EPiC Series in Computing

Volume 54, 2018, Pages 135–144

ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems

Numerical Verification of 10000-dimensional Linear Systems 10000x Faster

Stanley Bak

Safe Sky Analytics
stanleybak@gmail.com

Abstract

Tool Presentation: We evaluate an improved reachability algorithm for linear (and affine) systems implemented in the continuous branch of the Hylaa tool. While Hylaa's earlier approach required n simulations to verify an n-dimensional system, the new method takes advantage of additional problem structure to produce the same verification result in significantly less time. If the initial states can be defined in i dimensions, and the output variables related to the property being checked are o-dimensional, the new approach needs only $\min(i, o)$ simulations to verify the system, or produce a counter-example. In addition to reducing the number of simulations, a second improvement speeds up individual simulations when the dynamics is sparse by using Krylov subspace methods.

At ARCH 2017, we used the original approach to verify nine large linear benchmarks taken from model order reduction. Here, we run the new algorithm on the same set of benchmarks, and get an identical verification result in a fraction of the time. None of the benchmarks need more than tens of seconds to complete. The largest system with 10922 dimensions, which took over 24 hours using last year's method, is verified in 3.4 *seconds*.

1 Introduction

Reachability analysis attempts to compute, given a set of initial states, the set of states a system can enter. The verification problem can be solved by checking if an unsafe state is among the reachable states.

Reachability methods often target dynamics that are defined by linear or affine differential equations. Many tools exist to verify this class of systems [3], including Axelerator [13, 25], CORA [1, 4], Flow* [14, 15], Hylaa[10, 11], HyPro [30, 29], SpaceEx [19, 18], XSpeed [28, 22].

This class of systems is popular because it strikes a balance between the expressiveness of the models and the efficiency and accuracy of reachability computation. Solutions for linear systems can be written exactly using the matrix exponential, which is used by reachability techniques. Further, more complicated dynamics can also be converted to piecewise non-deterministic linear systems using the well-studied hybridization technique [26, 8, 5, 7, 24].

Last year at ARCH 2017 [9], the Hylaa tool demonstrated the scalability of its approach by verifying a set of nine large linear systems benchmarks taken from model order reduction [31]. All models were successfully analyzed, with runtimes up to 24 hours for the largest, 10922-dimensional system.

Here, we evaluate an improved reachability algorithm [12] for the same class of systems, with significantly improved computation efficiency. For comparison, we verify the same set of nine benchmarks as before, and obtain an identical verification result, including counter-example generation in the unsafe cases, in significantly less time. In particular, on the largest model the new method is on the order of 10000 times faster, verifying the largest system in 3.4 seconds.

The key insight of the method is that a single numerical simulation can compute linear combinations of either the rows or columns of the matrix exponential. Previously in Hylaa, a simulation needed to be performed for each column, which were then combined to get the full matrix exponential. This approach needed n simulations to verify an n-dimensional system. However, by taking advantage of common problem structure, a lower-dimensional initial space or output space, we can greatly reduce the number of required simulations. In a sense, the new method achieves its speedup by only computing the portion of the matrix exponential relevant to the verification problem. As a second improvement, when the dynamics matrix is sparse, efficient Krylov subspace methods can be used to reduce the simulation time.

In this paper, we first provide the details of the new method in Section 2 along with a simple example. Next, we present the updated benchmark result with the new approach in Section 3. Some of these structure observations have been used before in earlier research on reachability. We review related work and provide a comparison with the new approach in Section 4.

2 Reachability Method

Before evaluating the new reachability method in the next section, we first present the improvements in more detail. The new reachability method [12] uses two types of common problem structure to improve efficiency: (i) the often lower dimension of the initial and output spaces, and (ii) the sparsity of the dynamics matrix. We first present the verification problem and a running example in Section 2.1, followed by the earlier verification approach in Section 2.2. The two improvements are then described, in turn, in Section 2.3 and Section 2.4.

2.1 Timed Harmonic Oscillator Example

A timed harmonic oscillator has the dynamics $\dot{\mathtt{x}} = \mathtt{y}$, $\dot{\mathtt{y}} = -\mathtt{x}$, $\dot{\mathtt{t}} = 1$. The trajectories in this system rotate clockwise around the origin on the $\mathtt{x}$-$\mathtt{y}$ plane. Initially, let $\mathtt{x} = -5$, $\mathtt{y} \in [0, 1]$, and $\mathtt{t} = 0$. Let the unsafe set be all states where $\mathtt{x} = 4$. In this paper, we deal with discrete time reachability (known methods can extend this to continuous time [19]), so we consider a fixed time step of $\frac{\pi}{4}$.

The reachable set of states for the system can be plotted on the $\mathtt{x}$-$\mathtt{y}$ plane, as shown in Figure 1. From the plot, clearly the unsafe states are reachable at time $\frac{3\pi}{4}$.

The verification problem is to check if there exists a state in the initial states, and a time which is a multiple of the step (and possibly less than some time bound), such that the solution to the ordinary differential equations (ODEs) from the initial set is an unsafe state. This can be stated more formally, using variables x (x is a vector of n state variables) with linear dynamics $\dot{x} = Ax$, an initial set of states $\mathcal{I}$ defined with linear inequalities $\mathcal{I} = \{x \mid \mathcal{I}_x x \leq \iota_x\}$, an unsafe set of states $\mathcal{U}$ defined with linear inequalities $\mathcal{U} = \{x \mid \mathcal{U}_x x \leq \upsilon_x\}$, a time step δ, and a time bound T. The verification problem is to find a pair (x_0, t) where $x_0 \in \mathcal{I}$ and $t = \delta i \leq T, i \in \mathbb{N}$ such that $e^{At}x_0 \in \mathcal{U}$, or to prove such an (x_0, t) does not exist. Here $e^{At}x_0$ is the solution to the linear ODEs $\dot{x} = Ax$ starting at state x_0 at time t, which uses the matrix exponential.

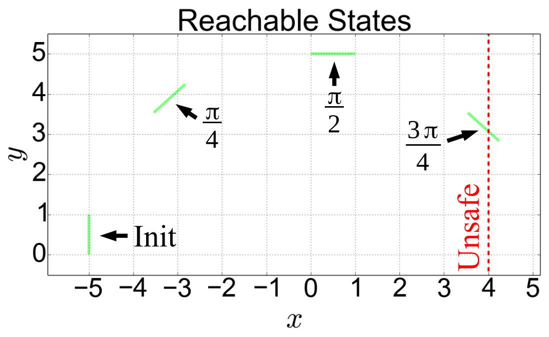

Figure 1: The timed harmonic oscillator system reaches an unsafe state at time $\frac{3\pi}{4}$.

The verification problem is defined by the matrices, A, $\mathcal{I}_x$, $\mathcal{U}_x$, the vectors ι_x, υ_x, and the scalars δ and T. In the timed harmonic oscillator case, since there is an affine term (the derivative of $\texttt{t}$ is 1), we first convert the dynamics to be linear by adding an additional variable $\texttt{a}$, which is initialized to 1, and remains constant at all time, $\dot{\texttt{a}} = 0$. We can then rewrite the derivative of $\texttt{t}$ to be $\dot{\texttt{t}} = \texttt{a}$ to get a linear system with the same solutions as the original affine system, when projected onto the original three variables. The system now has four variables, $\texttt{x}$, $\texttt{y}$, $\texttt{t}$, and $\texttt{a}$, with

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

2.2 Earlier Verification Approach

The previous verification approach used in Hylaa [10] computes the matrix exponential column-by-column using numerical simulations based on methods such as Runge Kutta [17]. This is possible because, as mentioned before, the solution of the linear system $\dot{x} = Ax$ from some initial state v is equal to $e^{At}v$. We can consider each column of the identity matrix as an initial state, run a numerical simulation, and then combine the results to construct the matrix exponential at each step. If e_j is the jth column of the identity matrix, $I = (e_1|e_2|\ldots|e_n)$, then $e^{At} = e^{At}I = (e^{At}e_1|e^{At}e_2|\ldots|e^{At}e_n)$, and each $e^{At}e_j$ can be computed by performing a numerical simulation starting from e_j up to time t. This method requires one simulation for each of the n columns of the identity matrix.

With the matrix exponential computed, safety can be checked at each discrete time step by constructing a set of linear constraints and using a linear programming (LP) solver. We construct an LP with two sets of variables, x at the current time step and x_0 at the initial time. We add constraints on the x_0 variables using $\mathcal{I}_x$ and ι_x, and constraints on the x variables from $\mathcal{U}_x$ and υ_x. Finally, we encode the linear relationship between x and x_0, which is just the values of the matrix exponential at the current time t. If the LP is feasible, then an unsafe state is reachable at time t, otherwise it is not. Further, if the LP is feasible, the solver provides an assignment to the variables that demonstrates the counter-example, namely the value of the start state x_0. This process of constructing the LP is repeated at each discrete time step up to the time bound, changing only the values of the matrix exponential at each step.

Basis Matrix

$$\begin{pmatrix} -1 & 0 & 0 & 0 & \boxed{-0.707} & 0.707 & 0 & 0 \\ 0 & -1 & 0 & 0 & -0.707 & -0.707 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 2.36 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 0 \\ 0 & \multicolumn{2}{c}{\textbf{Unsafe}} & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & \multicolumn{2}{c}{\textbf{Condition}} & 0 & \mathbf{0} & \mathbf{-1} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} \mathtt{x} \\ \mathtt{y} \\ \mathtt{t} \\ \mathtt{a} \\ \mathtt{x}_0 \\ \mathtt{y}_0 \\ \mathtt{t}_0 \\ \mathtt{a}_0 \end{pmatrix} \begin{matrix} = 0 \\ = 0 \\ = 0 \\ = 0 \\ \mathbf{= 4} \\ \mathbf{= -5} \\ \mathbf{\le 0} \\ \mathbf{\le 1} \\ \mathbf{= 0} \\ \mathbf{= 1} \end{matrix}$$

Initial State Conditions

(a) Full Linear Constraints (Earlier Method)

Basis Matrix

$$\begin{pmatrix} -1 & 0.707 & 3.54 \\ \mathbf{1} & 0 & 0 \\ 0 & \mathbf{-1} & \mathbf{0} \\ 0 & \mathbf{1} & \mathbf{0} \\ 0 & \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} o_{\mathtt{x}} \\ i_{\mathtt{y}} \\ i_{\mathtt{f}} \end{pmatrix} \begin{matrix} = 0 \\ \mathbf{= 4} \\ \mathbf{\le 0} \\ \mathbf{\le 1} \\ \mathbf{= 1} \end{matrix}$$

Initial State Conditions

(b) Using Initial and Output Spaces

Figure 2: The linear constraints for the timed harmonic oscillator example at time $\frac{3\pi}{4}$ can be encoded using the full set of linear constraints (left), or with initial/output spaces (right). Both methods generate the same unsafe counter-example.

The set of linear constraints for the timed harmonic oscillator example at time $\frac{3\pi}{4}$ are shown in Figure 2a. As mentioned before, only the values of the matrix exponential change between time steps. These values, which we call the *basis matrix*, are circled in red in the figure.

Since for small time steps the matrix exponential changes very little, it greatly benefits the performance of the method to use warm-start linear programming [16], where the previous solution is used as a starting point for the next LP, rather than constructing and solving a new LP from scratch at each time step. With this optimization, the majority of the computation time is usually spent computing the matrix exponential by running n simulations.

2.3 Leveraging Initial and Output Spaces

We can reduce the number of simulations required by taking advantage of initial and output spaces, which are often low-dimensional. If i is the dimension of the initial states and o is the dimension of the output space, we can reduce the number of simulations needed to $\min(i, o)$.

Formally, the output space is defined with an $o \times n$ matrix C. The unsafe states, then, need to be defined in the output space, $\mathcal{U} = \{x \mid y = Cx \wedge \mathcal{U}_y y \le \upsilon_y\}$. Similarly, the initial space is defined with an $n \times i$ matrix E. The initial states then must be written in the input space, $\mathcal{I} = \{x \mid x = Ez \wedge \mathcal{I}_z z \le \iota_z\}$.

Notice we can always take C and E to be the identity matrices, making $\mathcal{U}_y = \mathcal{U}_x$, $\upsilon_y = \upsilon_x$, $\mathcal{I}_z = \mathcal{I}_x$, and $\iota_z = \iota_x$. This means this approach does not lose any generality in expressing the initial or unsafe states. However, it is often the case that the output space is small, $o \ll n$, or the initial space is small, $i \ll n$. It is in these cases that we can reduce the number of required simulations.

In the timed harmonic oscillator example, for instance, we can express the system with $i = 2$ and $o = 1$. The initial space can be written in terms of two variables: $i_{\mathtt{y}}$, which corresponds to $\mathtt{y}$, and $i_{\mathtt{f}}$, which is used to constrain all the fixed initial terms of $\mathtt{x}$, $\mathtt{t}$, and $\mathtt{a}$. The E matrix is the 4×2 matrix $\left(\begin{smallmatrix} 0 & 1 & 0 & 0 \\ -5 & 0 & 0 & 1 \end{smallmatrix}\right)^T$, with the initial constraints $0 \le i_{\mathtt{y}} \le 1$ and $i_{\mathtt{f}} = 1$. For the output space, there is a single variable $o_{\mathtt{x}}$ which corresponds to the $\mathtt{x}$ variable. C is thus the 1×4 matrix $\left(\begin{smallmatrix} 1 & 0 & 0 & 0 \end{smallmatrix}\right)$, and the unsafe constraint is $o_{\mathtt{x}} = 4$.

In order to verify a system written with initial and output constraints, the LP at each step will contain a set of variables in the initial space and a set of variables in the output space at the current time. The linear constraints encode the initial constraints in the initial space, the output constraints in the output space, and the relationship between the initial and output variables, $Ce^{At}E$. This is the value which must be updated at each step, the basis matrix.

The constraints for the example using initial and output spaces is shown in Figure 2b. Once again, if the LP is feasible, an assignment will be provided to the initial space variables, from which we can recover the initial state that led to the unsafe state through $x = Ez$.

The value of $Ce^{At}E$ can be computed using either i or o numerical simulations. In the i case, we can first compute $e^{At}E$ by simulating the system $\dot{x} = Ax$ starting from each column of the E matrix, and then multiplying the result by C. Since E has i columns, if $i < n$ then we have reduced the number of required simulations. Alternatively, in the o case, we use properties of the matrix transpose. Since $Ce^{At}E = ((Ce^{At}E)^T)^T = (E^T(e^{At})^T C^T)^T = (E^T e^{A^T t} C^T)^T$, we can first compute $e^{A^T t}C^T$ by simulating the transpose dynamics $\dot{x} = A^T x$ starting from each column of C^T, and then multiplying the result by E^T. Once completed, a final transpose is done to get the basis matrix. Since each column of C^T is a row of C, and C has o rows, this means that the basis matrix can be computed with o simulations. Since either option can be used to compute the basis matrix $Ce^{At}E$, the number of required simulations is $\min(i, o)$.

In the timed harmonic oscillator example, since $i = 2$ and $o = 1$, we can find the basis matrix using a single numerical simulation using the transpose system dynamics. The basis matrix in Figure 2b can be computed by starting from the initial state from the single row of C, $(1, 0, 0, 0)^T$, and simulating using the transpose dynamics A^T up to time $\frac{3\pi}{4}$. This results in the state $(-0.707, 0.707, 0, 0)$ which is multiplied by $E^T = \left(\begin{smallmatrix} 0 & 1 & 0 & 0 \\ -5 & 0 & 0 & 1 \end{smallmatrix}\right)$ to get $(0.707, 3.54)^T$, the transpose of the values in the basis matrix in the figure. An LP solver can then solve these constraints and provide an assignment to the variables: $o_{\mathtt{x}} = 4$, $i_{\mathtt{f}} = 1$, $i_{\mathtt{y}} = 0.66$. Multiplying the initial variables with E provides the initial state for the counter-example in the original state variables, $(\mathtt{x}, \mathtt{y}, \mathtt{t}, \mathtt{a})^T = (-5, 0.66, 0, 1)^T$. Finally, simulating from this state for $\frac{3\pi}{4}$ time confirms that an unsafe state is reached at state $(4, 3.07, 2.36, 1)^T$. The accuracy of the computed counter-example can then be evaluated by comparing C multiplied by the simulated unsafe state with the values of the output variables given in the LP solution.

2.4 Leveraging Sparse Dynamics

When the dynamics matrix A is large and sparse (contains many zeros), numerical simulations can be done more efficiently by using Krylov subspace simulation methods [20]. These methods approximate $e^A v$ by computing its closest element in the k-dimensional Krylov subspace $K_k \equiv span\{v, Av, \ldots, A^{k-1}v\}$. This is done through the formula $e^{At}v \approx \|v\|\, V_k e^{H_k t} e_1$, where V_k and H_k are the outputs after k iterations of the well-known Arnoldi algorithm [6], starting from the normalized version of the initial vector v, and e_1, as before, is the first column of the identity matrix. Here, V_k is the projection from the Krylov subspace back to the original system, and H_k is the projection of the linear map A in the Krylov subspace. The approximation formula essentially takes the normalized projection of the initial vector in the Krylov subspace, e_1, performs the matrix exponential operation using A's projection H_k, and then projects the result back to the original space with V_k.

We omit many details here for space considerations, but do mention that one main challenge with Krylov simulation methods is to choose the dimension of the Krylov subspace k that gives a sufficiently accurate result. Using a small k is faster, but may result in a simulation with unacceptably large error. The main innovation with the new algorithm is to do a posteriori error analysis to determine when k is large enough, as opposed to earlier approaches [23] which use an a priori bound that can be overly pessimistic.

Two types of a posteriori analysis are possible. The simplest, which was used in this work, is to run Krylov simulations using both a k and $k+1$ dimensional Krylov subspace. The difference between the two simulations is compared in terms of relative error, until some acceptably small error is reached, for example 10^-6 (the numerical tolerance used in the LP solver). If the error is too large, k is increased and the process repeats. An alternative a posteriori error bound can be derived by examining the elements of the H_k matrix and using the eigenvalues of A [32]. This provides a guarantee on the error, but H_k must be first computed so a sufficient value of k is not known ahead of time. This method has similar performance to the one we evaluated here [12].

Notice these methods are numerical, and give the answer only up to some desired accuracy. This is similar to numerical methods in linear algebra for high-dimensional systems. This does mean that there is a small chance that the inaccuracy in the matrix exponential computation, leads to a missed error in the system. This positive side of this is that scalability and speed are significantly higher than for guaranteed approaches, to the point where a numerical approach may be the only option if a system is sufficiently large.

3 Benchmark Results

We evaluate the new reachability algorithm on a set of nine large linear systems benchmarks presented in ARCH16 taken from model order reduction [31]. This was the same set of benchmarks analyzed last year with Hylaa's previous approach [9], so we can directly compare the verification result and performance. We used the smallest time step considered before, 0.001. The measurements were performed on the same system as the earlier result, a system with an Intel i7-3930K processor (6 cores, 12 threads) running at 3.5 GHz. All nine systems were successfully verified in less than 20 seconds. The full results are shown in Table 1. Every model experienced a speedup, ranging from 1.3x to over 25000x for the largest system.

For each system, the output space has either one or two dimensions. This means that the alternative method of simulating with the transpose dynamic requires less simulations, and is therefore used for each system. This sometimes drastically reduces the number of required simulations compared with using the initial space approach. For example, in the Space Station system, all 273 dimensions are initially uncertain, so the initial space is 273-dimensional. Although the output property for this system involves many state variables, it is a single linear combination of them, and therefore a one-dimensional output space can be used.

The Krylov simulation method only benefits some of the systems. The dimension of the Krylov subspace k used for each simulation is shown in the table (some of the systems require two simulations). For some systems, like PDE, FOM, MNA1, and MNA5, the Krylov subspace dimension is significantly less than the number of system dimensions n, which saves computation time. For others, k is equal to n, and so using the Krylov method does not provide a benefit, and likely slightly slows the analysis due to the need to repeatedly check the error and increase k. The larger systems seem to benefit more from Krylov subspace simulation, which is promising for the scalability of the method.

Table 1: Benchmark results. The runtime and speedup columns use the safe specification for the systems. The CE Error column uses the unsafe specification for the systems.

Model	**Dims**	**Kry-Dims**	**New Runtime**	**Old Runtime** [9]	**Speedup**	**CE Error**
Motor	10	6, 6	1.5s	2.6s	1.7x	$1.3{\cdot}10^{-12}$
Building	49	49	5.9s	7.6s	1.3x	$7.2{\cdot}10^{-10}$
PDE	85	28	1.9s	13.4s	6.9x	$4.6{\cdot}10^{-13}$
Heat	201	201	2.1s	56.6s	26.6x	$6.6{\cdot}10^{-9}$
Space Station	273	273	18.8s	1m39s	5.3x	$7.5{\cdot}10^{-11}$
Beam	349	349	5.7s	4m24s	46.2x	$4.0{\cdot}10^{-11}$
MNA1	587	316	3.0s	20m41s	414.2x	$1.6{\cdot}10^{-9}$
FOM	1007	211	15.4s	45m15s	176.6x	$7.5{\cdot}10^{-11}$
MNA5	10922	63, 63	3.4s	24h2m	25207.2x	$1.1{\cdot}10^{-11}$

The result of the verification process (safe or unsafe) matches the previous method in every case. For the unsafe cases, a counter-example is found at exactly the same time step in the analysis as before. The `CE Error` column in the table shows the relative error between the counter-example found versus an external, high-accuracy simulation of the system. The relative errors are on the order of 10^{-9} or better, which shows that the a posteriori error analysis, which is included in the tool runtime, selects a sufficiently high dimension of the Krylov subspace.

One additional benefit of the new approach is that the basis matrix and LP at each step is significantly smaller than before. This likely benefits LP solving time as there are less variables and constraints, and also reduces the memory required to perform verification.

4 Related Work

The verification approach evaluated here uses problem two types of problem structure: (i) a lower-dimensional initial space or output space, and (ii) a sparse dynamics matrix A. Prior research has also used these structure assumptions, in various forms, to speed up analysis.

Zonotopes [21] are an example of a set representation which can represent low-dimensional initial sets in a higher dimensional space, when the number of generators is small. Zonotopes, however, are symmetric, making them more restrictive than the method here. Further, computing the reachable set at each time step uses a linear map operation that requires multiplying each generator by the full matrix exponential. This means the full matrix exponential must be computed, which may be slow for high-dimensional systems.

Similarly, support functions [27] can be seen as a way to take advantage of a low-dimensional output space. In particular, when using support functions to encode only the linear constraints used in the safety property, a highly-efficient method results. This approach was used by SpaceEx in the piece-wise affine reachability competition in ARCH17 for the building benchmark, where a single support function direction allowed the analysis to quickly complete [3]. Although having a small output space makes the linear map operation efficient, like zonotopes, the full matrix exponential must still be computed.

Reachability methods with affine representations [23] leverage both low-dimensional initial sets and the sparsity of the dynamics with Krylov subspace simulation. This means that only part of the matrix exponential needs to be computed. However, unlike our method, this approach does not alternatively leverage a low-dimensional output space using the transpose dynamics to reduce the number of simulations. For example, it would need to run a simulation for each of the n dimensions for the `Space Station` system in our evaluation, because each initial dimension is uncertain, even though the output space is one-dimensional. Further, their Krylov simulations use an a priori error condition to determine the necessary dimension of the Krylov subspace k, and then bloat the result by the computed error bound. On the one hand, this elegantly accounts for the simulation error. On the other hand, a priori error estimates for Krylov subspace methods are often extremely pessimistic [12], even when the simulations are actually accurate. This reduces the potential benefits of Krylov subspace simulation. This can be somewhat mitigated in reachability analysis [2] by using newer results on a priori error bounds for Krylov subspace simulations [32].

5 Conclusion

We presented an improved reachability algorithm for linear and affine systems implemented in Hylaa. The approach was used to analyze a set of large linear systems benchmarks taken from ARCH16, and demonstrated a significant speed up in verification time. The increase was greatest for the highest dimensional systems, implying that it may be possible to verify even larger systems in a reasonable time. One class of systems where such high-dimensional models arise is in spatial discretizations of partial differential equations (PDEs) [23]. For example, a 3-D discretization of a PDE on a 100x100x100 mesh would result in million dimensional ODE system. The method could also be adapted for use in hybrid systems. Guards of hybrid systems are often low-dimensional, so leveraging a low-dimensional output space could provide a speedup.

References

[1] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015.

[2] M. Althoff. Reachability analysis of large linear systems with uncertain inputs in the krylov subspace. *arXiv preprint arXiv:1712.00369*, 2017.

[3] M. Althoff, S. Bak, D. Cattaruzza, X. Chen, G. Frehse, R. Ray, and S. Schupp. Arch-comp17 category report: Continuous and hybrid systems with linear continuous dynamics. In G. Frehse and M. Althoff, editors, *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 48 of *EPiC Series in Computing*, pages 143–159. EasyChair, 2017.

[4] M. Althoff and D. Grebenyuk. Implementation of interval arithmetic in CORA 2016. In *Proc. of the 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 91–105, 2016.

[5] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4042–4048. IEEE, 2008.

[6] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics*, 9(1):17–29, 1951.

[7] E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43(7):451–476, 2007.

[8] S. Bak, S. Bogomolov, T. A. Henzinger, T. T. Johnson, and P. Prakash. Scalable static hybridization methods for analysis of nonlinear systems. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, HSCC '16, pages 155–164, New York, NY, USA, 2016. ACM.

[9] S. Bak and P. S. Duggirala. Direct verification of linear systems with over 10000 dimensions. In *4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, EPiC Series in Computing. EasyChair, 2017.

[10] S. Bak and P. S. Duggirala. Hylaa: A tool for computing simulation-equivalent reachability for linear systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, 2017.

[11] S. Bak and P. S. Duggirala. Rigorous simulation-based analysis of linear hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2017.

[12] S. Bak, H.-D. Tran, and T. T. Johnson. Numerical verification of affine systems with up to a billion dimensions. *arXiv:1804.01583*, 2018. https://arxiv.org/abs/1804.01583.

[13] D. Cattaruzza, A. Abate, P. Schrammel, and D. Kroening. Unbounded-time analysis of guarded lti systems with inputs by abstract acceleration. In *International On Static Analysis*, pages 312–331. Springer, 2015.

[14] X. Chen, E. Abraham, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *Proceedings of the 2012 IEEE 33rd Real-Time Systems Symposium*, RTSS '12, pages 183–192, Washington, DC, USA, 2012. IEEE Computer Society.

[15] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*, pages 258–263. Springer, 2013.

[16] J. W. Chinneck. Practical optimization: a gentle introduction. *Systems and Computer Engineering*, 2006.

[17] P. S. Duggirala and M. Viswanathan. Parsimonious, simulation based verification of linear systems. In *International Conference on Computer Aided Verification*, pages 477–494. Springer, 2016.

[18] G. Frehse, R. Kateja, and C. Le Guernic. Flowpipe approximation and clustering in space-time. In *Proc. Hybrid Systems: Computation and Control (HSCC'13)*, pages 203–212. ACM, 2013.

[19] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[20] E. Gallopoulos and Y. Saad. Efficient solution of parabolic equations by krylov approximation methods. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1236–1264, 1992.

[21] A. Girard. Reachability of uncertain linear systems using zonotopes. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, LNCS. Springer, 2005.

[22] A. Gurung, A. Deka, E. Bartocci, S. Bogomolov, R. Grosu, and R. Ray. Parallel reachability analysis for hybrid systems. In *Formal Methods and Models for System Design (MEMOCODE), 2016 ACM/IEEE International Conference on*, pages 12–22. IEEE, 2016.

[23] Z. Han and B. H. Krogh. Reachability analysis of large-scale affine systems using low-dimensional polytopes. In *HSCC*, volume 6, pages 287–301. Springer, 2006.

[24] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE transactions on automatic control*, 43(4):540–554, 1998.

[25] B. Jeannet, P. Schrammel, and S. Sankaranarayanan. Abstract acceleration of general linear loops. In *ACM SIGPLAN Notices*, volume 49, pages 529–540. ACM, 2014.

[26] N. Kekatos, M. Forets, and G. Frehse. Constructing verification models of nonlinear simulink systems via syntactic hybridization. 2017.

[27] C. Le Guernic and A. Girard. *Reachability Analysis of Hybrid Systems Using Support Functions*, pages 540–554. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[28] R. Ray, A. Gurung, B. Das, E. Bartocci, S. Bogomolov, and R. Grosu. Xspeed: Accelerating reachability analysis on multi-core processors. In *Haifa Verification Conference*, pages 3–18. Springer, 2015.

[29] S. Schupp and E. Abraham. Efficient dynamic error reduction for hybrid systems reachability analysis. In *Proc. of the 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'18)*, LNCS. Springer, 2018.

[30] S. Schupp, E. Abraham, I. B. Makhlouf, and S. Kowalewski. HyPro: a C++ library of state set representations for hybrid systems reachability analysis. In *NASA Formal Methods Symposium*, pages 288–294. Springer, 2017.

[31] H.-D. Tran, L. V. Nguyen, and T. T. Johnson. Large-scale linear systems from order-reduction (benchmark proposal). In *3rd Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Vienna, Austria, 2016.

[32] H. Wang and Q. Ye. Error bounds for the krylov subspace methods for computations of matrix exponentials. *SIAM Journal on Matrix Analysis and Applications*, 38(1):155–187, 2017.