Reachability of Black-Box Nonlinear Systems after Koopman Operator Linearization

Stanley Bak * **Sergiy Bogomolov** **
Parasara Sridhar Duggirala *** **Adam R. Gerlach** ****
Kostiantyn Potomkin **

* *Department of Computer Science, Stony Brook University, NY, USA*
** *Newcastle University, Newcastle Upon Tyne, UK*
*** *Department of Computer Science, University of North Carolina at Chapel Hill, NC, USA*
**** *Aerospace Systems Directorate, United State Air Force Research Laboratory, Wright–Patterson Air Force Base, OH, USA*

Abstract:
Reachability analysis of nonlinear dynamical systems is a challenging and computationally expensive task. Computing the reachable states for linear systems, in contrast, can often be done efficiently in high dimensions. In this paper, we explore verification methods that leverage a connection between these two classes of systems based on the concept of the Koopman operator. The Koopman operator links the behaviors of a nonlinear system to a linear system embedded in a higher dimensional space, with an additional set of so-called observable variables. Although, the new dynamical system has linear differential equations, the set of initial states is defined with nonlinear constraints. For this reason, existing approaches for linear systems reachability cannot be used directly. In this paper, we propose the first reachability algorithm that deals with this unexplored type of reachability problem. Our evaluation examines several optimizations, and shows the proposed workflow is a promising avenue for verifying behaviors of nonlinear systems.

Keywords: Reachability Analysis Nonlinear Systems Koopman Operator.

1. INTRODUCTION

Verification of cyber-physical systems requires algorithms that can reason over both software and the behavior of the physical world. In particular, computing the post operation for a set of states is an important fundamental operation. For software, the post operation computes how a set of variables is modified after executing a block of code. For the physical world, the post operation determines how state variables can change after some amount of time has elapsed. In hybrid systems terminology, the post operation is often called time-bounded reachability. Given a set of initial states, some model of the physical system, and a time bound, the goal is to compute how the set of states can change up to the time bound, possibly checking if some unsafe configuration is possible.

Reachability analysis for nonlinear systems is challenging, and is often the bottleneck for formal cyber-physical systems (CPS) analysis methods. In this work we investigate nonlinear reachability approaches based on *Koopman operator linearization* Mauroy et al. (2020). Koopman operator linearization is a process where a nonlinear system can be approximated as a linear system with a large number of so-called observable variables, each of which can be a nonlinear function of the original state variables. This linear system can be computed either symbolically from differential equations or—importantly for black-box systems—from data derived from real-world system executions or simulations Kutz et al. (2016). For reachability analysis, such a method is promising, as there exist highly-scalable methods to compute reachable sets for linear systems Girard (2005a); Le Guernic and Girard (2009); Bak et al. (2019). However, directly applying existing algorithms is not possible, as the observable variables are nonlinear functions of the original state variables. If the original initial set is described as a convex polytope, for example, the initial set of the Koopman linearized system may be nonconvex, which current algorithms do not support.

To apply the Koopman method for reachability analysis, two issues must be overcome. First, a Koopman linearized model of the nonlinear dynamics must be constructed whose behavior is a good approximation of the original system. Second, linear reachability analysis methods must be modified to support nonlinear initial state sets. The first problem is the focus of much current research on dynamical system analysis with the Koopman operator, and we do not focus on it here. Instead, this paper's primary contribution is on the second problem, developing efficient algorithms for analysis for the class of systems with linear dynamics and nonlinear initial sets. We first show the problem can be solved using a nonlinear satisfiability module theory (SMT) solver to enforce the initial state constraints. This Direct Encoding Algorithm is correct but may be slow in practice and even undecidable in theory. We improve analysis efficiency through zonotope overapproximations of the nonlinear initial sets constructed using interval arithmetic, as well as two abstraction-refinement techniques: (i) Hyperplane Backprogation and (ii) Zonotope Domain Splitting.

This paper is organized as follows. First, we provide a brief review of Koopman operator linearization in Section 3. Next, we describe our proposed verification algorithm and optimizations for Koopman linearized systems in Section 4. We then demonstrate the feasibility of the approach and evaluate each of the optimizations on a number of nonlinear systems in Section 5. We finish with related work and a conclusion.

2. PRELIMINARIES

Before detailing Koopman operator theory and our proposed reachability algorithm, we first introduce a few important mathematical preliminaries.

An interval $[a, b]$ (with $a \leq b$) denotes the set of numbers x such that $a \leq x \leq b$. Arithmetic operations such as addition and multiplication over intervals $[a_1, b_1]$ and $[a_2, b_2]$ are defined using *interval arithmetic* Jaulin et al. (2001) as follows.

$$\begin{aligned}[a_1, b_1] + [a_2, b_2] &= [a_1 + a_2, b_1 + b_2] \\ [a_1, b_1] \times [a_2, b_2] &= [\mathsf{min}\{a_1a_2, a_1b_2, b_1a_2, b_1b_2\}, \\ &\qquad \mathsf{max}\{a_1a_2, a_1b_2, b_1a_2, b_1b_2\}]\end{aligned}$$

The system under consideration evolves in the state space $\mathbb{R}^n$. Elements in the state space are denoted as $\boldsymbol{x}$, and scalars are denoted as x. Dynamical systems we consider in this paper evolve according to the differential equation:

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{x}). \tag{1}$$

Often, $\boldsymbol{f}$ is a nonlinear function of the state $\boldsymbol{x}$. We refer to such systems as non-linear dynamical systems. A trajectory of such nonlinear dynamical system described in Equation 1 is a function $\xi_{\boldsymbol{f}}(\boldsymbol{x}_0, t)$, the solution to the differential equation $\boldsymbol{f}$, that takes as input an initial state $\boldsymbol{x}_0$ and time t and returns the state of the system after time t. We often drop the subscript $\boldsymbol{f}$ from the solution function $\xi_{\boldsymbol{f}}$ when it is clear from context.

When the function $\boldsymbol{f}(\boldsymbol{x})$ is a linear function, i.e., $\boldsymbol{f}(\boldsymbol{x}) = A\boldsymbol{x}$ where $A \in \mathbb{R}^{n \times n}$, the trajectory has a closed-form solution. More specifically, $\xi(\boldsymbol{x}_0, t) = e^{At}\boldsymbol{x}_0$ where e^{At} is the matrix exponential that can be defined as:

$$e^{At} = I + \frac{At}{1!} + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \dots$$

For the dynamical system $\dot{x} = Ax$, the set of states that reach $c^T x \leq d$ at time t is given by the set of states $((e^{At})^T c)^T x \leq d$. That is, the backward propagation of a constraint $c^T x \leq d$ is obtained by right multiplying c with $(e^{At})^T$. We label this as **hyperplane backpropagation**.

An important part of Koopman linearization is the notion of observable functions, or observables, which are scalar functions from system's state space, $g : \mathbb{R}^n \to \mathbb{R}$. A vector of k scalar-valued observables maps the state space to $\mathbb{R}^k$ and is denoted as $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^k$. Sometimes, we denote the observables as $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x})$.

In this paper we focus on safety verification at *discrete* time instances with some step size $h > 0$, where h is assumed to evenly divide the analysis time bound T. This both simplifies the problem and allows us to generate counterexamples demonstrating safety violations. A trajectory on a nonlinear system can be safe in discrete time with respect to a given set of unsafe states.

Definition 2.1. Trajectory $\xi(\boldsymbol{x}_0, t)$ is **discrete time safe** if and only if $\forall_{0 \leq i \leq T/h}\, \xi(\boldsymbol{x}_0, ih) \notin U$, where U is an unsafe state set and $i \in \mathbb{Z}_{\geq 0}$ is the step number.

The main problem we want to solve is a discrete-time bounded nonlinear verification problem.

Problem 1. Given a set of initial states I, time step h, time bound T, and nonlinear dynamics $\boldsymbol{f}$, we want to prove the system is discrete-time safe, meaning that $\xi(\boldsymbol{x}_0, t)$ is discrete-time safe for every initial state $\boldsymbol{x}_0 \in I$.

We assume I and U are given as as conjunctions of linear constraints, unless indicated otherwise.

3. KOOPMAN OPERATOR LINEARIZATION

The modern study of dynamical systems is driven by Poincaré's *state space* view of the underlying system. By considering the evolution of points in a state space, this view enables intuitive tools for analyzing, designing, controlling, and verifying dynamical systems. However, it can be ill-suited for certain classes of problems such as uncertain systems Budišić et al. (2012); Gerlach et al. (2020) and systems without explicit equations describing their evolution (data-driven or black-box models) Jones (2001).

An alternative to this state space view is Koopman's *observable space* view of dynamical systems. In contrast to the state space view, the observable space view considers the evolution of observables, or functions, of the given state space Budišić et al. (2012) instead of the states themselves. This alternative view leads to the notion of the so-called Koopman operator Koopman (1931). For dynamical system $S \colon \Omega \to \Omega$ and observable $g \colon \Omega \to \mathbb{R}$, the Koopman operator $\mathcal{K}$ is defined by

$$\mathcal{K}g = g \circ S, \qquad \forall g \in L^\infty \tag{2}$$

As such, the Koopman operator is an infinite-dimensional linear operator on the space of scalar-valued functions of the state space Koopman (1931). The spectral properties of this linear operator describe the evolutionary properties of the underlying dynamical system S, similar to finite-dimensional linear state space models (*e.g.,* eigen decomposition of a state matrix). However, unlike a finite-dimensional linear state matrix which describes the evolution of system states, the Koopman operator describes the evolution of scalar-valued functions as driven by the dynamics of the underlying system Narasingam and Kwon (2019). For space reasons, we refer to our online appendix [1] for mor details on Koopman operator linearization.

[1] `https://www.dropbox.com/s/hu1f421n3jzgdxf/Appendix.pdf?dl=0`

As Eq. 2 is equally valid for linear and nonlinear systems, the observable space view enables the linear treatment of full nonlinear dynamics via the Koopman operator. Thus, the Koopman operator has the potential to bridge nonlinear systems and existing linear tools for analysis, design, control, and verification Brunton et al. (2016); Kutz et al. (2016) without sacrificing information, like with traditional linearization techniques Budišić et al. (2012). However, the Koopman operator linerization introduces the nonlinear initial state sets and, thus, the question arised on how to threat them efficiently. In the next section, we strive to answer this question.

4. VERIFYING LINEAR SYSTEMS WITH NONLINEAR OBSERVABLES

Koopman operator linearization creates approximations to non-linear, possibly black-box systems. The resulting systems have linear dynamics in the space of observables. Reachability analysis of linear systems is a well-studied topic, and existing methods are efficient even with thousands or more state variables Bak et al. (2019). However, the initial and unsafe constraints in the original problem are defined on the original system variables, not on the nonlinear observable variables. This nonlinear relationship creates two additional tasks: (1) projection of initial set from state space variables to the space of observables, and (2) projection of reachable set in the space of observables back into the state space. Problem (2) can be resolved by including the original state variables within the dictionary used during Koopman linearization. This means that the projection from the space of observables to the original state variable can be written as a simple linear transformation, $\boldsymbol{x} = M\boldsymbol{g}(\boldsymbol{x})$. Problem (1), however, is more difficult to handle, and the main focus of the rest of this section.

4.1 Direct Encoding of Nonlinear Constraints with SMT Solvers

Let the state space of the system to be $\boldsymbol{x}$ and the observables be $\boldsymbol{g}(\boldsymbol{x})$. Furthermore, the evolution of observables is determined by a linear differential equations, $\boldsymbol{g}(\boldsymbol{x}_t) = \mathcal{K}\boldsymbol{g}(\boldsymbol{x}_0)$. The state after time t, denoted as $\boldsymbol{x}_t$ is $M\boldsymbol{g}(\boldsymbol{x}_t)$. Given an initial set I, and unsafe set U, the safety verification can be formulated as satisfiability of constraints in Equation 3.

$$\begin{aligned} \boldsymbol{x}_0 \in I, y = \boldsymbol{g}(\boldsymbol{x}_0), \\ y_t = \mathcal{K}_t y, \\ \boldsymbol{x}_t = My_t, \boldsymbol{x}_t \in U. \end{aligned} \tag{3}$$

Given step size $h > 0$, for each time instant $t = i \times h$, an SMT solver can be invoked with the constraints in Equation 3. Often, the initial set I and unsafe set U are specified as conjunctions of linear constraints. For such cases, observe that the only nonlinear constraint in Equation 3 is $y = \boldsymbol{g}(\boldsymbol{x}_0)$. The complexity of finding an assignment of variables that satisfies these linear constaints dependd on the number and functions used for the observables.

For example, if the dictionary of observables are polynomials, SMT solvers can use algorithms like cylindrical algebraic decomposition Arnon et al. (1984) that are theoretically guaranteed to terminate with a correct result. In practice, this algorithm is doubly exponential in the number of variables, so a result may not be produced in a reasonable time. If the dictionary includes transcendental functions like sin or cos, the problem in general may not even be decidable.

In our implementation, we use the δ-decidability SMT solver dReal Dolzmann and Sturm (1997); Brown (2003); Gao et al. (2013), which theoretically always terminates but may produce an unknown result if the constraints are on the boundary of satisfiability (within a tolerance δ), which we can then still flag as potentially unsafe.

Although the direct encoding works, for the reasons stated above it can be slow, so we next focus on optimizations and efficiency improvements.

4.2 Overapproximating Nonlinear Constraints with Intervals

Suppose that the initial set I is given as hyperrectangles, as is often the case. That is, $I = [x_{0l}^1; x_{0u}^1] \times \ldots \times [x_{0l}^n; x_{0u}^n]$. Over such a domain, it is possible to compute conservative approximation of $y = \boldsymbol{g}(\boldsymbol{x}_0)$ where $\boldsymbol{x}_0 \in I$ using interval arithmetic. Alternatively, when I is defined with linear constraints, we can compute upper and lower bounds on each of the variables using linear programming (LP) to construct box bounds, and then use those to construct the conservative approximation of $y = \boldsymbol{g}(\boldsymbol{x}_0)$. Let the bounds we obtain on y be such that $y \in [y_l^1, y_u^1] \times \ldots \times [y_l^k, y_u^k]$. Substituting these in Equation 3 results in the following constraints.

$$\begin{aligned} y \in [y_l^1, y_u^1] \times \ldots \times [y_l^k, y_u^k], \\ y_t = \mathcal{K}_t y, \\ \boldsymbol{x}_t = My_t, \boldsymbol{x}_t \in U. \end{aligned} \tag{4}$$

This set of constraints can be solved efficiently using LP, as is done in linear systems reachability analysis using zonotopes Girard (2005a) or linear star sets Bak and Duggirala (2017).

While this method can be very efficient, the overapproximation of $\boldsymbol{g}(\boldsymbol{x})$ using interval arithmetic might yield a very coarse overapproximation. This would mean that spurious unsafe executions of the system may be found, due to the overapproximation. To overcome this the *Interval Encoding Algorithm* uses a hybrid approach, where an LP is first solved at each step according to Equation 4, and only if the LP is feasible will we then call the dReal SMT solver with the nonlinear constraints from Equation 3.

4.3 Hyperplane Backpropagation

One of the building blocks that helps us improve efficiency is the notion of **hyperplane backpropagation**. Consider the unsafe set given as $U = [U_l^1; U_u^1] \times \ldots \times [U_l^n; U_u^n]$, observables $\boldsymbol{g}(\boldsymbol{x})$, and $\boldsymbol{x} = M\boldsymbol{g}(\boldsymbol{x})$. Since we assumed the Koopman dictionary contained the original state variables, we can directly encode U as constraints in the observable observable space, which we write as $\boldsymbol{g}(U)$. If $q^T y \leq r$ is a halfplane constraint in $\boldsymbol{g}(U)$, then the corresponding constraint for propagating this constraint back by time

t is obtained by $(\mathcal{K}^T q)^T y \leq r$. We perform hyperplane backpropagation for all the constraints in $\boldsymbol{g}(U)$. In the Koopman linerization literature, this operation is called *pull-back* operation Meyers et al. (2019). Any state that satisfies all the constraints obtained by performing the hyperplane backpropagation will end up in $\boldsymbol{g}(U)$ after time t. We label these constraints as $\mathsf{PropCons}(U)$. While we have explicitly specified this only for unsafe sets that are intervals, this can be easily extended to unsafe sets specified as conjunction of half-spaces.

We use the above process to backpropagate each of the unsafe set constraints into the initial set, i.e., $\mathsf{PropCons}(U)$. We then compute the overlap between the projection of initial set into the observables ($\boldsymbol{g}(I)$) and compute its overlap with propagated constraints, i.e., $\boldsymbol{g}(I) \cap \mathsf{PropCons}(U)$, using interval arithmetic. We then project this set back into the initial set as $M(\boldsymbol{g}(I) \cap \mathsf{PropCons}(U))$. If this projected set I' is a strict subset of initial set I, then, one can specify the possible violation of safety with higher precision. We then repeat the process with the new set I'. If, during this process I' is empty, then, none of the trajectories go into the unsafe set; we can declare the system to be safe. If I' is same as the set I, then this approach does not further improve precision. In this case, the *Hyperplane Backpropagation Algorithm* would invoke dReal and instantiate the constraints given in Equation 3 with I' instead of I. The pseudocode of the core step of the algorithm is provided in Algorithm 1.

Algorithm 1 Hyperplane Backpropagation Algorithm

```
function BACKPROPAGATE($I, U, t, \boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x}), \mathcal{K}$)
    Output: A smaller initial set $I'$ after backpropagation, or $\emptyset$ if safe.
    $U_o = \boldsymbol{g}(U)$
    $\mathsf{PropCons} = \mathcal{K}^{-1} U_o$
    $I_o = \boldsymbol{g}(I)$
    $I' = M(I_o \cap \mathsf{PropCons})$
    while $I' \subset I$ do
        if $I' = \emptyset$ then
            return safe.
        $I_o = \boldsymbol{g}(I')$
        $I' = M(I_o \cap \mathsf{PropCons})$
    return $I'$
```

4.4 Zonotope Domain Splitting

The reachable states at each time step encoded by Equation 4 can be represented using a zonotope Girard (2005b), where the box domain of the zonotope is the initial set $[y_l^1, y_u^1] \times \ldots \times [y_l^k, y_u^k]$. The accuracy of this zonotope overapproximation can be improved by splitting each dimension's interval domain into several smaller subintervals and computing the new interval overapproximation in each of the subintervals.

For the proposed *Zonotope Domain Splitting Algorithm*, we use a heuristic that always splits the variable with the maximum range. After splitting, we then perform hyperplane backpropagation on each of the smaller intervals. This process can be repeated until either the overapproximation is safe for the current step, or some predetermined upperLimit is reached on the number of splits. Upon reaching the limit, we then invoke dReal using the nonlinear constraints from Equation 3. As a result, dReal is invoked with smaller initial sets, thus helping the numerical procedure to terminate faster. However, as we show later in the evaluation, performing refinement too often can harm the performance of the verification procedure, as splitting can increase the number of queries needed.

Notice that the order of applying hyperplane backpropagation and zonotope domain contraction can alter the performance of the verification procedure. While we prefer performing hyperplane backpropagation at each iteration, delaying the process until the interval under consideration becomes smaller could be a useful heuristic for some examples. In our evaluation, we consider various possible combinations of these two methods. In our experience, invoking dReal with the smaller initial sets succeeds to either prove safety or generate counterexample quicker than larger initial sets.

5. EVALUATION

In this section, we present experimental results of using the Koopman operator in reachability analysis. We implemented our algorithms in Julia, using the LazySets package of JuliaReach Bogomolov et al. (2019), the PyCall.jl[2] package to call dReal Gao et al. (2013), the DataDrivenDiffEq.jl[3] package for Koopman operator linearization via Extended DMD and the DifferentialEquations.jl Rackauckas and Nie (2017) package to generate numerical simulations. A Sobol sequence in the set of initial conditions is used to determine the initial conditions for the simulations. The resulting data matrices for each simulation are then combined via column-wise concatenation as in Tu et al. (2014).

For each system, the dictionary of observables for Koopman linearization included multivariate polynomial basis functions up to a fixed order for the original state variables, $\sin t$ and $\cos t$, and combinations of these (*e.g.*, $\boldsymbol{x} \sin^a t \cos^b t$). Lastly, SVD truncation was performed as described in Kutz et al. (2016) to remove any non-dominant modes.

Note that although we know the nonlinear differential equations for each system, we only used simulation data to perform Koopman linearization, treating each system as a black box. Since few approaches can work directly with black-box systems, this allows us to compare against other reachability methods that require knowing the system's dynamics.

5.1 System Definitions

We evaluate our algorithms on four benchmark nonlinear systems. As our algorithms are sensitive to the distance between the reachable set and the unsafe region, we consider parameterized unsafe regions, where a parameter i controls the distance of the reachable set to the unsafe region (if the value of i is big enough then the system is unsafe). The

[2] https://github.com/JuliaPy/PyCall.jl

[3] https://datadriven.sciml.ai/

nonlinear differential equations for each system are available on HyPro Schupp et al. (2017) benchmark website[4].

Roessler model We run the Roessler attractor[5] with the following initial set: $x(0) \in [-0.05, 0.05], y(0) \in [-8.45, -8.35], z(0) \in [-0.05, 0.05]$. In our experiments we use the following parameterized unsafe region: $y \geq 6.375 - 0.025 \cdot i$, where $i \in [0, 20]$. The Koopman linearized system contains 70 observables.

Steam model The steam governor Sotomayor et al. (2007) system[6] is modeled with the following initial set: $x(0) \in [0.95, 1.05], y(0) \in [-0.05, 0.05], z(0) \in [0.95, 1.05]$. In our experiments we use the unsafe region $y \leq -0.25 + 0.01 \cdot i$, where $i \in [0, 10]$. The Koopman linearized system contains 71 observables.

Coupled Van der Pol oscillator The composed model of two coupled Van der Pol oscillators Rand and Holmes (1980) system[7] is modeled with the following initial set: $x_1(0) \in [-0.025, 0.025], y_1(0) \in [4.975, 5.025], x_2(0) \in [-0.025, 0.025], y_2(0) \in [4.975, 5.025]$. In our experiments we use the following unsafe region: $x \geq 1.25 - 0.05 \cdot i$, where $i \in [1, 16]$. The linearized system contains 131 observables.

Biological model We run the biological Klipp et al. (2005) system[8] with the following initial set: $x_1(0), ..., x_7(0), \in [0.99, 1.01]$. In our experiments we use the following unsafe region: $x_4 \leq 0.883 + 0.002 \cdot i$, with $i \in [1, 10]$. The linearized system contains 104 observables.

Approximation Error Although the goal of this work is not to investigate the Koopman linearization process, it is important to show that accurate approximations are plausible for the workflow to make sense.

For this purpose, we run simulations from the corners and centers of the initial sets and compare the average and maximum errors at different points in time. Although error generally grows as the simulation time increases, it remains acceptable within the time bound. While for most of the models the and time bounds the error is less than 1%, the maximum relative error observed is in the Roessler model at time $0.6 * T$, where the error reaches 5.76%.

Further investigation into error bounds for the Koopman linearization process is a potential topic of future research. For general black-box systems where only data is provided, we do not expect guaranteed error bounds, although it may be possible to provide statistical guarantees. When the observables are derived symbolically from the differential equations, guaranteed error bounds may be possible, although existing approaches for this are fairly pessimistic Forets and Pouly (2017).

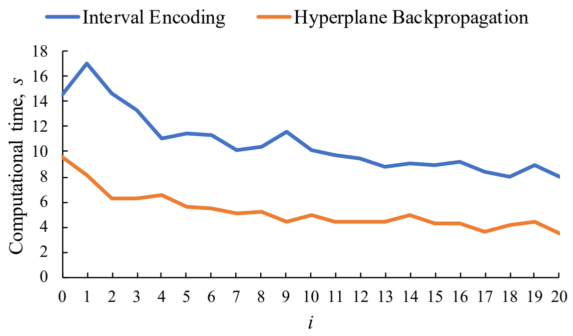

Fig. 1. Evaluation results of Interval Encoding and Hyperplane Backpropagation algorithms for the Roessler model for different values of i. Hyperplane Backpropagation is generally twice as fast for this problem.

5.2 Results

We run each of the models on a number of different instances. As mentioned earlier, for each model we parameterize unsafe regions using a single parameter i for each of the four systems. In this section, we evaluate each of the proposed improvements to the direct encoding algorithm, as well as provide a comparison with other verification methods. For space reasons, we do not present a detailed comparison of every optimization on every model instance, but instead elaborate on the overall trends.

Performance of Hyperplane Backpropagation We compare the performance of the Hyperplane Backpropagation (Section 4.3) against the Interval Encoding algorithm (Section 4.2). Figure 1 shows a comparison with the Roessler model. We observe that the algorithm with backpropagation is around two times faster than the Interval Encoding algorithm for all problem instances. In addition, the computational time gets smaller as i is increased. The main reason can be that we have a smaller time horizon when i is large, because an unsafe state is reachable. We need to compute up to $t = 2.93$ for $i = 1$ and up to $t = 2.81$ for $i = 21$. We also call dReal less when i is large for the same reason. For $i = 1$, with the Interval Encoding Algorithm we call dReal 14 times, and only 8 times for Hyperplane Backpropagation. For the last instance, $i = 21$, we call dReal 7 and 3 times for the two algorithms respectively, which leads to performance improvement.

Performance of Zonotope Domain Splitting We next evaluate Zonotope Domain Splitting (Algorithm 1) on the Coupled Van der Pol oscillator. We demonstrate the performance of the algorithm on two instances of the model: a safe case where $i = 4$ and an unsafe case where $i = 12$. We further evaluate using different values of max_level. We can observe that when the system is safe, Zonotope Domain Splitting with a large value of max_level generally benefits performance, whereas for $i = 12$ we see the opposite. The explanation is that for the safe instance we can save calls to dReal with backpropagation and splitting together. For $i = 12$, on the other hand, the system is unsafe and there are fewer steps where we can avoid calls to dReal. Performing splitting in this case is futile, as the overapproximation cannot prove safety of the system.

[4] https://ths.rwth-aachen.de/research/projects/hypro/benchmarks-of-continuous-and-hybrid-systems/
[5] https://ths.rwth-aachen.de/research/projects/hypro/roessler-attractor/
[6] https://ths.rwth-aachen.de/research/projects/hypro/steam-governor/
[7] https://ths.rwth-aachen.de/research/projects/hypro/coupled-van-der-pol-oscillator/
[8] https://ths.rwth-aachen.de/research/projects/hypro/biological-model-i/

Table 1. Evaluation results of the Zonotope Domain Splitting algorithm for the Coupled Van der Pol oscillator with $max\_level \in [0, 5]$. Safe corresponds to the $i = 4$ case and unsafe to the $i = 12$ case.

max_level	0	1	2	3	4	5
Safe	190.56	200.94	136.78	135.29	133.68	117.24
Unsafe	37.77	49.45	50.73	54.69	62.16	74.34

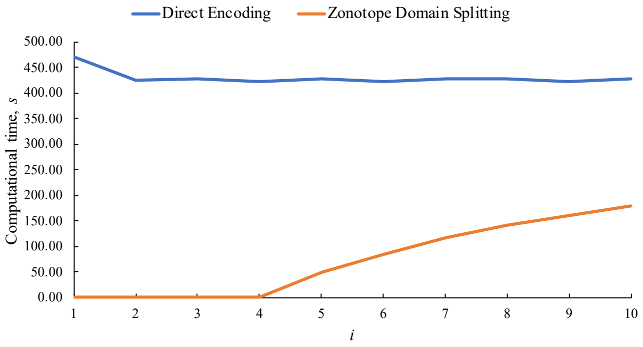

Fig. 2. Comparison of Direct Encoding and Zonotope Domain Splitting on the Biological model. When i is small, the optimized Zonotope Domain Splitting method verifies the system without calling dReal, and is over 1000 times faster.

Overall Algorithm Performance We next compare performance against the Direct Encoding algorithm (Section 4.1), using the Biological model. In Figure 2, for $i \in [1, 4]$ we observe the computation time is close to zero and we are 1000 times faster than when we use Direct Encoding to verify the system. We gain such a boost due to avoiding calls to dReal altogether, by only employing backpropagation. Put another way, the overapproximation is sufficient to verify the system as safe when the reachable states are far from the unsafe set. When $i \geq 5$ we can observe that the computational time for our approach increases due to dReal calls. However, since we still eliminate many unnecessary calls to dReal, our algorithm still verifies the system $4 - 8$ times faster. The runtime of the direct encoding method is much less sensitive to the i parameter.

Comparison with Other Methods We next summarize the verification time for each of the models using Zonotope Domain Splitting, Direct Encoding, the Flow* Chen et al. (2013) nonlinear reachability tool and the dReach tool Kong et al. (2015), which uses dReal to directly verify the original nonlinear systems. The result are shown in Table 2. The Flow* tool parameters were taken from the HyPro benchmark repository Schupp et al. (2017)—developed by the same group as Flow*—ensuring the measured performance is not due to a poor choice of parameters. The Zonotope Domain Splitting algorithm is up to 1000 times faster than both Flow* and Direct Encoding on many instances. However, there are instances (*e.g.* Steam model with $i = 0$) where Flow* performs better. Such cases outline directions for future research directions, for example trying to find ways to guide the search for the counterexample when splitting the zonotope domain, as opposed to always using the largest variable range. The dReach tool timed out on all of the models, meaning that a verification result was not produced within two hours. We confirmed the tool was being run correctly by severely reducing the analysis time bound until an output was produced.

Table 2. Computational time (seconds) comparing Flow*, Direct Encoding and the Zonotope Domain Splitting. The dReach tool timed out on all models.

	i	Flow*	Direct	Zono
Coupled VP	1	251.11	788.45	0.57
	8	497.61	680.61	53.91
	16	1665.16	557.24	18.52
Biological	1	260.69	470.59	0.59
	5	250.26	426.37	49.41
	10	238.56	427.00	179.25
Steam	0	61.06	197.08	182.62
	5	285.20	59.53	37.27
	10	77.68	29.21	18.52
Roessler	0	55.28	181.06	9.53
	10	78.33	177.92	5.01
	20	55.29	174.63	3.50

6. RELATED WORK

Domain contraction, the practice of narrowing the domain of possible solutions, has been used in solving constraint satisfaction over non-linear real arithmetic Benhamou and Granvilliers (2006); Fränzle et al. (2006); Gao et al. (2010); Granvilliers and Benhamou (2006). The same technique has been used in hybrid systems safety verification Ratschan and She (2007) and checking for intersection of flow-pipes with guard sets for discrete transitions Althoff and Krogh (2012); Chen et al. (2012). While in these works, the domain contraction is performed in a branch-and-bound manner, we propagate constraints and use a zonotope intersection method for computing the reduced domain.

Transforming a non-linear ODE into a linear ODE by performing change of variables has also been previously investigated Sankaranarayanan (2012). The main goal is to a) search for the change of variables transformation such that non-linear dynamical and hybrid systems become linear dynamical and hybrid systems and b) synthesize invariants for the linear system to prove safety. The current work differs in two ways. First, our method is purely data-driven and hence can also be applied to black-box systems. Second, instead of synthesizing invariants, we prove safety by computing the flow-pipe of the high dimensional linear system.

The process of approximating a nonlinear dynamical system as a piecewise linear system with uncertain inputs is called hybridization Asarin et al. (2003, 2007); Dang et al. (2010); Han and Krogh (2006). The inputs over-approximate the divergence between the linear and non-linear dynamics in a subspace defined by the invariant of each mode in the hybrid system. The first challenge in performing safety verification using hybridization is that it requires computing intersections of flow-pipes with the guards for discrete transitions, which might become expensive Girard and Le Guernic (2008). Some methods to avoid computing this intersections have been investigated Althoff and Krogh (2012); Bak et al. (2016). Secondly, for accurate hybridization, the number of modes in the hybrid system might be prohibitively large. This paper investigates techniques for approximating a nonlinear system as a high dimensional linear system for safety verification use observable variables, rather than linear approximations of the original nonlinear dynamics. While high dimensional linearization using Koopman operator has been used to analyze controllability in Goswami and Paley (2017) and propagating the probability distribution of initial set in Matavalam et al. (2020), these works do not explicitly construct flow-pipes for proving safety of nonlinear systems.

Finally, flow-pipe computation techniques for computing reachable set using Taylor Models Chen et al. (2013), polynomial Zonotopes Althoff (2015), sample trajectories Duggirala et al. (2015), and SMT solvers Kong et al. (2015) have all been investigated as potential techniques for nonlinear systems verification. These methods work directly on the nonlinear differential equations, which create scalability challenges, whereas we use Koopman Operator linearization prior to analysis which can leverage scalable methods to compute reachable states for linear systems.

7. CONCLUSION

Accurate reachability and verification of nonlinear dynamical systems is a grand challenge. Many methods have been proposed for this problem, and this paper has provided a new avenue to verification based on Koopman operator linearization. This process outputs a system of linear dynamics with nonlinear constraints on the initial state set. As far as we are aware, this class of systems has not been considered before for formal set-based reachability analysis. We have proposed and evaluated the first algorithm to solve such reachability problems, along with several optimizations that use overapproximation to reduce analysis time. We have demonstrated that on some systems our method can outperform existing mature reachability tools, and we expect with further research both performance and accuracy can be improved. Unique to our method, we can compute reachable states for black-box systems, as the Koopman linearization process uses data to create linear approximations in the space of observable functions. In the future we plan to further optimize the method, as well as to create a user-friendly tool in order to participate in the annual nonlinear reachability competition Geretti et al. (2020).

REFERENCES

Althoff, M. (2015). An introduction to cora 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*.

Althoff, M. and Krogh, B.H. (2012). Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, 45–54.

Arnon, D.S., Collins, G.E., and McCallum, S. (1984). Cylindrical algebraic decomposition i: The basic algorithm. *SIAM Journal on Computing*, 13(4), 865–877.

Asarin, E., Dang, T., and Girard, A. (2003). Reachability analysis of nonlinear systems using conservative approximation. In *International Workshop on Hybrid Systems: Computation and Control*, 20–35. Springer.

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

Bak, S., Bogomolov, S., Henzinger, T.A., Johnson, T.T., and Prakash, P. (2016). Scalable static hybridization methods for analysis of nonlinear systems. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, 155–164.

Bak, S. and Duggirala, P.S. (2017). Hylaa: A tool for computing simulation-equivalent reachability for linear systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, HSCC '17.

Bak, S., Tran, H.D., and Johnson, T.T. (2019). Numerical verification of affine systems with up to a billion dimensions. In *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, 23–32. ACM, New York, NY, USA.

Benhamou, F. and Granvilliers, L. (2006). Continuous and interval constraints. *Handbook of Constraint Programming*, 569.

Bogomolov, S., Forets, M., Frehse, G., Potomkin, K., and Schilling, C. (2019). JuliaReach: a toolbox for set-based reachability. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 39–44.

Brown, C.W. (2003). Qepcad b: a program for computing with semi-algebraic sets using cads. *ACM SIGSAM Bulletin*, 37(4), 97–108.

Brunton, S.L., Brunton, B.W., Proctor, J.L., and Kutz, J.N. (2016). Koopman Invariant Subspaces and Finite Linear Representations of Nonlinear Dynamical Systems for Control. 11(2), e0150171. doi:10.1371/journal.pone.0150171. URL `https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0150171`.

Budišić, M., Mohr, R., and Mezić, I. (2012). Applied Koopmanism. 22(4), 047510. doi:10.1063/1.4772195. URL `http://aip.scitation.org/doi/10.1063/1.4772195`.

Chen, X., Abraham, E., and Sankaranarayanan, S. (2012). Taylor model flowpipe construction for non-linear hybrid systems. In *2012 IEEE 33rd Real-Time Systems Symposium*, 183–192. IEEE.

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

Dang, T., Maler, O., and Testylier, R. (2010). Accurate hybridization of nonlinear systems. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, 11–20.

Dolzmann, A. and Sturm, T. (1997). Redlog: Computer algebra meets computer logic. *Acm Sigsam Bulletin*, 31(2), 2–9.

Duggirala, P.S., Mitra, S., Viswanathan, M., and Potok, M. (2015). C2e2: A verification tool for stateflow models. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 68–82.

Springer.

Forets, M. and Pouly, A. (2017). Explicit error bounds for carleman linearization. *arXiv preprint arXiv:1711.02552*.

Fränzle, M., Herde, C., Teige, T., Ratschan, S., and Schubert, T. (2006). Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *Journal on Satisfiability, Boolean Modeling and Computation*, 1(3-4), 209–236.

Gao, S., Ganai, M., Ivančić, F., Gupta, A., Sankaranarayanan, S., and Clarke, E.M. (2010). Integrating icp and lra solvers for deciding nonlinear real arithmetic problems. In *Formal Methods in Computer Aided Design*, 81–89. IEEE.

Gao, S., Kong, S., and Clarke, E.M. (2013). dreal: An SMT solver for nonlinear theories over the reals. In *International conference on automated deduction*, 208–214. Springer.

Geretti, L., Sandretto, J.A.D., Althoff, M., Benet, L., Chapoutot, A., Chen, X., Collins, P., Forets, M., Freire, D., Immler, F., et al. (2020). Arch-comp20 category report: Continuous and hybrid systems with nonlinear dynamics. *EPiC Series in Computing*, 74, 49–75.

Gerlach, A.R., Leonard, A., Rogers, J., and Rackauckas, C. (2020). The Koopman Expectation: An Operator Theoretic Method for Efficient Analysis and Optimization of Uncertain Hybrid Dynamical Systems. URL `http://arxiv.org/abs/2008.08737`.

Girard, A. (2005a). Reachability of uncertain linear systems using zonotopes. In M. Morari and L. Thiele (eds.), *Hybrid Systems: Computation and Control*, LNCS. Springer.

Girard, A. (2005b). Reachability of uncertain linear systems using zonotopes. In *International Workshop on Hybrid Systems: Computation and Control*, 291–305. Springer.

Girard, A. and Le Guernic, C. (2008). Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *International Workshop on Hybrid Systems: Computation and Control*, 215–228. Springer.

Goswami, D. and Paley, D.A. (2017). Global bilinearization and controllability of control-affine nonlinear systems: a koopman spectral approach. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 6107–6112. IEEE.

Granvilliers, L. and Benhamou, F. (2006). Algorithm 852: Realpaver: an interval solver using constraint satisfaction techniques. *ACM Transactions on Mathematical Software (TOMS)*, 32(1), 138–156.

Han, Z. and Krogh, B.H. (2006). Reachability analysis of nonlinear systems using trajectory piecewise linearized models. In *2006 American Control Conference*, 6–pp. IEEE.

Jaulin, L., Kieffer, M., and Didrit, O. (2001). *Applied interval analysis: with examples in parameter and state estimation, robust control and robotics*. Springer, London. URL `http://opac.inria.fr/record=b1097265`.

Jones, C.K.R.T. (2001). Whither Applied Nonlinear Dynamics? In B. Engquist and W. Schmid (eds.), *Mathematics Unlimited — 2001 and Beyond*, 631–645. Springer.

Klipp, E., Herwig, R., Kowald, A., Wierling, C., and Lehrach, H. (2005). *Systems biology in practice: concepts, implementation and application*. John Wiley & Sons.

Kong, S., Gao, S., Chen, W., and Clarke, E. (2015). dReach: δ-reachability analysis for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 200–205. Springer.

Koopman, B.O. (1931). Hamiltonian Systems and Transformation in Hilbert Space. 17(5), 315–318. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1076052/`.

Kutz, J.N., Brunton, S.L., Brunton, B.W., and Proctor, J.L. (2016). *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM-Society for Industrial and Applied Mathematics.

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

Matavalam, A.R.R., Vaidya, U., and Ajjarapu, V. (2020). Data-driven approach for uncertainty propagation and reachability analysis in dynamical systems. *arXiv preprint arXiv:2001.07668*.

Mauroy, A., Susuki, Y., and Mezić, I. (2020). Introduction to the Koopman Operator in dynamical systems and control theory. In *The Koopman Operator in Systems and Control*, 3–33. Springer.

Meyers, J.J., Leonard, A.M., Rogers, J.D., and Gerlach, A.R. (2019). Koopman operator approach to optimal control selection under uncertainty. In *2019 American Control Conference (ACC)*, 2964–2971. IEEE.

Narasingam, A. and Kwon, J.S.I. (2019). Koopman Lyapunov-based model predictive control of nonlinear chemical process systems. 65(11), e16743. doi:10.1002/aic.16743. URL `http://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.16743`.

Rackauckas, C. and Nie, Q. (2017). DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. 5(1), 15. doi:10.5334/jors.151. URL `http://openresearchsoftware.metajnl.com/article/10.5334/jors.151/`.

Rand, R. and Holmes, P. (1980). Bifurcation of periodic motions in two weakly coupled van der pol oscillators. *International Journal of Non-Linear Mechanics*, 15(4-5), 387–399.

Ratschan, S. and She, Z. (2007). Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(1), 8–es.

Sankaranarayanan, S. (2012). Change-of-bases abstractions for non-linear systems. *arXiv preprint arXiv:1204.4347*.

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

Sotomayor, J., Mello, L.F., and Braga, D.d.C. (2007). Bifurcation analysis of the watt governor system. *Computational & Applied Mathematics*, 26(1), 19–44.

Tu, J.H., Rowley, C.W., Luchtenburg, D.M., Brunton, S.L., and Kutz, J.N. (2014). On dynamic mode decomposition: Theory and applications. 1(2), 391. doi:10.3934/jcd.2014.1.391. URL `https://www.aimsciences.org/article/doi/10.3934/jcd.2014.1.391`.