Direct Verification of Linear Systems with over 10000 Dimensions

Stanley Bak and Parasara Sridhar Duggirala

Overview

- Description of Safety Verification Method
- Evaluation on Linear Benchmark Suite (9 benchmarks) taken from ARCH2016

Three-Question Quiz on Superposition

Q1: Given a 2-d linear ODE, $x' = Ax$, if an initial state **(1, 0)** goes to **(a, b)** after 10 seconds, where would **(2, 0)** go to after 10 seconds?

Three-Question Quiz on Superposition

Q1: Given a 2-d linear ODE, $x' = Ax$, if an initial state **(1, 0)** goes to **(a, b)** after 10 seconds, where would **(2, 0)** go to after 10 seconds?

A1: **(2a, 2b)**

Three-Question Quiz on Superposition

Q1: Given a 2-d linear ODE, x' = Ax, if an initial state **(1, 0)** goes to **(a, b)** after 10 seconds, where would **(2, 0)** go to after 10 seconds?

A1: **(2a, 2b)**

Q2: For the same system, if initial state **(0, 1)** goes to **(c, d)** after 10 seconds, where would **(2, 2)** go after 10 seconds?

Three-Question Quiz on Superposition

Q1: Given a 2-d linear ODE, x' = Ax, if an initial state **(1, 0)** goes to **(a, b)** after 10 seconds, where would **(2, 0)** go to after 10 seconds?

A1: **(2a, 2b)**

Q2: For the same system, if initial state **(0, 1)** goes to **(c, d)** after 10 seconds, where would **(2, 2)** go after 10 seconds?

A2: **(2a + 2c, 2b + 2d)**

Sets of Initial States

What if we want to know where a (linear) set of initial states goes to after 10 seconds?

Q3: If **(1, 0)** $\rightarrow$ **(a, b)**, where could **(x_0, 0)** go to, if $x_0 \in [3, 5]$?

Sets of Initial States

What if we want to know where a (linear) set of initial states goes to after 10 seconds?

Q3: If **(1, 0)** $\rightarrow$ **(a, b)**, where could **(x_0, 0)** go to, if $x_0 \in [3, 5]$?

A3: Anywhere between **(3a, 3b)** and **(5a, 5b)**.

Sets of Initial States

What if we want to know where a (linear) set of initial states goes to after 10 seconds?

Q3: If **(1, 0)** $\rightarrow$ **(a, b)**, where could **(x_0, 0)** go to, if $x_0 \in [3, 5]$?

A3: Anywhere between **(3a, 3b)** and **(5a, 5b)**.

Notice that all the conditions are linear. We can encode everything into a linear program (LP).

(LP Demo)

LP Formulation

- At each time t, we solve an LP with:
  - Variables at current time, $x(t)$
  - Variables at initial time, $x(0)$
  - Linear constraints on initial variables
  - (possibly) linear constraints defining unsafe states
  - Relationship between x and $x(0)$, $x(t) = \Phi(t) * x(0)$, where each column of $\Phi(t)$ is a simulation point

- But remember that the solution to a set of linear ODEs can also be given by:
  - $x(t) = e^{At} * x(0)$

- So $\Phi(t) = e^{At}$. Which computation method is better?

Overall Computation Steps

To check for safety at each time $t \in \{0, h, 2h, ..., t_{max}\}$:

1. Compute the basis matrix at time t
2. Solve an LP

We can compute the basis matrix by either:

- Running **N** simulations

  -or-

- Computing an **N**-dimensional matrix exponential (or, since, $e^{A2h} = e^{Ah} * e^{Ah}$, compute e^{Ah} once and then do **N**-dim matrix multiplication at each step)

Benchmarks

We made a tool, **Hylaa**, which uses this approach. We then evaluated the method on a Linear System Verification Benchmark Suite* presented at ARCH last year:

- Motor (11 dims)
- Building (50 dims)
- Partial Differential Equation (86 dims)
- Heat (202 dims)
- International Space Station (274 dims)
- Clamped Beam (350 dims)
- MNA1 (588 dims)
- FOM (1008 dims)
- MNA5 (**10923 dims**)

* "Large-scale linear systems from order-reduction", H. D. Tran, L. V. Nguyen, and T. T. Johnson, 3rd Applied Verification for Continuous and Hybrid Systems Workshop (ARCH 2016)

Results

- **Every model was successful analyzed!**

- The paper has a large table with all the results:

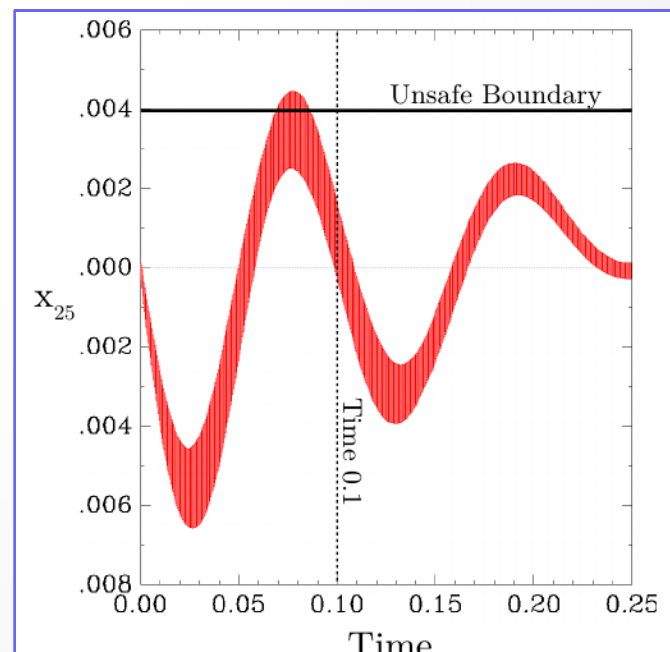
Table 1: Benchmark results. Stars (*) indicate each benchmark's original specification.

Model	Dims	Unsafe Error Condition	Method	Step Size	Runtime	Safe?	CE Error	CE Time
Motor*	11	$x_1 \in [0.35, 0.4] \wedge x_5 \in [0.45, 0.6]$	Simulation	0.1	0.5s	✓		
				0.01	0.8s	✓		
				0.001	2.6s	✓		
			Matrix Exp	0.1	0.4s	✓		
				0.01	0.7s	✓		
				0.001	3.9s	✓		
Motor	11	$x_1 \in [0.3, 0.4] \wedge x_5 \in [0.4, 0.6]$	Simulation	0.1	0.5s		$1.6 \cdot 10^{-11}$	0.1
				0.01	0.6s		$6.2 \cdot 10^{-13}$	0.04
				0.001	0.6s		$2.2 \cdot 10^{-12}$	0.037
			Matrix Exp	0.1	0.5s		$9.3 \cdot 10^{-13}$	0.1
				0.01	0.5s		$9.9 \cdot 10^{-13}$	0.04
				0.001	0.6s		$1.4 \cdot 10^{-12}$	0.037

Building (50 dims)

Building*	50	$x_{25} \geq 0.006$	Simulation	0.1	2.1s	✓		
				0.01	2.5s	✓		
				0.001	7.6s	✓		
			Matrix Exp	0.1	0.6s	✓		
				0.01	1.4s	✓		
				0.001	8.9s	✓		
Building	50	$x_{25} \geq 0.004$	Simulation	0.1	1.9s	✓ ?		
				0.01	2.0s		$1.2 \cdot 10^{-9}$	0.07
				0.001	2.8s		$2.2 \cdot 10^{-9}$	0.07
			Matrix Exp	0.1	0.5s	✓ ?		
				0.01	0.5s		$1.1 \cdot 10^{-9}$	0.07
				0.001	0.4s		$1.1 \cdot 10^{-9}$	0.07

- For both **Simulation** and **MatrixExp**, using a time-step of 0.1 seems to make the system safe



MNA1 (588 dims)

MNA1*	588	$x_1 \geq 0.5$	Simulation	0.1	9m49s	✓		
				0.01	10m22s	✓		
				0.001	20m41s	✓		
			Matrix Exp	0.1	22.6s	✓		
				0.01	3m0s	✓		
				0.001	30m8s	✓		
MNA1	588	$x_1 \geq 0.2$	Simulation	0.1	9m45s		$1.2 \cdot 10^{-10}$	16.6
				0.01	10m19s		$1.2 \cdot 10^{-10}$	16.56
				0.001	18m20s		$1.2 \cdot 10^{-10}$	16.554
			Matrix Exp	0.1	20.2s		$7.4 \cdot 10^{-11}$	16.6
				0.01	2m31s		$7.4 \cdot 10^{-11}$	16.56
				0.001	24m58s		$7.0 \cdot 10^{-11}$	16.554

- **MatrixExp** method runtime is almost linear with the number of steps (in the safe case)

FOM (1008 dims)

FOM*	1008	$y_1 \geq 45$	Simulation	0.1	7m3s	✓		
				0.01	9m39s	✓		
				0.001	45m15s	✓		
			Matrix Exp	0.1	30.3s	✓		
				0.01	4m51s	✓		
				0.001	48m49s	✓		
FOM	1008	$y_1 \geq 7$	Simulation	0.1	7m4s		$3.3 \cdot 10^{-10}$	0.2
				0.01	4m27s		$1.7 \cdot 10^{-10}$	0.07
				0.001	2m39s		$1.7 \cdot 10^{-10}$	0.069
			Matrix Exp	0.1	2.5s		$7.5 \cdot 10^{-12}$	0.2
				0.01	3.3s		$4.4 \cdot 10^{-12}$	0.07
				0.001	12.5s		$4.6 \cdot 10^{-12}$	0.069

- When a counter-example is found, however, **MatrixExp** terminates faster that **Simulation** (due to simulation batches).
- In the ARCH tool competition, **Hylaa** finds an error in one of the benchmarks in 0.02 seconds!

Clamped Beam (350 dims)

Beam	350	$x_{89} \geq 2100$	Simulation	0.1	4m38s	✓		
				0.01	4m47s	✓		
				0.001	7m0s	✓		
			Matrix Exp	0.1	5.2s	✓		
				0.01	28.7s	✓		
				0.001	4m24s	✓		
Beam*	350	$x_{89} \geq 1000$	Simulation	0.1	4m28s		$1.4 \cdot 10^{-12}$	16.1
				0.01	4m39s		$1.5 \cdot 10^{-12}$	16.05
				0.001	6m30s		$8.8 \cdot 10^{-13}$	16.041
			Matrix Exp	0.1	5.1s		$1.3 \cdot 10^{-11}$	16.1
				0.01	23.7s		$2.2 \cdot 10^{-11}$	16.05
				0.001	3m28s		$2.1 \cdot 10^{-12}$	16.041

- The original safety specification was created using simulations. For 8 of 9 models it was safe.
- For the Clamped Beam model, however, **it was not!** This shows that simulation can miss errors. The error was not known before analysis with **Hylaa**.

International Space Station (274 dims)

ISS*	274	$y_3 \notin [-0.0005, 0.0005]$	Simulation	0.1	7m23s	✓		
				0.01	7m14s	✓		
				0.001	7m44s	✓		
			Matrix Exp	0.1	2.9s	✓		
				0.01	11.5s	✓		
				0.001	1m39s	✓		
ISS	274	$y_3 \notin [-0.00017, 0.00017]$	Simulation	0.1	7m11s		$1.1 \cdot 10^{-6}$	0.5
				0.01	7m8s		$7.9 \cdot 10^{-7}$	0.5
				0.001	4m10s		$6.3 \cdot 10^{-7}$	0.498
			Matrix Exp	0.1	0.7s		$1.1 \cdot 10^{-6}$	0.5
				0.01	1.2s		$7.9 \cdot 10^{-7}$	0.5
				0.001	3.2s		$6.3 \cdot 10^{-7}$	0.498

- **Simulation** vs **MatrixExp**; which is better?

MNA5 (10923 dims)

MNA5*	10923	$x_1 \geq 0.2 \vee x_2 \geq 0.15$	Simulation	0.1	41m28s	✓		
				0.01	3h51m	✓		
				0.001	24h2m	✓		
			Matrix Exp	0.1	14h3m	✓		
				0.01	(>5d)			
				0.001	(>53d)			
MNA5	10923	$x_1 \geq 0.1 \vee x_2 \geq 0.15$	Simulation	0.1	5m57s		$9.2 \cdot 10^{-8}$	2.0
				0.01	21m40s		$2.0 \cdot 10^{-7}$	1.92
				0.001	2h11m		$2.0 \cdot 10^{-7}$	1.919
			Matrix Exp	0.1	1h25m		$9.2 \cdot 10^{-8}$	2.0
				0.01	13h29m		$2.0 \cdot 10^{-7}$	1.92
				0.001	(>5d)			

- For the largest models, **Simulation** seems to work faster. Why?
  - Euler simulation: x(t+1) := x(t) + A * x(t)

The Journey to 10000 Dimensions

- The benchmark model file is empty!
- SpaceEx Model Editor freezes! Use text editor. Gedit → Geany
- Hyst conversion (ANTRL Grammar Exception), 11k * 2 initial conditions
- Hyst stack overflow → internal expression tree unbalanced
- 800MB Python script → OS freezes (cannot run first line)
- OS freezes when swap is active
- Change Hyst to initialize matrix of zeros and assign entries (sparse repr)
- Out of memory while computing... 800 MB * 20000 steps = 16 TB!
  - Don't use explicit Jacobian in ODEINT
  - Python uses processes for parallelism... keep dynamics sparse
  - Run simulations a few steps at a time
- Random crashes “pickling” matrices, LP solving GLPK errors...
  bad memory stick!

Conclusion

Continuous systems with over 10000 dimensions can be verified in tens of minutes to tens of hours.

The **Hylaa** tool code, repeatability scripts, the earlier interactive demo, and videos are all available online:

stanleybak.com/hylaa

There will be a more complete talk about **Hylaa** at HSCC Wednesday afternoon*.

* "HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems", S. Bak and P. S. Duggirala, Hybrid Systems: Computation and Control (HSCC 2017)