T} - T) \quad \dot{x_T} = M(V_{cmd} - V)$
  - Each feedback loop has a state variable, increasing total dimensions to 11. This is a lot! Splitting a box in half along each dimension to increase accuracy creates 2048 boxes. Might need to do this recursively and for many boxes.

Verification Tools: HSolver

- Developed by Stefan Ratschan *et al.*
- State space is divided into boxes of varying size.
- Reachability Algorithm:
  - Compute coarse initial approximation of reachable states using large boxes, using constraint solver to determine whether there is a transition between two boxes.
    - Constraint has the form ($\exists\ s_1 \in \text{box}_1, s_2 \in \text{box}_2$ s.t. …).
  - Repeatedly increase accuracy by splitting boxes in selected areas of state space. This is called abstraction refinement.
  - Terminate when specified accuracy is reached.
- HSolver does not use any notion of time step.
- Use of adaptive accuracy in different regions of state space, and sound rounding, are attractive.

Verification Tools: HyCreate2

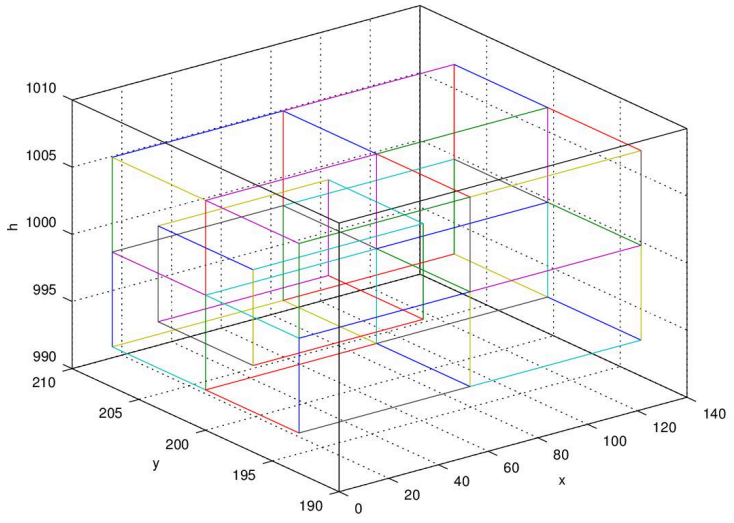
- Developed by Stanley Bak
- State space is divided into boxes using a fixed-size grid.
- User specifies grid size for each dimension, width of neighborhood (determines time step), …
- Reachability Algorithm:
  - Start with boxes containing initial states.
  - At each time step, compute newly reachable boxes by propagating faces of the existing boxes forward in time, based on time step and extremal value of derivatives in neighborhood of face. User specifies possible extremal points for derivative (default: corners of box).
  - If boxes get large (because faces propagate quickly), split the boxes, based on the grid size.

Verification Tools: Flow*

- Developed by Xin Chen *et al.*
- Approximate solution for each timestep is represented as a Taylor Model, consisting of:
  - local variables $\vec{v} \in [-1,1]^d$ and $t \in [0,\tau]$
  - polynomial $p_x(t,\vec{v})$ for each state variable x.
  - remainder R_x for each state variable x
- Better accuracy than boxes for non-rectangular behaviors.
- User specifies degree of polynomials, time step, remainder estimate, …
- Reachability Algorithm:
  - Start with Taylor Model containing initial states.
  - At each time step, construct a Taylor Model representing states that become reachable in that time step.

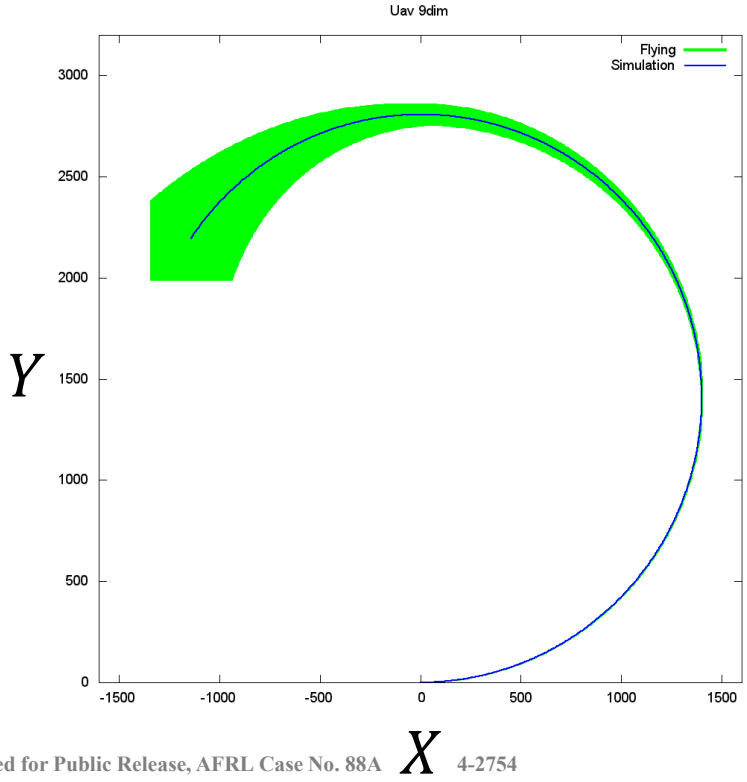
Experience with HSolver

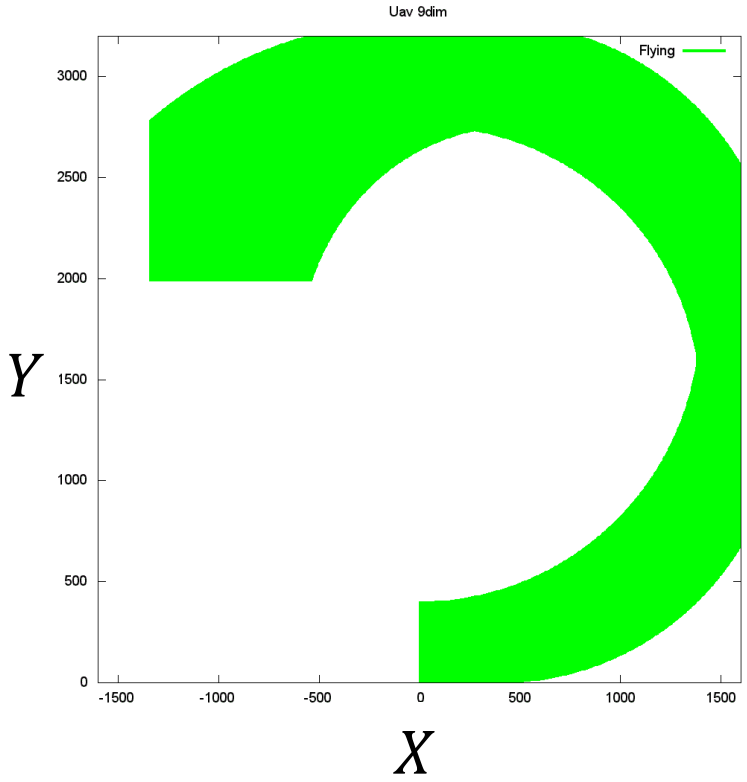
- Much slower than the other tools.
- Computation time for each step is high, presumably due to constraint solving and sound rounding.
- For the relatively complex differential equations in the benchmark, this cost outweighs the benefit of varying accuracy (box size) in different regions of the state space.



Experience with HyCreate2

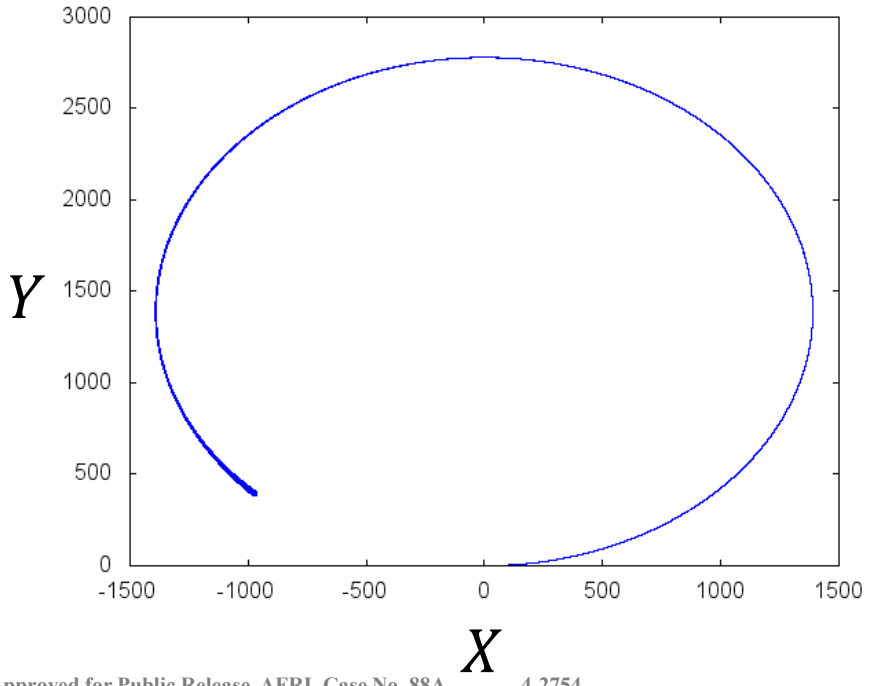
- Inaccuracy grows with each time step, because over-approximations of solution in each time step accumulate.
- Example: Behaviors with safety controller for 45 sec from one initial position (L) and 400' cube of initial positions (R). HyCreate2 has similar accuracy in both cases.

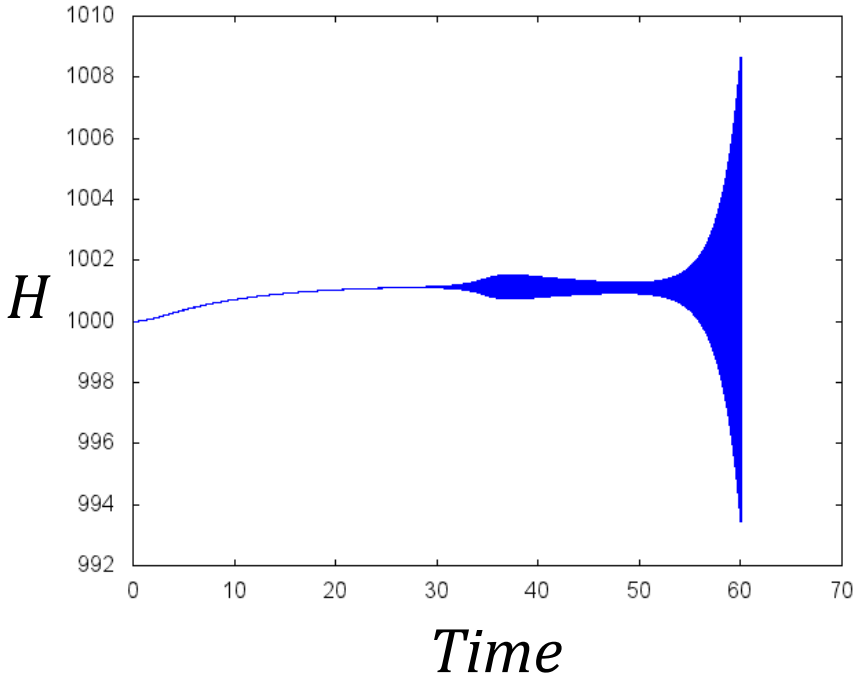




Experience with Flow*: One Initial Position

- Better accuracy than HyCreate2, because polynomials in Taylor Models allow more accurate representation of curved trajectories. Inaccuracy accumulates more slowly for some time but then degrades dramatically.
- Example: Behavior with safety controller for 60 sec from a single initial position. Good accuracy for about 53 sec.



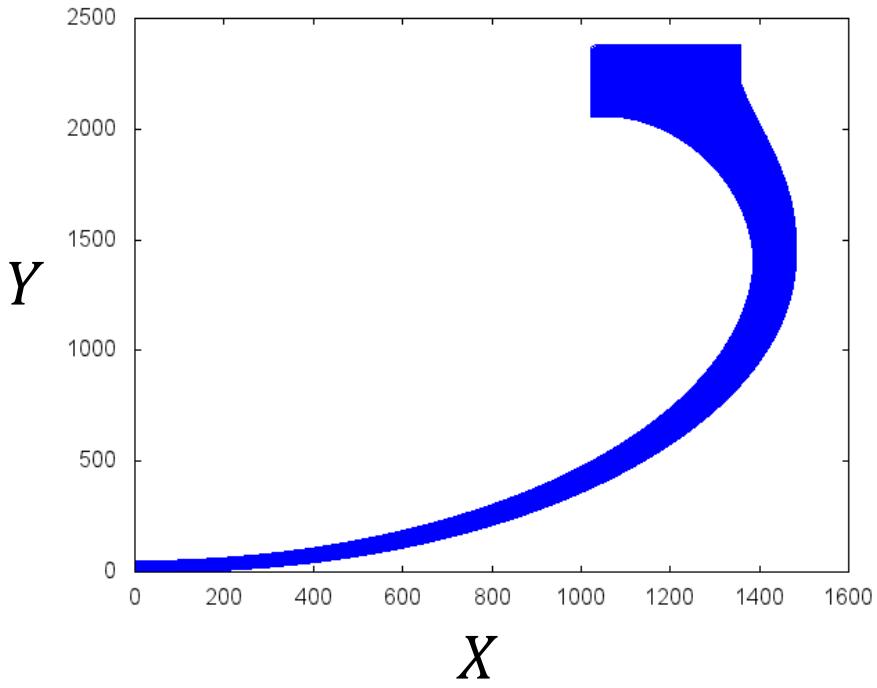


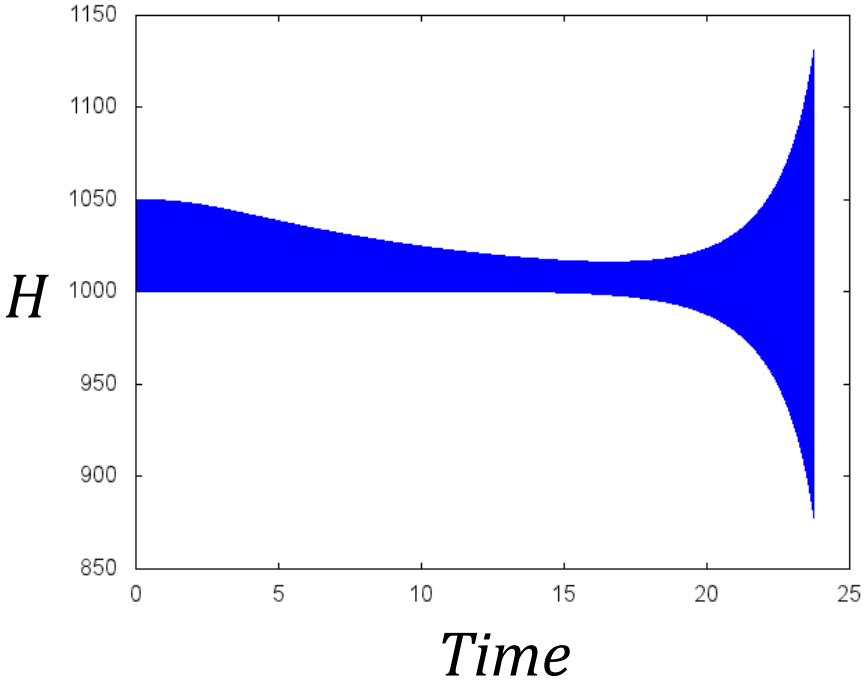
Experience with Flow*: Cube of Initial Positions

- For 400' cube of initial positions, accuracy degrades sharply after 12 sec of simulation time, and Flow* terminates the calculation (error estimate exceeds threshold) after 15 sec.
- Tried to tune algorithm parameters to improve results.
  - Decrease time step
  - Increase degree of polynomials
  - Decrease remainder estimate
  - Increase numerical precision (number of digits)
- These changes either provided little increase in accuracy, or increased computation time excessively.
- Limiting the initial positions to a smaller cube was the only effective strategy.
- In contrast, HyCreate2 had reasonable accuracy for 400’ cube.

Experience with Flow*: Cube of Initial Positions

- Example: Behavior with safety controller for 24 sec from a 50' cube of initial positions. Good accuracy for about 19 sec.
- This might be sufficient to compute switching condition for keep-out zone avoidance: the UAV has reversed direction.





Conclusions and Future Work

- HyCreate2 and Flow* can accurately compute reachability for model of UAV with simple safety controller or non-deterministic advanced controller, for limited but reasonable time horizons and regions of initial positions.
- If similar accuracy and performance are achieved when considering regions of initial values for all dimensions (position, thrust, lift, bank angle, velocity, …), then it should be feasible to use these tools to compute the switching condition for the keep-out zone avoidance benchmark.
- In summary, using such tools to compute switching conditions appears promising for short-term safety properties (e.g., keep-out zone avoidance, operating envelope safety), but less suitable for long-term safety properties (e.g., sufficient fuel to return to base).