Simulation-guided Lyapunov Analysis for Hybrid Dynamical Systems

James Kapinski[*]
Toyota Technical Center

Jyotirmoy V. Deshmukh
Toyota Technical Center

Sriram Sankaranarayanan
Univ. of Colorado

Nikos Aréchiga
Carnegie Mellon University

ABSTRACT

Lyapunov functions are used to prove stability and to obtain performance bounds on system behaviors for nonlinear and hybrid dynamical systems, but discovering Lyapunov functions is a difficult task in general. We present a technique for discovering Lyapunov functions and barrier certificates for nonlinear and hybrid dynamical systems using a search-based approach. Our approach uses concrete executions, such as those obtained through simulation, to formulate a series of linear programming (LP) optimization problems; the solution to each LP creates a candidate Lyapunov function. Intermediate candidates are iteratively improved using a global optimizer guided by the Lie derivative of the candidate Lyapunov function. The analysis is refined using counterexamples from a Satisfiability Modulo Theories (SMT) solver. When no counterexamples are found, the soundness of the analysis is verified using an arithmetic solver. The technique can be applied to a broad class of nonlinear dynamical systems, including hybrid systems and systems with polynomial and even transcendental dynamics. We present several examples illustrating the efficacy of the technique, including two automotive powertrain control examples.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Control theory*; I.6 [**Simulation and Modeling**]: Simulation Output Analysis; G.1.6 [**Numerical Analysis**]: Optimization—*Global optimization*

[*]The authors can be contacted at the following email addresses, respectively: `jim.kapinski@tema.toyota.com`, `jyotirmoy.deshmukh@tema.toyota.com`, `srirams@colorado.edu`, and `narechig@ece.cmu.edu`.

HSCC'14, April 15–17, 2014, Berlin, Germany.
Copyright 2014 ACM 978-1-4503-2732-9/14/04 ...$15.00.
http://dx.doi.org/10.1145/2562059.2562139.

Keywords

Lyapunov functions, Stability, Invariant Sets, Barrier certificates, Simulation

1. INTRODUCTION

Analysis techniques for hybrid systems range from formal techniques that can provide mathematical proofs of correctness to testing-based techniques that rely on a large number of simulations to gain confidence in system correctness. Formal techniques provide better guarantees but are often intractable for large, complex system designs. On the other hand, simulation-based methods work well for systems of arbitrary complexity but cannot be used for verification. In this paper, we describe our effort to bridge this gap by formally addressing prominent analysis problems for hybrid systems while leveraging data obtained from simulations. In particular, we address the problems of proving stability of a system, characterizing performance bounds by computing forward invariant sets, and proving system safety by automatically synthesizing barrier certificates.

It is well-known that each of these problems can be effectively addressed if the designer is able to supply a function v that satisfies the following *Lyapunov conditions* in a given region of interest: (1) v is positive definite, and (2) the Lie derivative of v along the system dynamics is negative (semi-)definite. While the search for a Lyapunov function is widely recognized as a hard problem, sum-of-squares (SoS) optimization-based techniques have been used successfully to obtain Lyapunov functions for systems with polynomial [17, 21] , nonpolynomial [16], and hybrid [18] dynamics. While these techniques have mature tool support [20, 14], they often involve solving problems that are numerically sensitive. For instance, a function computed by such a technique may not strictly satisfy the Lyapunov conditions for all points in the region of interest.

Our key contribution is a novel technique to exploit the results obtained by simulating a system to obtain a *provably correct* and *numerically robust* certificate of stability or safety for the system. The decision to use simulation data and test results is natural in the context of complex dynamical systems, such as those in industrial control systems. In such systems, simulations are often used to validate system designs and increase confidence in system performance. Powerful tools for performing simulation are readily available and are commonly used in, for example, the automotive industry to perform model-based design (e.g., Simulink from the MathWorks [1]).

We now give a brief overview of our technique. We assume that the desired Lyapunov function has a certain parameterized template form: an SoS polynomial of fixed degree. We derive a set of linear constraints on the parameters in the Lyapunov function from concrete execution traces. Given a set of such constraints, the search for a Lyapunov function then reduces to solving a linear program (LP) to obtain a *candidate Lyapunov function*. A key step is then to use a stochastic global optimizer to search the region of interest for states that violate the Lyapunov conditions for the given candidate. The search is guided by a cost function that is based on the Lie derivative of the candidate Lyapunov function; if the minimum cost is less than zero, then the minimizing argument provides a witness (which we call a *counterexample*) that the candidate Lyapunov function is invalid. After the global optimizer obtains counterexamples, the associated linear constraints are included in the LP problem and the candidate Lyapunov function is updated. The process terminates when the global optimizer is unable to identify counterexamples.

As global optimization is not exhaustive, it is imperative to validate any analysis based on the candidate Lyapunov function obtained by the counterexample-guided iterative technique described above. To do so, we use an ensemble of solvers: SMT solvers with nonlinear capabilities such as z3 [6] and dReal [8] and symbolic tools such as quantifier elimination as implemented by the `Reduce` command in Mathematica [24].

Using the candidate Lyapunov function and a suitable solver, we can perform various types of analysis: showing Lyapunov stability or producing a forward invariant set or a barrier certificate. To show Lyapunov stability, we employ one of the solvers to verify the soundness of the candidate Lyapunov function. To produce a forward invariant set, we generate a *sublevelset* of the candidate Lyapunov function S_ℓ of v (i.e., the set $\{\mathbf{x} \mid v(\mathbf{x}) \leq \ell\}$) and then validate that S_ℓ is an invariant using one of the solvers mentioned above.This can be formulated as a single convex optimization problem. Given an initial set of states X_0, and a set of unsafe states U, we can also use it to obtain a *barrier certificate* that includes the initial states while excluding the unsafe set. In this instance, we formulate the barrier certificate as a suitable *levelset* of v that separates X_0 from U.

To demonstrate the efficacy of our techniques, we present examples of dynamical systems, ranging from simple nonlinear systems and systems with transcendental, time-varying dynamics, to switched and nonpolynomial dynamical systems. Our examples include two systems inspired by the automotive engine control domain. The first automotive example is an Air-to-Fuel ratio (A/F) control system with nonlinear, nonpolynomial dynamics, and we construct a forward invariant, which provides performance bounds for the system. The second automotive example is a closed-loop model-predictive control system, modeled as a switched-mode system with piecewise affine dynamics in each of its 69 modes. For this system, we are able to obtain a Lyapunov function for the region of interest (27 modes), thus providing a proof of stability as well as a means to compute performance bounds.

The use of simulations to obtain Lyapunov functions and estimate the maximal region of attraction has been investigated in the past by Topcu et al. [21]. Their approach uses simulation traces to estimate the region of attraction (ROA) for a dynamical system: converting a set of bilinear matrix inequalities (which are computationally expensive to solve) into linear matrix inequalities, which are computationally less expensive. We provide the following extensions to that work: a.) We provide a procedure that uses a guided approach to iteratively improve the quality of the candidate Lyapunov functions, and b.) Our technique is not restricted to the class of systems with polynomial dynamics.

Related work was proposed by Gupta et al. [9] for program analysis. Their approach uses traces of discrete programs to compute termination proofs in the form of ranking functions and linear invariants.

The layout of the paper is as follows: We review the theoretical background in Sec. 2. In Sec. 3, we present our technique for generating candidate Lyapunov functions and for iteratively improving the candidates. In Sec. 4, we explain how SMT solvers can be used to verify the soundness of the candidates and we also explain how counterexamples can be used to further improve the candidate Lyapunov functions. We demonstrate our technique on interesting nonlinear and hybrid examples in Sec. 5, and finally conclude with a discussion of future work in Sec. 6.

2. PRELIMINARIES

Continuous-time switched-mode systems (CSMS). A CSMS is a dynamical system described by a set of ODEs:

$$\dot{\mathbf{x}}(t) \quad = \quad f_i(\mathbf{x}(t)),\ \forall \mathbf{x}(t) \in X_i, \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state of the system at time t, and X_i, $i = 1, \ldots, I$ is a partition of the state space $\mathcal{X} \subseteq \mathbb{R}^n$. Each f_i is a nonlinear vector field that is Lipschitz-continuous. We abuse notation and take $\mathbf{x} \in \mathbb{R}^n$ to be a singleton and $\mathbf{x}(\cdot)$ to be a differentiable function $\mathbf{x} : \mathbb{R} \to \mathbb{R}^n$.

Given an initial condition $\mathbf{x}_0 \in \mathbb{R}^n$, a *trace* of a CSMS is a function $\mathbf{x}(t) : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, where $\mathbf{x}(0) = \mathbf{x}_0$ and (1) holds for all $t \in \mathbb{R}_{\geq 0}$.

We assume that the system has no Zeno behavior, that is, we assume that there are finite switches in finite time. Given an initial condition $\mathbf{x}(0)$, a unique solution $\mathbf{x}(t)$ to (1) exists.

We define $\phi(t)$ as a discrete-time *trace* of system (1). That is, $\phi(t)$ is a function $\phi : T \to \mathbb{R}^n$, where $T = \{t_1, \ldots, t_N\} \subset \mathbb{R}$, where $N \in \mathbb{Z}_{>0}$, and there exists an $\mathbf{x}(t)$, such that for each $1 \leq j < N$, $\phi(t_j) = \mathbf{x}(t_j)$ and (1) holds for all $t \in [t_j, t_{j+1}]$.

Definition 2.1 (Equilibrium Point). *A state $\mathbf{x}^*$ is called an* equilibrium point *of a CSMS if a trace of the system with $\mathbf{x}(0) = \mathbf{x}^*$ is given by $\mathbf{x}(t) = \mathbf{x}^*$ for all time t.*

In standard fashion, we use $\|\mathbf{x}\|$ to denote the Euclidean norm $\sqrt{\mathbf{x}^T\mathbf{x}}$, i.e. the distance of a point in $\mathbb{R}^n$ from the origin. A^T denotes the transpose of the matrix A.

Definition 2.2 (Forward Invariant Set). *A set of states $I \subseteq \mathbb{R}^n$ of a CSMS is called a* forward invariant set *if for all $\mathbf{x}(0) \in I$ and for all $t \geq 0$, $\mathbf{x}(t) \in I$.*

The goal of Lyapunov's direct method is to show stability of a system by identifying a *Lyapunov function*.

Definition 2.3 (Lyapunov function). *Given a* csms*, a function $v : \mathcal{X} \to \mathbb{R}_{\geq 0}$ is called a Lyapunov function if the following holds for all i [13]:*

$$\forall \mathbf{x} \in X_i \backslash \mathbf{0} : v(\mathbf{x}) \ > 0, \ v(\mathbf{0}) = 0 \tag{2}$$

$$\forall \mathbf{x} \in X_i : \nabla v(\mathbf{x}(t))^T \cdot f_i(\mathbf{x}(t)) \ \leq 0. \tag{3}$$

The existence of a Lyapunov function, as specified above, guarantees non-asymptotic stability. For switched systems, such a Lyapunov function can take the form of a single, continuous and differentiable *common Lyapunov function*. When a common Lyapunov function cannot be found, it is sometimes possible to define a *piecewise Lyapunov function*, where a unique Lyapunov-like function is defined for each mode, with additional conditions on the behavior of the Lyapunov-like functions at the switching instances [4].

For certain classes of systems, such as stable linear time-invariant systems, techniques exist to identify Lyapunov functions and invariant sets. For continuous systems with dynamics given by polynomial equations, relaxations based on SoS techniques exist that allow Lyapunov functions and invariant sets to be identified for certain cases. For general nonlinear systems, however, no such techniques exist. For hybrid systems with linear continuous dynamics, several techniques exist for identifying Lyapunov functions, such as LMI solutions for simultaneous Lyapunov functions and piecewise quadratic Lyapunov functions, but these techniques are not complete (i.e., they can fail to identify a Lyapunov function even when one exists).

The *sublevelset* of a Lyapunov function $v(\mathbf{x})$ is the set $\{\mathbf{x} \mid v(\mathbf{x}) \leq \ell\}$. It is well known that sublevelsets of Lyapunov functions are forward invariant sets. While forward invariant sets can be used to characterize performance bounds of a given csms, the closely related notion of a *barrier certificate* can be used to verify safety of a given system.

Definition 2.4 (Barrier Certificate). *Given a* csms*, an initial set $X_0 \subset \mathcal{X}$, and an unsafe set $U \subset \mathcal{X}$ such that $U \cap X_0$ is empty, a function $B : \mathcal{X} \to \mathbb{R}$ is called a barrier certificate if it satisfies the following conditions for all i:*

$$B(\mathbf{x}) \leq 0 \qquad \forall \mathbf{x} \in X_0 \tag{4}$$

$$B(\mathbf{x}) > 0 \qquad \forall \mathbf{x} \in U \tag{5}$$

$$\nabla B(\mathbf{x})^T \cdot f_i(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in X_i \ s.t. \ B(\mathbf{x}) = 0. \tag{6}$$

Note that a Lyapunov function can be used to construct a barrier certificate as follows. Given an $l \in \mathbb{R}_{>0}$, if we select $B(\mathbf{x}) = v(\mathbf{x}) - l$ and define $S_\ell = \{\mathbf{x} \mid B(\mathbf{x}) = 0\}$, then $B(\mathbf{x})$ satisfies (6) if $S_\ell \subseteq \mathcal{X}$. As long as (4) and (5) hold, then $B(\mathbf{x})$ is a barrier certificate.

Discrete-time Switched-Mode Systems (dsms). We also consider discrete-time switched-mode systems (DSMS), where $\mathbf{x}[k+1] = f_i(\mathbf{x}[k])$. When discussing a discrete-time context, we use $\hat{\mathbf{x}} = f_i(\mathbf{x})$. The notions defined above for csms carry over for dsms. For example, an invariant set for a dsms is defined as a set I such that for all $\mathbf{x}[0] \in I$, and for all $k \in \mathbb{Z}_{>0}$, $\mathbf{x}[k] \in I$. Similarly, the Lyapunov conditions for a dsms are:

$$\forall \mathbf{x} \in X_i \backslash \mathbf{0} : v(\mathbf{x}) \ > 0, \ v(\mathbf{0}) = 0 \tag{7}$$

$$\forall \mathbf{x} \in X_i : v(\mathbf{x}) - v(\hat{\mathbf{x}}) \ > 0. \tag{8}$$

3. ITERATIVELY IMPROVED LYAPUNOV CANDIDATES

We present a technique to compute candidate Lyapunov functions for switched-mode systems using simulation traces. The technique relies on a *falsification* tool to produce a series of successively improved candidate functions. The falsification tool is a global optimizer that is guided by direction information provided by the intermediate Lyapunov candidates. The falsifier adds constraints to a series of LPs by selecting initial conditions for simulation traces. We go on to describe how to use the resulting candidates and automated reasoning tools to: a.) show that the candidates are Lyapunov functions, or b.) produce invariant sets and barrier certificates.

Topcu et al. [21] employ simulation traces to formulate a convex optimization problem to compute candidate Lyapunov functions and invariant sets. The goal for their work is to characterize a region of attraction of a given continuous-time dynamical system. In this paper, we go further and provide a technique to iteratively improve the candidates using a stochastic global optimization-based approach that is guided by a cost function based on the Lie derivative of the candidate Lyapunov function.

3.1 Constructing Candidates

In the following, we assume the system has a stable equilibrium point, which is, without loss of generality, at the origin. Let Φ_i be a collection of p traces within mode i. We assume we can obtain discrete-time traces, $\phi_i(t)$.

We obtain candidates for functions v that satisfy conditions (2) and (3) by using the following alternate conditions:

1. We restrict each v to the class of polynomials of some fixed degree;
2. We require that a necessary condition for constraints (2) and (3) hold for every trace in Φ_i.

We impose condition (1) by requiring $v(\mathbf{x}) = \mathbf{z}^T \mathbf{P} \mathbf{z}$, where $\mathbf{z}$ is some vector of m monomials in $\mathbf{x}$ and $\mathbf{P} \in \mathbb{R}^{m \times m}$ is symmetric. We impose condition (2) by requiring the following:

$$v(\phi(t_j)) \ > \ 0 \tag{9}$$

$$v(\phi(t_j)) - v(\phi(t_{i+j})) - \gamma \|\phi(t_j)\|^2 \ > \ 0 \tag{10}$$

$$\gamma \ > \ \epsilon, \tag{11}$$

for all $\phi_i \in \Phi$, $j \in \{1, \ldots, N-1\}$. The parameter $\epsilon \in \mathbb{R}_{\geq 0}$ is a fixed positive value. Note that (9) is a series of necessary conditions for constraint (2) to hold. For (3) to hold, it must be that $v(\phi(t_j)) - v(\phi(t_{i+j})) > 0$; constraint (10) is stronger in that it bounds $v(\phi(t_j)) - v(\phi(t_{i+j}))$ away from zero. We call any v that satisfies (9) through (11) a *Lyapunov candidate*. To distinguish between a Lyapunov function and a Lyapunov candidate, we use the term *proper Lyapunov function* to refer to a Lyapunov function.

Remark 3.1. *We reiterate that constraints (9) and (10) impose necessary* but not sufficient *conditions for (2) and (3) to hold. Therefore, to enforce (2) and (3), we have to perform a formal validation of our final Lyapunov candidate, as discussed in Section 4.1. In practice, we find that adding more simulation traces, and thus more constraints (9) and (10) improves the likelihood that a Lyapunov candidate also satisfies (2) and (3).*

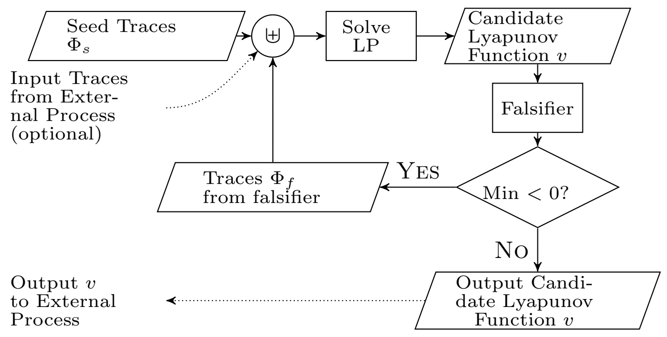
Seed Traces Φ_s
$\uplus$
Solve LP
Candidate Lyapunov Function v
Input Traces from External Process (optional)
Falsifier
Traces Φ_f from falsifier
YES
Min < 0?
NO
Output v to External Process
Output Candidate Lyapunov Function v

Figure 1: Procedure to iteratively improve candidate Lyapunov functions for system (1).

Note that (9) and (10) are linear constraints since they are linear in the matrix variable $\mathbf{P}$. Conditions (9) through (11) represent a set of linear constraints for which a feasible solution can be found using a standard LP solver.

Note that (2) could be imposed directly by replacing the constraint (9) with the alternative linear matrix inequality (LMI) constraint $\mathbf{P} \succ 0$, but this would require solving a potentially large scale semidefinite programming (SDP) problem (if, for example, there are a large number of simulation traces). Our experience indicates that SDP solvers are not as mature as LP solvers and are more prone to numerical difficulties. Thus, we elect to use the linear constraint (9).

REMARK 3.2. *Given a Lyapunov template $v(\mathbf{x}) = \mathbf{z}^T\mathbf{P}\mathbf{z}$, the set of feasible solutions that satisfy (2) and (3) is convex in the decision variable $\mathbf{P}$. Adding constraints (9) to (11) bounds the feasible set with linear constraints. Thus the feasible set lies on the interior of the linear constraints. If an interior point algorithm is used to provide a feasible solution to the LP, the solution returned is the* analytic center *of the LP problem [23]. We rely on this, since our intuition is that for many problems, the analytic center of the LP problem coincides with the interior of the feasible set for (2) and (3).*

3.2 Iterative Candidate Improvement

We present procedures to automatically select execution traces for system (1) to iteratively improve the quality of a series of Lyapunov candidates. We equate the quality of a Lyapunov candidate with the amount of time it takes a search procedure to identify a point that violates constraint (10). We rely on a falsification tool to automatically identify points within some domain $\mathcal{D} \subseteq \mathcal{X}$ that violate (10). Our falsification tool is a global optimizer that minimizes the LHS of (10) to find examples of points that violate the constraint.

The inputs to the procedures are:

– A parameter $\beta \in \mathbb{R}_{\geq 0}$ representing the size of an open ball centered around the origin, $B_\beta = \{\mathbf{x} \mid \|\mathbf{x}\| < \beta\}$, that will *not* be included in the analysis;

– A domain of interest, $\mathcal{D}$;

– A time step $T \in \mathbb{R}_{>0}$;

– A bound on the degree of the Lyapunov candidate function;

– A description of system (1) and a mechanism for generating concrete execution traces, such as simulations.

Also, we provide an initial collection of traces, Φ_s, to seed the procedure. The step-by-step process of the algorithm to construct candidate Lyapunov functions is shown in Figure 1. We elaborate on important steps in the procedure below.

Solve LP. In this stage, we obtain a Lyapunov candidate by solving the feasibility problem given by (9) through (11), based on the the set of all simulation traces explored by a.) the manually selected set of seed traces and b.) the falsification tool. If the LP is successfully solved, then we move to the Falsification Stage. If the LP is deemed infeasible, then we halt and report that the technique failed to find a Lyapunov candidate; this could occur due to a.) no Lyapunov function of the given template form exists or b.) numerical problems.

Falsifier. In this step we use a non-convex, global optimizer, which we call a *falsifier*, to search for a simulation trace that violates the Lyapunov conditions for the candidate Lyapunov function. The optimization problem is given by:

$$J^* = \min_{\phi(t_0)\in\mathcal{D}} \left(\min_{\substack{i\in\\ \{1,\ldots,N-1\}}} v(\phi(t_i)) - v(\phi(t_{i+1})) - \gamma\|\phi(t_i)\|^2 \right). \tag{12}$$

If $J^* < 0$, then the minimizing trace, ϕ_f, demonstrates that the Lyapunov candidate, v, does not satisfy condition (10). We call such a trace a *counterexample*. Note that the cost function in (12) is based on an estimate of the Lie derivative (i.e., $v(\phi(t_i)) - v(\phi(t_{i+1}))$ is proportional to the Lie derivative). If counterexamples Φ_f are found, then we add the linear constraints corresponding to the counterexamples in Φ_f to the set of LP constraints and return to the Solve LP stage. If no counterexample is found, we halt and return the candidate Lyapunov function.

A prototype of the technique has been implemented in the MATLAB programming environment, using the freely available SeDuMi and YALMIP optimization packages [15, 20, 14]. Our implementation of the falsification tool uses a Nelder-Mead algorithm for the global optimizer.

REMARK 3.3. *Note that our search for a Lyapunov function can also be used for black-box systems, where we have no analytic representation of system dynamics because the system is either proprietary or is modeled in a graphical language with obscure semantics, such as Simulink [1]. In such a scenario, the Lyapunov candidate we obtain cannot be formally vetted, but can be used to give semi-formal guarantees. For some gray-box systems, where we have only limited knowledge of the system dynamics, such as the Lipschitz constant for the dynamics and the maximum absolute value of the vector field within the region of interest, we can give formal guarantees by using a dense sampling of the region of interest as the set of initial states and a small enough simulation time-step. We omit these results for brevity.*

4. VERIFICATION WITH SOLVERS

In this section, we describe how we use a variety of solvers to formally validate the results of the simulation-guided Lyapunov analysis techniques.

4.1 Formal validation

Verifying Lyapunov conditions. Let the predicate $R(\mathbf{x}) < 0$ be true when $\mathbf{x}$ is in the region of interest X. We formulate the query for checking positive definiteness conditions (2,7) within a given region of interest for a CSMS or DSMS as follows:

$$\exists \mathbf{x} : (\mathbf{x} \neq \mathbf{0}) \wedge (R(\mathbf{x}) < 0) \wedge (v(\mathbf{x}) \leq 0). \tag{13}$$

If the above query is unsatisfiable, it proves positive definiteness of v. For checking if the Lie derivative $\dot{v}$ is negative definite (in the region of interest), we formulate the following queries for CSMS (14) and DSMS (15) respectively:

$$\exists \mathbf{x} : (\mathbf{x} \neq \mathbf{0}) \wedge (R(\mathbf{x}) < 0) \wedge (\nabla v^T \cdot f_q(\mathbf{x}) > 0) \tag{14}$$

$$\exists \mathbf{x} : (\mathbf{x} \neq \mathbf{0}) \wedge (R(\mathbf{x}) < 0) \wedge (v(\mathbf{x}) - v(f_q(\mathbf{x})) < 0). \tag{15}$$

If such a query is unsatisfiable, then it proves that for all points within the region of interest, the Lie derivative is nonincreasing.

Verifying Barrier certificate conditions. Recall that we use a barrier function B of the form $v(\mathbf{x}) - \ell$. Let the predicate $I(\mathbf{x}) < 0$ be true when $\mathbf{x} \in X_0$ (the set of initial states). Similarly, let the predicate $U(\mathbf{x}) < 0$ be true when $\mathbf{x} \in U$ (the set of unsafe states). We use ϵ to denote a small positive real constant (e.g., 0.00001). The unsatisfiability of each of the first three queries below respectively establishes the barrier conditions (4-6):

$$\exists \mathbf{x} : (I(\mathbf{x}) < 0) \wedge (v(\mathbf{x}) - \ell > 0) \tag{16}$$

$$\exists \mathbf{x} : (U(\mathbf{x}) < 0) \wedge (v(\mathbf{x}) - \ell < 0) \tag{17}$$

$$\exists \mathbf{x} : (v(\mathbf{x}) - \ell = 0) \wedge (\nabla v^T \cdot f_q(\mathbf{x}) > -\epsilon). \tag{18}$$

While the above treatment may seem like an obvious translation of the Lyapunov conditions or the barrier certificate conditions, we wish to point out that each of these queries is essentially a satisfiability query formulated in a suitable theory. If the candidate Lyapunov function that we obtain is a polynomial (or SoS) expression, and if the system dynamics are also polynomial, then each of these queries is a sentence in the decidable theory of real closed fields. If the system dynamics are nonpolynomial, then the query may belong to an undecidable theory. Nevertheless, advanced nonlinear solver technologies can often provide answers for these cases. We now briefly discuss the solvers that we use and their underlying technical principle.

4.2 Solver Engines

Symbolic solvers. The most popular algorithm for deciding sentences over algebraic expressions uses Partial Cylindrical Algebraic Decomposition (PCAD) [5]. A number of tools either directly implement PCAD or CAD based algorithms, or use them for specific sub-tasks. Examples include Mathematica [24], the Conflict-driven Clause Learning style search used by z3 [6], and QEPCAD. While algebra-based solvers seem to perform well with a single polynomial inequality, in our experience these solvers do not scale when faced with a conjunction of polynomial inequalities (they either exceed a generous time-out that we specify, or run out of memory). Another interesting solver in this space is MetiTarski [2]. This is a resolution-based theorem prover modified to call a decision procedure for the theory of real closed fields. Nonpolynomial functions are approximated by upper and lower bounds that are rational functions derived from Taylor expansion representations.

Solvers based on optimization and numeric techniques. SoS techniques have been employed to synthesize and to check the validity of Lyapunov functions for dynamical systems with polynomial dynamics. Basically, an SoS problem is formulated to show that the negative of the Lie derivative is in the set of SoS polynomials (thus, guaranteeing that the Lie derivative is always decreasing). This can be accomplished for polynomial systems, as the Lie derivative is polynomial in the state variables [17, 21]. Further, for some dynamical systems with nonpolynomial dynamics, variable transformations can be performed, which allow the test of the Lie derivative to again be posed as an SoS problem [16]. For some hybrid systems, the test of the validity of a candidate Lyapunov function may be performed, as in [18].

SoS techniques address an important class of dynamical systems, but it should be noted that even if the Lie derivative is negative definite, an SoS certificate is not guaranteed to exist. Further, even if an SoS certificate for the Lie derivative exists, an SDP solver will sometimes fail to generate the desired result. This is due to the lack of maturity in SDP solvers, which can often fail due to numerical problems.

Solvers based on interval methods. Interval constraint propagation (ICP) is a technique for contracting interval domains associated with a set of variables without removing any value that is consistent with a set of constraints. When combined with a branch and bound algorithm, ICP can be used to obtain quick but approximate results for satisfiability of nonlinear constraints. For example, dReal [8] and iSat [7] are such solvers, and we focus on using dReal for our validation problems. dReal supports various nonlinear elementary functions in the framework of δ-complete decision procedures, and returns "unsat" or "δ-sat" for a given query, where δ is a precision value specified by the user. When the answer is "unsat", dReal produces a proof of unsatisfiability; when it returns "δ-sat", it gives an interval of size δ, which contains points that may possibly satisfy the query. We remark that when using dReal, for some queries, it often helps if we add additional constraints bounding the free variables in the queries to intervals. This often produces less conservative results.

4.3 Solver-guided Improvement and Validation

We use the above-mentioned solvers in a procedure to verify the soundness of results that are based on the Lyapunov candidates produced by the procedure in Figure 1. The procedure utilizes counterexamples from the solver to iteratively improve the quality of the Lyapunov candidate functions. The result is a proof of soundness for either a.) a Lyapunov function b.) a forward invariant set or c.) a barrier certificate.

Figure 2 illustrates a process that incorporates the formal validation techniques discussed in Sec. 4.2 with the iterative Lyapunov candidate improvement procedure shown in Figure 1. The following describes the important aspects of the procedure.

Formulate Solver Query. This operation creates one of the queries described in Section 4.1 to validate the result of the Lyapunov candidate analysis produced by the proce-

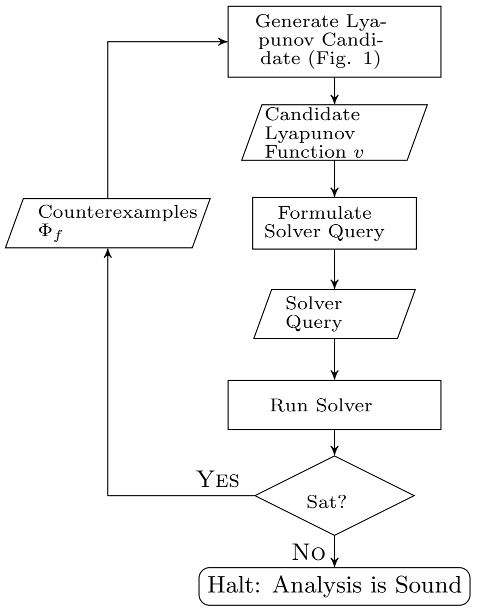

Figure 2: Incorporating solver technologies to verify soundness of the Lyapunov anlyses.

dure described in Section 3. When the desired output is a proper Lyapunov function, the Lyapunov candidate v may be tested directly using the procedure described in Section 4.1. When the desired output is a forward invariant set or a barrier certificate, a candidate certificate must first be generated. This is done by selecting a levelset size of the Lyapunov candidate v (i.e., a size l such that the sublevelset is given by $\{\mathbf{x}|v(\mathbf{x}) \leq l\}$). An appropriate levelset size may be computed by maximizing the levelset size such that the levelset remains within $\mathcal{D}$. This can be formulated as a convex optimization problem that can be solved efficiently using an SDP solver. The result is a candidate barrier certificate, which can be validated using the technique described in Section 4.1.

Run Solver. This operation applies one of the technologies described in Section 4.2 to the generated query. If the result is that the query is unsatisfiable, then the Lyapunov analysis (based on the candidate Lyapunov function and the construction produced by the Formulate Solver Query operation) is sound, and the procedure may halt. If the result is that the query is satisfiable, then a counterexample may be used to refine the Lyapunov candidate on which the query was based. Note that all of the technologies described in Section 4.2 produce some form of counterexamples except for the SoS-based techniques; for this case, some other method of refinement must be selected (e.g., selecting a different Lyapunov function template or adding several new simulation traces randomly).

5. EXAMPLE CASE STUDIES

We present several examples involving nonlinear and hybrid dynamical systems. In some cases, the analysis task is to produce a Lyapunov function within some designated domain; in other examples, the analysis task is to produce a forward invariant set. Our examples include systems with ODEs that are polynomial, transcendental, and switched. For systems with ODEs, traces are produced by the `ode45` numerical integration algorithm provided in MATLAB®.

A summary of the results for the examples is given in Table 2. For each example, the table lists the following: the example name, the number of continuous state variables, the computation time taken for the procedure in Figure 1 to produce a candidate Lyapunov function, the number of simulation points explored by the falsification tool, the computation time required by the arithmetic solver, and the arithmetic solver used to verify the result.

5.1 Example 1: Normalized Pendulum

Consider a standard pendulum system with normalized parameters:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\sin(x_1) - x_2 \end{bmatrix}.$$

Here, x_1 represents angular position and x_2 angular velocity. The system has only one mode of operation. The continuous dynamics contain a transcendental function, which we note is difficult for most other techniques to handle. This system is guaranteed to be stable, as it is a representation of a passive physical system with damping (i.e., the system will tend to a zero-energy state over time).

The task for this example is to identify a Lyapunov function for the system that is valid within the domain $\mathcal{D} = \{\mathbf{x}|\mathbf{x}^T\mathbf{x} \leq 1\}$ and also to identify a forward invariant set. We select $\mathbf{z} = \mathbf{x}$, that is, the Lyapunov candidates are quadratic.

The procedure from Figure 1 produces the candidate Lyapunov function $v(\mathbf{x}) = \mathbf{x}^T\mathbf{P}\mathbf{x}$, where, after rounding:

$$\mathbf{P} = \begin{bmatrix} 100.0 & 24.0 \\ 24.0 & 92.0 \end{bmatrix}.$$

The procedure takes 74.22 seconds. A total of 300 simulation traces were explored by the falsification tool, each with 10 time steps of 0.1 seconds each.

A query of the form given by (13) and (14) was posed to the Mathematica arithmetic solver and was able to prove that the query is unsatisfiable in 7.72 seconds, thus proving that the above candidate Lyapunov function is a proper Lyapunov function.

A convex optimization provides the size of the largest levelset of the Lyapunov function that is contained within the domain. The resulting levelset size was $l = 71.51$, where the invariant set is given by $\{\mathbf{x}|v(\mathbf{x}) \leq l\}$. The SDP solver returns this result in 1.36 seconds. Figure 3 illustrates the results. Simulation traces explored by the falsification tool appear as dotted lines, with the associated initial conditions marked with an asterisk. The dashed line indicates the domain for the example (the unit ball). The dash-dotted line represents the invariant set.

5.2 Example 2: Constrained Pendulum

Consider the following constrained pendulum example [22]:

$$\begin{aligned} \dot{x}_1 &= \begin{cases} \frac{1}{2}x_2 & x_1 \geq -\frac{\pi}{18} \\ x_2 & \text{otherwise} \end{cases} \\ \dot{x}_2 &= -g\sin(x_1) - x_2, \end{aligned}$$

where x_1 is the angular position, x_2 is the angular velocity, and $g = 9.8$ is the acceleration due to gravity. The behavior is similar to the previous example, except a pin constrains

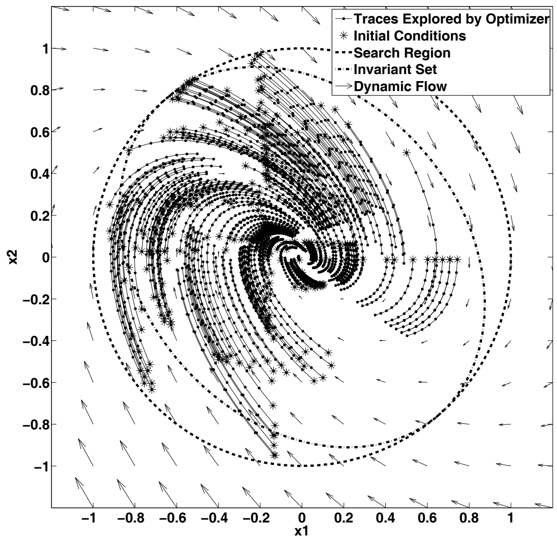

Figure 3: Optimization results of analysis for the Normalized Pendulum example.

the swing of the pendulum. Thus, the system has two modes of operation. If $x_1 \geq \frac{\pi}{18}$, the pendulum is unconstrained by the pin, and the effective length of the pendulum is 2.0 m. When $x_1 < \frac{\pi}{18}$, the pin constrains the pendulum swing, and effective length of the pendulum is 1.0 m.

As in the previous example, the system is guaranteed to be stable as it is a physical system with damping. For this example, we consider the task of constructing a forward invariant set for the system. To highlight the feature that we can supply different templates during the search for candidate Lyapunov functions, in this example, we specify a piecewise Lyapunov function template. In [12], Chapter 4.4, the author proposes a way to frame a piecewise Lyapunov function for piecewise linear systems that is continuous across the switching boundary. We can extend this idea to general switched systems; here, we show how it can be applied to this example. Consider a CSMS with two modes. The basic idea is to search for a Lyapunov function that has the following form:

$$v(\mathbf{x}) = \mathbf{z}_1^T Q \mathbf{z}_1 + \mathbf{z}_2^T P_i \mathbf{z}_2 \cdot (h(\mathbf{x}))^2, \tag{19}$$

where $i \in \{1, 2\}$. Here, $\mathbf{z}_1$ and $\mathbf{z}_2$ are monomial vectors, where the degree of $\mathbf{z}_1$ is higher than the degree of $\mathbf{z}_2$ [1].

The expression $h(\mathbf{x})$ is a function such that $h(\mathbf{x}) = 0$ specifies the switching surface separating the two modes. For this example, $h(\mathbf{x}) = x_1 + \frac{\pi}{18}$. Observe that on the switching surface the right summand evaluates to 0, and hence the Lyapunov function becomes continuous at the switching boundary.

The search procedure returns a candidate Lyapunov candidate after 2,308.54 seconds. The resulting Q and P_i matrices are omitted for brevity. After the candidate Lyapunov function is returned, a search returns a levelset size, which is used to define a candidate forward invariant set. dReal returns a verification result for the forward invariant set in 0.084 seconds.

[1]This constraint is necessary to prevent the terms in h from making the Lyapunov candidate trivially positive.

5.3 Example 3: Damped Mathieu System

Consider the damped Mathieu system (page 315 in [10]):

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_2 - (2 + \sin(t))x_1 \end{bmatrix}.$$

The task for this example is to identify a Lyapunov function within the domain given by the unit ball centered at the origin. Note that the Mathieu dynamics are time varying. That is, $\dot{\mathbf{x}} = f(t, \mathbf{x})$. To construct Lyapunov candidates for this system, we use a variation on Lyapunov's direct method. We invoke Barbalat's Lemma, as in [19] (page 125). This requires that a.) $\dot{v} < 0$ for all $t \geq 0$ and b.) the second derivative of v be uniformly continuous in time. We apply condition (10) over simulations of duration 6 seconds (the intuition being that 6 seconds is representative of the dynamics for all $t > 0$). Also, it can be shown that the second derivative of v is continuous.

Again, we select a quadratic form for the candidate Lyapunov function. The procedure from Figure 1 produces the candidate Lyapunov function $v(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$, where, after rounding:

$$\mathbf{P} = \begin{bmatrix} 98.0 & 24.0 \\ 24.0 & 55.0 \end{bmatrix}.$$

The above result was returned after 216.61 seconds and a total of 200 simulation traces were explored by the falsification tool, each with 60 time steps of 0.1 seconds each.

A query of the form given by (13) and (14) was posed to the dReal SMT solver and was able to prove that the query is unsatisfiable in 0.044 seconds, thus proving that the above candidate Lyapunov function is a proper Lyapunov function.

5.4 Example 4: Switched-Mode System

Consider the following CSMS system, which is a modified version of an example from Johansson[12]:

$$\dot{\mathbf{x}} = \begin{cases} \begin{bmatrix} -0.1 & 1.0 \\ -10 & -0.1 \end{bmatrix} \mathbf{x} & \begin{array}{c} (x_1 \geq 0 \wedge x_2 \geq g(x_1)) \vee \\ (x_1 \leq 0 \wedge x_2 \leq g(x_1)) \end{array} \\ \\ \begin{bmatrix} -0.1 & 10 \\ -1 & -0.1 \end{bmatrix} \mathbf{x} & \begin{array}{c} (x_1 < 0 \wedge x_2 > g(x_1)) \vee \\ (x_1 > 0 \wedge x_2 < g(x_1)) \end{array} \end{cases}, \tag{20}$$

where $g(x_1) = 0.1e^{x_1} - 0.1$. The task for this example is to identify an invariant set within the unit ball.

Invariant sets can be obtained from Lyapunov functions for switched-mode systems, and there are techniques that attempt to obtain a common Lyapunov function by solving the convex optimization feasibility problem $\mathbf{P} \succ 0$, $A_i^T \mathbf{P} + \mathbf{P} A_i \prec 0$ for every mode i. Note, however, that no solution (i.e., a common Lyapunov function) for the continuous dynamics in this system exists, as shown in [12]. While techniques such as LMI solutions based on the so called *S-procedure* [3] could succeed for the original example in [12], these techniques fail to capture the transcendental switching surface in this example. As we show, our technique can compute an invariant set for this system, indicating that our technique offers a viable alternative when other techniques fail.

The falsification tool uses traces of the system with a step size of 0.02 seconds, out to 1.0 seconds. We select a quadratic Lyapunov function template. The falsification tool produces

$$\mathbf{P} = \begin{bmatrix} 11.0 & 1.0 \\ 1.0 & 100.0 \end{bmatrix}$$

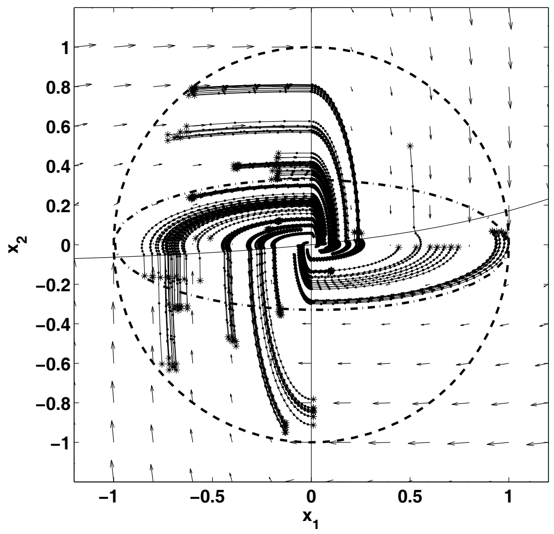

Figure 4: Optimization results of analysis for the Switched-Mode example.

in 170.75 seconds. A total of 196 simulation traces were explored by the falsification tool. The SDP solver returned a levelset size of $l = 10.8$ in 0.79 seconds. Mathematica is able to verify the candidate invariant set in 1.476 seconds.

Figure 4 illustrates the results. The arrows indicate the direction of the vector field. The dashed surface represents the domain of interest (the unit ball). The solid traces are the simulations that were used to compute a Lyapunov candidate function. The vertical axis and the curved surface passing through the origin define the switching boundaries.

5.5 Example 5: Powertrain Control System

We consider a fuel controller for an automotive application, and evaluate its ability to maintain the air-fuel (A/F) ratio within a given range of an optimal value. Environmental concerns and government legislation require that the rate of emissions (e.g., hydrocarbons, carbon monoxide, and nitrogen oxides) be minimized; control of automobile engine A/F ratio is crucial to minimize emissions. Ideal A/F levels are given by the *stoichiometric* value; we present an A/F control system model whose purpose is to maintain the A/F ratio to within 10% of the stoichiometric value when running under normal operating conditions.

The experiment that we model involves an engine connected to a dynamometer, which is a device that can control the speed of the engine and measure the output torque. For our experiment, the dynamometer maintains the engine at a constant rotational velocity, as the engine is tested.

The dynamical system we consider is a CSMS representing the parallel composition of a plant subsystem with a controller subsystem. This system has four state variables: two associated with the plant and two associated the controller. The two states associated with the plant represent the manifold pressure p and the normalized A/F ratio r (this is the ratio of the actual air-fuel ratio to the stoichiometric value 14.7). The controller implements a feedforward open-loop estimator to observe the state p of the plant; the output of the estimator is the state p_{est}. It also implements a feedback PI control law, and the state i represents the internal state of the integrator. We present the system dynamics in Fig. 5 and then tabulate the model parameters in Table 1.

Table 1: Model Parameters for the PTC Example.

Parameter	Value
c_1	0.41328
c_2	200.0
c_3	-0.366
c_4	0.08979
c_5	-0.0337
c_6	0.0001
c_7	2.821
c_8	-0.05231
c_9	0.10299
c_{10}	-0.00063
c_{11}	1.0
c_{12}	14.7
c_{13}	0.9
c_{14}	0.4
c_{15}	0.4
c_{16}	1.0
$\hat{u_1}$	23.0829

We translate the system so that the origin coincides with the equilibrium point $p \approx 0.8987$, $r = 1.0$, $p_{est} \approx 1.077$, $i \approx 0.0$ and call the translated variables $\hat{p}$, $\hat{r}$, $\hat{p}_{est}$, and, $\hat{i}$, respectively. Using $\mathbf{x} = [\hat{p}\ \hat{r}\ \hat{p}_{est}\ \hat{i}]^T$, we define the following unsafe set:

$$U = \{\mathbf{x} |\ \|\hat{r}\| > 0.1\},$$

which corresponds to unacceptable A/F ratio values. The requirement is that the system should never enter U, given an initial condition in X_0, where

$$X_0 = \{\mathbf{x} |\ \|\mathbf{x}\| \leq 0.02\}.$$

We apply our technique to identify a barrier certificate for this system to verify that this system satisfies the requirement. We select a domain of $\mathcal{D} = \{\mathbf{x} |\ \|\mathbf{x}\| \leq 0.1\}$, which we note does not intersect with U. We use a candidate Lyapunov function of the form $v(\mathbf{x}) = \mathbf{z}^T \mathbf{P} \mathbf{z}$, where $\mathbf{z}$ is a vector of all monomials of degree ≤ 2 of the state variables $\hat{p}$, $\hat{r}$, $\hat{p}_{est}$ and $\hat{i}$. Note that $\mathbf{z}$ thus contains 14 monomials, and the $\mathbf{P}$ that we wish to find is a 14x14 matrix. The procedure from Figure 1 produces a candidate Lyapunov function of the desired form. We omit the resulting $\mathbf{P}$ matrix for brevity but note that the minimum and maximum eigenvalues are approximately 7.6 and 489.0, respectively.

The computation takes $1,413.73$ seconds. An appropriate levelset size of the Lyapunov candidate was found to be 0.07. The candidate barrier certificate is thus $B(\mathbf{x}) = \mathbf{z}^T \mathbf{P} \mathbf{z} - l$, where $l = 0.07$. The dReal solver is used to prove that B is a barrier certificate using a query similar to (16) through (18). dReal is able to prove that the query is unsatisfiable in $1,157.42$ seconds.

5.6 Example 6: MPC for Engine Control System

Lastly, we consider a representation of a model predictive control (MPC) system for a turbocharged diesel engine application. This system has been the subject of recent experimental work in the automotive industry and has appeared

$$
\begin{aligned}
\dot{p} &= c_1 \left(2\hat{u}_1 \sqrt{\frac{p}{c_{11}} - \left(\frac{p}{c_{11}}\right)^2} - \left(c_3 + c_4 c_2 p + c_5 c_2 p^2 + c_6 c_2^2 p\right) \right) \\
\dot{r} &= 4 \left(\frac{c_3 + c_4 c_2 p + c_5 c_2 p^2 + c_6 c_2^2 p}{c_{13}(c_3 + c_4 c_2 p_{est}^2 + c_5 c_2 p_{est}^2 + c_6 c_2^2 p_{est})(1 + i + c_{14}(r - c_{16}))} - r \right) \\
\dot{p}_{est} &= c_1 \left(2\hat{u}_1 \sqrt{\tfrac{p}{c_{11}} - \left(\tfrac{p}{c_{11}}\right)^2} - c_{13} \left(c_3 + c_4 c_2 p_{est} + c_5 c_2 p_{est}^2 + c_6 c_2^2 p_{est}\right) \right) \\
\dot{i} &= c_{15}(r - c_{16})
\end{aligned}
\tag{21}
$$

Figure 5: System dynamics for the Powertrain Control System.

recently in the literature [11]. There has been interest in adopting MPC in the automotive industry, but several hurdles remain, such as the ability to prove safety properties of the closed-loop system. A technique that provides a means to, for example, prove stability or to provide guarantees on performance bounds would help to ease the way for this new technology to find application in industry. Below, we apply our technique to prove this system is stable by discovering a discrete-time Lyapunov function that is valid over a given domain.

The purpose of the MPC system for this application is to regulate the manifold pressure (MAP) and exhaust gas recirculation (EGR) rate. The MAP affects the amount of air injected into the cylinder for the combustion phase of the engine; accurately controlling the MAP directly affects the power output of the engine as well as the efficiency. The EGR subsystem allows some portion of the exhaust gas to be reinjected into the cylinder, with the ultimate effect of increasing efficiency and decreasing the rate of emissions.

The actuators are the variable geometry turbine (VGT) and the EGR valve. The VGT controls how much air is forced into the manifold due to pressure from the exhaust gases. The EGR valve regulates the rate at which exhaust gases are recirculated into the intake manifold.

The model we consider is a DSMS. The continuous-valued dynamics are given by affine difference equations. The plant is highly nonlinear with at least eight state variables. The version of the plant that we consider is first linearized, producing a linear time-invariant model with eight state variables. Then the number of state variables is reduced by applying a model-order reduction technique. The controller has 69 modes of operation (27 modes within the domain of interest); in each mode a unique linear feedback law is applied. The resulting closed-loop model of the system has three continuous-valued state variables and 27 modes.

The control space is divided into so called *controller regions*, $X_i \subset \mathbb{R}^3$ for $i \in \{1, \ldots, 27\}$, where each region is associated with a unique mode of the controller. The collection of regions partitions the domain of interest; the boundaries of each region are polyhedral sets. The dynamics are given by:

$$
\mathbf{x}[k+1] \;=\; \mathbf{A}_i \mathbf{x}[k] + \mathbf{b}_i, \ \forall \mathbf{x}[k] \in X_i,
$$

where $\mathbf{A}_i \in \mathbb{R}^{n \times n}$ and $\mathbf{b}_i \in \mathbb{R}^n$.

Note that existing LMI techniques failed to identify a Lyapunov function for this example, but as we describe below, our search-based technique is able to discover a solution.

We use a quadratic Lyapunov template and define the domain as the ball of radius 20.0 centered at the origin. The search procedure produces the following Lyapunov candidate in 107.29 seconds:

$$
\mathbf{P} \;=\; \begin{bmatrix} 1.625 & -0.309 & 0.740 \\ -0.309 & 0.886 & 0.208 \\ 0.740 & 0.208 & 1.688 \end{bmatrix}.
$$

A query to dReal takes 133 seconds to prove that the resulting candidate Lyapunov function is a proper Lyapunov function over the domain. This provides a proof of stability as well as a mechanism to produce forward invariant sets for the MPC system.

6. CONCLUSIONS

We presented a Lyapunov-based technique for the analysis of systems based on simulation data. The technique leverages numerical optimization and automated reasoning technologies such as SMT solvers and can be used to demonstrate stability and to provide performance bounds and safety guarantees for nonlinear and hybrid dynamical systems. This technique directly targets industrial applications, where simulation data is easily obtainable but application of traditional formal methods is not yet feasible.

The foundation of our analysis is a technique to automatically generate and iteratively improve upon candidate Lyapunov functions. The candidates are generated based on linear constraints provided by simulation traces; the feasible solution to an LP problem provides the Lyapunov candidates. We iteratively improve upon a series of candidates by using a tool that we call a falsifier, which is a global optimizer guided by a cost function that is based on the Lie derivative of the candidate Lyapunov function. An SMT solver is then used to validate the soundness of the resulting analysis, which can be a stability proof, a forward invariant set, or a barrier certificate. If necessary, we refine the candidate Lyapunov functions using counterexamples from the SMT solver.

We provided several examples, including two examples from the automotive engine control domain. No guarantees exist that our procedure will terminate with a sound analysis result, but our examples show that the technique can be applied to challenging industrial problems.

[1]For a Intel Xeon E5606 2.13GHz Dual Processor machine, with 24 GB RAM, running Windows 7, SP1

[2]For a 4x Intel Core i7 at 2.7 GHz with 8 GB RAM, running Ubuntu 13.04

Table 2: Results from Lyapunov analysis for various examples.

Model	Name	Degree	Candidate Time (sec.)^1	No. Sim. Points	Verif. Time (sec.)^2	Solver
1	Pendulum	2	74.22	3,000	7.72	Mathematica
2	Constrained Pendulum	2	2,308.54	57,240	0.084	dReal
3	Mathieu	2	216.61	12,000	0.044	Mathematica
4	Switched-Mode	2	170.75	9,800	1.476	Mathematica
5	PTC	3	1,413.73	258,078	1,157.42	dReal
6	MPC	3	107.29	4,480	133	dReal

Future work will consider non-autonomous systems and will explore alternative search strategies based on, for example, machine learning.

Acknowledgements

The authors would like to thank Ken Butts from the Toyota Technical Center and Mike Huang from the University of Michigan for their help with the automotive control models, Ufuk Topcu from the University of Pennsylvania for insightful discussions on numerical techniques for obtaining Lyapunov functions, and the anonymous reviewers for their constructive comments.

7. REFERENCES

[1] *Using Simulink*. The MathWorks, 2007.

[2] B. Akbarpour and L. C. Paulson. Metitarski: An automatic theorem prover for real-valued special functions. *Journal of Automated Reasoning*, 44(3):175–205, 2010.

[3] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *SIAM Studies in Applied Mathematics*. SIAM, 1994.

[4] M. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):475–482, April 1998.

[5] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, 1991.

[6] L. De Moura and N. Bjørner. Z3: An efficient smt solver. In *Proc. of Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, 2008.

[7] M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *JSAT*, 1(3-4):209–236, 2007.

[8] S. Gao, J. Avigad, and E. M. Clarke. δ-complete decision procedures for satisfiability over the reals. In *Automated Reasoning*, pages 286–300. Springer, 2012.

[9] A. Gupta, R. Majumdar, and A. Rybalchenko. From tests to proofs. In *Proc. of Tools and Algorithms for Construction and Analysis of Systems*, pages 262–276, 2009.

[10] W. Haddad and V. Chellaboina. *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach.* Princeton University Press, 2011.

[11] M. Huang, H. Nakada, S. Polavarapu, R. Choroszucha, K. Butts, and I. Kolmanovsky. Towards combining nonlinear and predictive control of diesel engines. In *American Control Conference, 2013. Proceedings of the 2004*, pages 2852–2859. IEEE, 2013.

[12] M. Johansson. *Piecewise Linear Control Systems*, volume 284 of *Lecture Notes in Control and Information Sciences*. Springer, 2003.

[13] D. Liberzon. Basic problems in stability and design of switched systems. *IEEE Control Systems*, 19(5):59–70, Oct. 1999.

[14] J. Löfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proc. of Computatational Aspects of Control System Design*, Taipei, Taiwan, 2004.

[15] MATLAB. *version 7.12.0 (R2011a)*. The MathWorks Inc., Natick, Massachusetts, 2011.

[16] A. Papachristodoulou and S. Prajna. Analysis of Non-polynomial Systems Using the Sum of Squares Decomposition. In D. Henrion and A. Garulli, editors, *Positive Polynomials in Control*, volume 312 of *Lecture Notes in Control and Information Sciences*, pages 23–43. Springer Berlin / Heidelberg, 2005.

[17] P. A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization.* PhD thesis, California Institute of Technology, 2000.

[18] S. Prajna. *Optimization-based methods for nonlinear and hybrid systems verification.* PhD thesis, California Institute of Technology, Caltech, Pasadena, CA, USA, 2005.

[19] J. Slotine and W. Li. *Applied nonlinear control.* Prentice Hall, 1991.

[20] J. F. Sturm. Using SeDuMi 1.02, A MATLAB Toolbox for Optimization over Symmetric Cones. *Optimization Methods and Software*, 11/12(1-4):625–653, 1999.

[21] U. Topcu, P. Seiler, and A. Packard. Local stability analysis using simulations and sum-of-squares programming. *Automatica*, 44:2669–2675, 2008.

[22] A. van der Schaft and J. Schumacher. *An Introduction to Hybrid Dynamical Systems.* Lecture Notes in Control and Information Sciences. Springer, 2000.

[23] L. Vandenberghe and S. Boyd. Semidefinite Programming. *SIAM Review*, 38(1):49–95, March 1996.

[24] S. Wolfram. *The Mathematica® Book, Version 4.* Cambridge University Press, 1999.