# Convex Optimization and Machine Learning for Scalable Verification and Control

by

Shen Shen

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2020

Author . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Department of Electrical Engineering and Computer Science
July 31, 2020

Certified by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Russ Tedrake
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

# Convex Optimization and Machine Learning for Scalable Verification and Control

by

Shen Shen

Submitted to the Department of Electrical Engineering and Computer Science
on July 31, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

## Abstract

Having scalable verification and control tools is crucial for the safe operation of highly dynamic systems such as complex robots. Yet, most current tools rely on either convex optimization, which enjoys formal guarantees but struggles scalability-wise, or black-box learning, which has the opposite characteristics. In this thesis, we address these contrasting challenges, individually and then via a rapprochement.

First, we present two scale-improving methods for Lyapunov-based system verification via sum-of-squares (SOS) programming. The first method solves compositional and *independent* small programs to verify large systems by exploiting natural, and weaker than commonly assumed, system interconnection structures. The second method, even more general, introduces novel quotient-ring SOS program reformulations. These programs are multiplier-free, and thus smaller yet stronger; further, they are solved, provably correctly, via a numerically superior finite-sampling. The achieved scale is the *largest* to our knowledge (on a 32 states robot); in addition, *tighter* results are computed 2–3 orders of magnitude faster.

Next, we introduce one of the *first* verification frameworks for partially observable systems modeled or controlled by LSTM-type (long short term memory) recurrent neural networks. Two complementary methods are proposed. One introduces novel integral quadratic constraints to bound general sigmoid activations in these networks; the other uses an algebraic sigmoid to, without sacrificing network performances, arrive at far simpler verification programs with fewer, and exact, constraints.

Finally, drawing from the previous two parts, we propose SafetyNet, which via a novel search-space and cost design, jointly learns readily-verifiable feedback controllers and *rational* Lyapunov candidates. While leveraging stochastic gradient descent and over-parameterization, the theory-guided design ensures the learned Lyapunov candidates are positive definite and with "desirable" derivative landscapes, so as to enable direct and "high-quality" downstream verifications. Altogether, SafetyNet produces sample-efficient and certified control policies—overcoming two major drawbacks of reinforcement learning—and can verify systems that are provably beyond the reach of pure convex-optimization-based verifications.

Thesis Supervisor: Russ Tedrake
Title: Professor of Electrical Engineering and Computer Science

# Acknowledgments

I am deeply grateful to my advisor Russ Tedrake for his support and guidance — his investment in me, his interest and encouragement, his tremendous patience and faith, his technical inputs and constructive advices, his way of developing my skills and perspectives, and his overall help turning my inkling of what is research into the work presented here. I feel fortunate for having his deep insight, broad vision, and acute sense of technical substance versus mere cleverness to draw on. More broadly, his work ethic, open-mindedness, and disciplined and healthy lifestyle, still to this day, inspire me day-to-day. It has truly been an honor and a pleasure to work with Russ.

I would like to thank my thesis committee: Sasha Megretski and Pablo Parrilo. The bi-weekly meetings with them and Russ over the past two years have been nothing short of inspiration. I fondly remember zigzagging with them to fit ideas in between trivial and impossible, and how their quick pointers easily nudge me out of deadzones. Their debates and critique sessions reflect a balanced taste and historical perspective that took many years of effort to develop, and to be able to simply feed off those is a privilege I deeply cherish. Their questions and advices helped improve the clarity of many parts in this thesis. I would like to particularly thank Pablo for his direct inputs to Chapter 4 and Sasha for his inputs to Chapter 3.

I would also like to thank all the Robot Locomotions Group members for making the lab so stimulating and collaborative. Hongkai Dai, Robin Deits, and Twan Koolen effectively guided me into the lab. Knowledgeable and patient, they introduced to me the tools, problems, and culture in the lab, and greatly flattened the learning curve for me. They, along with Russ, happen to all be programming wizards, and imparted to me many good software engineering practices as well. I would like to thank Sadra Sadradinni for leading a collaboration on a verification paper; and thank him and Tobia Marcucci and Jack Umenberger for many helpful control and optimization-oriented discussions, and Yunzhu Li and Lucas Manuelli for many learning-related ones. I would also like to thank Tao Pang, Greg Izatt, and Wei Gao for keeping us

informed of the advancements in mechanics, manipulation, and computer vision. And thanks to Mieke Moran, Stacie Ford (from TRI), and Gretchen Jones for being so awesome at supporting our lab.

Over the years, thanks to the help of many, I was fortunate to get familiar with various topics that would otherwise be too time-consuming or peripheral for me to invest in. In particular, I would like to thank Diego Cifuentes for tutorial and many helpful discussions on algebraic geometry, which are instrumental for Chapter 4. Thanks to Osbert Bastani for many discussions on safe reinforcement learning via our collaboration, which helped accelerate my overall understanding of the field. Thanks to Aleksander Madry for initiating the CDML weekly meetings, which demystified for me many statistical and causal learning researches, and brought to my attention the studies of over-parameterization which later inspired a part of the designs in Chapter 6. Thanks to Hadas Kress-Gazit and Jacopo Banfi for many infrastructure and perception-in-the-loop discussions for the Periscope MURI. Thanks to Micah Fry for many interesting conversations on transferring the work to the test-bed in the Lincoln Lab, which taught me quite a lot about ground vehicle hardwares.

Thanks to all the professors whom I have had the chance to TA for: Devavrat Shah, David Sontag, Suvrit Sra, Jeff Lang, Karl Berggren, Gerry Sussman, and Qing Hu. Teaching has always been a passion of mine. To be able to get suggestions, tips, and directions from them to more confidently and effectively engage my class for the teaching, and, moreover, to be able to get a greater appreciation of the technical material through these teaching opportunities for my own learning, it is deeply rewarding.

Special thanks to my graduate counselor Asu Ozdaglar. Prior to working with Russ, my PhD journey was rather bumpy, when I would often feel overwhelmed (sometimes even lost) by all the possible research labs to join and classes to take. Asu generously shared her time, experience, and wisdom, and encouraged and guided me back to the right track.

My master's years were spent at the Media Lab, and I would like to take this opportunity to thank my then advisor Andrew Lippman. His eloquence, wit, wisdom,

and humor made my time at the lab not only fulfilling but enjoyable. I learned the importance of crafting an argument and the art of sales from losing many casual and fun debates to Andy. I am also grateful to Andy for connecting with Chess Grandmaster Maurice Ashley for collaboration; I feel very fortunate to have the chance to work on a project that mixes teaching, chess, software engineering, and social studies, all of which I truly enjoy.

This is also a fitting opportunity to thank my teachers and friends from my undergrad years at the Harbin Institute of Technology. My professors, in both Aero/Astro and English Literature departments, helped prepare me for transitioning into this academically and culturally different post-grad life. I would like to particularly thank Huijun Gao and Lixian Zhang for providing for me my first research home, and Ye Zhao for leading the project that would turn into my first paper. I would also like to thank Yibo, Yue, Peng, and Dan for being wonderful friends through those years and for the kind consolation to me and my mom when my dad suddenly passed away.

Thanks to my friends at MIT, Ying, Mitra, Lei, Yehua, and Ying-zong. You are the big brothers and sisters I never had; thank you for all your life and career advices, for putting up with my "shen-anigans" and laughing at my dumb jokes.

Thanks to my maternal grandparents for nurturing my love of comedy, literature, and opera. At times of inevitable frustration and anxiety, like now, as I write this thesis amidst this pandemic, those have been a great escape and reminder of the broader beauty in the world.

To my mother, thank you for just being you, strong, caring, and independent. Thank you also for always being there for me, for being a constant joy in my life, and for showing me how to love life. To my late father, I do wish you were alive to share this journey and moment, but I like to believe that you knew already I would have enjoyed what I experienced at MIT. After all, it was the fun we had playing with Legos, skateboards, and RC planes that sparked this dream in engineering. Thank you and love you, mom and dad. I am blessed to be your daughter.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The recent years witnessed a momentous growth of impressive robotics applications: obstacle-avoiding drones, back-flipping humanoids, Rubiks-solving and Lego-playing robotic hands, (semi)autonomous cars, not to mention the "purpose-built robots" for food delivering or warehouse packing, which were once a fantasy and now a somewhat mundane reality.

While a unifying theme of "safely and more reliably perform more dexterous and complex real-world tasks" clearly emerges from these advancements, under the hood, such progress in verification and control is made possible by two fundamentally different approaches. One approach is deeply rooted in control theory and convex optimization, as represented by Lyapunov theory and sum-of-squares (SOS) programs; and the other is data-driven, as represented by deep neural networks and supervised or reinforcement learning.

At the heart of the co-existence of the two approaches are their contrasting pros and cons. The optimization-based offer provable performance guarantee, but the core assumption of convexity restricts their scale and generality. On the contrary, despite the ever-increasing popularity and stellar empirical performance, analysis of learning-based methods is elusive due to the models' black-box nature.

## 1.1 Contribution

In this thesis, we address these contrasting challenges of the two approaches, first individually and then via a rapprochement. In particular, we propose to improve the scalability of the optimization-based; bridge the analytical gap of the learning-based; and design a balanced (structured) mixture of the two.

First in Part I, we present two methods to address the well-acknowledged scalability challenge for Lyapunov-based stability verification via sum-of-squares (SOS) programming. The first method exploits that large-scale systems are often natural interconnections of smaller subsystems, and solves *independent* and compositional small programs to verify the large systems. Compared with existing compositional methods, the proposed procedure does not rely on commonly-assumed special structures (e.g., cyclic or triangular), and results in significantly smaller programs and faster computation.

The second method, even more general, proposes novel sampling quotient-ring SOS programs. The method starts by identifying that inequality constraints and Lagrange multipliers are a major, but so far largely neglected, culprit of creating bloated SOS verification programs. In light of this, we exploit various inherent system properties to reformulate the verification problems as quotient-ring SOS programs. These new programs are multiplier-free, smaller, sparser, less constrained, yet less conservative. Their computation is further improved, significantly, by leveraging a recent result on sampling algebraic varieties. Remarkably, solution correctness is guaranteed with just a finite (in practice, very small) number of samples. The achieved scale is the *largest* to our knowledge (on a 32 states robot); in addition, *tighter* results are computed 2–3 orders of magnitude faster.

Next in Part II, we introduce one of the *first* verification frameworks for partially observable systems modeled or controlled by LSTM-type (long short term memory) recurrent neural networks. Formal guarantees for such systems are elusive due to two reasons. First, the de facto activations used in these networks, the sigmoids, are not directly amenable to existing verification tools. Moreover, the networks' internal looping structures make straightforward analysis schemes such as "unrolling" impractical for long-horizon reasoning.

Recognizing these challenges, we propose two complementary techniques to handle the sigmoids, and also to enable a connection with tools from control theory and convex optimization to handle the long horizon. One method introduces novel integral quadratic constraints to bound arbitrary sigmoid activations in LSTMs; the other proposes the use of an algebraic sigmoid to, without sacrificing network performances, arrive at far simpler verification with fewer, and exact, constraints.

Finally in Part III, drawing from the previous two parts, we design SafetyNet, a new algorithm that jointly learns readily-verifiable feedback controllers and *rational* Lyapunov candidates. While built on two cornerstones of deep learning—stochastic gradient descent and over-parameterization—SafetyNet more importantly takes cues from optimization and control theory for a purposeful search-space and cost design. Specifically, the design ensures that (i) the learned rational Lyapunov candidates are positive definite by construction, and that (ii) the learned control policies are empirically stabilizing over a large region, as encoded by "desirable" Lyapunov derivative landscapes. These two properties, importantly, enable direct and "high-quality" downstream verifications (those developed in Part I).

Altogether, thanks to the careful mixture of learning and optimization, SafetyNet has advantages over both components. In particular, it produces sample-efficient and certified control policies—overcoming two major drawbacks of reinforcement learning—and can verify systems that are provably beyond the reach of pure convex-optimization-based verification schemes.

# Chapter 2

# Background

In this chapter we provide a brief background on the key theoretical and computational tools that will be employed throughout this thesis.

## 2.1 Lyapunov Functions and Linear Matrix Inequalities

Lyapunov theory is perhaps the most fundamental tool in system stability analysis. Prior to its introduction, the only way to analyze the stability property was to explicitly solve for the system trajectory, which is potentially very hard, if at all possible. Lyapunov proposes to instead search for a scalar surrogate function of the state, often associated or intuitively understood as an energy function, whose value always decreases along the trajectory (implicitly) and thus eventually reaches the minimum, local or global depending on the searching criterion.

For example, for LTI systems, quadratics are necessary and sufficient, and the search problem can be straightforwardly formulated as semi-definite programming (SDP) or equivalently (only differ in terminology), linear matrix inequality (LMI) [13].

To give a more concrete illustration, suppose we are interested in checking if a system $x^+ = f(x)$ is globally asymptotically stable with respect to the origin, i.e., if for all possible initial states, $x_{+\infty} = 0$. Then what Lyapunov proposed is that we look for a scalar function $V(x)$ that satisfies: $V(x) = 0, x = 0$; $V(x) > 0, \forall x \neq 0$; and $V(x^+) - V(x) = V(f(x)) - V(x) < 0, \forall x$. If we can find one such function, it would be sufficient to make the stability claim. Notice how in the last condition, $f(x)$ only appears implicitly, which spared us the difficulty of keeping tabs on what the state realization is at any given specific time (except for the initial and final states which are both given as problem data).

In addition to offering the theoretically powerful alternative viewpoint, Lyapunov theory also gained popularity because it links nicely with the efficient semi-definite programming (SDP), a type of convex optimization. This connection makes the search of Lyapunov function a very automated and systematic procedure. For instance, if the system dynamics is $x^+ = f(x) = Ax$ where $A$ is a constant matrix, then there exists a $V$ satisfying all the Lyapunov conditions if and only if the SDP below is feasible:

$$\begin{aligned} \text{find} \quad & P \succ 0 && \text{(2.1a)} \\ \text{s.t.} \quad & APA' - P \prec 0 && \text{(2.1b)} \end{aligned}$$

The sufficiency should be obvious once recognizing it is by parameterizing $V = x'Px$. We omit the necessity details and refer the readers to the wonderful book [13] for details.

Granted, Eq. (2.1) and its immediate variants are only suitable for global analysis of linear time invariant, deterministic, and non-constrained systems. It nonetheless opened up doors to numerous extensions, contributed from both systems and optimizations, to address more general problem settings.

We describe three relevant ones here: sum-of-squares programming, which brings in specialized polynomial dynamics and polynomial Lyapunov parameterization; S-procedure, which adds the capability of local analysis and state constraints analysis; and integral quadratic constraints, which offers the treatment of a broad class of bounded nonlinearities.

## 2.2 Sum-of-Squares (SOS) Programming

For polynomial dynamics, direct application of Lyapunov theory requires checking non-negativity of polynomials, which is unfortunately NP-hard in general. However, the problem of checking if a polynomial is sum-of-squares (SOS) - sufficient for non-negativity, is computationally approachable. A scalar multivariate polynomial $F(x) \in \mathbb{P}[x]$ is called SOS if it can be written as $F(x) = \sum_{i=1}^{m} f_i^2(x)$ for a set of polynomials $\{f_i\}_{i=1}^{m}$. If $\deg(F) = 2n$, this SOS condition is equivalent to $F(x) = m'(x)Qm(x)$ where $m(x)$ is a vector whose rows are monomials of degree up to $n$ in $x$, and the constant matrix called the Gram matrix $Q \succeq 0$. Thus, the search of a SOS decomposition for $F$ can be equivalently cast as an SDP on $Q$ [51].

In short, SOS program computationally generalizes the linear dynamics Lyapunov analysis and greatly expands the use cases. Additionally, the generalization is so clean and under-the-hood that for an end user everything conceptual remains the same. The only change is a swap of the sign conditions (or equivalently the positive-definitenesses) with SOS conditions.

**S-procedure** For general nonlinear systems we are interested in, global analysis is doomed to fail in all but very special cases [35]. We need a tool that can encode the local information, which ideally would also be compatible with the Lyapunov theory and computation for all the aforementioned benefits. S-procedure is such a tool.

Abstractly, S-procedure is a sufficient condition that handles the implication of the signs of quadratic functions for the sign of another quadratic function. Specifically, let $Q_1, \dots Q_m$ be quadratic functions of $x \in \mathbb{R}^n : Q_i(x) = x'U_ix + V_ix + w_i, i = 1, 2, \dots m$, and suppose we are interested in finding out this: for all the $x$ such that all these $m$ quadratic functions are non-negative, would these $x$ make another quadratic function $T(x)$ non-negative as well? (This is in general not trivial to answer, details can be found in [13, 14, 84].) S-procedure says, obviously, if there exists some $\lambda_1 \geq 0, \lambda_2 \geq$

$0, \ldots, \lambda_m \geq 0$ such that:

$$T(x) - \sum_{i=1}^{m} \lambda_i Q_i(x) \geq 0, \ \forall x \in \mathbb{R}^n \tag{2.2}$$

then the statement is true, that indeed:

$$T(x) \geq 0, \ \forall x \in \{x : Q_i(x) \geq 0, i = 1, 2, \ldots, m\} \tag{2.3}$$

This process links to our analysis as follows: if we let the Lyapunov difference condition Eq. (2.1b) be $-T(x)$, the target quadratic whose sign we hope to investigate, and if we let $\{x : Q_i(x) = x'U_ix + V_ix + w_i \geq 0\}$ encode the "not the entire $\mathbb{R}^n$" information, then by solving problems like Eq. (2.2), we could make claims such as "for all the states satisfying $\{x : Q_i(x) \geq 0\}$, the Lyapunov difference condition is met". We defer till the next subsection for a concrete example leveraging this process.

## 2.3 Integral Quadratic Constraints (IQC)

**Quadratic Constraints** The core idea of IQC is to relax the nonlinear terms with some chosen *fixed* quadratic constraints that those nonlinearities are known to satisfy, in order to simplify analysis. Figure 2-1 shows a contrived example to illustrate the main idea. For both systems, the dynamics can be written as $x^+ = (f(x) + x)/3$: a linear term plus a state-dependent nonlinearity expressed as $f(x)$. The distinction is that in system I, $f = \tanh(x)$ plotted as the red line is a difference equation; whereas in system II, $f(x)$ is a difference inclusion that is allowed to take any possible value as long as it satisfies $\{(f(x) - x) f(x) \leq 0\}$, which is plotted as the blue region. This difference inclusion is, in fact, very widely-used. It is due to Lur'e and is called the sector condition, and we will revisit it later.

Note that since any $x$ and $f = \tanh(x)$ pair respects the inequality $(f(x) - x)f(x) \leq 0$ defining the sector set, the hyperbolic tangent nonlinearity belongs to the sector. Visually, the blue region encompasses the red line. This containment means that any trajectory that system I could possibly produce is also a member in the set of trajectories system II could produce. Now if system II is stable or $\ell_2$ stable locally or globally, then any subset of those trajectories under consideration must also be stable or $\ell_2$ stable, including the subset that corresponds to system I. By this rationale, system II becomes a relaxation of system I in terms of stability/robustness analysis and serves as a surrogate for our analysis. We note that the full version with an integral involves a bit more reasoning but follows the same high level relaxation logic.

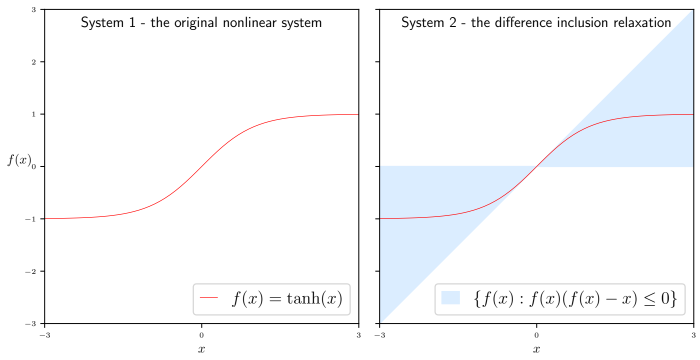


Figure 2-1: IQC tutorial example. For both systems, the dynamics is $x^+ = (x + f(x))/3$ where $f(x)$ is a difference equation for system I and a difference inclusion for system II.

A natural question arises as to why would one take such a detour in analysis. The virtue lies in that the system II is easier to analyze. The constraints, now quadratic, are directly compatible with the aforementioned framework. Specifically, direct application of the Lyapunov condition Eq. (2.1) for global stability analysis and S-procedure Eq. (2.2) requires solving:

$$\begin{aligned} \text{find} \quad & P > 0, \lambda > 0 \\ \text{s.t.} \quad & \underbrace{Px^2 - P(x+f)^2/9}_{-\Delta V} + \underbrace{\lambda(f-x)f}_{\substack{\text{S-procedure} \\ \text{constraining } x \text{ and } f}} > 0 \quad \forall x, f \end{aligned} \tag{2.4}$$

Simple algebra shows $P = 4, \lambda = 1$ makes Eq. (2.4) feasible, meaning system II is stable. And by the relaxation logic, this implies that system I is stable.

Notice here the Lyapunov difference plays the role of $-T$ and the sector condition plays the role of $Q_1$ in Eq. (2.2). At first glance, it may be surprising that the blue sector region, which is non-convex, could be encoded into a convex optimization problem. The process is indeed subtle: the non-convex constraint is on $x$ and $f(x)$, neither appears as decision variable in Eq. (2.4); instead, the decision variable is the multiplier $\lambda$ whose only constraint, the sign condition, is convex. Peeling one layer deeper, note that the implication of Eq. (2.2) $\implies$ Eq. (2.3) is true regardless of the convexity of both $T$ and $Q_i$ on $x$[1].

[1]In fact, the implication is true for non-quadratic functions as well; the assumption of everything quadratic is again for computational reasons. Quadratic is believed to be the sweet spot between conservatism (rich enough a class of function) and computational cost; it also comes with the benefit of easily describable as an SDP [45].

# Part I

# Scalable Optimization-Based Verification

# Chapter 3

# Compositional Verification

This chapter is adapted from work previously published in [65].

## 3.1 Introduction

As described in Section 2.1, LMIs are ubiquitous in system analysis, largely due to their clean connection to Lyapunov theory. It is widely known that most of the common LTI systems analysis and synthesis tasks directly translate through Lyapunov argument into LMIs [13]. More recently, the development of sum-of-squares (SOS) programming makes it possible to essentially apply this technique in polynomial systems as well [51], expanding the applicability even further.

Practically, however, LMIs and, by extension, SOS do not scale very well, and the computational cost is immense for large-scale systems. This computational challenge of SOS-based approaches for large-scale systems, combined with the natural decomposition structure arising from many such systems, motivates research areas such as compositional analysis [4, 78] and distributed and decentralized control [59, 6]. The common theme there is to not investigate the high-dimensional system directly. Instead, the large system is first divided and studied in parts, and the implications of these individual results would then be reasoned about collectively for the original system.

In the context of stability analysis for LTI systems, the compositional idea is typically materialized as the search of block-diagonal Lyapunov matrices. Various necessary and sufficient conditions on their existence have been given in the literature, but they are either restrictive or non-constructive. The restriction is usually on the dynamics matrix $A$ having a special structure such as being a Metzler matrix [48], block triangular, or cyclic [5], or on the Lyapunov matrix having particular block-diagonal patterns such as strictly diagonal [8, 67, 81, 72] or being limited to a 2-by-2 partition [69]. For the more general conditions, such as in [18], there is no simple recipe for computing the desired Lyapunov matrix from the sufficient rank conditions given. Our work offers sufficient constructive conditions without any of these structural limitations.

For compositional analysis of the more general polynomial systems via the SOS framework, we mention [78, 68, 3], which are most similar to ours. In previous work, the Lyapunov value constraints are untangled but their time derivatives are not; ours completely decouples both. We also note in particular that [3] focuses more on revealing a latent modular structure using graph partition ideas; we, on the other hand, assume the system has a given decomposition structure or one obvious enough by visual inspection and strive purely for efficiency. Our views are thus complementary, and combining them can solve a larger class of problems faster, as will be shown later by an example.

In Section 3.2, we formalize the problem and introduce technical background. In Section 3.3, we focus on finding block-diagonal Lyapunov matrix for LTI system, and present two algorithms for dynamics matrices of arbitrary size, structure, and partition pattern. In Section 3.4, we address the extension to polynomial dynamics and formulate a much smaller SOS programming. Finally, we demonstrate on numerical and practical examples in Section 3.5 the efficiency of the proposed algorithms.

**Notation** For a real vector $x \in \mathbb{R}^n$, the usual Euclidean 2-norm is denoted as $\|x\|$, the weighted 2-norm is denoted as $\|x\|_A^2 := x'Ax$ with $A \in \mathbb{R}^{n\times n}$, and the time derivatives are denoted as $\dot{x}$. If $x_i \in \mathbb{R}^{n_i}$, $i = 1, 2, \ldots, m$, then $(x_1, x_2, \ldots, x_m)$ denotes their column catenation. For a matrix $A \in \mathbb{R}^{m\times n}$, $\sigma_1(A)$ is its largest singular value and $A'$ its transpose. $A \succ 0$ (resp. $A \succeq 0$) implies $A$ is square, symmetric, and positive definite (resp. positive semidefinite). If $A_i \in \mathbb{R}^{k_i \times k_i}$, $i = 1, 2, \ldots, m$, then $\bigoplus_{i=1}^{m} A_i := A_1 \oplus A_2 \cdots \oplus A_m$ denotes the block diagonal matrix with diagonal blocks $A_1, A_2, \ldots, A_m$. $I$ denotes the identity matrix of appropriate size. Symbol $\setminus$ denotes set complement. $\mathbb{R}[x]$ denotes the ring of scalar polynomial functions in indeterminate $x$ with real coefficients, and $\mathbb{R}[x]^{m \times n}$ denotes an $m$ by $n$ matrix whose elements are scalar polynomials in $\mathbb{R}[x]$.

## 3.2 Problem Statement

Consider a time-invariant polynomial system described by $\dot{x} = f(x)$ where the state $x \in \mathbb{R}^n$ and the dynamics $f \in \mathbb{R}[x]^n$. We restrict ourselves to time-invariant systems and will drop all time dependencies. Let the state $x$ be partitioned into $m$ components: $x = (x_1, x_2, \ldots x_m)$, where $x_i \in \mathbb{R}^{n_i}$ constitutes the states of a subsystem. We assume the partition is one such that no more than two subsystems are coupled, i.e., no terms like $x_{11}x_{22}x_{32}$ ($x_{11}$ being the first state in the first subsystem and so on) exist in $f$. This is not a restrictive assumption as it can always be satisfied by regrouping (e.g., one can merge $x_1$ and $x_2$ into a new sub-system should terms like $x_{11}x_{22}x_{32}$ appear).

With the partition and assumption above, $\dot{x} = f(x)$ can be rearranged into a component-wise expanded form:

$$\dot{x}_i = f_i(x_i) + \sum_{\substack{j=1 \\ j \neq i}}^{m} g_{ij}(x_i) h_{ij}(x_j) \tag{3.1}$$

where $f_i \in \mathbb{R}[x_i]^{n_i}$ describes the internal dynamics of sub-state $x_i$, $g_{ij} \in \mathbb{R}[x_i]^{n_i \times l_{ij} n_j}$ and $h_{ij} \in \mathbb{R}[x_j]^{l_{ij} n_j}$ captures the coupling between sub-state $x_i$ and $x_j$. The newly introduced dimension $l_{ij}$ is due to the possibility of more than one linearly independent coupling terms involving $x_i$ and $x_j$, for instance, say $\dot{x}_1 = -{x_1}^3 + x_1 x_2 + 3{x_1}^2{x_2}^2$, then $l_{12} = 2$. For the special LTI case, $l_{ij} = 1, \ \forall i, j$.

We are interested in making claims such as asymptotic stability to the origin and invariance for the entire system states $x$, but ideally by examining one sub-system state $x_i$ at a time. To this end, we associate the system with a Lyapunov-like function $V$ such that:

$$V(x) = \sum_{i=1}^{m} V_i(x_i) \geq 0, \forall x \in \mathbb{R}^n \tag{3.2a}$$

$$\dot{V}(x) = \sum_{i=1}^{m} \dot{V}_i < 0, \forall x \in \mathbb{D} \tag{3.2b}$$

where the equality in Eq. (3.2a) holds only at the origin, and the region $\mathbb{D}$ in Eq. (3.2b) varies with the task in hand. For instance, when dealing with global stability to the origin, $\mathbb{D} = \mathbb{R}^n \backslash \{0\}$, whereas in local analysis the region is usually a sub-level set of $V$ and part of the decision variables.

The aim in this section is to find the set of $\{V_i\}_{i=1}^m$ functions *independently* so as to form as small an LMI or SOS as possible. Eq. (3.2a) is already in a decoupled form, and one can simply require $V_i \geq 0, \forall i$. Eq. (3.2b) may look decoupled too, and one may be tempted to claim that $\dot{V}_i < 0, \forall i$ is also a set of independent constraints; this is not true. Note that

$$\dot{V}_i = \frac{\partial V_i(x_i)}{\partial x_i} f_i + \sum_{\substack{j=1 \\ j \neq i}}^{m} \frac{\partial V_i(x_i)}{\partial x_i} g_{ij}(x_i) h_{ij}(x_j) \tag{3.3}$$

while the first term is only dependent on $x_i$, the second term that is the summation involves $h_{ij}$, a function of sub-states $x_j$, and the summing over all $j \neq i$ makes $\dot{V}_i$ dependent on possibly the entire states $x = (x_1, x_2, \ldots x_m)$. This is a direct consequence of sub-states coupling from the dynamics $\dot{x}_i$, in other words, the set of $\{\dot{V}_i\}_{i=1}^m$ are inherently entangled.

Our compositional approach thus avoids dealing with $\{\dot{V}_i\}_{i=1}^m$ head-on. Instead, we resort to finding an upper bound of $\dot{V}$ that is by design a sum of functions each dependent on one $x_i$ only. We then require this upper bound to be non-positive to sufficiently imply Eq. (3.2b). While this detour leads to more conservative results, it allows the parallel search we desire and can bypass the computational hurdle of direct optimizations. The details of our approach are in Section 3.3 and 3.4.

## 3.3 LTI Systems and Block-Diagonal Lyapunov Matrix

We first study the most fundamental LTI systems. Though the technical result in this section can be reduced from the polynomial systems' result, some more intuitive aspects of it can only be or are better appreciated in this limited setting and hence it merits the separate elaboration here.

Under the LTI assumption, Eq. (3.1) takes a clean form

$$\dot{x}_i = A_{ii}x_i + \sum_{\substack{j=1 \\ j\neq i}}^{m} A_{ij}x_j \tag{3.4}$$

where $A_{ii} \in \mathbb{R}^{n_i\times n_i}$, $A_{ii}x_i$ corresponds to the $f_i$ term, $A_{ij} \in \mathbb{R}^{n_i\times n_j}$ corresponds to $g_{ij}$ with dimension $l_{ij} \equiv 1$, and $x_j$ corresponds to the $h_{ij}$ term.

For LTI systems, it only makes sense to consider global asymptotic stability (to the origin) as all convergences in LTI systems are in the global sense. A quadratic parameterization of Lyapunov function $V = x'Px$ such that $P \succ 0$ and $AP + PA' \prec 0$ is both necessary and sufficient for this task. Naturally then, when considering Lyapunov functions for the subsystems, we use this quadratic parameterization as well and let $V_i = x_i'P_ix_i$. This is equivalent to imposing a block-diagonal structure constraint on $P \succ 0$ as $P = \bigoplus_{i=1}^{m} P_i \succ 0$. Substituting the parameterization into condition Eq. (3.2) yields:

$$\begin{aligned}
&\text{find} \quad \{P_i\}_{i=1}^{m} && \text{(3.5a)}\\
&\text{s.t.} \quad P_i \succ 0, \forall i && \text{(3.5b)}\\
&\qquad \begin{bmatrix} A_{11}P_1 + P_1A_{11}' & \dots & A_{1n}P_n + P_1A_{n1}' \\ A_{21}P_1 + P_2A_{12}' & \dots & A_{2n}P_n + P_2A_{n2}' \\ \vdots & \ddots & \vdots \\ A_{n1}P_1 + P_nA_{1n}' & \dots & A_{nn}P_n + P_nA_{nn}' \end{bmatrix} \prec 0, && \text{(3.5c)}
\end{aligned}$$

Note that Eq. (3.5c) gives a more tangible sense of the latent intertwined nature of

*Eq.* (3.2b). The clean block-diagonal structure in $P$ is deeply buried here as the left hand side is a full matrix with $A_{ij}P_j + P_i A'_{ji}$ at all the off-diagonal spots.

Our goal in this section is to find the set of $\{P_i\}_{i=1}^m$ independently for each $i$. We start with an LMI-based algorithm that is still somewhat coupled, and then gradually get to the truly decoupled algorithm which is based on Riccati equations.

### 3.3.1 Sparse LMIs

**Theorem 1.** *For an $n$-dimensional LTI system written in the form Eq.* (3.4)*, if the optimization problem*

$$\begin{aligned}
\textit{find}\quad & \{P_i\}_{i=1}^m, \{M_{ij}\}_{i,j=1,i\neq j}^m && (3.6a)\\
\textit{s.t.}\quad & P_i \succ 0, \forall i && (3.6b)\\
& M_{ij} \succ 0, \forall i,j, i \neq j && (3.6c)\\
& A_{ii}P_i + P_i A'_{ii} + \sum_{\substack{j=1\\ j\neq i}}^{m} A_{ij}M_{ij}A'_{ij} + P_i M_{ji}^{-1} P_i \prec 0, \forall i && (3.6d)
\end{aligned}$$

*is feasible, then the set of $\{P_i\}_{i=1}^m$ satisfies problem Eq.* (3.5) *with $\mathbb{D} = \mathbb{R}^n \backslash \{0\}$, and the original system is strictly asymptotically stable.*

*Proof.* Two proofs, one from the primal perspective and the other the dual, are included in the chapter appendix. Theorem 1 can also be reduced from Theorem 2 (in Subsection 3.4), whose proof is in fact less involved. The appended proofs, however, offer a control and optimization connection that the simple proof lacks. □

**Remark 1.** *Theorem 1 can be viewed as a generalization of the sufficient direction of Lyapunov inequality for LTI systems. Particularly, if the state is not partitioned, $P$ has no structural constraint, the summation in Eq.* (3.6d) *disappears, and the condition reduces to the ordinary Lyapunov inequality. Furthermore, Theorem 1 gives an explicit procedure to construct a Lyapunov matrix with user specified structures including the extreme case of pure diagonal structure.*

**Remark 2.** *Theorem 1 also closely resembles the small-gain theorem (which in fact is the inspiration for our results). Notice that setting $m = 2$ (two subsystems scenario) reduces Theorem 1 to the matrix version of the bounded real lemma, which proves the product of the two subsystems' $\ell_2$-gains less than or equal to one and implies stability. The intuition behind the connection is this: think of diagonal blocks $A_{11}$ and $A_{22}$ as describing two disconnected "nominal" plants, and the off-diagonal blocks are pumping feedback disturbance from one nominal system to the other. The compound system admitting a block-diagonal Lyapunov matrix indicates it is stable whether or not the disturbance blocks are present, and this is exactly what small-gain theorem implies. The feedback disturbance interpretation obviously carries over to more than two interconnected systems even though there is no extension of small-gain theorem in those settings. (The structured disturbance setup arising from $\mu$-synthesis is not such a generalization. While it does admit multi-dimensional disturbances, all the disturbances are to the central nominal plant but not to one another.) We illustrate the system assumptions difference between the two versions of small-gain and ours in Figure 3-1.*

**Remark 3.** *One might wonder what is the virtue of studying the bare bone Lyapunov inequality; after all, checking stability can easily be done through an eigenvalue computation and that is uniformly faster than LMIs. We believe the value lies in that LMIs is more general and clean than eigen-based methods. For instance, LMI formulation leads to extensions such as robustness analysis via common Lyapunov functions which eigen-based method fails to handle; or to a straightforward formulation of $\ell_2$-gain bound, for which the eigen-based method leads to very messy computation. Therefore, $AP + PA'$, the most basic building block appearing in almost every control LMI, deserves a close examination.*

**Remark 4.** *Computationally, Eq.* (3.6d) *with the nonlinear term $P_i M_{ji}^{-1} P_i$ can be equivalently turned into an LMI via Schur complement. Hence, Theorem 1 requires solving $m$ coupled LMIs of the original problem size, but all of them enjoy strong sparsity as all non-zero terms only appear, in the block sense, at one row, one cor-*

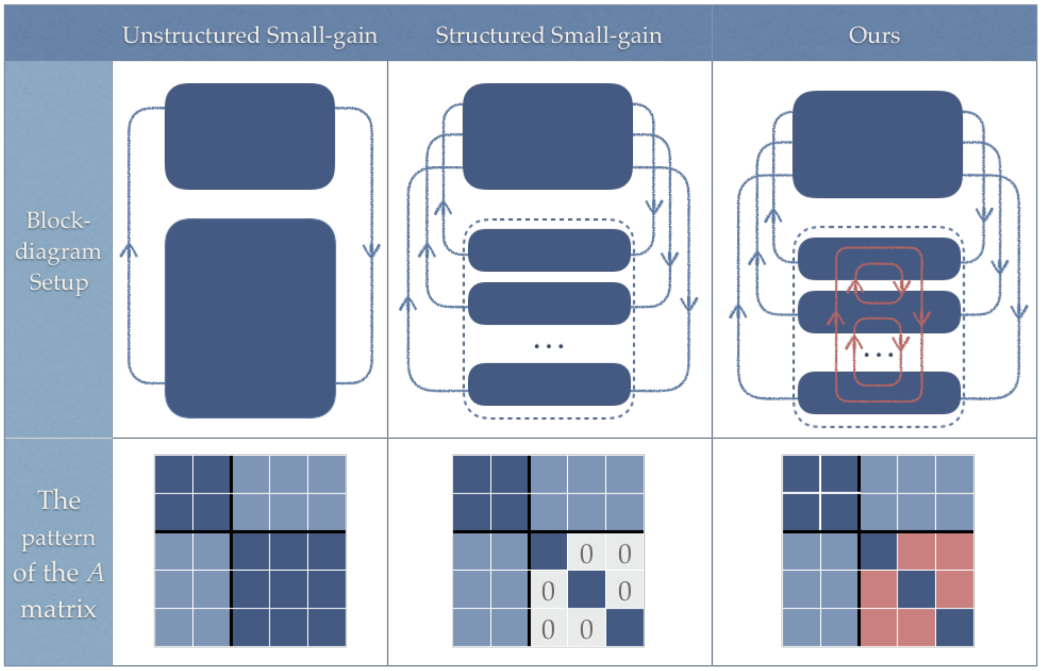


Figure 3-1: Our setup compared with the unstructured and structured small-gain setup. Our setup takes advantage of the 'weak' internal interconnection, thus achieve a 'balance' between the unstructured and structured cases.

*responding column, and the main diagonal. For instance, when $m = 8$, Figure 3-2 shows the sparsity pattern of three out of the eight LMIs. The sparsity in practice might already be a worthy trade off, and we test the claim in Section 3.5. Further, if we fix the set of $M_{ij}$ rather than searching for them, constraints Eq.* (3.6) *become decoupled low-dimensional LMIs. Better yet, they can be solved by an even faster Riccati equation based method below.*

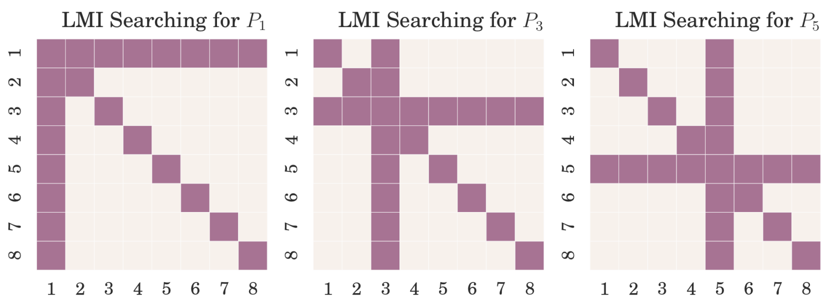


Figure 3-2: Sparsity pattern of three LMIs for $M = 8$ example

### 3.3.2 Riccati Equations

If $M_{ij}$ are fixed, the set of decoupled low-dimensional LMIs can be solved by a method based on Riccati equations, which has near-analytical solutions and by implication far better scalability and numerical stability than LMIs. Specifically, if we replace the inequality with equality, then Eq. (3.6) are precisely Riccati equations with unknown $P_i$. The feasibility of LMIs like Eq. (3.6) is equivalent to the feasibility of the associated Riccati equations. That is, the (unique) positive definite solution to the Riccati equation lives on the boundary of the feasible set of the LMI [13]. So by nudging the right hand side in the constraint slightly in the positive direction, e.g., replacing zero with $\epsilon I$ for some small $\epsilon > 0$, we get a solution strictly in the interior and one precise to the original LMI. We note that when the Riccati equations return a feasible solution, it is much faster than solving the sparse LMIs Eq. (3.6), and even more significantly so than the original LMI Eq. (3.5).

The choice of the set of positive definite $M_{ij}$ scaling matrices can be arbitrary but would largely affect the feasibility. Identity scaling is one obviously valid choice, and it is very likely to succeed in cases such as when the off-diagonal blocks are very close to zeros. In general though, identity scaling is not guaranteed to always work, it is then desirable to have some other heuristics at our disposal. Inspired by the small-gain theorem connection in Remark 2, we propose another educated guess that we call $\sigma_1$-scaling. The procedure is to first initialize a set of scalars $\gamma_{ij} = \sigma_1({A_{ii}}^{-1}A_{ij})$; then keep $\gamma_{ij}$ as is if $\gamma_{ij}\gamma_{ji} \leq 1$, otherwise, say $\gamma_{ij} > \gamma_{ji}^{-1}$, then keep only $\gamma_{ji}$ and shrink $\gamma_{ij}$ down to $\gamma_{ji}^{-1}$; and finally set $M_{ij} = \gamma_{ij} I$. The justification of the heuristic is that $\sigma_1$ operator of a dynamics matrix loosely reflects the input-output signal magnification by the system, and $\sigma_1({A_{ii}}^{-1}A_{ij})$ can therefore serve as a barometer of the relative energy exchange between an internal $A_{ii}$ subsystem and the coupling $A_{ij}$ term from system $j$. Of course, there is no guarantee on the performance of $\sigma_1$-scaling either. Empirically though, they succeed roughly 7 times out of 10. Plus, the time it takes to test these scalings is negligible compared with solving any LMIs, so it is well worth a try.

## 3.4 Polynomial Systems and Compositional SOS Lyapunov Functions

Extending the LTI decoupling idea to polynomial systems is conceptually straightforward: we again want to upper bound $\dot{V}$. Technically, a few nice properties from the LTI case would vanish. We will discuss these issues when they appear, and for now start with the result for global asymptotic stability (g.a.s.).

**Theorem 2.** *For a polynomial system described in the expanded form Eq.* (3.1)*, if*

$$
\begin{aligned}
\textit{find}\quad & \{V_i\}_{i=1}^m, \{M_{ij}\}_{i,j=1,i\neq j}^m && (3.7a)\\
\textit{s.t.}\quad & M_{ij} \succ 0, \forall i, j, i \neq j && (3.7b)\\
& V_i - \epsilon\|x_i\| \quad \textit{is SOS}, \forall i && (3.7c)\\
& -\frac{\partial V_i}{\partial x_i} f_i - \frac{1}{2}\sum_{\substack{j=1\\ j\neq i}}^m \|\frac{\partial V_i}{\partial x_i} g_{ij}\|_{M_{ij}}^2 - \frac{1}{2}\sum_{\substack{j=1\\ j\neq i}}^m \|h_{ji}\|_{M_{ji}^{-1}}^2 - \epsilon\|x_i\| \quad \textit{is SOS}, \forall i && (3.7d)
\end{aligned}
$$

*is feasible for some $\epsilon > 0$, then the set of polynomial functions $\{V_i(x_i)\}_{i=1}^m$ satisfies Eq.* (3.2) *with $\mathbb{D} = \mathbb{R}^n \backslash \{0\}$ and the system is (g.a.s.) at the origin.*

*Proof.* Eq. (3.7c) obviously implies Eq. (3.2a). Then introducing invertible matrices $m_{ij} \in \mathbb{R}^{l_{ij}nj \times l_{ij}nj}$ and let $M_{ij} = m_{ij}m'_{ij}$, we can have $\dot{V}$ upper bounded:

$$
\begin{aligned}
\dot{V} =& \sum_{i=1}^m \left( \frac{\partial V_i}{\partial x_i} f_i + \sum_{\substack{j=1\\ j\neq i}}^m \frac{\partial V_i}{\partial x_i} g_{ij} h_{ij} \right) && (3.8a)\\
=& \sum_{i=1}^m \frac{\partial V_i}{\partial x_i} f_i + \sum_{i=1}^m \sum_{\substack{j=1\\ j\neq i}}^m \left( \frac{\partial V_i}{\partial x_i} g_{ij} m_{ij} m_{ij}^{-1} h_{ij} \right) && (3.8b)\\
\leq& \sum_{i=1}^m \frac{\partial V_i}{\partial x_i} f_i + \frac{1}{2}\sum_{i=1}^m \sum_{\substack{j=1\\ j\neq i}}^m \left( \|g_{ij} \frac{\partial V_i}{\partial x_i}\|_{M_{ij}}^2 + \|h_{ij}\|_{M_{ij}^{-1}}^2 \right) && (3.8c)\\
=& \sum_{i=1}^m \left( \frac{\partial V_i}{\partial x_i} f_i + \frac{1}{2} \sum_{\substack{j=1\\ j\neq i}}^m \left( \|\frac{\partial V_i}{\partial x_i} g_{ij}\|_{M_{ij}}^2 + \|h_{ji}\|_{M_{ji}^{-1}}^2 \right) \right) && (3.8d)
\end{aligned}
$$

Eq. (3.8c) is due to the elementary inequality of arithmetic and geometric means (AM-GM inequality), and the exchange of summation index at Eq. (3.8d) is due to the symmetry between $i$ and $j$. Eq. (3.7d) implies the negation of Eq. (3.8d) is SOS, which directly leads to that the negation of $\dot{V}$ is SOS, and sufficient to imply Eq. (3.2b). □

**Remark 5.** *Our method can be extended to handle coupling terms such as $x_i x_j x_k$ that involves more than two sub-states. The key step is to use the generalized version of AM-GM inequality with $n > 2$ variables at Eq.* (3.8c).

**Remark 6.** *Similar to the LTI case, the scaling matrices $M_{ij}$ brings coupling across the constraints. Eliminating these constants as decision variables could again untangle the entire set of constraints. However, we do not believe a trivial extension of $\sigma_1$-scaling developed for the LTI case would be as convincing a heuristic for hand-picking these constants in the polynomial settings. This is mainly due to the lack of a notion of 'coupling strength' in the polynomial sense. Specifically, for LTI systems, the coupling can only enter as $A_{ij}x_i x_j$, so at least intuitively, for a 'normalized' $A_{ii}$ the strength of the coupling is quantified by $A_{ij}$. Polynomial systems with the additional freedom of degrees, however, can have coupling terms like $4x_i x_j$ and $x_i x_j^2$. It is then hard to argue, even hand-wavingly, if the coefficients play a more important role or if the degrees do. Therefore, we settle with just fixing all the $M_{ij}$ to identity.*

**Remark 7.** *Even with $M_{ij} = I$, the term $\|\frac{\partial V_i}{\partial x_i} g_{ij}\|^2$ in Eq.* (3.7d) *is still not directly valid for a SOS program because of the quadratic dependency on $V_i$. We develop below what can be considered the generalization of Schur complement in the polynomial settings to legalize the constraint.*

**Lemma 1.** *Given a scalar SOS polynomial $q(x) \in \mathbb{R}[x]$ of degree $2d_q$ and a vector of generic polynomials $s(x) \in \mathbb{R}[x]^n$ of maximum degree $d_s$, let $y$ be a vector of indeterminates whose elements are independent of $x$, then $q(x) - s'(x)s(x) \in \mathbb{R}[x]$ is SOS if and only if $q(x) + 2y's(x) + y'y \in \mathbb{R}[x, y]$ is SOS.*

*Proof.* Define $m(x)$ and $n(x, y)$ respectively as the standard monomial basis of $x$ and $(x, y)$ up to degree $d = \max(d_q, d_s)$. It is always possible to rewrite $s(x) =$

$C'm(x)$ for some coefficients $C$, and $q(x) = m'(x)\left[Q + L(\alpha_1)\right]m(x)$, where Q is a constant symmetric matrix such that $q(x) = m'(x)Qm(x)$, $L(\alpha_1)$ is a parameterization of the linear subspace $\mathcal{L} := \{L = L' : m'(x)L(\alpha)m(x) = 0\}$, and $Q + L(\alpha_1) \succeq 0$. Denote $q(x) - s'(x)s(x)$ as $\Pi_1$ and plug in these parameterizations, $\Pi_1 = m'(x)\left[Q + L(\alpha_1) - C'C\right]m(x)$.

If $\Pi_1$ is SOS, then there exists an $L(\alpha_2) \in \mathcal{L}$ (possibly different from $L(\alpha_1)$) such that $Q + L(\alpha_1) - C'C + L(\alpha_2) \succeq 0$. This implies via Schur complement that $V := \left[\begin{smallmatrix} Q+L(\alpha_1)+L(\alpha_2) & C' \\ C & I \end{smallmatrix}\right] \succeq 0$. Denote $q(x) + 2y's(x) + y'y$ as $\Pi_2$ and notice that it is precisely $(m(x), y)'V(m(x), y)$, and therefore $\Pi_2$ is SOS.

If $\Pi_2$ is SOS, then there exists a $\beta_1$ such that $\Pi_2 = n'(x,y)\left[T + M(\beta_1)\right]n(x,y)$ where $T$ is a constant symmetric matrix such that $n'(x,y)Tn(x,y) = \Pi_2$, $M(\beta_1)$ is a parameterization of the linear subspace $\mathcal{M} := \{M = M' : n'(x,y)M(\beta)n(x,y) = 0\}$, and $T + M(\beta_1) \succeq 0$. Since the elements of $m(x)$ and $y$ form a strict subset of those in $n(x,y)$, the ordering $n(x,y) = (m(x), y, k(x,y))$ where $k$ encapsulates the $x, y$ cross term monomials is possible. Accordingly, $T + M(\beta_1)$ can be partitioned as $\left[\begin{smallmatrix} T_{11}+M_{11} & T_{12}+M_{12} & T_{13}+M_{13} \\ * & T_{22}+M_{22} & T_{23}+M_{23} \\ * & * & T_{33}+M_{33} \end{smallmatrix}\right]$ ($\beta_1$ from now on dropped for concision). Then, since $\Pi_2$ has no cross terms of second or higher order in $y$, it must be that $k'(x,y)\left[T_{33} + M_{33}\right]k'(x,y) = 0$, and since $y$ and $x$ are independent, $T_{33} + M_{33} = 0$. Similar arguments imply that $T_{23} + M_{23} = T'_{32} + M'_{32} = 0$, and $T_{22} + M_{22} = I$. Once these four blocks are fixed, by the equality constraint in the Schur complement of $T_{11} + M_{11}$, it must be the case that $T_{13} + M_{13} = T'_{31} + M'_{31} = 0$. In other words, $\Pi_2$ in fact admits a more compact expansion $\Pi_2 = (m(x), y)'\left[\begin{smallmatrix} T_{11}+M_{11} & T_{12}+M_{12} \\ * & I \end{smallmatrix}\right](m(x), y)$ where by matching the terms and invoking the independence of $x$ and $y$, $m'(x)\left[T_{11} + M_{11}\right]m(x) = q(x)$, $m'(x)\left[T_{12} + M_{12}\right] = s(x)$, and the gram matrix is positive semi-definite. The Schur complement of the $I$ block therefore gives an explicit SOS parameterization of $\Pi_1$ in $m(x)$. $\square$

*Lemma 1 trivially extends to "$q(x) - \sum_{j=1}^{m} \|s_j(x)\|^2$ is SOS", again via Schur complement of the Gram matrix. The extended condition can be mapped to Eq.* (3.7d), *with* $-\frac{\partial V_i}{\partial x_i} f_i - \frac{1}{2}\sum_{\substack{j=1 \\ j \neq i}}^{m} \|h_{ji}\|^2$ *as* $q(x)$, *and* $\frac{\partial V_i g_{ij}}{\sqrt{2}\partial x_i}$ *as* $s_j(x)$. *Now the constraint Eq.* (3.7d) *is linear in* $V_i$ *and can be readily handled by a SOS program.*

## 3.5 Experiments and Examples

The examples are run on a MacBook Pro with 2.9GHz i7 processor and 16GB memory. The LMI problem specifications are parsed via CVX [25], SOS problems are parsed via SPOTLESS [76], and both are then solved via MOSEK [47]. The source code is available online.[1]

**Randomly Generated LTI Systems** We randomly generate 1000 candidate $A$ matrices of various sizes that admit block-diagonal Lyapunov matrices of various block sizes. We facilitate the sampling process by biasing $A$ towards negative block-diagonal dominance (to make it more likely an eligible candidate), and then pass this sample $A$ into the full LMI Eq. (3.5) to check if the LMI (a necessary and sufficient condition) produces a block-diagonal Lyapunov matrix, if not, the sample is rejected. Table 3.1 records the average run time comparison of this full LMI Eq. (3.5) and our proposed sparse LMIs Eq. (3.6) and Riccati equations algorithms. It also records the success rate of identity scaling and $\sigma_1$-scalings for hand-picking the $M_{ij}$ term in the Riccati equations.

Table 3.1: Run-time and success rates for LTI systems

| | Size of $A$ | 50 | 100 | 100 | 1000 |
|---|---|---|---|---|---|
| | Block Size | 25 | 10 | 25 | 25 |
| Run-time (seconds) | Full LMI | 0.2 | 1.03 | 2.26 | – |
| | Sparse LMIs | 0.18 | 0.83 | 1.76 | – |
| | Riccati eqns | 1.44e-2 | 1.1e-3 | 1.89e-2 | 0.5 |
| Success Rate (Riccati eqns) | Identity | 70% | 56% | 68% | – |
| | $\sigma_1$-scaling | 78% | 69% | 65% | – |

The first two "–"s in the last column indicate the LMI-based methods run into memory issues and the solver is forced to stop. Riccati equations are still able to solve some of these sampled problems, but without the baseline LMI feasibility, its success rate is not available either, hence the last two "–"s. We note that when the

[1]codes available at https://github.com/shensquared/ComposableVerification

Riccati equations are feasible, they are the fastest method to check stability of the generated $A$ matrices. The computational saving becomes more significant as the dimension goes up thanks to scalable algorithms solving Riccati equations. We also note that there is no clear winner between identity scaling and $\sigma_1$ scaling when it comes to producing feasible Riccati equations. In practice, one may want to try both as there is very little added real time computational cost of doing so.

**Lotka-Volterra System** We take from [3] the Lotka-Volterra example and verify its stability. It is a 16-dimensional polynomial system[2], and is thus beyond the reach of direct SOS optimization. In that paper, the authors handle the task by first developing a graph partition based algorithm, which finds a 3-way partition scheme for this example, and then a non-sparse or sparse SOS programs for computing a Lyapunov function for each subsystem and a composite Lyapunov function for the original system.

Their graph partition algorithm is of particular interest to us, since our compositional algorithm requires a partition but lacks the ability to search for one. Reusing their resulting 3-way partition scheme, we are also able to find a composite Lyapunov function. However, our underlying SOS formulations and hence run-time are significantly different.

Table 3.2: Run-time comparison for Lotka-Volterra system

| | Non-sparse alg.[3] | Sparse alg. [3] | Proposed |
|---|---:|---:|---:|
| $V_1$ | 0.25s | 0.38s | 0.1023s |
| $V_2$ | 0.25s | 0.37s | 0.0587s |
| $V_3$ | 0.44s | 0.59s | 0.0568s |
| $V$ | 1415.23s | 688.54s | 0.2178s |

For both non-sparse and sparse algorithms in [3], after the individual $\{V_i\}_{i=1}^{3}$ are found by low-dimensional SOS, it relies on an additional *search* of $\alpha_i > 0$ such that $\sum_i \alpha_i V_i$ satisfies the derivative condition in $x$. This has to be done with yet another

[2]we omit explicitly listing here the dynamics for saving space and refer the reader to the source code for details

SOS program of considerable size since the indeterminate is the high dimensional $x$. Also, there is no guarantee such an $\alpha_i > 0$ always exists; it heavily depends on how compatible $\{V_i\}_{i=1}^3$ are (in our test, the $\{V_i\}_{i=1}^3$ we found do not produce feasible $\alpha_i$, so we copy the run-time reported in the paper for comparison).

Our sufficient condition Eq. (3.7), in contrast, guarantees that $\sum_i V_i$ would *automatically* satisfy the derivative condition. This is done by imposing more restrictive conditions on $\{V_i\}_{i=1}^3$ at the their independent construction stages. In other words, once we get these ingredients, no extra work is necessary and the time spent searching for $V$ is just the sum of time spent on each $\{V_i\}_{i=1}^3$ in much lower dimensions. Consequentially, as shown in Table 3.2, our final run-time is 3-4 orders of magnitude faster in finding the composite $V$.

This example showcases the combined power of graph partition-like algorithms such as [3] and the technique proposed in this paper: the former as a prepossessing step can extend the use case, and the latter can facilitate a much faster optimization program.

## 3.6 Discussion and Future Work

In this paper, general and constructive compositional algorithms are proposed for the computationally prohibitive problem of stability and invariance verification of large-scale polynomial systems. The key idea is to break the large system into several sub-systems, construct *independently* for each subsystem a Lyapunov-like function, and guarantee that their sum *automatically* certifies the original high-dimensional system is stable or invariant. The proposed algorithms can handle problems beyond the reach of direct optimizations, and are orders of magnitude faster than existing compositional methods.

We are interested in exploring extensions of our work to compositional safety verification of multi-agent networks by leveraging the barrier certificate idea [56]. We believe the connection is immediate both technically, as barrier certificates are natural extensions of Lyapunov function, and practically, as the multi-agent network has, by definition, a compositional structure.

# Appendix

## Primal Proof of Theorem 1

*Proof.* Let us denote the left hand side of Eq. (3.5c) as $U$. Let $U_k$ be the k-th leading principal sub-matrix of $U$ in the blocks sense (e.g., $U_1$ equals $A_{11}P_1 + P_1A'_{11}$ instead of the first scalar element in $U$), and let $\tilde{U}_k$ be the last column-blocks of $U_k$ with its last block element deleted, i.e.,

$$\tilde{U}_k := \begin{bmatrix} A_{1k}P_k + P_1A'_{k1} \\ A_{2k}P_k + P_2A'_{k2} \\ \vdots \\ A_{(k-1)k}P_k + P_{k-1}A'_{k(k-1)} \end{bmatrix}$$

Also, define a sequence of matrices:

$$N_k := \bigoplus_{i=1}^{k} \left( \sum_{j=k+1}^{m} A_{ij}M_{ij}A'_{ij} + P_iM_{ji}^{-1}P_i{}' \right)$$

for $k = 1, 2, \dots, m-1$, and $N_k = 0$ for $k = m$. Let $\bar{N}_k$ be the largest principal minor of $N_k$ in the block sense. It's obvious then that by construction $N_k \succeq 0, \forall k$.

We will use induction to show that $U_k + N_k \prec 0, \forall k$, so that in the terminal case $k = n$, we would arrive at the desired Lyapunov inequality $U = U_n + N_n \prec 0$. For $k = 1$, $U_1 + N_1 \prec 0$ is trivially guaranteed by taking $i = 1$ in Eq. (3.6d). Suppose $U_k + N_k \prec 0$ for a particular $k \leq n-1$, let us now show that $U_{k+1} + N_{k+1} \prec 0$.

First, notice that for $k \leq n-1$, the sequence of $N_k$ satisfies this recursive update: $N_k = n_k + \bar{N}_{k+1}$ where

$$n_k := \bigoplus_{i=1}^{k} \left( A_{i(k+1)}M_{i(k+1)}A'_{i(k+1)} + P_iM_{(k+1)i}^{-1}P_i{}' \right)$$

Notice also that $n_k = L_k S_{k+1} L_k'$ where

$$L_k := \left[ \bigoplus_{i=1}^{k} P_i \quad , \quad \bigoplus_{i=1}^{k} A_{i(k+1)} \right]$$

$$S_{k+1} := \left[ \bigoplus_{i=1}^{k} M_{(k+1)i}^{-1} \oplus \bigoplus_{i=1}^{k} M_{i(k+1)} \right]$$

From the assumption that $U_k + N_k \prec 0$, we have $U_k + N_k = U_k + L_k S_{k+1} L_k' + \bar{N}_{k+1} \prec 0$. Rearrange the terms:

$$-\left(U_k + \bar{N}_{k+1}\right) \succ L_k S_{k+1} L_k' \tag{3.9}$$

Next, let $D_{k+1}$ be the left hand side of constraint Eq. (3.6d) at $i = k+1$ with the summation truncated to be only over indicies $k+2$ to $n$ (as opposed to be over all indices other than $k+1$), i.e.,

$$\begin{aligned} D_{k+1} :=& A_{(k+1)(k+1)} P_{k+1} + P_{k+1} A'_{(k+1)(k+1)} \\ &+ \sum_{j=k+2}^{m} A_{(k+1)j} M_{(k+1)j} A'_{(k+1)j} \\ &+ \sum_{j=k+2}^{m} P_{k+1} M_{(k+1)j} P_{k+1} \end{aligned}$$

and let

$$T_{k+1} := [A_{(k+1)1}, A_{(k+1)2}, \ldots, A_{(k+1)k}, \underbrace{P_{k+1}, \ldots, P_{k+1}}_{\text{repeat k times}}]$$

Then by Schur complement, constraint Eq. (3.6d) with $i = k+1$ is equivalent to:

$$\begin{bmatrix} D_{k+1} & T_{k+1} \\ \\ T'_{k+1} & -S_{k+1} \end{bmatrix} \prec 0 \tag{3.10}$$

Use Schur complement on Eq. (3.10) again, this time from the opposite direction, it is also equivalent to:

$$S_{k+1} \succ -T'_{k+1} D_{k+1}^{-1} T_{k+1} \tag{3.11}$$

Because $P_i \succ 0, \forall i$, $L_k$ has full row-rank, then pre- and post-multiplying Eq. (3.11) with $L_k$ and $L_k'$ preserves the positive definite order:

$$\begin{aligned} L_k S_{k+1} L_k' &\succ -L_k T_{k+1}' D_{k+1}^{-1} T_{k+1} L_k' && (3.12a) \\ &= -\tilde{U}_{k+1} D_{k+1}^{-1} \tilde{U}_{k+1}' && (3.12b) \end{aligned}$$

The last equality is due to

$$L_k T_{k+1}' = \begin{bmatrix} A_{1(k+1)} P_{k+1} + P_1 A_{(k+1)1}' \\ A_{2(k+1)} P_{k+1} + P_2 A_{(k+1)2}' \\ \vdots \\ A_{k(k+1)} P_{k+1} + P_k A_{(k+1)k}' \end{bmatrix} = \tilde{U}_{k+1}$$

Finally, combining Eq. (3.9) and Eq. (3.12), we have

$$-\left(U_k + \bar{N}_{k+1}\right) \succ -\tilde{U}_{k+1} D_{k+1}^{-1} \tilde{U}_{k+1}' \tag{3.13}$$

By again taking Schur complement, this is equivalent to:

$$\begin{bmatrix} U_k + \bar{N}_{k+1} & \tilde{U}_{k+1} \\ \\ \tilde{U}_{k+1}' & D_{k+1} \end{bmatrix} = U_{k+1} + N_{k+1} \prec 0 \tag{3.14}$$

which completes the induction. □

## Dual Proof of Theorem 1

We'll make use of Theorem of Alternatives [7] below.

**Theorem 3.** *Let $\mathcal{V}$ (resp. $\mathcal{S}$) be a finite-dimensional vector space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{V}}$ (resp. $\langle \cdot, \cdot \rangle_{\mathcal{S}}$). Let $\mathcal{A} : \mathcal{V} \to \mathcal{S}$ be a linear mapping, and $\mathcal{A}^{adj} : \mathcal{S} \to \mathcal{V}$ be the adjoint mapping such that $\forall x \in \mathcal{V}$ and $\forall Z \in \mathcal{S}$, $\langle \mathcal{A}(x), Z \rangle_{\mathcal{S}} = \langle x, \mathcal{A}^{adj}(Z) \rangle_{\mathcal{V}}$, and let $A_0 \in \mathcal{S}$. Then exactly one of the two statements is true:*

*1. There exists an $x \in \mathcal{V}$ with $\mathcal{A}(x) + A_0 > 0$.*

*2. There exists a $Z \in \mathcal{S}$ with $Z \gneq 0$, $\mathcal{A}^{adj}(Z) = 0$, and $\langle A_0, Z\rangle_{\mathcal{S}} \leq 0$.*

The theorem can be intuitively thought of as a generalization of Farkas' Lemma to non-polyhedral convex cones. A complete proof can be found in [7]. Tailored to our need, $\mathcal{V}$ and $\mathcal{S}$ are taken as the cone of positive semidefinite matrices equipped with $\langle A, B\rangle = trace(AB)$, $>$ (resp. $\geq$) therefore means $\succ$ (resp. $\succeq$), and $Z \gneq 0$ means $Z \succeq 0, Z \neq 0$.

*Proof.* Now, let us first match Eq. (3.5) to the first statement to get the linear mapping $\mathcal{A}$ and the adjoint. Then Eq. (3.5) is infeasible if and only if there exists a $Z$ with the same size and partition of $A$ (as well as $P$) such that:

$$Z \gneq 0 \tag{3.15a}$$

$$Z_{ii}A_{ii} + A'_{ii}Z_{ii} + \sum_{\substack{j=1 \\ j\neq i}}^{m} A'_{ij}Z_{ij} + Z'_{ij}A_{ij} \gneq 0, \forall i \tag{3.15b}$$

Finding the alternative of Eq. (3.6) is less straightforward. It turns out it is easier to work with an equivalent form of Eq. (3.6):

$$P_i > 0, M_{ij} > 0, \forall i, j, i \neq j \tag{3.16a}$$

$$P_iA_{ii} + A'_{ii}P_i + \sum_{\substack{j=1 \\ j\neq i}}^{m} P_iA_{ij}M_{ij}A'_{ij}P_i + M_{ji}^{-1} < 0, \forall i \tag{3.16b}$$

We then match Eq. (3.16) also to the first statement. It is infeasible if and only if there exists a set of $\{T_i\}_{i=1}^{m}$ where each $T_i$ is with the same size and partition of $A$

(as well as $P$) such that:

$$T_i \gneq 0, \forall i \tag{3.17a}$$

$$A_{ii}T_{iii} + T_{iii}A'_{ii} + \sum_{\substack{j=1 \\ j\neq i}}^{m} T_{iij}A'_{ij} + A_{ij}T'_{iij} \gneq 0, \forall i \tag{3.17b}$$

$$T_{iii} \leq T_{j_{ii}}, \forall i, j, i \neq j \tag{3.17c}$$

It is clear that the feasibility of Eq. (3.15) implies that of Eq. (3.17) because one can simply let $T_i = Z^{-1}$, $\forall i$ (the reverse implication does not necessarily hold; it is possible the set $\{T_i\}_{i=1}^{m}$ can not be 'squashed' into a single $Z^{-1}$). Therefore, the infeasibility of Eq. (3.5) implies the infeasibility of Eq. (3.16) which is equivalent to Eq. (3.6). Flipping both sides of the last statement, Eq. (3.6) is feasible implies Eq. (3.5) is feasible. $\square$

# Chapter 4

# Sampling Quotient-Ring SOS Programs

This chapter is adapted from work previously published in [66].

To connect the dots, three limitations in the compositional work presented in Chapter 3 motivated this work. In particular, we would like to i) avoid implicitly assuming (non-general) system structure; ii) not sacrifice solution quality, and most importantly, iii) address the "true" culprit that creates large-scale SOS/SDP programs in the first place.

## 4.1 Introduction

We consider the fundamental verification problem of region-of-attraction (ROA) approximations for polynomial, polynomial with generalized Lur'e uncertainty, and multi-rigid-body systems. Sum-of-squares programs are widely accepted as a standard approach to this problem. Powered by semidefinite program, SOS provides a systematic way to optimize over polynomial Lyapunov functions' sub-levelsets for these approximation tasks [51, 55].

Despite the popularity and rich theories, the problems solved by these approaches are still of only modest dimension (10–15 states) [41]. This is limiting, as many interesting real-world applications, e.g., mechanical systems consisting of many rigid

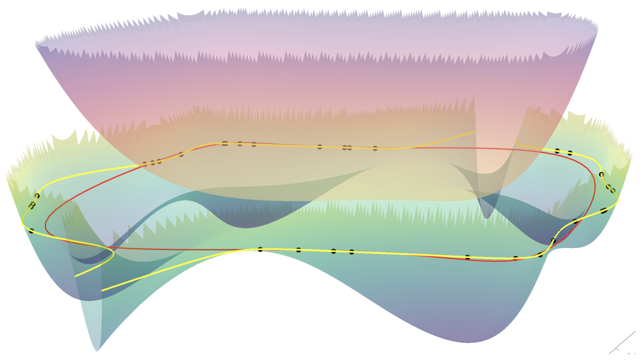


Figure 4-1: The proposed method significantly reduces *both* formulation and computation overhead. One resulting improvement is visualized above on the ROA approx. of the Van der Pol. Traditional methods typically involve conditions on, e.g., the set of all states enclosed within the yellow line, and solve an optimization globally. Our method, *provably* correct and less conservative, only needs to examine few random samples, shown as blue dots, on the yellow line.

bodies, are well beyond that scale.

What, then, could be causing the scalability challenge?

Typical scale-improving techniques, rightly so, identifies the low-level SDPs as a computational bottleneck. However, the SDPs are far from the only issue; in fact, we argue that they are, to a large extent, a scapegoat for the inefficient high-level problem formulations.

Specifically, traditional formulations heavily rely on the recipe of (in)equality implication, S-procedure, and auxiliary high-degree Lagrange multipliers. These multipliers not only introduce a large number of auxiliary decision variables and possibly extra expensive constraints, they inflate the problem dimension or degree as well, all of which responsible for creating bloated SDPs. If the dynamics are not exactly polynomial, like the Lur'e-uncertain or rigid-body dynamics that we consider, auxiliary indeterminates are additionally necessary, aggravating the complexity even further [54].

Motivated to eliminate all these multipliers (and most of the auxiliary indeterminates), we exploit inherent system properties — continuity in polynomial, convexity in Lur'e uncertain, and implicit algebraic structure in rigid-body systems — and reformulate the ROA approximation problems as quotient-ring SOS programs. These are programs that directly reason on algebraic varieties (objects defined by polynomial equations; for example, the yellow line in Figure 4-1 is a variety) without relying on multipliers. Basic algebraic geometry properties imply these reformulated programs are smaller, sparser, less constrained, yet less conservative.

The computation of the new quotient-ring SOS programs is further improved, significantly, by leveraging a sampling algebraic variety approach. The method, recently introduced in [20], reduces a quotient-ring SOS program to sampled instances on the defining variety, resulting in small SDPs with low-rank data and better numerical conditions. Remarkably, solution correctness is guaranteed with just a finite (in practice, very small) number of such samples.

Combining the new formulations and sampling, the proposed method can verify systems well beyond the reach of existing SOS-based approaches (32 states). On smaller problems where a baseline is available, it computes tighter solution 2-3 orders faster.

Finally, while this paper focuses on ROA verification, extensions to the closely related problems such as reachability analysis or barrier certification [55] are immediate.

Our general contributions are:

(i) We present three new quotient-ring SOS programs, one for each ROA approximation problems considered. Different inherent system structures are exploited, all leading to *smaller* yet *stronger* formulations.

(ii) We apply the efficient sampling variety approach from polynomial optimization to the context of general nonlinear system verification.

(iii) More specifically for the Lur'e-type uncertain systems, our new formulation is novel, independent from the application of the underlying quotient-ring structure used for ROA analysis.

To our knowledge, all of these are proposed for the first time.

### 4.1.1 Related Work

SOS programs have had success in verification for a wide variety of systems and tasks, from polynomial to hybrid, deterministic to stochastic, and stability to robustness to safety [51, 55, 70, 9, 54], and not only theoretically but demonstrated on hardware [41] as well.

All these work essentially follow a standard pipeline, as illustrated in Fig. 4-2. While we follow the same pipeline, our ingredients differ from the beginning, e.g. our Lyapunov or Lur'e conditions are not the usual inequality implications. These new conditions lead us to smaller yet stronger quotient-ring SOS programs; multipliers, traditionally needed to segue to conformed convex and polynomial but bloated programs, are thereby eliminated entirely.

These new quotient-ring SOS programs also allow us to take advantage of [20] and improve the downstream computation differently. Existing methods commonly assume special structures such as compositional [65], admitting low-rank solution [15], or chordal sparse [44] in SDPs, or symmetric or sparse in polynomials [52, 53]. Other methods, while general, either approximate the semidefinite cone with linear or second-order cones [41], or rely on first-order methods such as the augmented Lagrangian [62]; scalability are therefore achieved at the cost of conservatism or accuracy.

In contrast, via sampling, [20] exploits the inherent geometric structure in our quotient-ring SOS programs. It constructs orthogonal and low-dimensional (implicit) Gröbner basis, and produces SDPs that are small, better conditioned, and of low-rank (data, not solution). Remarkably, efficiency is significantly increased, without sacrificing the program's generality, correctness, and less-conservatism.

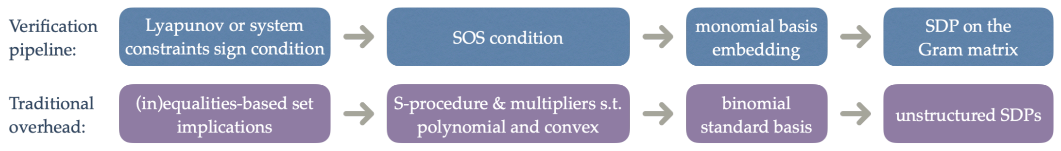


Figure 4-2: Standard SOS-based verification pipeline and the traditional overhead. We follow the same pipeline but use different ingredients *throughout*. Thus, unlike most scale-improving methods that are SDP-oriented, we reduce all these overhead.

## 4.2 Problem Statement and Approach

Given a continuous-time closed-loop nonlinear system with dynamics $\dot{x} = f(x)$ and a fixed positive definite polynomial Lyapunov candidate $V(x)$, we consider the task of quantitatively verifying if the system is locally asymptotically stable around the fixed point (assumed to be the origin). Concretely, we are interested in finding a sub-levelset $\mathcal{E}(V, \rho) := \{x \mid V(x) < \rho\}$ whose volume grows with $\rho$. The connected component of $\mathcal{E}$ that includes the origin is an inner approximation of the ROA if the constraint in

$$\begin{aligned} \max \quad & \rho \\ \text{s.t.} \quad & \dot{V}(x) = \frac{\partial V}{\partial x} f(x) < 0, \ \forall x \in \mathcal{E}(V, \rho) \setminus \{0\}, \end{aligned} \tag{4.1}$$

is satisfied. The cost on $\rho$ encourages enlarging the sub-levelset, thus providing a tighter approximation.

We consider solving the ROA approximation on three sub-problems; they differ in the dynamics characterization.

*Polynomial problem*: the "vanilla" case where $f(x)$ is polynomial in $x$.

*Lur'e problem*: The dynamics is a nominally polynomial $f_0(x)$ subject to additive uncertainty

$$f(x) = f_0(x) + \delta(x),$$

where $\delta(x)$ satisfies a generalized Lur'e type condition $(\alpha(x) - \delta(x))(\beta(x) - \delta(x)) \leq 0$ with $\alpha(x)$ and $\beta(x)$ both polynomial in $x$. Note that $\delta(x)$ may not be polynomial.

*Rigid-body problem*: The dynamics of rigid-body mechanical system, which come from the equations of motion, are given as

$$f(x) = M^{-1}(x)F(x),$$

where both $M(x)$ and $F(x)$ include terms like $\sin(x)$, thus $f(x)$ is rational trigonometric.

The overall approach in this paper is two-pronged: reformulate the three ROA verification problems as simpler yet stronger quotient-ring SOS programs, and apply the efficient sampling algebraic variety method to solve them.

We begin by describing the complete solution to the polynomial problem. For Lur'e and rigid-body problems, we focus on illustrating their tailor-made formulations only, as the sampling subroutine is identically applied.

## 4.3 Formulation - Polynomial Problem

### 4.3.1 Existing Formulations

There are two known SOS programs formulations for the polynomial problem. The more popular one, which we call program (IE), is based on a straightforward inequality implication $V \leq \rho \implies \dot{V} \leq 0$, S-procedure and multipliers:

$$\text{(IE)} \qquad \begin{aligned} \text{find} \quad & \lambda(x) \\ \text{s.t.} \quad & \lambda(x)(V(x)-\rho) - \dot{V}(x) \text{ is SOS} \\ & \lambda(x) \text{ is SOS} \end{aligned}$$

Note that, this is a feasibility program and requires a line-search of a fixed $\rho$ since otherwise the program would be bilinear (non-convex) in $\rho$ and $\lambda$.

An alternative equality constrained formulation can be found in [51, 41]. In particular, under the assumption that the Hessian of $\dot{V}$ is negative definite at the origin, the following is also sufficient for problem Eq. (4.1):

$$\text{(E)} \qquad \begin{aligned} \max_{\rho, Q, \lambda(x)} \quad & \rho \\ \text{s.t.} \quad & (x'x)^d \left(V(x)-\rho\right) - \lambda(x)\dot{V}(x) = m'(x)Qm(x) \ \forall x \end{aligned}$$

Here we explicitly write out the SOS factorization constraint on the right-hand side (for easy reference later); $m(x)$ denotes the standard monomial basis of appropriate degree.

Both formulations need to optimize over auxiliary multipliers $\lambda(x)$. When the $\lambda$ are of the same degree choices, the SOS programs translate to SDPs of similar dimension and lead to similar optimal $\rho$. However, the equality constrained (E) is much simpler to solve due to the elimination of the SOS condition on the multipliers and the line-search.

## 4.3.2 Proposed Formulation

The proposed formulation is closely related to (E). However, since it was given in the references without proof, some important and subtle questions were left un-addressed. For example, what is the formulation based on? and what is the purpose of the $(x'x)^d$ term? To answer these, we first reverse-engineer the formulation to discover its underlying implication, described below.

**Theorem 4.** *Under the assumption that the Hessian of $\dot{V}$ is negative definite at the origin, the implication condition*

$$\dot{V}(x) = 0 \implies V(x) \geq \rho \text{ or } x = 0 \tag{4.2}$$

*is a sufficient condition for Eq.* (4.1).

*Proof.* The Hessian condition ensures that $\dot{V}(0) = 0$ is a local maximum. Therefore, locally around the origin, we must have $\dot{V} < 0$. If $\dot{V}$ is negative definite, the system is globally asymptotically stable, and Eq. (4.2) gives $\rho = \infty$ which in turn correctly implies Eq. (4.1). The more interesting case is when the system is locally stable, implying that at some states $\dot{V} > 0$. Since $V$ and $f$ are both polynomial by assumption, so is $\dot{V}(x)$ and it is thus continuous. Given this continuity, and that $\dot{V}$ changes sign eventually, zero-crossing event(s) must have occurred at some states.

If at all such states where $\dot{V}(x) = 0$, the evaluation of $V \geq 0$ or it is precisely the origin, as encoded by Eq. (4.2), then by contraposition, it is equivalent to

$$x \in \{x \mid V(x) < \rho, x \neq 0\} \implies \dot{V}(x) \neq 0$$

Given the local behavior of $\dot{V}$ around the origin, the connected component of the $\rho$ sub-levelset that includes the origin, must have $\dot{V} < 0$ (except for the origin itself). □

**Remark 8.** *Note that a discrete-time counterpart to Theorem 4 can be straightforwardly stated and similarly proven as: The condition*

$$\Delta V(x) = V(f(x)) - V(x) = 0 \implies V(x) \geq \rho \text{ or } x = 0 \tag{4.3}$$

*is a sufficient for the sub-levelset $\mathcal{E}(V, \rho) := \{x \mid V(x) < \rho\}$ to be an inner-approximation of the true ROA to the system with dynamics $x^+ = f(x)$.*

An interactive visualization of the proof idea is available online[1]. Figure 4-1 shows a snapshot of it, where the yellow line precisely defines those important non-origin zero-crossings $\dot{V}(x) = 0$.

With Theorem4 in place, it should be clear that Formulation (E) is a multiplier-based sufficient condition for Eq. (4.2), therefore sufficient for Eq. (4.1) as well. Note the importance of the negative definite Hessian condition[2], it sufficiently implies the local maximum condition needed in the proof. Note also the importance of the $(x'x)^d$ term, where $d$ is a strictly positive integer user chooses. Without this term, the optimization (E) is meaningless because it would always return the trivial solution $\rho = 0$. To see this, plug in $x = 0$. The left-hand side become $0 - \rho - 0$ which has to match a non-negative right-hand side; the maximal value of $\rho$ must be zero.

Our formulation is a direct application of algebraic geometry on Eq. (4.2), using basic objects such as affine variety, quotient ring, and Gröbner basis (due to space limitation, we prioritize making the high-level idea clear, and only define variety below, and refer to [21] Chapter 1 for the other background and definitions).

**Definition 1.** *Let $k$ be a field, and let $f_1, \ldots, f_s$ be polynomials in $k[x_1, \ldots, x_n]$. Then we set*

$$\mathcal{V}(f_1, \ldots, f_s) = \{(a_1, \ldots, a_n) \in k^n : f_i(a_1, \ldots, a_n) = 0 \text{ for all } 1 \leq i \leq s\}$$

*We call $\mathcal{V}$ the variety whose defining equations are $f_1, \ldots, f_s$.*

[1] http://web.mit.edu/shenshen/www/VDP-animation.html

[2] even though one reference does not make this assumption explicit and the other has a sign flip

In rough terms, a variety is a set of roots to polynomial equations. Then, simply by defining an algebraic variety $\mathcal{V} := \{x|\dot{V}(x) = 0\}$, a sufficient condition to Eq. (4.2) is given by the following quotient-ring SOS program:

$$
\text{(Q)} \qquad \begin{aligned} \max_{\rho, Q} \quad & \rho \\ \text{s.t.} \quad & (x'x)^d (V - \rho) = n'(x) Q n(x), \ \forall x \in \mathcal{V} \end{aligned}
$$

where $n(x)$ is a Gröbner basis.

(Q) and (E) may seem trivially equivalent and only differ in terminology; after all, they stem from the same high-level polynomial equality constraint Eq. (4.2). However, there are four facts that make the reformulation (Q) more appealing.

(i) The decision variable $\lambda(x)$ is eliminated

(ii) The basis $n(x)$ in (Q) is of lower dimension than $m(x)$ in (E), due to Gröbner basis (see [21], Chapter 2);

(iii) The fixed degree $d$ can be lower in (Q), due to the elimination of the $\lambda(x)\dot{V}(x)$ term;

(iv) (Q) is intrinsically stronger than (E), i.e., optimal solution of (E) is in general only suboptimal to (Q).

The last fact is important but subtle. It is due to that (E) relies on degree-bounded multiplier whereas (Q) relies on geometric description of the variety. An explicit example to make this distinction clear:

**Degree-bounded multipliers are "bounded"** Suppose we need to check if this implication $x + 1 = 0 \implies x^2 - 1 \leq 0$ is true. Multiplier-based formulation would search for a $\lambda(x)$ such that $(x^2 - 1) + \lambda(x)(x + 1) \leq 0, \forall x$. This optimization can not be feasible if $\lambda$ is limited to be a constant, even though the implication is true. It takes at least an affine multiplier, for example $\lambda(x) = -(x - 1)$ to make the problem feasible. In contrast, quotient-ring formulation interprets the left-hand side of the implication as $x \equiv -1$, so the right-hand side becomes $1^2 - 1 = 0 \leq 0$ which is trivially true.

To recap, facts (i)-(iii) mean that the quotient-ring SOS program (Q) leads to a much smaller SDP; yet it is also stronger (unless the multipliers can be of infinite degree) due to fact (iv). Therefore, (Q) is a strictly better formulation, in theory. The only downside, in practice, is that Gröbner basis themselves may be challenging to find, especially when the defining equations for the variety get complicated or high-dimensional. To overcome this potential difficulty, we leverage an efficient sampling-based method.

## 4.4 Sampling on Algebraic Varieties

We apply the sampling algebraic varieties method introduced in [20] to solve the quotient-ring problem (Q). The high-level idea is rather straightforward: instead of solving the optimization for all real-valued $x$ with Gröbner basis, solve it at only a set of sampled numerical instances $\{x_i\}$:

$$\text{(S)} \qquad \begin{aligned} \max_{\rho, Q} \quad & \rho \\ \text{s.t.} \quad & \dot{V}(x_i) = 0, \ \forall x_i \\ & \left(x_i' x_i\right)^d \left(V(x_i) - \rho\right) = \tilde{n}'(x_i) Q \tilde{n}(x_i), \ \forall x_i \end{aligned}$$

using $\tilde{n}(\cdot)$, which *can be* a standard basis or an implicit Gröbner basis; this is to be described later.

As hinted, there are certain numerical benefits of solving the sampled version (S). But given that the ultimate goal is to produce stability certificate, we should be immediately asking: a solution to (S) is necessarily a solution to (Q), does there exist guarantee regarding sufficiency (as required to claim correctness)? Also, what is the sample complexity and sampling procedure? The detailed answers and rigorous treatments can be found in [20], we include a brief high-level summary for completeness.

### 4.4.1 Correctness Guarantee on Finite Samples

The sampled program (S) is equivalent to the original program (Q), with probability one, if the samples $x_i$ are generic. The genericity condition can be interpreted as checking if *enough* samples are drawn *randomly*. In theory, there exists a *finite* sample bound. This bound depends on many factors including the problem size, variety structure, etc. Concretely, genericity is checked by a simple rank test of a matrix whose elements are simple monomial evaluations at the samples.

Through this practical case-by-case numerical rank check, we accumulated enough empirical evidence that the samples needed are in fact, very small. Usually, this number is far less than the number of elements in the Gram matrix. In Section 4.7, we document the number of samples used for each program, which could serve as an empirical reference.

An intuitive explanation might be helpful; after all, "with probability one guarantee" is usually stated in the asymptotic regime. Very loosely, here, the combination of "being exactly on the variety" and "degree-bounded polynomial parameterization" imposes a constraint so strong that finite samples are capable of capturing it. To some extent, it is similar to polynomial interpolation, where a finite number of samples can faithfully recover the coefficients of a degree-bounded polynomial.

We finally point out that, the sampling procedure itself involves finding roots to polynomial equation(s). In the simple case where dynamics itself is polynomial, sampling means finding roots of a single multi-variate polynomial $\dot{V}$, which can be easily done via open-source packages (in our case, we use shooting method and numpy). As the variety gets more complicated (usually so when having more defining equations), so will the sampling process. Fortunately, sampling is a trivially parallelizable process, where each thread only comes with very low processing and memory requirement.

### 4.4.2 Computational Benefits

The computational gains come from the paradigm shift: whereas traditional methods match polynomial coefficients, sampled approach matches polynomial evaluations.

One direct consequence is the low-dimensional numerical basis $\tilde{n}(x_i)$. First, this basis *can* be chosen as *standard* monomials evaluations, because the generic samples numerically capture the underlying variety. In comparison, problem (Q) must rely on explicit Gröbner basis to symbolically encode the variety. This in and of itself is a huge improvement.

$\tilde{n}(x)$ can be further simplified (from e.g. the standard basis) to an even lower-dimensional implicit Gröbner basis by leveraging the underlying variety and a simple SVD procedure. These implicit basis can be thought of as the orthogonal basis with respect to a natural inner product supported on the samples. Orthogonalization, as a byproduct of this size-reduction procedure, has been shown to improve SDP numerical condition as well [38].

Finally, (S) results in an SDP with low-rank data structures, which can be readily exploited by solvers. Note that the right-hand side of (S) is a scalar evaluation. Via the trace cyclic property, $\tilde{n}'(x_i)Q\tilde{n}(x_i) = \operatorname{tr}(Q, \tilde{n}(x_i)\tilde{n}'(x_i))$, where $\tilde{n}(x_i)\tilde{n}'(x_i)$, the problem data in the SDP, is of rank at most one by construction (because recall that $\tilde{n}(x_i)$ is a numerical vector). Such low-rank data does not appear in traditional SOS programs, since $n(x)$ there are symbolic monomials. Note that it is the problem data (rather than the decision variable $Q$) that is of low-rank, which a lot of solvers can readily take advantage of.

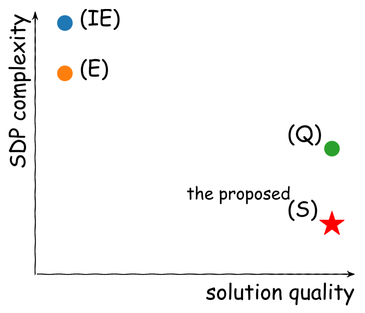


Figure 4-3: Qualitative comparison of the four programs.

**Comparison of the four SOS programs.** We have presented four different SOS programs for the polynomial ROA problem. Figure 4-3 summarizes a qualitative comparison of the solution quality and underlying SDP complexity. Note that the relative scale of the gap varies case-by-case. For example, on very simple problems, all programs might be overkill and arrive at the same solution. On the other hand, the proposed method achieves more significant computational gain for more complex systems; Section 4.7 includes these quantitative comparisons.

## 4.5 Formulation - Lur'e Problem

Consider dynamics with generalized Lur'e uncertainties:

$$f(x) = f_0(x) + \delta(x) \tag{4.4}$$

where $f_0(x)$ is the nominal polynomial dynamics; the uncertainty $\delta(x)$ satisfies $(\alpha(x) - \delta(x))(\beta(x) - \delta(x)) \leq 0$, where $\alpha(x)$ and $\beta(x)$ are both polynomial (generalized from the standard linear). A one-dimensional example of $\delta(x)$ is visualized in Figure 4-4.

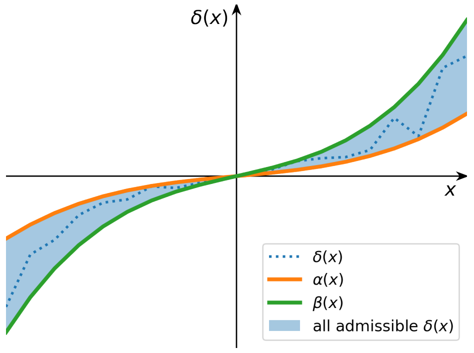


Figure 4-4: Generalized Lur'e type sector uncertainty. $\alpha(x)$ and $\beta(x)$, both polynomial, define the "boundaries" of the sector; the uncertainty $\delta(x)$ can take any function "in between".

**Existing Formulation** For Lur'e problem, we would like to eliminate a standard S-procedure dedicated to encoding the uncertainty $\delta$. This is a separate issue / overhead from those arising from encoding sub-levelset (discussed in Section 4.3). To isolate the two and highlight the new improvements here, we first present the standard and proposed formulations for global analysis. Local extension is discussed later.

The standard way to verify global asymptotically stable (g.a.s.) is via an IQC-type treatment [46]:

$$\begin{aligned}
\text{find} \quad & \xi(x,\delta) \\
\text{s.t.} \quad & \xi(x,\delta) > 0 \\
& \underbrace{\frac{\partial V}{\partial x}(f_0(x)+\delta)}_{\dot{V}(x,\delta)} - \underbrace{\xi(x,\delta)(\alpha(x)-\delta)(\beta(x)-\delta)}_{\text{S-procedure encoding } (x,\delta) \text{ dependency}} < 0,
\end{aligned} \tag{4.5}$$

where $\delta$ is an auxiliary indeterminate, independent from $x$ (thus the notation $\dot{V}(x,\delta)$); its true dependency on $x$ is incorporated using multiplier $\xi(x,\delta)$ and the S-procedure.

**Proposed Formulation** The proposed formulation is *simpler* yet *stronger* (in aspect different from quotient-ring structure): it eliminates the auxiliary multiplier $\xi(x,\delta)$ and indeterminate $\delta$, and allow us to analyze all admissible dynamics $f$ (defined in Eq. (4.4)) by examining the boundaries $f_\alpha := f_0(x) + \alpha(x)$ and $f_\beta := f_0(x) + \beta(x)$ with a less conservative condition.

**Lemma 2.** *For a given positive definite $V(x)$, define*

$$\begin{cases} \dot{V}_\alpha(x) := \frac{\partial V(x)}{\partial x} f_\alpha < 0 \\ \dot{V}_\beta(x) := \frac{\partial V(x)}{\partial x} f_\beta < 0, \end{cases} \tag{4.6}$$

*then* (4.5) $\implies$ (4.6) $\implies$ *g.a.s.* $\not\Longrightarrow$ (4.6) $\not\Longrightarrow$ (4.5) *(slight abuse of notation here,* (4.5) *denotes that the optimization is feasible).*

*Proof.* (4.5) $\implies$ (4.6): If (4.5) holds, it holds for all admissible $\delta$. It must hold when $\delta = \alpha(x)$, plug this in and (4.5) reduces to exactly $\dot{V}_\alpha < 0$. Similarly $\dot{V}_\beta < 0$ is implied.

(4.6) $\implies$ g.a.s.: First, note that for any *fixed* $x$, $\delta(x)$ can be written as a *convex* combination of $\alpha(x)$ and $\beta(x)$. Second, $\dot{V}(x) = \frac{\partial V}{\partial x}(f_0(x) + \delta(x))$ is linear with respect to $\delta(x)$. Combining the two observations, $\dot{V}$ can also be written as a convex combination of $\dot{V}_\beta$ and $\dot{V}_\alpha$ (with generally state-dependent combination coefficients). Therefore, (4.6) implies $\dot{V}(x) < 0$ for all admissible $f$.

g.a.s $\not\Longrightarrow$ (4.6): Well-known fact. Finally, (4.6) $\not\Longrightarrow$ (4.5) is a consequence of the multiplier limitation (Section 4.3). □

Lemma 2 shows that (4.6) is *stronger* than (4.5). Also, in terms of computation, not only is the multiplier $\xi(x, \delta)$ and S-procedure eliminated, so is the auxiliary indeterminate $\delta$. These simplifications are preserved, and also combined with those in Section 4.3 (e.g. elimination of multipliers $\lambda$) when extended to local analysis via the following.

**Remark 9.** *It is worth pointing out that the Lemma 2 and its proof do not generally translate directly into the discrete-time (DT) case. This is because the Lyapunov difference $\Delta V = V(x^+) - V(x)$ (the DT counterpart of $\dot{V}$) is* not *linear in the $\delta$; for example, if $V(x)$ is quadratic in $x$, then $\Delta V$ is quadratic in $\delta$. So unless $V(x)$ itself is linear/affine in $x$ (or very precise cancellation happens for very particular $f$ and $V$ pair), $\Delta V$ can not be. However, a linear/affine $V$ cannot be positive definite (unless it is trivially a constant, which is never a valid Lyapunov function).*

**Theorem 5.** *Given a positive definite $V(x)$, if program (S) is feasible for dynamics $f_\alpha$ and $f_\beta$, with optimal solutions $\rho_\alpha$ and $\rho_\beta$, then $\dot{V}(x) < 0$, $\forall x \in \mathcal{E}\left(V, \underline{\rho}\right) := \{x \mid V(x) \leq \min(\rho_\alpha, \rho_\beta)\}$, and for all admissible $f$ defined by Eq.* (4.4).

*Proof.* Given Theorem 4, the optimal solutions of (S) imply that $\dot{V}_\alpha(x) < 0$, $\forall x \in \mathcal{E}\left(V, \underline{\rho}\right)$, and similarly for $\dot{V}_\beta$. For all admissible $f$, $\dot{V}(x) < 0$ within this set can be almost identically proved as "(4.6) $\Longrightarrow$ g.a.s." part in Lemma 2. □

Finally, the new formulation affords additional high-level insights into application too; details in Section 4.7.2.

## 4.6 Formulation - Rigid-Body Problem

Multi-body rigid-body systems are challenging to analyze, not only because of their often large scale, but more importantly because the Equations of Motions (EoM) and rotation and potential energy make their dynamics complicated rational trigonometric. Shown in Figure 4-5 is an N-link pendulum on a cart system, whose dynamics exhibit such characteristic. We use it to illustrate more explicitly the challenge and the existing and proposed approach.

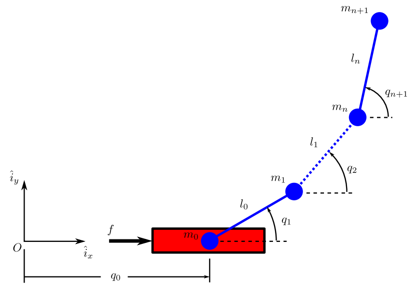


Figure 4-5: N-link pendulum on a cart. [3]

Without loss of generality, assume $N = 1$ (the classical cart-pole). Also, suppose the cart and pole both have unit mass and the pole has unit length.

Let the states be $y := [q_0, q_1, \dot{q}_0, \dot{q}_1]$, where $q_0$ is the cart position, $q_1$ the pole angle, and $\dot{q}_0, \dot{q}_1$ the corresponding velocities. Let the force on the cart be $u$ and the gravitational constant be $g$. The EoM is:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & -\sin q_1 \\ 0 & 0 & -\sin q_1 & 1 \end{bmatrix}}_{M(y)} \underbrace{\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \ddot{q}_0 \\ \ddot{q}_1 \end{bmatrix}}_{\dot{y}} = \underbrace{\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_1^2 \cos q_1 + u \\ -g \cos q_1 \end{bmatrix}}_{F(y,u)} \tag{4.7}$$

where $M(y)$ is the (positive definite) mass matrix, $F(y, u)$ the force matrix, and $\dot{y}$ the dynamics. Since $\dot{y} = M^{-1}F$, the dynamics is explicitly rational trigonometric.

**Existing Formulations** Taylor expansion is commonly used to handle the rational trigonometric dynamics. This approach has two major limitations. One is the error; or the complication introduced by error-bounding. The other limitation is Taylor expansion's own scalability; expansion can be challenging when the dimension or

[3]figure taken from the python PyDy package.

degree gets non-trivial. For example, to expand the dynamics of the N-linked cart example to third order, Sympy fails at link 4, whereas Matlab fails at link 5. If higher-orders are needed to reduce the error, the scalability is even worse.

A "less lossy" transformation-type technique based on [50] has also been applied [54, 9]. This technique deals with the dynamics in two steps. First, a change-of-variables recasting technique can turn the trigonometric components into polynomials and the dynamics are simplified as rationals. The second step is to clear the rationals' denominator, perhaps not surprising by now, via multipliers, which carry all the complications discussed so far.

**Proposed Formulation** The proposed method to deal with the rational trigonometric dynamics is a combination of change of variable, differential algebraic equations (DAE), and implicit algebraic variety.

We start with the standard change of variable. Let $x := [q_0, s_1, c_1, \dot{q}_0, \dot{q}_1]$ where $s_1 \equiv \sin q_1$ and $c_1 \equiv \cos q_1$; suppose a feedback controller $u(x)$ is given to close the loop. The new coordinates first must satisfy the unit circle condition $s_1^2 + c_1^2 = 1$, which is equivalent to $x'Sx = 1$ where $S = \text{diag}(0, 1, 1, 0, 0)$.

Secondly, simple variable substitution of the recast states $x$ into the EoM Eq. (4.7) gives $M(x)\dot{y} = F(x)$. This is the key step, where we do not explicitly write out the $\dot{y}$ but rather leave it *implicit*. In other words, instead of dealing with ordinary differential equations, we describe the dynamic via DAEs, which are in fact more general as well.

Thirdly, due to the coordinate transformation, the dynamics of the recast $\dot{x}$ follows a transformation from $\dot{y}$ as:

$$\dot{x} = \begin{bmatrix} \dot{q}_0 \\ \dot{s}_1 \\ \dot{c}_1 \\ \ddot{q}_0 \\ \ddot{q}_1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_1 & 0 & 0 \\ 0 & -s_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{T(x)} \underbrace{\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \ddot{q}_0 \\ \ddot{q}_1 \end{bmatrix}}_{\dot{y}} = T(x)\dot{y}$$

where $T(x)$ is the recasting transformation matrix (purely dependent on $x$). Consequentially, the derivative of the Lyapunov function is then

$$\dot{V}(x) = \frac{\partial V}{\partial x}\dot{x} = \frac{\partial V}{\partial x}T(x)\dot{y} \tag{4.8}$$

With these preparations, sampling quotient-ring can be now readily applied. In the 'vanilla' case where the dynamics is polynomial, the variety has only one component that is $\dot{V} = 0$. Here, due to the recasting, the variety $\mathcal{V}$ is more involved $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \mathcal{V}_3$ where the defining equations are:

$$x'Sx = 1 \tag{4.9a}$$

$$M(x)\dot{y} = F(x) \tag{4.9b}$$

$$\frac{\partial V(x)}{\partial x}T(x)\dot{y} = 0 \tag{4.9c}$$

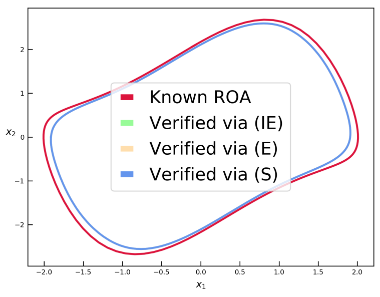
Note that $\dot{y} = [\dot{q}_0, \dot{q}_1, \ddot{q}_0, \ddot{q}_1]$ may seem necessary to be included in the SOS program basis. However, because the first half of the elements (first-order derivatives of $q$) is included in $x$, whereas the second half (second-order derivatives of $q$) is otherwise independent of $x$, the numerical samples via the variety capture all dependencies between $\dot{y}$ and $x$. In other words, it is not necessary for $\dot{y}$ to appear explicitly in the basis, which is another major advantage over multiplier-based formulation.

## 4.7 Experiments and Examples

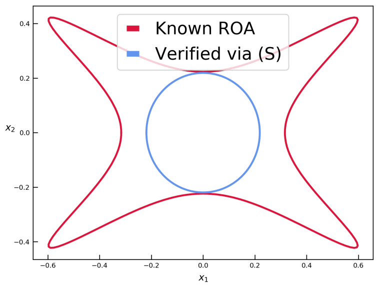
### 4.7.1 Polynomial Problems

We first consider three polynomial systems: Van der Pol oscillator, Ninja star, and Pendubot. Programs (IE), (E), and (S) are compared. Across all three examples, the proposed method (S) demonstrates speed improvement of up to 2-3 orders (Table 4.1). On Pendubot and Ninja star example, the proposed method also produces better solutions.

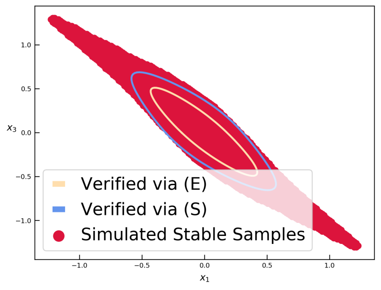
Figure 4-6: ROA approximations of polynomial systems. Qualitatively, for VanderPol, all three programs return identical result; for Ninja star, only the proposed method (S) succeeds; for Pendubot, the proposed method is tighter.



(a) Van der Pol



(b) Ninja Star



(c) Pendubot

| | Van der Pol | | | Ninja Star | | | Pendubot | | |
|---|---|---|---|---|---|---|---|---|---|
| | (IE) | (E) | (S) | (IE) | (E) | (S) | (IE) | (E) | (S) |
| PSD variable dim | 45 | 45 | 15 | 220 | 220 | 55 | 495 | 495 | 70 |
| num. scalar var. | 46 | 46 | 1 | 221 | 221 | 1 | 496 | 496 | 1 |
| num. constraints[a] | 165 | 152 | 29 | 540 | 632 | 78 | 4844 | 4840 | 118 |
| time (*sec*) | 0.09 | 0.07 | 0.01 | err[b] | err | 0.13 | err | 217.96 | 0.33 |

[a]equals the number of samples on the variety for method (S)

[b]"err" indicates the solver encounters numerical error

Table 4.1: Numerical comparison of three methods for ROA verification.

**Van der Pol** (Time-reversed) Van der Pol is a 2 state, degree 3 polynomial systems. It has a known ROA, and has thus been a staple benchmark. Using a degree 6 candidate $V$, all programs produce almost tight approximation (Fig. 4-6a), though the proposed method is the fastest.

**Ninja star** To showcase the efficacy of our method in numerically challenging situations, we purposely create a system with "badly conditioned" dynamics:

$$\dot{x}_1/16 = -25x_1^3 + 2500x_1^7 + 48x_2^2x_1 - 14400x_2^2x_1^5 + 28432x_2^4x_1^3 - 19200x_2^6x_1$$

$$\dot{x}_2/16 = -100x_2^3 + 40000x_2^7 + 48x_2x_1^2 - 4800x_2x_1^6 + 28432x_2^3x_1^4 - 57600x_2^5x_1^2$$

The system, with known true ROA that resembles a Ninja star shown in Figure 4-6, is low dimensional (2 states) but of high degree (7 degree). Further, the coefficients are very unbalanced (relative scale difference is $10^3$); and the dynamics linearization $A$ matrix at the origin are precisely zero. We supply a unit quadratic function $V = x'x$, therefore coefficients of $V$ and $\dot{V}$ are even more unbalanced than the those in the dynamics. Among the three programs, only the proposed succeeds at producing a result (and the result is tight).

**Pendubot** We take from [79] an LQR-controlled four dimensional Pendubot system, Taylor expanded to degree three at the fixed point. A degree six Lyapunov function is provided. Figure 4-6 shows the produced approximation at slice $(x_1, x_3)$. Our method not only produces better approximation, it does so about $10^3$ times faster (Table 4.1).

### 4.7.2 Lur'e Problem - Path-Tracking Dubins Vehicle

Consider a Dubins car defined in the error frame relative to the virtual vehicle along a path to be tracked, illustrated in Figure 6-2. The model is: $\dot{\psi}_E = u_1 - k_v \ell, \dot{X}_E = u_1 Y_E + u_2 - \ell \cos \psi_E, \dot{Y}_E = -u_1 X_E + \ell \sin \psi_E$ where $\psi_E, \dot{X}_E, \dot{Y}_E$ are the angle error and linear displacements, $l$ and $k_v$ are the target speed and path curvature, and $u_1$ and $u_2$ are the angular and linear torques. Stabilization at zero error means the car achieves perfect tracking.

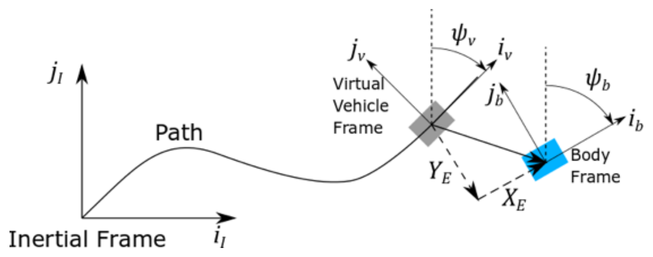


Figure 4-7: Path-tracking Dubins vehicle in the virtual error frame

An LQR controller is designed for a constant nominal tracking curvature $k_v = 1$. The true curvature can be between $[0.8, 1.2]$, and potentially time-varying. The task is to find an ROA approximation robust to this run-time parameter variation. Formulation presented in Section 4.5 is applied to this problem, which allows us to verify the robust ROA by checking the two extreme cases. The result is shown in Figure 4-8.

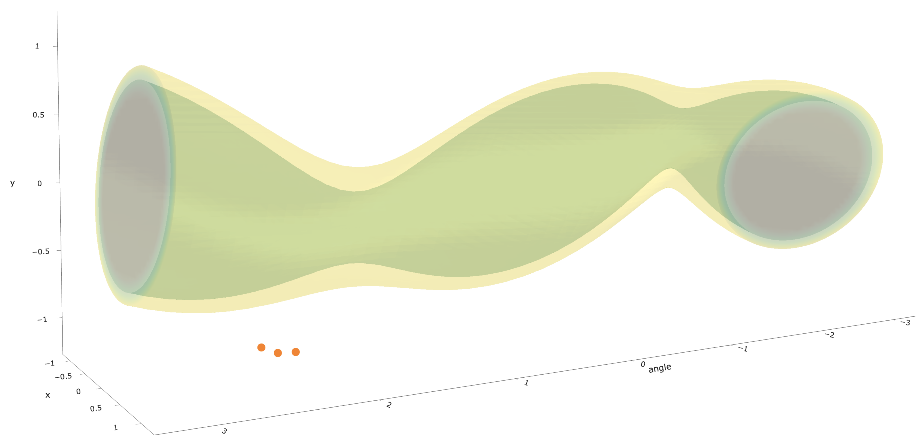


Figure 4-8: Robust ROA analysis for Dubins vehicle tracking a path of varying curvature. The yellow outer tube corresponds to $k_v = 0.8$ (straighter path). The green inner-tube corresponds to $k_v = 1.2$. The inner-tube is also the robust ROA for any $k_v$ varying within the given range. The red dots are counter-examples that do not converge to the origin; they show the tightness of the approximation.

We would like to point out an extra bit of insight the proposed method affords (which the existing formulation does not). Comparing the two ROAs produced for the extreme cases, the one corresponding to a straighter $k_v = 0.8$ path has larger volume than $k_v = 1.2$. In fact, we approximated the ROA for the nominal constant curvature $k_v = 1$ (note that this is not required for the robust ROA per se), and the result is sandwiched in between the two shown in Fig. 4-8 (so smaller than straighter-path one), even though the controller is designed based on this parameter. These findings agree with our intuition that tracking a "curvier" path feels dynamically more demanding, and it is interesting to have it, as a byproduct, emerging from the result.

### 4.7.3 Rigid-Body Problem - Cart with N-Link Pole

Consider the problem of N-link pendulum on a cart, illustrated in Figure 4-5. The system states are the position and (angular) velocity of the cart and the $N$ links. There are $N$ inputs, one is a force applied on the cart, and the rest are torques

Table 4.2: Numerical results of the ROA problem with different number of links on the cart.

| Link | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| dim$(x)$ | 5 | 8 | 11 | 14 | 17 | 20 | 23 | 26 | 29 |
| $\dim Q$[a] | 21 | 45 | 78 | 120 | 171 | 231 | 300 | 378 | 465 |
| cstrs[b] | 45 | 89 | 179 | 298 | 390 | 531 | 690 | 890 | 1390 |
| time $(s)$ | 0.03 | 0.48 | 23.67 | 91.87 | 178.78 | 338.56 | 478.37 | 598.20 | 723.03 |

[a]The matrix variable in the SDP is $\dim Q$-by-$\dim Q$.
[b]The number of constraints, which is the same as the number of samples.

applied on the $N-1$ links, starting from the attached on the cart (farthest one from the cart is not controlled directly). The task is to balance all the links upright.

We first produce an LQR controller and a quadratic Lyapunov function in the original coordinate. Then with small-angle approximations, and techniques introduced Section 4.6, transform them all into the recast $x$ coordinate. Figure 4-9 shows the ever-growing complexity disparity for the proposed method and its multiplier-based counterpart (E). Applying techniques detailed in, we are able to verify system of dimension 32, well beyond the scale of current SOS-based method. The numerical comparisons for different N are documented in Table 4.2.

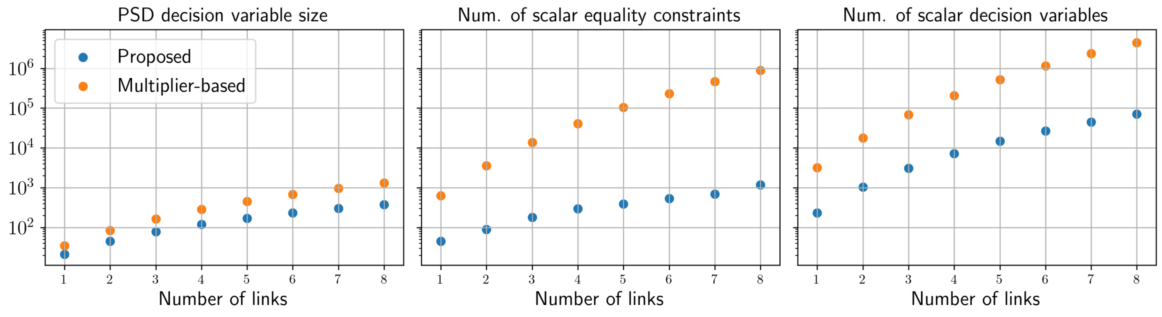


Figure 4-9: SDP complexity as the number of links in the N-link on cart system Figure 4-5 grows. Note the log-scale.

## 4.8 Discussion and Future Work

We presented a novel framework, combining smaller yet stronger problem reformulations and sampling, to address the scalability issues of SOS-based verification. The three new ROA formulations each relies on (different) intrinsic system structures, and are thus general. The subsequent quotient-ring SOS programs leverage geometric problem description rather than algebraic and are thus smaller, sparser, less constrained, yet less conservative. Their computation is further improved via the application of sampling variety method. Altogether, scale and speed are significantly improved.

Future work includes extending the techniques to more applications. A direct example is the multi-contact example described in [54]. Where the mass matrix can be handled via DAE technique as described in Section 4.6, contact condition can be simplified as described in Section 4.5.

An important related question we did not elaborate on here is how to find a high-quality Lyapunov candidate to feed into this new verification pipeline. This is the motivation of a sister project, which we discuss in Chapter 6.

# Part II

# Verification of Neural Networks Systems

Machine learning and neural networks have disrupted many engineering and scientific fields for their flexibility, simplicity and effectiveness. In the context of dynamical systems control, there has been a great deal of success using feedforward neural networks (FFNNs) or recurrent neural networks (RNNs) to mine controllers for challenging tasks such as grasping, autonomous driving, and walking.

Despite these successes, learning-based approaches have had limited applicability in the real world due to the lack of performance guarantees. Traditionally, these neural networks are tested on a large set of sample points for empirical performance evaluation, but since it is impossible to exhaustively test every "point" on the real line, let alone in the real world, what was not tested can never be truly trusted. Indeed, as evidenced by the so-called "adversarial attacks", even on a network with consistent high performance, a test input slightly disturbed can lead to a bafflingly different output. Placing this sort of solution as controller in a closed loop would imply that even if a neural network controller produces a safe trajectory from a given initial state, a tiny perturbation can cause the trajectory to become unsafe.

This is clearly concerning, and calls for *provable* certificates for *set*-based closed-loop behaviors, which can be exactly captured by the notion of stability and reachability. However, while there are rich tools for analyzing these properties, it is quite challenging to apply them directly to systems with machine learning components in the loop.

The challenges can be characterized by the complex system dynamics. It is well-known that an appropriately trained FFNN can approximate an arbitrary nonlinear function; modern RNNs are even more intricate due to their internal states and the interconnected gating mechanism introduced by the popular Long Short Time Memory (LSTM) design. In addition to these analytic challenges, since the neural networks achieve their performance by tuning a huge amount of parameters, their sheer sizes pose practical difficulty for computational tools. These are exactly the issues we propose to address in this part.

# Chapter 5

# Verification of Systems Modeled or Controlled by RNNs

This chapter, except for Method II (described in Section 5.4) and the corresponding results, has been presented publicly [64].

## 5.1 Introduction

Recurrent neural networks (RNNs) are a widely-used type of neural networks. Unlike feedforward networks (FFNNs), RNNs have internal dynamical cells, and are best known as the standard solution to learning tasks that require memory or inference, such as video captioning, text from/to speech or natural languages translations.

While may not be immediately obvious, RNNs is not a foreign concept in the context of control and identification; specifically, they are conceptually no different from the classical dynamic observers or dynamic output feedback compensators. These observers and compensators are the traditional approach to the well-known difficult partially observable problems; likewise, RNNs have been successfully applied for various such tasks [26], and substantially outperform static feedforward neural networks (FFNNs).

As tempting as it is to just throw an RNN at partially observable problems and claim it solved though, one should be cautious about the necessity of formally analyzing such systems, especially because, in direct contrast to their FFNN counterpart, formal verification or analysis on RNNs are scarce.

The lack of verification would certainly make sense if RNN were out of fashion, or intrinsically transparent to interpret, or immune to attacks. The reality, however, is quite the contrary. For sequential and temporal tasks, RNN remains the de facto standard and out-performs feedforward nets [49]; how to interpret and improve the RNN structure itself motivates an entire active research area [27]; and RNN has been shown to be quite vulnerable to adversarial attacks as well [17]. So then, why the void? What is the challenge and why cannot the techniques developed for feedforward nets simply be applied here?

In principle, an RNN can be unrolled into a sequence of feedforward blocks and then verified block by block using the existing methods, see Fig. 5-1; in practice, however, this is currently either impossible or impractical. The obvious reason is that the majority of the feedforward net verification work assumes ReLU activation [82, 32], and simply cannot handle the family of sigmoidal activations used in the prevailing RNN structure, the LSTMs. And even if we were to assume ReLU is good for RNN, direct application of feed-forward techniques is still, subtly, impractical.

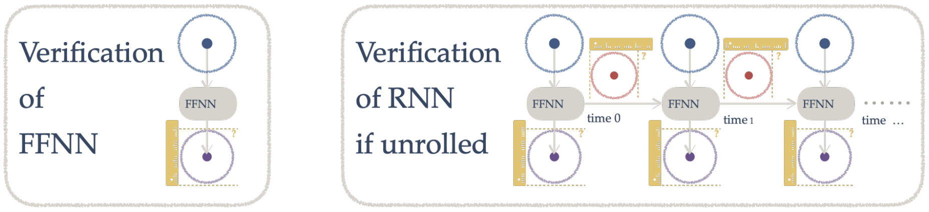


Figure 5-1: Standard verification setup for FFNN (left) and direct application of this idea to an unrolled RNN(right)

The subtlety has to do with time. For feed-forward nets, the existing techniques which reason about $\ell_p$-balls around a nominal sample in input and output spaces is 'one-and-done' and rightfully so. For unrolled RNN, at any specific time, the dependency of the network's current output on the previous state forces the verification process to additionally keep track of the $\ell_p$-ball around the previous nominal state. In other words, the verification has to be carried out sequentially for all time steps, each dependent on the verified result from the step before and so on. Given that at every single time, the existing techniques rely on solving either an exact but very expensive problem (mixed-integer or SMT), or a cheaper but relaxed one (convex optimization or nonlinear program), the accumulated computational cost would lead to finishing verifying an entire trajectory way too late, and the accumulated conservatisms would lead to getting hit by a zero volume $\ell_p$-ball way too early. Either way, "unrolling" practically destroys any hope of long-horizon reasoning.

Recognizing the challenge imposed by unrolling, we believe the technical gap can be and perhaps should be fulfilled with a viewpoint shift. We propose instead to embrace the unique loop structure in RNN and treat it as, quite naturally, a dynamical system. This view enables connection with control theory, Lyapunov theory in particular, which avoids explicit reasoning in time and connects elegantly with convex optimization.

But even from a control-theoretic viewpoint, verifying systems with RNN components is still quite challenging due to the networks' highly nonlinear dynamics. How to deal with this challenge constitutes the major part of our contribution below.

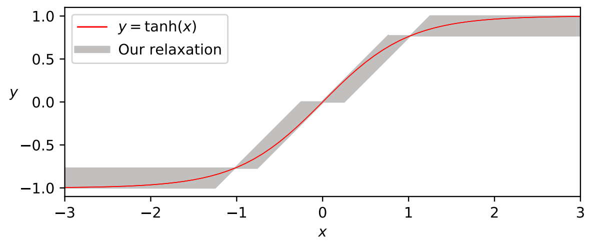


Figure 5-2: Our Method I relies on a relaxation for general activations in the sigmoidal family; hyperbolic tangent is used here for illustration. The relaxation is the conjunction of four *quadratic* constraints and the key ingredient in our control theory and convex optimization powered verification framework.

**Contribution** To the best of our knowledge, this is one of the first work to formally verify systems containing modern RNN components. Two methods are proposed for the verification task:

- Method I proposes tailored novel QC/IQC building blocks to make the general sigmoidal activation family amenable (previewed in Figure 5-2, using tanh as an example) to analysis tools. Our relaxation can achieve increasingly tighter bound with more computational resources (adding more multipliers).

- These proposed QC/IQCs are not limited to a learning setting and the sigmoidal nonlinearities; they more broadly apply to arbitrary functions that are odd and value-bounded.

- Method II proposes the use of an algebraic sigmoid as the activation function in the network. While achieving competitive network performance, the proposed activation scheme results in significant verification simplification compared with the more general Method I.

### 5.1.1 Related Work

From the system and controls viewpoint, the internal dynamical states make RNNs no different from the dynamic output feedback compensators. Designning these compensators are well-known to be difficult even for general linear time-invariant plants. Traditionally, the difficulty lies in the correct-by-construction requirement, because a simultaneous synthesis and verification task does not admit a convex formulation in all but very special cases such as the full-order linear dynamic compensator [63]. In our setting, the synthesis is decoupled from the correctness verification, and we rely on learning to propose a decent controller and verify it post-hoc. While the synthesis part becomes straightforward, the verification is consequentially more complex since it has to deal with a black-box model.

From the learning viewpoint, shortly after the exposure of the vulnerability of deep neural networks in image classification [24], which raised serious concerns over if and how such data-driven approaches can be reliably deployed, there has been an arms race between the attack and verification of the feedforward neural networks using various techniques [10, 33, 28, 83, 75, 58, 23, 71]. In striking contrast to this rapid development, essentially no formal verification framework exists for RNNs, largely due to the reason we laid out in the introduction.

A relevant work close to the problem we consider is [2], where systems controlled by RNNs with ReLU activation are studied via a simplified version of LTL on bounded executions. However, since ReLUs are rarely used as activation in RNNs in practice, the applicability of the work is very limited.

We instead consider the de facto standard RNN architecture, the Long Short Time Memory (LSTM)s. This type of RNN has a unique gating mechanism and relies on sigmoidal activations, whose gradients (softer than, e.g., ReLUs) effectively battle the notorious exploding or vanishing gradient problem commonly found in networks with long history or deep (many) layers. While the intricate design improves on predication criteria such as the horizon and accuracy, it introduces to the already challenging task of system analysis additional difficulty of complex dynamics, and is another reason for the lack of work in this area.

Note that at the time of this thesis writing, two new work had since emerged from the literature. One relies on explicit unrolling in time [30], a scheme similar to what was described in the introduction (Section 5.1); therefore, it inherently struggles in scaling-up in terms of the verification horizon. The work presented in [60] is more similar in spirit to ours, where an IQC-based bounding scheme is applied on dissipation inequality for robustness guarantee. Both of these work achieve similar scale as the proposed Method I but are dwarfed by Method II.

## 5.2 Problem Statement

We are interested in analyzing closed-loop behavior of systems with RNN in the loop. Similar to Chapter 4, our goal is to solve the fundamental region-of-attraction (ROA) approximation problem for this closed-loop system: maximize the inner approximation of the set of initial states which eventually converge to the locally stable fixed point (without loss of generality, assumed to be zero).

More specifically, we are interested in two problem settings: partially observable closed-loop systems modeled by RNNs, or partially observable open-loop systems controlled by RNNs.

### 5.2.1 System Controlled by RNNs

Suppose a known plant is given, with states $x$, inputs $u$, and outputs $y = g(x)$, and the dynamics are $x^+ = f(x, u)$. The plant is feedback-connected with an RNN system whose states $c$ have dynamics of $c^+ = h(c, y)$; note $c$ depend on the plant through plant output $y$ and not the explicit plant states $x$. The RNN then produces the control command $u = l(c)$ for the plant. Once the plant and RNN controller are connected, the closed-loop system states incorporate both components; we denote it by $s := [c, x]$, the column concatenation of the vectors $c$ and $x$.

We assume that $f(\cdot)$, $g(\cdot)$, and $l(\cdot)$ are all polynomials, and $g(\cdot)$ leads to partial observation of $x$. Additionally, the RNN structure is assumed to be the standard LSTM-type. In particular, while our verification techniques apply to any of these variations, we focus on the JANET (Just Another Net). The choice is motivated by the potential computational savings in both training and verification, and justified by JANET's competitive performance on many benchmark datasets [80].

Specifically, the RNN update rule $c^+ = h(\cdot)$ can be written as[1]:

$$\begin{aligned} c^+ &= h(c, y) \\ &= \frac{1}{2}\left(c + c \odot \tau_f + \tau_c - \tau_f \odot \tau_c\right), \end{aligned} \tag{5.1}$$

where $\odot$ denotes element-wise multiplication, and $\tau_f$ denotes the tanh function, taken element-wise as well:

$$\tau_f := \tanh\left(w_f' s\right), \tag{5.2}$$

$w_f$ are the trainable matrix weights. Similarly, $\tau_c := \tanh\left(w_c' s\right)$ simply uses a different trainable matrix weights for its inner-product argument. We sometime collectively refer to all the trainable weights stacked as $w := [w_f, w_c]$.

[1]The dynamics as presented in the original paper has both tanh and logistic sigmoidal activations. Since the two are a scaled and shifted version of each other, i.e. $\tanh(x) = 2\text{logistic}(2x) - 1$, we unify the two for clarity.

The overall dynamics can be written as:

$$s^+ = f_{cl}(s) = \begin{bmatrix} x^+ \\ c^+ \end{bmatrix} = \begin{bmatrix} f(x, l(c)) \\ h(c, g(x)) \end{bmatrix} \tag{5.3}$$

### 5.2.2 System Modeled by RNNs

Just as RNNs are suitable for controlling partially observable open-loop systems, they are also a natural parameterization for system identification of partially observable closed-loop systems.

Figure 5-3 illustrates the training process for the system identification task. The goal of training is to produce the RNN weights $w_f$, $w_c$ and and linear RNN output mapping $w_{out}$ so that the RNN is capable of producing similar output trajectories. The output trajectories from the RNN depend on the initial internal state $c_{\text{init}}$ of the RNN. In order to train the model, we allow the optimization to tune these initial conditions for each trajectory sample in order to optimize the cloning objective. In part because it will be helpful for our analysis, we choose to parameterize these initial conditions as a simple affine function mapping a short segment of outputs to the initial conditions, $w_{init}$.

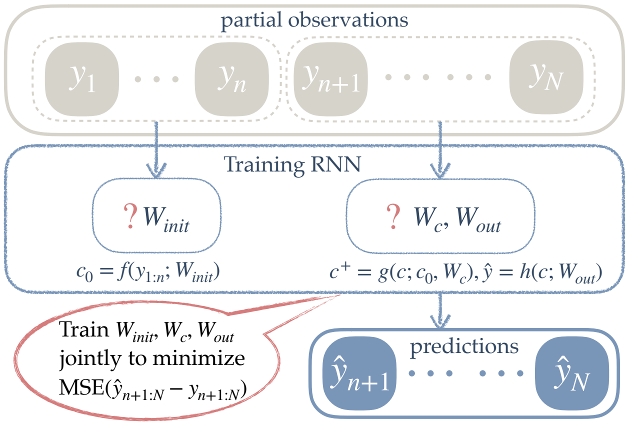
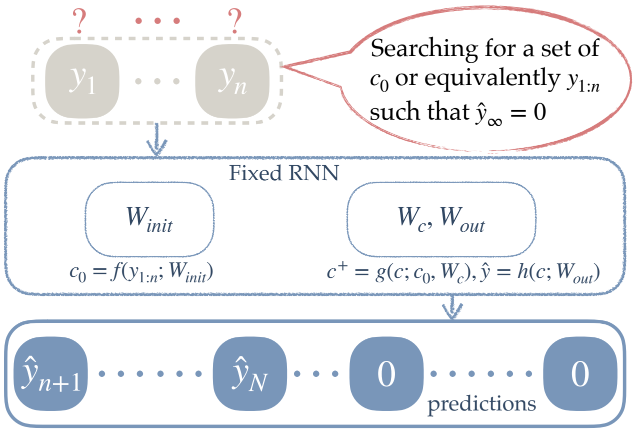

partial observations
$y_1$
$y_n$
$y_{n+1}$
$y_N$
Training RNN
? $W_{init}$
? $W_c$, $W_{out}$
$c_0 = f(y_{1:n}; W_{init})$
$c^+ = g(c; c_0, W_c), \hat{y} = h(c; W_{out})$
Train $W_{init}$, $W_c$, $W_{out}$ jointly to minimize MSE$(\hat{y}_{n+1:N} - y_{n+1:N})$
predictions
$\hat{y}_{n+1}$
$\hat{y}_N$

Figure 5-3: The training process.

More precisely, from the sampled $N$-step observations of the true system outputs, three sets of weights are jointly trained: the initialization weights $w_{init}$, mapping a short fraction at the beginning of the observation tape $y_{[1:n]}$ to an initial RNN cell state $c_{\text{init}}$; the cell update weights $w_f$ and $w_c$, mapping the internal state $c$ to the next time step state $c^+$; and the output weights $w_{out}$, mapping the internal state $c$ to RNN output $\hat{y}$. The training objective is to minimize the mean square error between the observations and the RNN output prediction for the remaining $N - n$ steps.

The verification task is at a high-level similar to Subsection 5.2.1. In particular, we are interested in finding the set of $c_{\text{init}}$ which is inside the region of attraction for the zero fixed point.



?
?
$y_1$
$y_n$
Searching for a set of $c_0$ or equivalently $y_{1:n}$ such that $\hat{y}_\infty = 0$
Fixed RNN
$W_{init}$
$W_c, W_{out}$
$c_0 = f(y_{1:n}; W_{init})$
$c^+ = g(c; c_0, W_c), \hat{y} = h(c; W_{out})$
$\hat{y}_{n+1}$
$\hat{y}_N$
0
0
predictions

Figure 5-4: The verification objective.

Practically, the RNN internal states is most likely hard to interpret or does not associate straightforwardly to the quantities we really care about or are directly measuring, e.g., physical or semantical ones. It may therefore be more convenient or meaningful to equivalently map the set found in the RNN's initial state space to those interpretable space. In our setting, this means a reverse mapping to the short history space we directly observed for initialization. Figure 5-4 illustrates the verification objective.

In the next two sections, we present two methods to address the problem formulated here. At a high level, inherent in these two methods is a generality versus efficiency trade-off.

## 5.3 Method I: Integral Quadratic Constraints

Method I applies to all sigmoidal activation variants, including but not limited to: tanh, arctan, logistic, ElliotSig, as well as the algebraic sigmoidal we consider in Method II. For brevity, we use tanh activation as the running example to describe this proposed method.

In principle, problem stated in Section 5.2 can be handled by the standard tools introduced in the background chapter (Chapter 2) as is. The issue we see is that each and every one of those computational tools only provides a sufficient condition for the problem, so the resulting optimization, even if feasible, is unlikely to provide any meaningful solution.

Realistically, most of these sufficient-only gaps are terribly hard to close. For example, the gap between SOS polynomials and non-negative polynomials, and the gap between linear and nonlinear dynamics each constitutes an entire research field. On the other hand, the gap between the sector condition relaxation and the exact tanh (see Figure 2-1, the relaxed blue region versus the actual red line), is so tangibly loose and so problem specific that there should be hope to tighten it up. In light of this, we propose below a novel, tailored relaxation for the tanh (applicable to general sigmoids).

**IQCs work in "conjunction"** The tutorial example in Subsection 2.3 demonstrates how the IQC idea materializes with one single quadratic constraint. What is perhaps not immediately obvious is that, multiple QCs/IQCs can be combined in a conjunctive way to tighten the relaxation. This is most easily shown visually with QCs.

For notational clarity, we use generic scalars $a$ and $b = \tanh(a)$ for illustrating our tailored solution. Figure 5-5 plots on the left the sector condition, which we discussed in the background tutorial (see Figure 2-1). Plotted in the middle is another widely-used constraint, properly named the saturation condition. The saturation is a valid QC for the $b = \tanh(a)$ the same way the sector is: it encompasses the red hyperbolic tangent line. Recall that the sector condition can be folded in the Lyapunov condition using S-procedure Eq. (2.4); or on a lower level act as one $Q_1$ term in Eq. (2.2), this saturation condition can likewise act as a $Q_2$ term and be attached using another multiplier. The implied condition Eq. (2.3) is then for all the states such that both $Q_1$ and $Q_2$ sign conditions are met, namely the conjunction of those two individual feasible sets, which is plotted on the right of Figure 5-5.

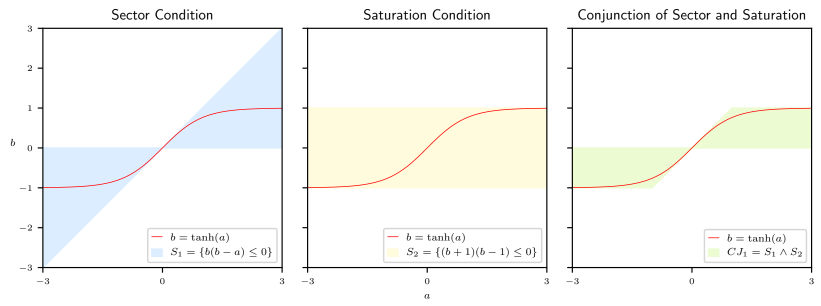


Figure 5-5: The sector and saturation conditions and their conjunction.

By joining forces of two QCs, we effectively improved the relaxation — the conjunction on the right of Figure 5-5 is strictly tighter than either of its two ingredients to its left. Yet, compared with the precise red hyperbolic tangent line, this relaxation is still too loose and not satisfying. Of course, one can play the same game again and stack on more IQCs to further tighten up the regions; the question then becomes where to find these ingredient IQCs, which is usually the hard part in the analysis and calls for problem-specific solutions.

**Proposed quadratic constraints** To the best of our knowledge, the sector and saturation conditions are the only off-the-shelf solutions for the hyperbolic tangent. Our proposed tailored solution is inspired by these two. Specifically, we discover that shifted versions of the sector can also be valid IQCs for our problem. This can be proven by the picture Figure 5-6 or by simple algebra invoking the bounded derivative property of the hyperbolic tangent. These tailored relaxations, after passing through the same conjunctive reasoning from the S-procedure, produce a tighter relaxation on the right in Figure 5-6.

The shifting constants $\alpha$ and $\beta$ can be chosen arbitrarily as long as they satisfy $\alpha = \beta - \tanh^{-1}(\beta)$. Our choice of $\beta = 0.77$ and the corresponding $\alpha$ are simply taken

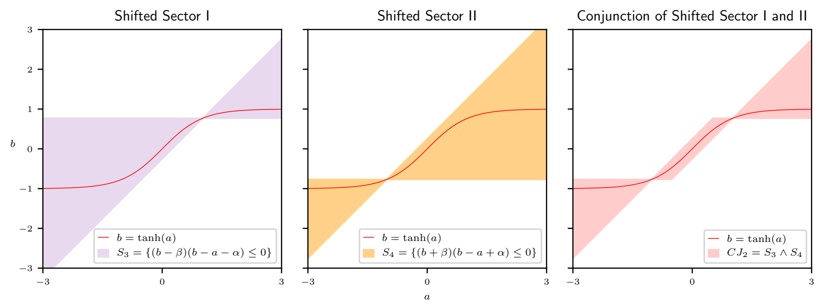


Figure 5-6: Two shifted sector conditions and their conjunction. The shift constants $\alpha$ and $\beta$ can be arbitrarily chosen as long as $\alpha = \beta - \tanh^{-1}(\beta)$; the plot corresponds to the choice of $\beta = 0.77$, and $\alpha = 0.77 - \tanh^{-1}(0.77)$

from the visual cue that they "mix well" with the sector and saturation condition in a complementary sense, and would produce an overall tight relaxation. This is shown in Figure 5-7. In our experiment, we found that $\beta$ in the range of $[0.65, 0.8]$ are similarly well-balanced—as is visually clear why—and produce similarly tight solutions.

For reference, the QCs proposed for the function $b = \tanh(a)$ are $\text{QC}(a, b; \alpha, \beta) < 0$ where

$$\text{QC}(a, b; \alpha, \beta) := \begin{bmatrix} (b-a) \odot b \\ (b+1) \odot (b-1) \\ (b-\beta) \odot (b-a-\alpha) \\ (b+\beta) \odot (b-a+\alpha) \end{bmatrix}, \tag{5.4}$$

$\alpha = \beta - \tanh^{-1}(\beta)$, and ">" is to be interpreted element-wise. Here, $a, b$ are both scalars but can be trivially extend to vector arguments via concatenation.

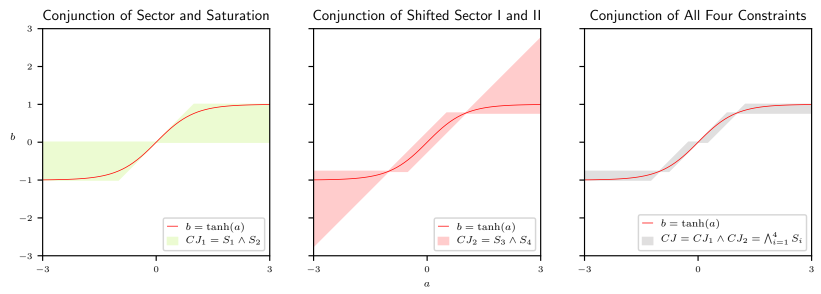


Figure 5-7: Hierarchical conjunctions of various IQCs.

**Proposed integral quadratic constraints** The QCs took advantage of two important properties of the tanh function: that it is odd and value-bounded. However, note that these QCs are static; they only constrain all the particular snapshot instances of the function realization, but do not capture the transient behavior, which presents a missed opportunity in tightening the relaxation.

To take advantage of the dynamic nature of the system, we propose below an integral quadratic constraint using a simple dynamics $a^+ = \tanh(a)$ where $a \in \mathbb{R}$ as an example. Consider this function: $V(a) = a \tanh(a)$; simple algebra shows that:

$$\begin{aligned}
\Delta V(a) &= V(a^+) - V(a) \\
&= \tanh(a) \big(\tanh\big(a^+\big) - a\big) \\
&\leq \tanh(a) \left(\tanh(a) - a\right) \\
&< 0,
\end{aligned} \tag{5.5}$$

which illustrates an upper-bounding scheme for the update difference of the proposed $V(a) \succ 0$. Applying this idea back to the RNN dynamics, a finally building block to be added to the verification will simply be:

$$\text{IQC}(s, \tau; w) := (w's) \odot \tau, \tag{5.6}$$

and the corresponding $\Delta$IQC can be similarly upper-bounded by invoking properties of tanh as in Eq. (5.5).

### 5.3.1 Verification Programs

Recall the closed-loop system has states $s = [x, c]$ with dynamics $f_{cl}$ defined in Eq. (5.3). Given the assumptions, the RNN dynamics Eq. (5.1) is the only component that is not polynomial.

However, if we first treat all the $\tau$ terms as new indeterminates, independent from $s$, then the closed-loop dynamics is polynomial in $[s, \tau]$. Of course, the introduction of new indeterminates in itself loses all the information that $\tau_f = \tanh\big(w_f'[c, y]\big), \tau_c = \tanh(w_c'[c, y])$ and creates a gap. This gap is to be taken care of by the QC/IQC process, which links these new indeterminates $\tau$ back to their arguments.

Following the standard procedure (refer to Section 2.3) to assemble all the pieces together, the verification program can be transcribed as:

$$\max_{V,\rho,l,m} \ \rho \tag{5.7a}$$

$$\text{s.t. } \rho > 0 \tag{5.7b}$$

$$l, m \text{ are SOS} \tag{5.7c}$$

$$-\Delta V(s) + l' \underbrace{(\text{(I)QCs}([s;\tau])}_{\text{Eq. (5.4),(5.6)}} + m' (\underbrace{V(s) - \rho}_{\text{sublevel-set}}) \text{ is SOS} \tag{5.7d}$$

**Remark 10.** *Method I was developed before the techniques presented in Chapter 4 were conceived; but this is not the reason that we do not invoke the "simplified" condition for Lur'e-type nonlinearity, proved in Lemma 2, Section 4.5. The reason is because the setup and proof there rely on the system being continuous-time; the lemma does not hold in discrete-time settings we are dealing here, see Remark 9 for details.*

**Remark 11.** *Method I was also developed before the techniques to be presented in Chapter 6 were conceived. Those techniques could indeed further simplify the verification process here. In particular, it is useful for finding a "high-quality" Lyapunov candidate $V$, and the bilinear alternation we do here could be avoided.*

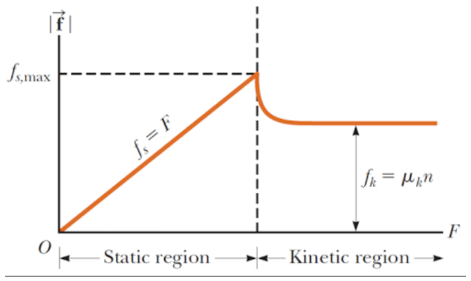


Figure 5-8: Canonical friction model in mechanical systems, where $F$ is an applied force, $f$ is the friction force, $\mu$ is the dynamic coefficient of friction, $n$ is the magnitude of the normal force. Such nonlinearity can be similarly handled via the proposed QCs/IQCs. Figure taken online from Quora.com.

**Remark 12.** *The proposed QCs Eq.* (5.4) *and IQC Eq.* (5.6) *are not limited to the* tanh *or sigmoidal function; they are valid for any nonlinear function that is odd* *and value-bounded. In fact, many of the nonlinearities commonly found in physical systems can also be handled via conjunction of (potentially only a subset of) the proposed QCs/IQCs, one such example is the friction model, shown below.*

## 5.4 Method II: Algebraic Sigmoid (AlgSig)

Method II directly replaces the generally trouble-making sigmoidal activation with a particular one in the family: Algebraic Sigmoidal (AlgSig).

$$\operatorname{AlgSig}(x) := \frac{x}{\sqrt{1+x^2}} \tag{5.8}$$

While not as famous as its siblings like the logistic or tanh when it comes to neural network activations, AlgSig has actually been quite widely used in music signal processing applications [37]. There, not unlike the learning applications, smooth saturation and fast computations are desired, and AlgSig offers both.

Back to our learning application, some slightly different justification has to be made for this swapping proposal; similarly the motivation is slightly different as well.

The justification can be summarized as that AlgSig has all the nice sigmoidal properties that have been hypothesized as the key to their success as RNN activation; and that no empirical network performance is sacrificed due to the swap.

As shown in Fig. 5-9, the function curve of the standard tanh and AlgSig closely resemble each other. In particular, both have the key property of "saturating at $\pm 1$" and the "S"-shape properties.

Second, a nice property of the sigmoidal family that made it so popular is that the evaluation of their gradients are very cheap. For example, for tanh, it involves a simple multiplication of the function value itself:

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(\tanh(x)\right) = 1 - \tanh^2(x) \tag{5.9}$$

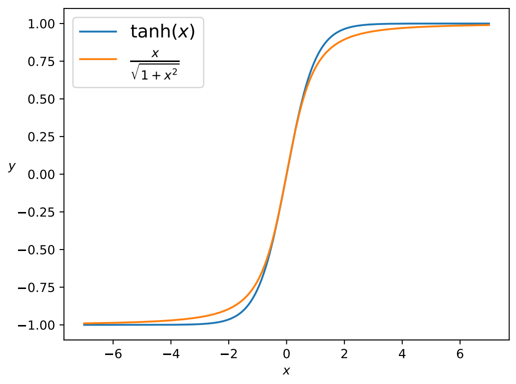


Figure 5-9: $\tanh(x)$ versus the proposed AlgSig nonlinear activation

AlgSig is equipped with this convenient property as well:

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{x}{\sqrt{1+x^2}}\right)=\frac{1}{\left(\sqrt{1+x^2}\right)^3} \tag{5.10}$$

In fact, as we will discuss in Section 5.7, some brief investigation has shown that, the evaluation of Eq. (5.10) is a few times faster than that of the evaluation of Eq. (5.9). Though this potential benefit is not the main focus of our work.

Thirdly, the soft gradient of tanh and logistic, which is the key to battle vanishing and exploding gradient problems in RNNs, is also preserved from the swap.

As the a justification, not only are the activations similar in and of themselves, when placed in networks, the resulting performance are quite comparable as well. Subsection 5.6.1 shows that, on a model trained with the standard tanh activation, merely swapping the activation with AlgSig already lead to similar performance; re-training after the activation swap easily offer further training and testing error improvements.

But why bother with a different activation? Similar to the numerous attempts at simplifying or modifying the "vanilla" LSTM networks, we are computationally motivated as well. However, it has more to do with the computation of the verification certificate than the training (though AlgSig has indeed huge potential for training savings as well; this is to be discussed in the future work). In particular, AlgSig leads to a significantly simplified verification process, where the sampling quotient-ring SOS framework discussed in Chapter 4 is directly applicable, with no approximation or relaxation required. This has important implication in both the verification scale and quality, to be described in Subsection 5.5.

### 5.4.1 Verification Programs

Suppose the number of units in the RNN is $n_c$, then the ROA problem can be written as maximizing over $\rho$, $V(s) \leq \rho$ for all $s, \tau$ such that (recall Eq.(4.3) in Section 4.3.2):

$$\begin{aligned} \Delta V(s) &= V\left(s^{+}\right) - V(s) \\ &= V(f_{cl}(s)) - V(s) = 0 \end{aligned} \tag{5.11}$$

and

$$\tau_i = \operatorname{AlgSig}(w_i' s) = \frac{w_i' s}{\sqrt{1 + (w_i' s)^2}} \tag{5.12}$$

where $\tau_i$ is the generic $i$-th element in the stacked $\tau$ vector and similarly $w_i$ is the $i$-th element in the stacked trainable weights $w$.

Note that constraint (5.11) is already in polynomial form, whereas (5.12) can be recast so:

$$\tau_i = \frac{w_i' s}{\sqrt{1 + (w_i' s)^2}} \quad \Longleftrightarrow \quad \begin{cases} \tau_i^2 + \tau_i^2 (w_i' s)^2 = (w_i' s)^2 \\ \tau_i w_i' s \geqslant 0 \end{cases} \tag{5.13}$$

These now polynomial constraints and cost can be translated into an SOS program amenable for the sampling quotient-ring SOS method described in Chapter 4. In particular, we formulate this problem:

$$\max_{\rho, \{l_i\}} \rho \tag{5.14a}$$

$$\text{s.t. } \rho > 0 \tag{5.14b}$$

$$l_i \text{ are SOS}, \ i = 1, 2, \ldots, 2n_c \tag{5.14c}$$

$$(s's)\left(V(s) - \rho\right) + \sum_{i=1}^{2n_c} l_i \tau_i w_i' s \text{ is SOS}, \ \forall (s, \tau) \in \mathcal{V} \tag{5.14d}$$

$$\mathcal{V}(s, \tau) := \left\{(s, \tau) : \tau_i^2 + \tau_i^2 (w_i' s)^2 = (w_i' s)^2, i = 1, 2, \ldots, 2n_c\right\} \tag{5.14e}$$

where $n_c$ is the number of units in the network, $l_i$ is a scalar polynomial multiplier, $\tau_i$ is the $i$-th element in the all-stacked AlgSig nonlinearity. The size $2n_c$ is due to that each cell introduces 2 AlgSig nonlinearities (both $\tau_c$ and $\tau_f$).

Note that Eq.(5.14d) encodes the $\tau w's \geq 0$ constraint, and the defining equations for the variety Eq.(5.14e) encodes the squaring of the algebraic sigmoidal; both due to Eq.(5.13).

## 5.5 Methods I and II Comparison

We have presented two methods, geared towards the same high-level problem. Method I is more general but comes with the price of having more complex constraints in the verification. Method II requires a specific activation, but in turn, results in a significantly simpler verification program with much less constraints; more importantly, these fewer constraints are all exact.

To be more precise, for Method I, the inequality constrained (I)QCs are each introduces at least a sign-constrained multiplier. And since all of the (I)QCs are only "bounding" the nonlinearity, to get a feasible or meaningful verification result, one needs to add as many such bounding constraints as possible (e.g., all 5 of the proposed).

Method II on the other hand relies on an algebraic activation that can be described by one polynomial equality and one single polynomial inequality, Eq. (5.13). Once the equality constraint is efficiently taken care of via the techniques presented in Chapter 4, the remaining single polynomial inequality is enough to "exactly" capture the nonlinearity.

So, the comparison boils down to "one constraint to exactly describe the nonlinearity" versus "five to just bound", these constraints and size differences add up very quickly when the RNN size grows. Method II, therefore, is perhaps preferable when applicable, especially if verification is of the highest priority. But if, for example, the RNN to be verified for uses non-AlgSig activation and cannot be modified or retrained, then the more geneal Method I is useful despite the higher verification computational cost.

## 5.6 Experiments and Examples

### 5.6.1 AlgSig Is Effective for Training

We test the performance of AlgSig against the vanilla LSTM and the simplified Janet models, both of which rely on the standard tanh sigmoid. Table 5.1 shows the testing accuracies of these models on benchmark datasets.

Table 5.1: Accuracies [%] on benchmark datasets for different RNN architectures. All networks have a single hidden layer of 128 units. "Janet w. AlgSig (swap)" are pre-trained Janet networks with all their tanh activation function swapped with the AlgSig activation, while keeping all weights (therefore the arguments in the activation as well) fixed. "Janet w. AlgSig (retrain)" are the swapped models with weights updated from further training. The means and standard deviations from 10 independent runs are presented.

| Model | MNIST[a] | pMNIST | IMDB [40] | MIT-BIH |
|---:|---|---|---|---|
| Vanilla LSTM | 98.9±0.13 | 92.3± 0.22 | 95.2± 0.18 | 85.4± 0.18 |
| Janet | 98.3±0.11 | 93.8± 0.13 | 93.2± 0.12 | 87.9± 0.21 |
| Janet w. AlgSig (swap) | 95.3±0.19 | 91.8± 0.29 | 91.3± 0.24 | 82.1± 0.25 |
| Janet w. AlgSig (retrain) | 97.8±0.10 | 92.9± 0.18 | 94.8± 0.14 | 84.8± 0.15 |

[a]each row (or column) of the pixels is fed into the net sequentially; this is different from how common CNNs treat the inputs.

### 5.6.2 Verification Examples

**Partially observable Van der Pol modeled by an RNN** Following the problem setup in 5.2.2, we train an RNN to approximate the dynamics of the benchmark system, time-reversed Van der Pol oscillator (TRVDP). TRVDP is a 2-dimensional continuous time polynomial system: $\dot{x}_1 = -x_2, \dot{x}_2 = x_1 - x_2 + x_2 x_1^2$, the output is simply the partial state, i.e. $y = x_2$. It has a known true ROA that's also easy to visualize, and therefore it has been a staple benchmark for ROA analysis techniques.

The RNN is trained to approximate the output $y = x_2$. For training data, we sample $10,000$ of initial states $[x_1; x_2]$ within the 6-by-6 square centered at the origin. For each sample, we simulate the TRVDP plant by integration for 12 seconds, and discard the unstable samples and the $x_1$ dimension. We then tick along the 1D simulated $y$ trajectories at every 0.12 second interval to get 100 discrete time steps. The first 10 time steps is used to train a mapping to the initial cell state $c_{\text{init}}$ of the RNN, and the rest 90 time steps is the target the RNN output attempts to approximate. The loss function is the mean square error between the true and predicted 90 steps, the training uses adam for gradient descent. The RNN has a single Janet layer with 10 units, an affine initialization mapping, and a linear output layer mapping.

Figure 5-10 compares four samples of the true system output $y$ versus the network prediction $\hat{y}$ drawn from the test set. Notice that, since the output $y$ is just one state $x_2$ of a 2-states system, some of the trajectories cross each other, implying that a single snap-shot measurement of partial state alone does not fully determine the future trajectory. This highlights the necessity and advantage of RNN, for that its internal cell states is capable of compensating the loss of full observability, as evidenced by the small error of the predicted and simulated trajectories.

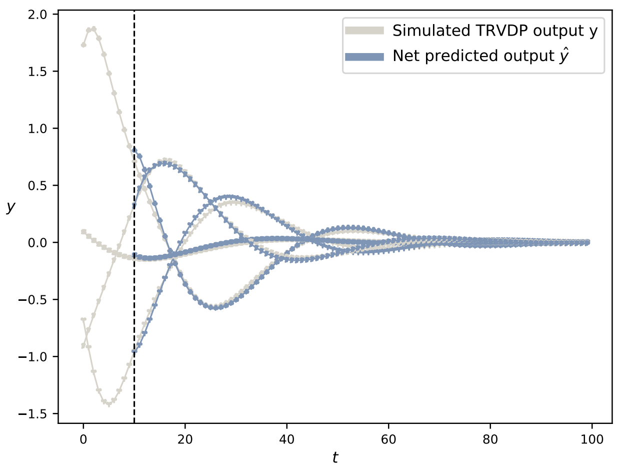


Figure 5-10: True output $y$ v.s. predicated output $\hat{y}$ trajectories, sampled from the test data set.

For verification, we use a quadric parameterization for the Lyapunov function and all the multipliers, and obtain an ellipsoidal ROA in $\mathbb{R}^{10}$ for $c_{\text{init}}$. Since directly plotting this ellipsoid does not give us any insight on the quality of the approximation with respect to the true ROA in $\mathbb{R}^2$, we instead densely sample, again from the sampling square for initial states. For each sample of the initial TRVDP states, we repeat what was done in the training to find its image in the RNN initial state space: forward simulate the TRVDP state for 10 steps, and then affine transform this short history using the trained initialization weights $w_{init}$. Once obtain this RNN initial states in $\mathbb{R}^{10}$, we simply check if it is within the verified ellipsoid.

Figure 5-11 shows samples in and out the verified ROA versus the true ROA boundary.

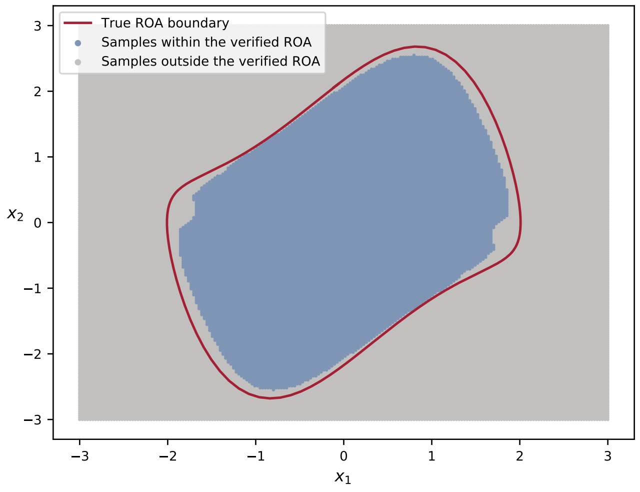


Figure 5-11: Verified samples v.s. the true ROA.

Note that the conservatism of our estimation is to be expected, and there could be two sources for the gap. First, the RNN is only an approximation of the true dynamics, and second, our verified ROA is also only an (inner) approximation. Nonetheless, that our estimation can capture the true ROA this closely is very encouraging.

**Partially observable double integrator controlled by an RNN** Consider the discrete-time double integrator, whose open-loop dynamics are: $x_1^+ = x_1 + x_2, x_2^+ = x_2 + u$, where $x_1$ is the position, $x_2$ is the velocity, and $u$ the input. We are interested in using only single shot of the position $x_1$, not a history of $x_1$ or any direct measurement of $x_2$, to produce control $u$ that can steer the integrator to the origin (zero position and velocity). Simple classical eigenvalue analysis can show that, a linear static output feedback in the form of $u = Kx_1$ where $K$ is the feedback gain is not capable of completing the task (in a more descriptive language, not knowing my past positions nor current velocity, it is impossible to come up with simple linear control strategy that gets me back to the origin). It is necessary to have at least an estimation of $x_2$, either explicitly or implicitly.

We train an RNN in a reinforcement learning fashion to accomplish this control task. We minimize, with the RNN in the feedback loop, the $\ell_2$ norm of $x_1$ measurement at the terminal time step. The closed-loop system dynamic in this setting are therefore $x_1^+ = x_1 + x_2, x_2^+ = x_2 + u = w'c, c^+ = h(c, x_1)$, where $h$ is the RNN dynamics, and $w$ is the trainable output weights. As for the initialization of the network's internal state, since it is only reasonable to assume the network knows nothing about the system before the first measurement, we always initialize all cells to zero. This is also the common practice in the adaptive control community.

The verification task is to find an inner approximation of the ROA, in the $s := [x_1; x_2; c]$ state space, for the closed-loop system. Figure 5-12 shows an example of the two-dimensional slice of the verified ROA using Method I. The network has ten internal cells, and we used quadratic parameterization for Lyapunov function and all the multipliers. The ellipse is a slice of our verified six-dimensional ellipsoid at time zero where it is known $c_{\text{init}} = 0 \in \mathbb{R}^{10}$. Since we do not have a known true ROA of the closed-loop system to compare against, we sample initial states and forward simulate to record the "True" ROA.

Once again, the plot showcases the tightness of our approximation. It is also worth pointing out that, in the training stage, all samples of the initial $[x_1; x_2]$ states are drawn from the 6-by-6 square centered at the origin, and we are able to verify a space much larger than that. Of course, we expect the network learns to generalize; this is the basic assumption and motivation underlying any learning approach. But being able to have a definitive and formal answer as to how much and how reliably the network is capable of generalizing, through our proposed verification framework, is very encouraging.

Additional implication of the result has to do with improving sample complexity,

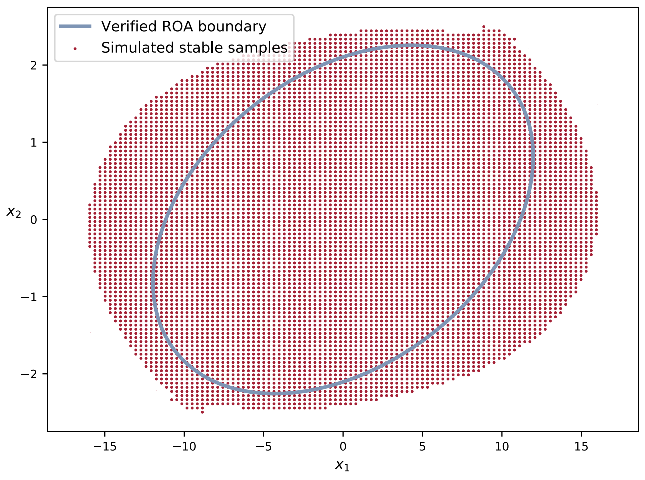


Figure 5-12: Verified ellipsoid ROA for the closed-loop double integrator system using RNN as a partial state feedback controller

which has been a major bottleneck in reinforcement learning paradigm, especially the model-free version. There, the performance quality is gained painstakingly through quantity — learning is encouraged to explore and assess as many sample points as possible, so as to get good test coverage. Given our verification, the exhaustive testing is much alleviated. For example, we might not have visited a sample at, say, $(10, 1)$. But as Figure 5-12 shows, exploration to this sample is not necessary as the verification proves formally that the controller would behave well there.

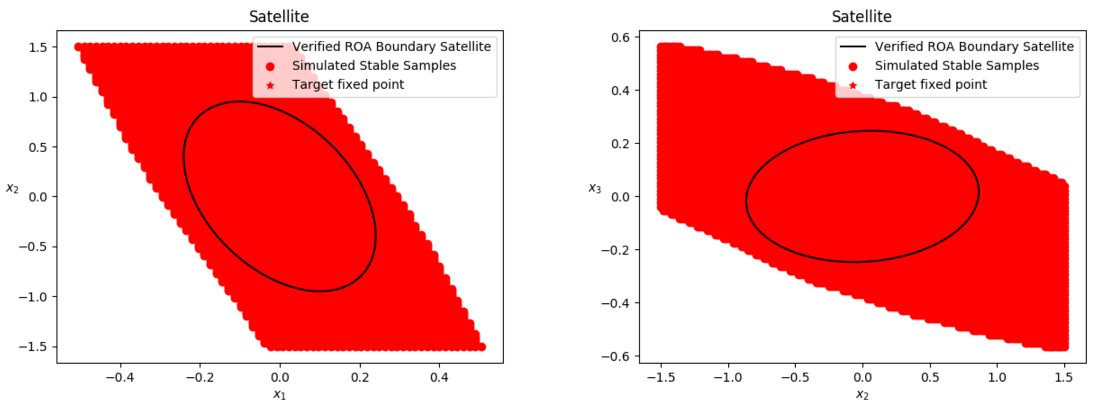
**Partially observable satellite controlled by an RNN** We take from [42] a 6-dimensional satellite model. The states are the configuration and the velocities, and the system admits a 3-dimensional input.

$$\begin{aligned} H\dot{\omega} &= -\Omega(\omega)H\omega + u \\ \dot{\psi} &= \frac{1}{2}\left(I + \Omega(\psi) + \psi\psi^T\right)\omega \end{aligned} \tag{5.15}$$

where $\omega \in \mathbb{R}^3$ are the angular velocities in the body-frame, $\psi \in \mathbb{R}^3$ represent the attitude as modified Rodriguez parameters, and let $x = [\omega; \psi]$ denote the 6-dimensional plant states. $\Omega : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ is the matrix defined so that $\Omega(\psi)\omega =$ is the cross product $\psi \times \omega$, and $H = \text{diag}([2, 1, 1])$ is the inertia matrix.

The plant output is selected to be the full state leaving out $x_6$, the dynamics are discretized via forward Euler with time-step of $1e^{-3}$ for the purpose of training, and a single-layer RNN with 8 AlgSig units is trained in a reinforcement learning fashion, similar as the double integrator example.

Due to the use of AlgSig activation, Method II is applicable to verify the closed-loop dynamics. Two slices of the verified ROA and the sampled stable initial states are shown in Figure 5-13



(a) Slice at $x_1$ and $x_2$ (b) Slice at $x_2$ and $x_3$

Figure 5-13: The verified ROA and the sampled stable initial states for the satellite plant controlled by RNN with AlgSig activation.

## 5.7 Discussion and Future Work

**Other (algebraic) sigmoid activations** Investigating whether other sigmoidal activations could offer comparable performance as the standard logistic and tanh could help test the hypothesis that the three key properties described in Section 5.4 are indeed crucial for RNN training (since they are also all satisfied by this function). A good reference on this topic is [22].

In the process of such investiagtion, we might also discover other functions that could further simplify the verification programs. An example, suggested by Alexandre Megretski

$$f(x) = \frac{x}{1+|x|},$$

might be a good starting point (for its analytical simplicity); the reference above also has brief discussion on this particular nonlinearity.

**Extension to ReLUs** Techniques used in this chapter inspire some ideas for analyzing FFNN using SOS/SDPs instead of the mixed integer linear programming or linear programming hierarchy. Two particular ones are:

- Encode the PWA ReLU with polynomial constraints.

  Recall that $y = \text{ReLU}(x) = \max(0, x)$, which can be equivalently written as:

$$\begin{cases} y(y-x) = 0 \\ y \geq x \end{cases} \tag{5.16}$$

- Use rational approximation of ReLU.

  In [74], it is shown that, rational functions are an ideal approximator to the ReLU activation function. Given the straightforward mechanism to convert between polynomial and rational functions, and consequentially the direct application of the something variety approach for verification given these polynomial or rational constraints, this could also be a fruitful direction.

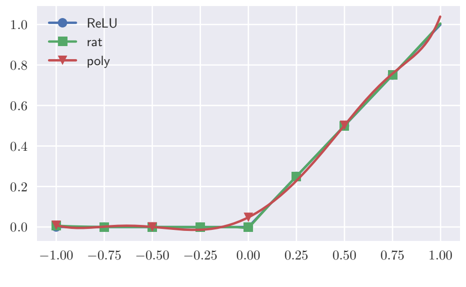
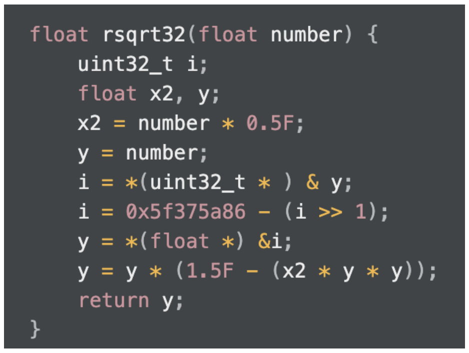


*Figure 5.* Polynomial and rational fit to $\sigma_r$.

Figure 5-14: ReLU and polynomial and rational approximations [74]

**Better understanding of the fast InvSqrt algorithm and its implications to RNN training/evaluation** A very preliminary report [16] shows that, even for the purpose of training alone, evaluations of AlgSig are 3x to 6x faster than the standard tanh.

This fast evaluation is due to the famous Fast Inverse Square Root (InvSqrt) algorithm, which efficiently calculates the function $y = \frac{1}{\sqrt{x}}$, the central operation in AlgSig. The code snippet of InvSqrt is shown below [39]. As a side note, curiously, this code snippet relies on the "magical" Hex constant 0x5F3759DF, which is shown to outperform some other constants rigorously chosen from pure math; unfortunately, the original author and derivation (or a trial-and-error procedure) of the magic are both unknown.



```
float rsqrt32(float number) {
    uint32_t i;
    float x2, y;
    x2 = number * 0.5F;
    y = number;
    i = *(uint32_t * ) & y;
    i = 0x5f375a86 - (i >> 1);
    y = *(float *) &i;
    y = y * (1.5F - (x2 * y * y));
    return y;
}
```

Figure 5-15: The fast InvSqrt() source code

It would be interesting to understand better the ideas behind this algorithm, investigate whether, and how, it could further improve the training and evaluation of RNNs with AlgSig activation. More broadly, brief reading on this topic suggests that the magic constant has close and clever connection with Newton iteration, and it would also be interesting to seek further pure optimization-related connection there.

# Part III

# SafetyNet: Structured Learning and Optimization Knit Together

We have discussed in Part I the optimization-based system verification framework, its main scalability challenge and two methods to improve the computation. Then in Part II, we showed how black-box learning models can be challenging to analyze, and how systematic relaxations and approximations, or clever tricks are needed to bridge the gap.

Given these contrasting strengths and weaknesses, a natural question to ask is, how can we combine them in a complimentary way? Admittedly, this is a huge open question, but some basic high-level considerations should be immediately clear.

First, learning-based methods are particularly good at mining out complex relationships of data pairs; whereas optimization can offer a final seal of approval for any of those candidates. In other words, learning is good at “searching” while optimization is good at “proving”. Therefore, an upstream learning-based module followed by a downstream optimization-based certification is one, and perhaps the only, logical choice.

Second, given the verification difficulties caused by the complex network structures and activations, it is desirable to have the learning methods produce rich yet nicely-parameterized outputs, so as to ease off such challenges.

The question remains as what parameterization to choose, and what properties to offer at the learning stags. The answer to both questions requires insights from the verification programs’ point of view, as to what would the programs directly benefit from. Our answers and the thinking behind are described in the next chapter.

# Chapter 6

# Readily-Verifiable Learned Controllers and Lyapunov Candidates

## 6.1 Introduction

Intrigued by the compelling success of (deep) reinforcement learning as an approximation to the computationally intractable dynamic programming and optimal control, we consider the central question in this chapter:

> What are some other important but hard control and verification problems that could benefit from data-driven searches? and how?

The "what" part may have many possible answers, but two obviously arise from our discussions in the previous chapters, i.e., the search of Lyapunov candidates and the synthesis of stabilizing controllers. While we (hopefully) have established their importances, the challenges therein, which we only alluded to or at best glossed over, e.g., in Chapters 4 and 5, has to do with non-convexities in traditional optimization setup.

To see this non-convexity more clearly, recall the Lyapunov sub-levelset used for approximating the ROA (see Section 3.4) is jointly defined by the Lyapunov function $V$ and the level $\rho$. This means that a simultaneous search of both $V$ and $\rho$ would require multipliers to encode the corresponding $\dot{V}$ condition, and result in a non-convex bilinear problem, which are known to be NP-hard in general [77]. If, in addition, a feedback controller $u = \pi(x)$ is to be searched for simultaneously, the non-convexity is even worse, i.e., escalated from bilinear to tri-linear.

Traditional optimization-based approaches address this non-convexity by fixing some variables so as to convexly search for the rest, and alternate between what to fix and what to search. As we will review in detail in Subsection 6.1.1, these alternations not only each inherits the traditional scalability challenges described in Part I), but they collectively introduce ill-conditioned iterations.

Various data-driven approaches have been proposed in order to bypass exactly these non-convexity and numerical issues. However, these previous work share common shortcomings. Specifically, the search space is either too limited/structured, e.g. strictly convex search-space, that it cannot accommodate control synthesis. Or it is too rich/complex, e.g., support vector machine, that formal set-based guarantee becomes elusive and analysis has to resort to statistical tools, which in itself introduces sample complexity scalability issues.

To overcome all these major limitations is to answer the "how" part in our motivating question, and these previous work highlight two key considerations. Most importantly, the search-space parameterization should be well-balanced between expressiveness and analytical cleanness (so as to facilitate a downstream verification). Moreover, as with any data-driven or learning approaches, it is also important to keep in mind how to encourage generalizability of the model to unseen data points/sets.

Given the large volume of successful polynomial controller in classical control literature, and the strong evidence that linear control functions are all we need for solving highly nonlinear control problems through reinforcement learning [43], polynomial parameterization for the feedback controller is a natural and well-justified choice. For the Lyapunov candidate, however, theoretical [1] and practical [9] evidence have indicated that polynomial family might be limited, thus we resort to the much richer rational family. Besides the theoretical justifications, these choices more practically enable connection with the techniques presented in Part I which allow us to carry out verification at scale.

The question remains what property should these learned functions have. Recall the conditions such as in Theorem 5 has positive definiteness (PD) of the Lyapunov candidate as a prerequisite so this is a necessary property to satisfy. However, PD is not a cheap constraint—the precise reason why SDP is the most expensive class of convex optimization. Our solution to this is based on the flexible stochastic gradient descent over carefully chosen search-spaces, e.g., the factorization space of the Gram matrix of an SOS polynomial. With some minor regularizing terms, it is guaranteed that the candidate is PD by construction.

As for the objective, motivated to generalize to unseen samples, we take cues from system theory and set the cost to be the worst-case Lyapunov exponential decay rate (instead of formulating the problem as, e.g., an ostensible classification).

These design choices ensure that: (i) the Lyapunov candidate is positive definite by construction, and that (ii) the Lyapunov derivatives indicate that controller is "empirically stabilizing" on samples over a large region. The first property then enables a direct application of the verification subroutine presented in Chapter 4, and the second implies the candidate is of "high-quality" thus encourages good (tight) verification results.

**Contributions.**

(i) We propose SafetyNet, a mixture of learning and optimization procedure for verification and control. SafetyNet generates sample-efficient and verified control policies, and thus overcomes two major drawbacks of reinforcement learning.

(ii) SafetyNet also offers the capability to directly search for *rational* Lyapunov candidate at scale; this is strictly more general than the polynomial family that most existing convex-optimization-based methods are limited to. Consequentially, SafetyNet can verify systems that are provably beyond the reach of pure convex-optimization-based verifications.

### 6.1.1 Related Work

**Completely convex-optimization-based via alternation** Traditionally, the bilinear or trilinear non-convex problem is solved via an alternation of SOS programs [73, 41]; the high-level idea is quite similar to the expectation maximization (EM) alternation in classical learning. Specifically, using linearization and Lyapunov quadratic equations or LQR, an initial Lyapunov candidate and controller (when necessary) pair can be fixed to make the optimization convex.

One major limitation here is that the linearization could be very local, especially for systems with highly nonlinear behavior. In some cases, with this rather weak initialization, it may take many iterations of alternations to arrive at a meaningful volume (because the shape and orientation of Lyapunov candidate largely determine the quality of the certificate). In worse situations, such as when the closed-loop linearization is marginally stable, the quadratic initialization may itself fail; we will demonstrate this via an explicit example In Section 6.4.

A more subtle issue has to do with numerical conditions. Through the alternations, each iteration solves for an optimization problem, and pushes the intermediate solution as far as possible to the boundary of the PSD cone, deteriorating the numerical condition for the subsequent iterations. In this sense, this is more challenging than the unconstrained EM alternation.

**Data-driven search (of polynomial $V$ only) via convex optimization** In the same spirit of convexifying the search by fixing some variable, an alternative line of work relies on samples to generate those to-be-fixed variables. For instance, [79] uses simulation to first find a set of stable states, and uses SDP to fit a candidate on those samples; the verification step is then carried out following the standard practice. The major disadvantage of this method is that it would require solving a potentially large-scale SDP for when there are a large number of samples, which is necessary to capture the nonlinear behavior or spread over a large region of the state space.

Such scalability limitation in generating the candidate motivated the work presented in [31], which instead uses an LP for the search of the candidate. Also, it uses a guided search of counter examples that violate the derivative (difference) condition to iteratively improve the quality of the candidate Lyapunov functions.

Compared with [79], the scalability (and quality) of the candidate search is improved, but this is at the cost of the complexity of the downstream verification: the resulting candidate is only positive semidefinite on the samples, as opposed to globally. Therefore, it requires an additional PSD constraint in the verification step to close the gap, and the overall scalability remains an issue.

Another common disadvantage of researches along this line is that their reliance on convex optimization limits the search to be for Lyapunov candidate only, as opposed to the more practical simultaneous search of a candidate $V$ and controller $\pi$.

**Learning-based search (of generic $V$ only) with statistical guarantee** Most researches along this learning-type search of Lyapunov candidate formulate the stability verification problem as a binary classification. For instance, [36] solves the classification via support vector machines (SVM) with polynomial kernels, so at the low level, it needs to solve a quadratic program (QP). The quality of the resulting candidate is only assessed empirically, either through straightforward generalization error on discrete samples, or a slightly more involved k-fold evaluation. Hidden in the method is a scalability issue: essentially, to fit the separating hyperplane "snuggly", SVM requires a large number of support vectors, and natural more general data. and the inherent scalability issue of QPs.

A recent approach to learning Lyapunov function which has some philosophical connections to the method we propose is that of [61]. It carries out a parameterized search of the factorization $V = NN'$ using gradient descent, and uses a loss function based on binary labels of whether a sample is simulated to be stable. The major issue there is that, because the $N$ component are themselves complicated neural networks, they can not be exactly verified; instead, the guarantee of the validity of the candidate is based on Lipschitz constant upper-bounds and discretization relaxation. Such discretization-based verification is either very conservative, or relies on very fine resolution which scales badly. This shortcoming casts doubt on the necessity and justification of using a black-box neural network to represent the Lyapunov candidate.

**Other related work** Our core idea of searching over factorization space is somewhat similar to the Burer-Monteiro (BM) method in optimization [15], and both are computationally motivated by avoiding solving expensive SDPs. However, the underlying assumption in BM, that the PSD decision variable has a low rank, is irrelevant in our work. In fact, we intentionally over-parameterize the factorization space, so the solution is generally explicitly of full rank. Another major difference is that our cost is highly non-convex. Therefore we do not have, at the candidate-generating stage, convergence guarantee as many BM method does; such guarantee is to be provided by the downstream verification though.

Another related and popular direction is to approximate the dynamics using Gaussian processes [57, 12]. The disadvantage is that the safety is guaranteed statistically in terms of probability bound and distributional parameters such as the covariance matrices; and generally such work suffer from scalability issues.

A more recent and more closely related work is [19], which produces both a controller and a Lyapunov candidate. The authors use a similar approach as [61], but the training set contains counterexample states (states where the Lyapunov condition is violated). These counter-example states are generated using an SMT solver at each iteration. Due to numerical issues stemming from the SMT solver, the numerical accuracy of is not high (around 0.01), especially around the equilibrium state, where a guarantee of stability is most critical.

## 6.2 Problem Statement

Given a continuous-time dynamical system $\dot{x} = f(x, u)$ (discrete-time counterpart is straightfowrd), we consider the task of synthesizing a feedback controller $u = \pi(x)$, such that the resulting closed-loop dynamics $f(x, \pi(x))$ is locally or globally asymptotically stable around a fixed point, which is without loss of generality assumed to be at the origin. Note that the setup includes as a special case the verification of a given closed-loop. That is, if a feedback controller $\pi(x)$ is given a priori or if the system effectively admits zero inputs.

The stability properties are to be verified by a Lyapunov candidate $V$ satisfying the value condition ($V \succ 0$) and derivative condition ($\dot{V} \prec 0$) globally or locally. The task in this chapter is to produce, via sampling and SGD, "empirically good" control law $\pi(x)$ and Lyapunov candidate $V(x)$, which are to be fed into downstream verification programs (those presented in Part I) for correctness guarantees.

By empirically good, we mean at the least, the value and derivative conditions should hold on a large fraction, if not all, of the training samples. Further, given that the end-goal is to produce an as large (volume) of an ROA approximation or even global certificate, the "error margins" should be large enough to accommodate for generalization, i.e., sign condition is satisfied on samples outside the training set.

These considerations, however, do not translate directly into an actionable cost. In particular, the straightforward sign function is not suitable for training cost since the gradients are zero almost everywhere. Nonetheless, this serves as a high-level description, the precise (mathematical) objective is to be presented in Section 6.3.

### 6.2.1 Assumptions

We assume the Lyapunov function $V(x)$ is rational in the states $x$; this includes the common polynomial feedback and/or Lyapunov parameterization. Further, we assume $f(x, u)$ is polynomial in $u$ (the math in fact can work for rational dependence in $u$, since singularity can be easily pitted out by parameterization tricks, but practical systems rarely depend on the control input rationally), and this includes the large class of control-affine systems, particularly mechanical systems. Due to similar practical consideration, even though the math works for rational parameterization, we assume the control law $\pi(x)$ is polynomial in $x$.

Unlike the controls, the dependence of the dynamics $f$ on the states $x$ is often times quite rich, therefore, we assume it to be rational. This is quite a rich assumption in itself, but techniques presented in Section 4.6 and in [50] can be applied to recast an even larger class of nonlinear analytical systems losslessly into polynomial or rational form.

## 6.3 Generate Control Policy $\pi(x)$ and Lyapunov Candidate $V(x)$ via SGD

This section describe our algorithm to generate via sampling and SGD a polynomial control law $\pi(x)$ and Lyapunov candidate $V(x)$ for the resulting closed-loop dynamics. We first introduce our search space and cost design, as justified from a mathematical or theoretical viewpoint.

### 6.3.1 Over-Parameterized Search Space Design

Given and a degree bound $d_n$ and $d_d$ for the Lyapunov candidate ($d_n$ for the numerator, and $d_d$ for the denominator), and a degree bound $d_\pi$ for the controller, we first construct the monomial basis $\varphi(x)$, $\phi_n(x)$, and $\phi_d(x)$, which are the standard monomial basis of $x$ up to degree $d_\pi$, $d_n/2$ and $d_d/2$, respectively.

The feedback controller is straightforwardly parameterized as

$$\pi(x) = w_u'\varphi(x), \tag{6.1}$$

where $w_u$ is the search variable (similar to notion of controller gain in linear feedback).

The Lyapunov function parameterization takes some consideration. We need to rely on an unconstrained, first-order optimization tool to guarantee for a positive definite rational function. Our solution below draws ideas from i) Padé rational parameterization, ii) PSD matrix definition, and iii) SOS polynomial decomposition:

$$V(x) = \frac{V_n(x)}{V_d(x) + 1} \tag{6.2a}$$

$$= \frac{\phi_n'(x)L_n'L_n\phi_n(x) + \delta x'x}{\phi_d'(x)L_d'L_d\phi_d(x) + 1} \tag{6.2b}$$

where $L_n$ and $L_d$ are the *unconstrained* parameters to be search for via SGD, and $\delta$ is a fixed small positive constant.

It should be obvious that $V(x) \succ 0$ (i.e., $V(0) = 0; V(x) > 0, x \neq 0), \forall L_n, L_d$, thus fulfilling our requirement. However, instead of directly searching over these $L_n$ and $L_d$ factorizations, strong evidences from the learning community, summarized below, suggest that an over-parameterization—with essentially no additional computational costs—might be desirable.

**Over-parameterization may be a key to the deep learning success** Here we first take a detour to look at what makes neural networks effective; this topic is beyond the scope of our work, and we will be merely presenting existing results. Nonetheless, the current thinking in that community not only shed interesting light on optimization in general but in fact motivated part of our search space design, and thus a brief summary is appropriate.

As referenced in Part II, there are many cases in deep learning where we can reliably solve seemingly very high-dimensional and non-convex optimization problems. Overall, the success is a mixture of many techniques maturing and emerging at the right moment, e.g., hardware GPUs that enable parallel computing coupled with the large quantities of data available. From an algorithmic point of view, several elements are absolutely indispensable for the success and popularization. Theoretical understanding of what makes neural networks so powerful is rapidly evolving; but it is fair to summarize the consensus thus far (which is by no means a complete list) as:

Non-convexity is not a vice; high-dimension is a blessing in disguise.

Specifically, the belief is that many of these success stories are happening in the so-called "interpolating regime" [11] where we have more decision variables than data, and the search space is dense with solutions that can fit the data perfectly.

Of course, the underlying non-convexity introduces many local minima that may not be global minima, and global minima that may not be robust, the arrival at a "good" interpolating solution relies on the optimization engines such as stochastic gradient descent, batch normalization [29] for some form of implicit regularization.

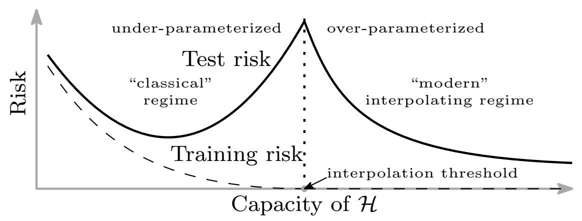


Figure 6-1: The double descent risk curve proposed by Belkin et al. [11]. It incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high capacity function classes (i.e., the "modern" interpolating regime).

This idea is very relevant in our case, since at candidate generation stage, we are only interested in a "good enough solution". Moreover, the crucial ingredient in our parameterization, and SOS polynomial is naturally amenable to over-parameterization. An explicit example is given below.

**Over-parameterize** $L$ Consider a univariate polynomial $p(x)$ written in factorization form:

$$\begin{aligned} p(x) &= \left(x+x^3\right)^2 + \left(x^2\right)^2 \\ &= \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{L} \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}. \end{aligned}$$

However, the factorization is not unique. Another decomposition, composed of more "summing" terms, naturally comes with more parameters in the $L$ matrix:

$$\begin{aligned} p(x) &= x^2 + 3x^4 + x^6 \\ &= \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sqrt{3} & 0 \\ 0 & 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \sqrt{3} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{L} \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix} \end{aligned}$$

As should be clear, the row size of the factorization $L$ is the number of terms that appear in the expanded SOS polynomial. Over-parameterize $L$ to be of more rows easily accommodate more possible factorizations.

An additional straightforward over-parameterization scheme is inspired by the layered architecture in modern neural networks. In particular, we can write the factorization $L$ itself as products of intermediate $l_m$ terms, each of which over-parameterized:

$$L = \prod l_m$$

While this second type of over-parameterization is less theoretically justified, it empirically improves the "smoothness" of the resulting Gram matrix, as will be shown in Subsection 6.5. It is not the focus of the this work to investigate why that is the case (but it is a topic we are interested in for future work), we simply include those evidence for completeness and an empirical justification.

Note that both over-parameterization strategies accommodate more possibilities of factorization choices, without the loss of correctness or tightness. In particular, the important positive definiteness property remains intact, and the Lyapunov derivative condition is guarded by the verification subroutine. Also, the added computation due to the over-parameterization is negligible given that we are using SGD as the computational engine.

### 6.3.2 Cost Design

Taking cues from control theory, our cost is designed to be the worst case exponential decay rate among all the samples:

$$\min_{w,L,\sigma} \max_{i} (\dot{V}_i / V_i)$$

This may look strange at first because a natural cost seems to be simply the sum or the mean of the Lyapunov derivatives; below are why we do not choose either.

First, it should be noted that one can simply scale a candidate which does not change the final verifiable result but would change $\sum \dot{V}_i$ or mean$(\dot{V}_i)$; so at least a scaling like $\dot{V}_i/V_i$ is necessary.

Even then, the scaled sum or the mean of the Lyapunov derivatives still have shortcomings. In particular, they do not distinguish whether a sample is close to the origin (so potentially have smaller, i.e., more negative $\dot{V}$) or close to the true boundary of the ROA (so only slightly negative $\dot{V}$), which is an important piece of information as it reveals which sample should contribute more into shaping the landscapes of $V$ and $\dot{V}$.

This information is nicely encoded in the proposed cost. By pushing from the above on all the samples, and penalizing only the worst-case, the general average samples do not incur cost and thus the shaping is much more largely contributed by the near-boundary ones.

### 6.3.3 Overall Algorithm

The overall algorithm is summarized in Algorithm 1.

To summarize, the algorithm is designed to efficiently generate a pair of polynomial controller $\pi(x)$ that is empirically stabilizing over a large region, and a corresponding $V(x) \succ 0$ for the closed-loop, with good $\dot{V}$ landscape. The efficiency is achieved by SGD; positive definiteness is guaranteed by searching in the factorization space, and the empirical performance goal is encoded in the cost.

## 6.4 Experiments and Examples

### 6.4.1 Closed-Loop Verification

Recall we showed Van der Pol oscillator and Pendubot verification results in Subsection 4.7.1. Those Lyapunov candidates are in fact generated using Algorithm 1. Figure 4-6a and Figure 4-6c demonstrate the tightness of the candidates.

To show the advantage of the proposed algorithm over the pure optimization-based methods, let us consider two challenging examples below.

**Dubins with marginally stable linearization.** Consider a Dubins car defined in the error frame relative to the virtual vehicle along a path to be tracked, illustrated

**Algorithm 1** Data-driven generation of Lyapunov candidate $V$ and controller $\pi$

```
Require: $f(x, u)$    ▷ The open- or closed-loop system dynamics
Require: $k^{\max}, n, \delta, d_n, d_d, d_u, \mathcal{D}, \gamma$    ▷ The number
    of training epochs, number of samples, fixed small positive real number (used in
    $V_n$), polynomial degree of $V_n$, polynomial degree of $V_d$, polynomial degree of the
    controller $\pi$, the domain of interest to control over (if the system is open-loop),
    negative $\dot{V}$ percentage
 1: if $f$ is closed-loop then
 2:     Randomly sample initial states, forward simulate and add the stable ones to
        the training set $\{x_i\}$
 3: else
 4:     Randomly sample $x_i \in \mathcal{D}$ to be added to the training set
 5: Construct symbolic standard basis $\psi(x)$, $\phi_n(x)$ and $\phi_d(x)$
 6: $k = 0, \gamma = 0$, and randomly set the weights $w, L$    ▷ Initialization
 7: while $k \leq k^{\max}$ and $\gamma < 1$ do
 8:     for all $x_i$ do
 9:         Evaluate $\pi_i$, $V_i$, and $\dot{V}_i$ with current weights    ▷ Via Eq. (6.1),(6.2)
10:         Calculate $\dot{V}_i/V_i$    ▷ Evaluate the output
11:     Update $w, L$ by decreasing the cost $\max_i(\dot{V}_i/V_i)$    ▷ Using SGD
12:     Calculate $\gamma = -\sum_i \operatorname{sign}(\dot{V}_i)/n$    ▷ the negative $\dot{V}$ percentage
13: Reconstruct $\pi(x)$, $V(x)$, $\dot{V}(x)$ from the weights $w, L$
14: return $\pi(x)$, $V(x)$, $\dot{V}(x)$
```

in Figure 6-2. The model is:

$$\begin{aligned}\dot{\psi}_E &= u_1 - k_v \ell \\ \dot{X}_E &= u_1 Y_E + u_2 - \ell \cos \psi_E \\ \dot{Y}_E &= -u_1 X_E + \ell \sin \psi_E\end{aligned} \tag{6.3}$$

where $\psi_E, \dot{X}_E, \dot{Y}_E$ are the angle error and linear displacements, $l$ and $k_v$ are the target speed and path curvature, and $u_1$ and $u_2$ are the angular and linear torques. We assume the path is straight $k_v = 0$ and the target speed $l = 2$. Stabilization at zero error means the car achieves perfect tracking. We close the loop with a simple hand-

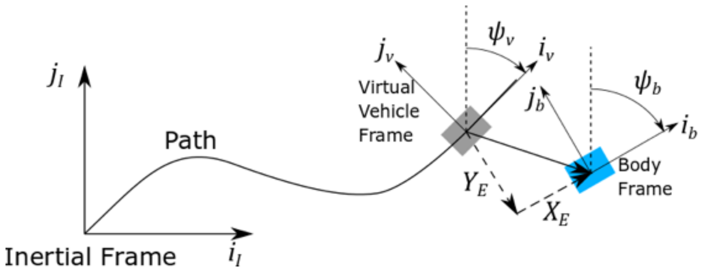


Figure 6-2: Virtual error frame for Dubins path tracking

designed feedback controller $u_1 = \pi_1(x) = -u_2 Y_E$ and $u_2 = \pi_2(x) = X_E^2 + Y_E^2 + l$. We then use a 3rd-degree Taylor expansion to take care of the sin and cos terms of the vehicle heading, and the resulting polynomial dynamics is the our verification target.

The traditional bilinear alternation fails this verification task due to an analytical limitation. In particular, one needs to linearize the dynamics for the local $A$ matrix, and then solves the Lyapunov equation for a quadratic Lyapunov function, in order to initialize the alternation. A requirement for this procedure to succeed is that $A$ is a Hurwitz matrix. In this example, the linearized $A$ matrix has eigenvalues $[0 + 2j, 0 - 2j, 0 + 0j]$ and is only marginally stable. As a result, the Lyapunov equation does not produce a qualifying solution for initialization, and the bilinear alternation is doomed to fail at the first iteration. By contrast, the proposed method succeeds because the generated Lyapunov candidate directly captures the nonlinear dynamics.

**System that admits *NO* polynomial Lyapunov function** In [1], a 2-dimensional polynomial system is given:

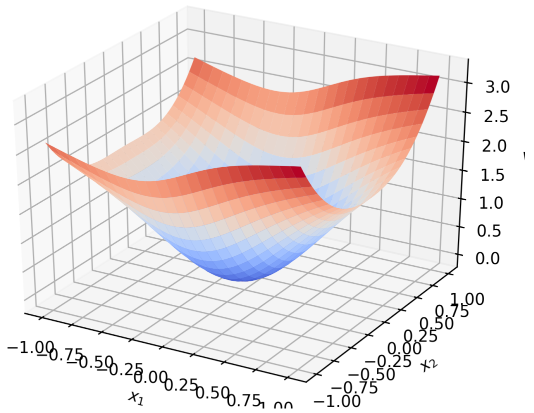
$$\begin{aligned}\dot{x}_1 &= -x_1 + x_1 x_2 \\ \dot{x}_2 &= -x_2\end{aligned} \tag{6.4}$$

This system is interesting because it is globally asymptotically stable (g.a.s.) about the origin, as proven by a hand-designed Lyapunov function of logarithm and polynomial mixtures. However, it also *provably* does not admit a polynomial Lyapunov function that can certify its g.a.s. property.

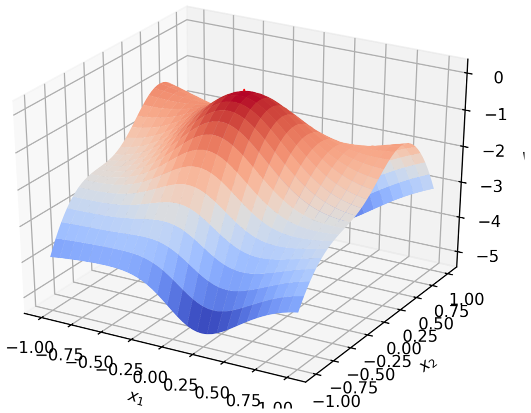
This implies that the traditional optimization-based method can not generate a suitable candidate. Using Algorithm 1, we search over the family of rational functions, and explicitly construct the following candidate $V = V_n/(V_d + 1)$ where

$$\begin{aligned}V_n &= 0.41652x_1^4x_2^2 + 0.997x_1^4 + 3.9694x_1^2x_2^4 + 3.544x_1^2x_2^2 + 3.46x_1^2 + 0.6408x_2^4 + 2.78x_2^2 \\ V_d &= 0.234x_1^4 + 2.23x_1^2x_2^2 + 1.18x_1^2 + 0.36x_2^2,\end{aligned} \tag{6.5}$$

Figure 6-3 shows the candidate $V$ and $\dot{V}$, whereas Figure 6-4 plots the contour of these two functions.



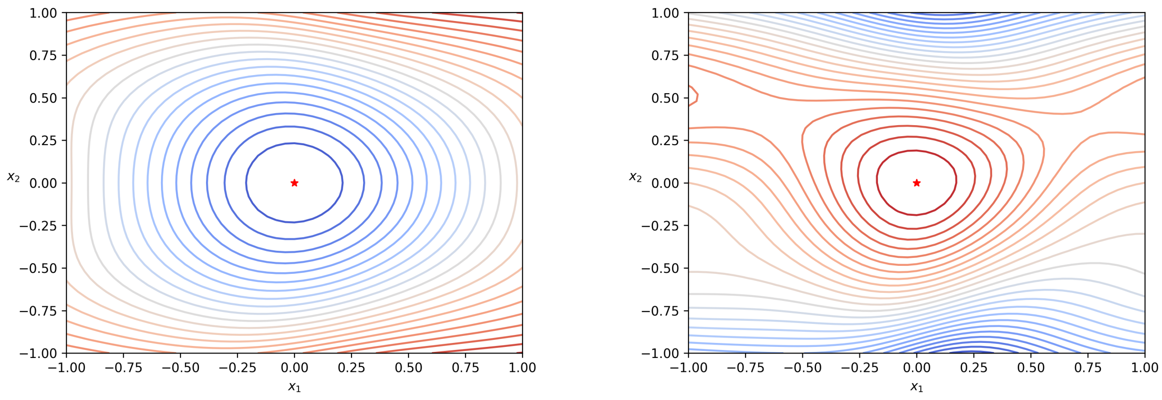
(a) Rational candidate $V$



(b) The corresponding $\dot{V}$

Figure 6-3: Rational candidate $V$ and the corresponding $\dot{V}$ for system Eq. 6.4

The numerator of the corresponding $-\dot{V}$ is verified to be SOS, thereby proving the g.a.s. property (this is because once the candidate is provided, the denominator of $\dot{V}$ is the fixed $(V_d + 1)^2$ and positive definite, so the verification of $-\dot{V}$ boils down



(a) Contour of the rational candidate $V$ (b) Contour of $\dot{V}$

Figure 6-4: Contour of the rational candidate $V$ and the corresponding $\dot{V}$ for system Eq. 6.4

to verifying its numerator only.

Note that this example highlights again the advantage of SafetyNet over the pure convex-optimization-based verification scheme. In particular, since for any rationally parameterized $V = \frac{V_1}{V_2}$, its time-derivative is

$$\dot{V} = \frac{V_2\dot{V}_1 - V_1\dot{V}_2}{V_2^2},$$

a simultaneously search of both components breaks the fundamental convex search assumption.

### 6.4.2 Simultaneous Generation of $\pi$ and $V$

In this subsection, we consider simultaneous synthesis and verification. We highlight that the proposed method generates controller that relies on only simple sampling and no domain knowledge, yet generalizes well.

**Inverted Pendulum Recast.** Consider the task of controlling and balancing a damped pendulum (dynamics $ml^2\ddot{\theta} = u - mgl\sin\theta - b\dot{\theta}$ with states $[\theta, \dot{\theta}]$) at the upright fixed point $[\pi, 0]$. We recast the states into $[s, c, \dot{\theta}]$ where $s \equiv \sin\theta$ and $c \equiv \cos\theta$, such that the dynamics is polynomial:

$$\begin{bmatrix} \dot{s} \\ \dot{c} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} c\dot{\theta} \\ -s\dot{\theta} \\ (u - mgls - b\dot{\theta})/(ml^2) \end{bmatrix} \tag{6.6}$$

subject to the unit circle constraint $s^2 + c^2 = 1$.

In the recast coordinate, the target fixed point is $[0, -1, 0]$. We set $\deg V = 4$, $\deg u = 1$ to search for a controller and a candidate. The algorithm produces the control law:

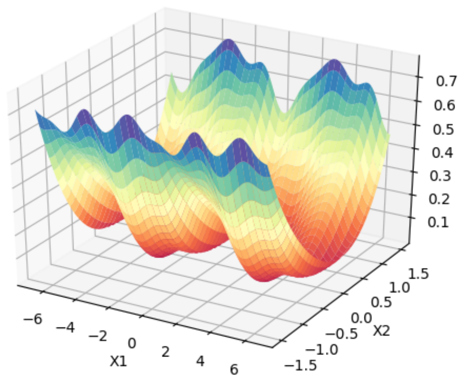
$$\pi(x) = 6.8679447 \sin(\theta) + 0.36002526(\cos(\theta) + 1) - 4.6468804\dot{\theta} \tag{6.7}$$

and a corresponding degree four Lyapunov function whose derivative $\dot{V}(x)$ is globally negative semidefinite on the unit circle in $\mathbb{R}^3$, as certified by a SOS decomposition.

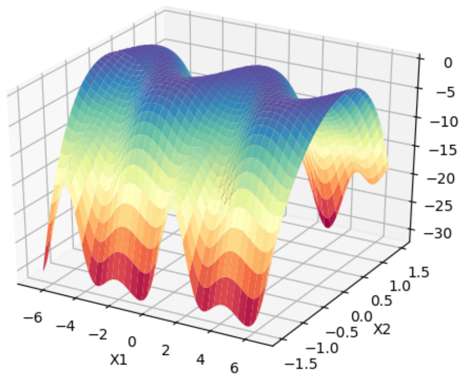
By LaSalle's invariance principle, this semi-definitness implies that all states converge to the invariant set where $\dot{V} = 0$. Note that all states where $\dot{V} = 0$ are transient except for the fixed points, and the resulting closed-loop using the control law Eq. (6.7) has two fixed points: one is the target $[0, -1, 0]$, and the other $[s = -0.354883, c = 0.9349107, \dot{\theta} = 0]$. The linearization at this other fixed point has one strictly positive eigenvalue, thus non-stable [34]. Combining this fact with the global $-\dot{V}$ certificate and LaSelle, the produced controller is guaranteed to be globally stabilizing, except for the measure-zero set of initial conditions on the stable manifold (Chapter 2, Perko book) of the non-stable fixed point. The results are visualized in Figure 6-5.

It is worth pointing out that compared with the well-known energy-shaping controller, the proposed method synthesizes a simple controller that is also verifiably globally stabilizing; but the procedure does not relying on any domain knowledge.

Another advantage of the proposed method is its generalization capability. In particular, the controller is "trained" on only 10,000 samples, uniformly grid-sampled within the square $\{|\theta| \leq \pi/2, |\dot{\theta}| \leq 1\}$, yet it generalizes *globally*, a major improvement



(a) $V$



(b) $\dot{V}$

Figure 6-5: $V$ and $\dot{V}$ of the pendulum plant (mass $m = 1$, length $l = 0.5$, damping $b = 0.1$ and gravity $g = 9.81$) controlled by the sampling-generated feedback controller $\pi(x) = 6.8679447\sin(\theta) + 0.36002526(\cos(\theta)+1) - 4.6468804\dot{\theta}$. Both $V$ and $\dot{V}$ are recast back from the $[s, c, \dot{\theta}]$ coordinate to the original $[\theta, \dot{\theta}]$ for plotting.

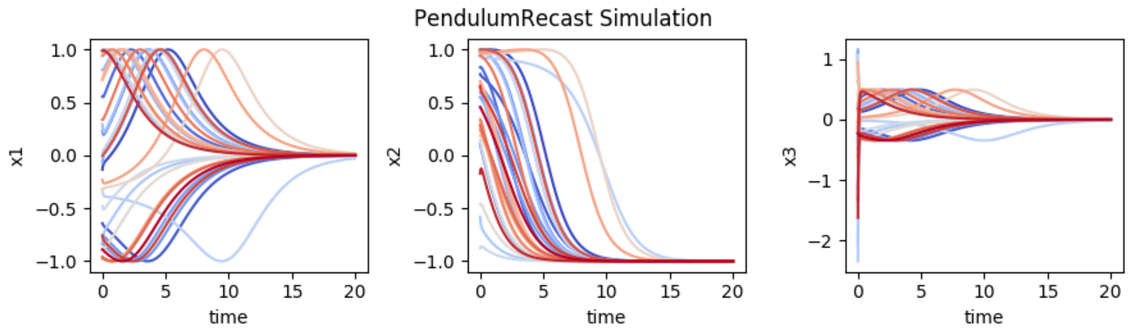


Figure 6-6: Forward simulations of 30 random initial conditions for the Pendulum Recast plant with sampling-based trigonometric feedback controller.

over most learning-based methods such as reinforcement learning that are well-known to suffer from sample complexity issues.

We interpret intuitively that this is due to the simultaneous search of a corresponding $V$, with good $\dot{V}$ landscape encouraged by the cost. In other words, while searching for the controller, we are already implicitly "improving" the downstream verification quality.

**Virtual Dubins Recast.** Consider the Dubins plant again, in this example, the controller is not given a-priori but to be synthesized. We recast the sin and cos terms, such that the model becomes:

$$
\begin{aligned}
\dot{s}_E &= c_E\left(u_1 - k_v l\right) \\
\dot{c}_E &= -s_E\left(u_1 - k_v l\right) \\
\dot{X}_E &= u_1 Y_E + u_2 - l c_E \\
\dot{Y}_E &= -u_1 X_E + l s_E
\end{aligned}
\tag{6.8}
$$

where all the parameters are as before, expect the fixed point becomes $[0, 1, 0, 0]$.

Similar with the pendulum case, by parameterizing a linear controller with feedback of the sin and cos, we again achieve a globally stabilizing controller, certified by having a SOS certificate of the accompanying 4-dimensional 4-degree $\dot{V}$ on the unit circle. A simulation of 30 random initial conditions are shown in Figure 6-7.

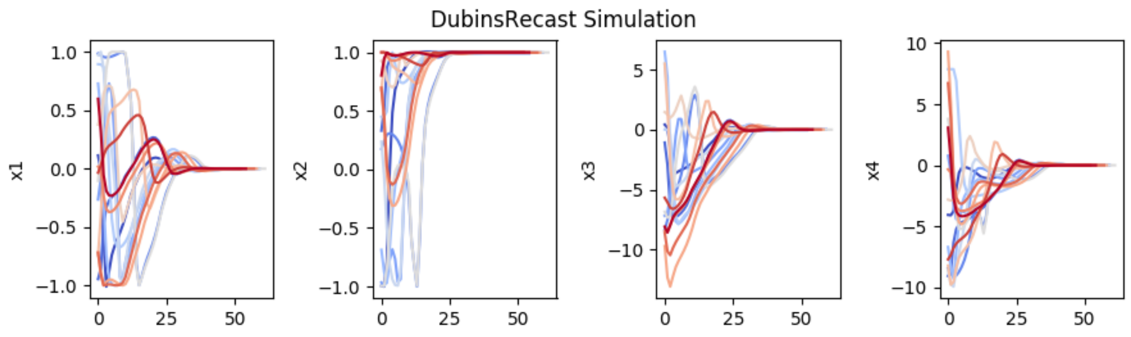


Figure 6-7: Forward simulations of 30 random initial conditions for the Dubins Recast plant with sampling-based trigonometric feedback controller.

Compared with the given Dubins with hand-designed controller, this example also demonstrates the power of not only synthesizing but also verifying on the sin and cos instead of angels themselves. Analysis involving angles has two limitations. First, due to $2\pi$ period of angles, only local certificates are achievable. Second, angles mostly appear in the dynamics in trigonometric forms, and as such would there is a gap between the true dynamics in angles and polynomial analysis.

The recasting technique can overcome these two limitations, but it introduces constraints such as unit circle. Just as in the ROA analysis formulation, such constraints also relies on Lagrange multipliers, which results in a similarly bloated SDP. The verification method proposed in Chapter 4, by taking advantage of the sampling variety subroutine, addresses this computational overhead as well.

## 6.5 Discussion and Future Work

**Understand why over-parameterization is numerically "smoothing"** Preliminary experiments show that over-parameterization leads to more "balanced" or "smoother" Gram matrix.

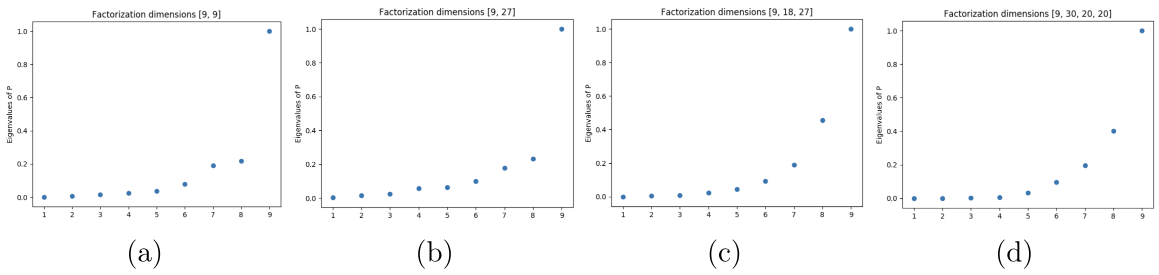


Figure 6-8: Eigenvalues of four 9-by-9 Gram matrix using different factorization sizes. For example, the Gram matrix $Q_a$ in plot (a) is constructed from the factorization $Q = L_1' L_1$ where $L_1 \in \mathbb{R}^{9\times 9}$, the one in plot (b) $Q_b$ is constructed from the factorization $Q = L_1' L_2' L_2 L_1$ where $L_1 \in \mathbb{R}^{9\times 9}$, and $L_1 \in \mathbb{R}^{9\times 27}$, and similarly for the next two plots.

Shown in Figure 6-8 is the eigenvalue plots of four 9-by-9 Gram matrix, found using different factorization schemes. Visually, the one with the most "components" and the most parameters has the smoothest distribution of eigenvalues.

The plots above are for particular Gram matrix and only offers visual cues, and we would like to examine the trend of the eigenvalue distribution versus the parameterization richness. Therefore, we run 1000 tests for each parameterization, and resort to entropy of the eigenvalue array to get a better overview. Recall that the uniform distribution, among discrete distributions, has the largest entropy, therefore, the identity matrix has the largest entropy of a fixed-size PSD matrix. For each parameterization scheme, we run Algorithm 1 1,000 times, and average out over this 1000 trials the entropies of the reconstructed Gram matrix.

It is interesting to observe that with more parameters, the entropy also in general increases, indicating a more central Gram matrix. Of course, the benefit, if any, is only numerical and investigation into that is left for future work.

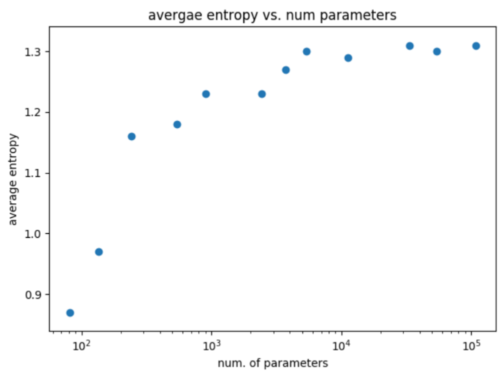


Figure 6-9: Average entropy over 1,000 Gram matrix for each fixed parameterization scheme

**Incorporate control saturation in controller synthesis** While we use samples to synthesize a globally stabilizing controller for the simple pendulum in the previous section, no input saturation was considered, which is rather unrealistic.

With very little modification, control saturation can be incorporated into SafetyNet, e.g., by any of the following procedure: (i) passing the polynomial control law $u$ through a saturation layer like the tanh function we are so familiar with (the main topic discussed in Chapter 5), (ii) simply clipping $u$ at saturation, and (iii) anding a, e.g., quadratic cost, on the control output $u$. Experiments on these ideas are left for future work.

Also, it is important to consider possible discontinuities necessary to enlarge the stabilizing area, for example, at the downward fixed point, should the controller swing left or right? Such discontinuous decision-making is most straightforwardly handled via an integer state/decision variable. However, this inevitably complicates the downstream verification; given our main computational tool is the SOS programming, a mix-integer SOS program is not likely to scale well—even with the help of the scale-improving techniques we developed in Part I. How to balance this tension between "easy to control" is "easy to verify", the central consideration for SafetyNet, but in a discrete decision-making context, is an important open (as far as we know) question.

# Bibliography

[1] Amir Ali Ahmadi and Bachir El Khadir. A globally asymptotically stable polynomial vector field with rational coefficients and no local polynomial Lyapunov function. *Systems and Control Letters*, 121(0):50–53, 2018.

[2] Michael E Akintunde, Andreea Kevorchian, Alessio Lomuscio, and Edoardo Pirovano. Verification of RNN-Based Neural Agent-Environment Systems. 2018.

[3] James Anderson and Antonis Papachristodoulou. A Decomposition Technique for Nonlinear Dynamical System Analysis. *IEEE Transactions on Automatic Control*, 57(6):1516–1521, 2012.

[4] Mituhiko Araki. Stability of Large-Scale Nonlinear Systems-Quadratic-Order Theory of Composite-System Method Using M-Matrices. *IEEE Transactions on Automatic Control*, 23(2):129–142, 1978.

[5] Murat Arcak and Eduardo D. Sontag. Diagonal stability of a class of cyclic systems and its connection with the secant criterion. *Automatica*, 42(9):1531–1537, 2006.

[6] Lubomír Bakule. Decentralized control: An overview. *Annual Reviews in Control*, 32(1):87–98, 2008.

[7] Venkataramanan Balakrishnan and Lieven Vandenberghe. Semidefinite programming duality and linear time-invariant systems. *IEEE Transactions on Automatic Control*, 48(1):30–41, 2003.

[8] G. P. Barker, A. Berman, and R. J. Plemmons. Positive diagonal solutions to the Lyapunov equations. *Linear and Multilinear Algebra*, 5(4):249–256, 1978.

[9] Andrew J. Barry, Anirudha Majumdar, and Russ Tedrake. Safety verification of reactive controllers for UAV flight in cluttered environments using barrier certificates. *ICRA*, 2012.

[10] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In *Advances in neural information processing systems*, pages 2613–2621, 2016.

[11] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[12] Felix Berkenkamp, Riccardo Moriconi, Angela P. Schoellig, and Andreas Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes. *2016 IEEE 55th Conference on Decision and Control, CDC 2016*, pages 4661–4666, 2016.

[13] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. Linear Matrix Inequalities in System and Control Theory. In *Studies in Applied Mathematics*, volume 15, page 203. Society for Industrial and Applied Mathematics (SIAM), 1994.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*, volume 25. Cambridge University Press, 2010.

[15] Samuel Burer and Renato D. C. Monteiro. A Nonlinear Programming Algorithm for Solving Semidefinite Programs via Low-Rank Factorization. 357:329–357, 2003.

[16] Brad Carlile, Guy Delamarter, Paul Kinney, Akiko Marti, and Brian Whitney.

Improving Deep Learning by Inverse Square Root Linear Units (ISRLUs). 0(1):1–8, 2017.

[17] Nicholas Carlini and David Wagner. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. 2018.

[18] David Carlson, Daniel Hershkowitz, and Dafna Shasha. Block diagonal semistability factors and Lyapunov semistability of block triangular matrices. *Linear Algebra and Its Applications*, 172(C):1–25, 1992.

[19] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural Lyapunov Control. (NeurIPS), 2019.

[20] Diego Cifuentes and Pablo A. Parrilo. Sampling Algebraic Varieties for Sum of Squares Programs. *SIAM Journal on Optimization*, 27(4):2381–2404, 2017.

[21] David Cox, John Little, and Shea Donal. *Ideals, Varieties, and Algorithms.* Springer, third edition, 2000.

[22] Włodzisław Duch and Norbert Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, 2:163–212, 1999.

[23] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai 2: Safety and robustness certification of neural networks with abstract interpretation.

[24] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. pages 1–11, 2014.

[25] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[26] Matthew Hausknecht and Peter Stone. Deep Recurrent Q-Learning for Partially Observable MDPs. 2015.

[27] Bo-jian Hou and Zhi-hua Zhou. Learning with Interpretable Structure from RNN. pages 1–26, 2018.

[28] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*, pages 3–29. Springer, 2017.

[29] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015.

[30] Yuval Jacoby, Clark Barrett, and Guy Katz. Verifying Recurrent Neural Networks using Invariant Inference. 2020.

[31] James Kapinski, Sriram Sankaranarayanan, Jyotirmoy V. Deshmukh, and Nikos Aréchiga. Simulation-guided Lyapunov analysis for hybrid dynamical systems. In *Hybrid Systems: Computation and Control*, pages 133–142, 2014.

[32] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. *Computer Aided Verification*, pages 97–117, 2017.

[33] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[34] khalil. Lecture notes in nonlinear systems and control.

[35] Hassan K Khalil. *Nonlinear Systems - Third Edition.* Prentice Hall, 2002.

[36] Alexandar Kozarev, John Quindlen, Jonathan P. How, and Ufuk Topcu. Case Studies in Data-Driven Verification of Dynamical Systems. In *Hybrid Systems: Computation and Control*, pages 81–86, 2016.

[37] Raph Levien. A few of my favorite sigmoids. https://raphlinus.github.io/audio/2018/09/05/sigmoid.html, Sep 2018.

[38] Johan Löfberg and Pablo A. Parrilo. From coefficients to samples: A new approach to SOS optimization. In *IEEE Conference on Decision and Control*, volume 3, pages 3154–3159, 2004.

[39] Chris Lomont. Fast inverse square root. https://www.lomont.org/Math/Papers/2003/InvSqrt.pdf, Feb 2003.

[40] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[41] Anirudha Majumdar. *Funnel Libraries for Real-Time Robust Feedback Motion Planning*. PhD thesis, MIT, 2016.

[42] Anirudha Majumdar, Ram Vasudevan, Mark M. Tobenkin, and Russ Tedrake. Technical Report: Convex Optimization of Nonlinear Feedback Controllers via Occupation Measures. 2013.

[43] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS):1800–1809, 2018.

[44] Richard P Mason and Antonis Papachristodoulou. Chordal Sparsity, Decomposing SDPs and the Lyapunov Equation.

[45] Alexandre Megretski. Relaxations of quadratic programs in operator theory and system analysis. *Systems, Approximation, Singular Integral Operators, and Related Topics*, pages 365–392, 2001.

[46] Alexandre Megretski and Anders Rantzer. System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, 1997.

[47] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 8.*, 2017.

[48] Kumpati S. Narendra and Robert Shorten. Hurwitz stability of Metzler matrices. *IEEE Transactions on Automatic Control*, 55(6):1484–1487, 2010.

[49] OpenAI. Learning Dexterous In-Hand Manipulation. *CoRR*, pages 1–27, 2018.

[50] Antonis Papachristodoulou and Stephen Prajna. Analysis of Non-polynomial Systems Using the Sum of Squares Decomposition. 2005.

[51] Pablo A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, Caltech, 2000.

[52] Pablo A. Parrilo. Exploiting Algebraic Structure in Sum of Squares Programs. *LNCIS*, 312:181–194, 2005.

[53] Frank Noble Permenter. *Reduction Methods in Semidefinite and Conic Optimization*. PhD thesis, MIT, 2017.

[54] Michael Posa. *Optimization for Control and Planning of Multi-contact Dynamic Motion*. PhD thesis, MIT, 2017.

[55] Stephen Prajna. *Optimization-based methods for nonlinear and hybrid systems verification*. PhD thesis, Caltech, 2005.

[56] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. A Framework for Worst-Case and Stochastic Safety Verification Using Barrier Certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428, 2007.

[57] John F. Quindlen, Ufuk Topcu, Girish Chowdhary, and Jonathan P. How. Active Sampling for Constrained Simulation-based Verification of Uncertain Nonlinear Systems. *ACC*, pages 6259–6265, 2018.

[58] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pages 10900–10910, 2018.

[59] Anders Rantzer. Dynamic dual decomposition for distributed control. *Proceedings of the American Control Conference*, pages 884–888, 2009.

[60] Max Revay, Ruigang Wang, and Ian R. Manchester. Convex Sets of Robust Recurrent Neural Networks. 2020.

[61] Spencer M. Richards, Felix Berkenkamp, and Andreas Krause. The Lyapunov Neural Network: Adaptive Stability Certification for Safe Learning of Dynamical Systems. (CoRL), 2018.

[62] R. T. Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1(2):97–116, 1976.

[63] Carsten Scherer, Pascal Gahinet, and Mahmoud Chilali. Multiobjective output-feedback control via LMI optimization. *IEEE Transactions on Automatic Control*, 42(7):896–911, 1997.

[64] Shen Shen. Verification of recurrent neural networks via systems and control theory. Quest Symposium on Robust, Interpretable Deep Learning Systems, 2018.

[65] Shen Shen and Russ Tedrake. Compositional Verification of Large-Scale Nonlinear Systems via Sums-of-Squares Optimization. In *ACC*, 2017.

[66] Shen Shen and Russ Tedrake. Sampling Quotient-Ring Sum-of-Squares Programs for Scalable Verification of Nonlinear Systems. In *CDC*, 2020.

[67] Robert N Shorten and Kumpati S. Narendra. On a theorem of Redheffer concerning diagonal stability. *Linear Algebra and Its Applications*, 431:2317–2329, 2009.

[68] Christoffer Sloth, Rafael Wisniewski, and George J. Pappas. On the existence of compositional barrier certificates. In *IEEE Conference on Decision and Control*, pages 4580–4585, 2012.

[69] Aivar Sootla and James Anderson. On existence of solutions to structured Lyapunov inequalities. *Proceedings of the American Control Conference*, 2016-July(0):7013–7018, 2016.

[70] Jacob Steinhardt and Russ Tedrake. Finite-time regional verification of stochastic non-linear systems. *The International Journal of Robotics Research*, 2012.

[71] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

[72] Takashi Tanaka and Cedric Langbort. The Bounded Real Lemma for Internally Positive Systems and H-Infinity Structured Static State Feedback. *IEEE Transactions on Automatic Control*, 56(9):2218–2223, 2011.

[73] Russ Tedrake, Ian R. Manchester, Mark M. Tobenkin, and John W. Roberts. LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification. *IJRR*, 29(8):1038–1052, 2010.

[74] Matus Telgarsky. Neural networks and rational functions. *34th International Conference on Machine Learning, ICML 2017*, 7:5195–5210, 2017.

[75] Vincent Tjeng, Kai Y Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. 2018.

[76] Mark M. Tobenkin, Frank Permenter, and Alexandre Megretski. Spotless polynomial and conic optimization. https://github.com/spot-toolbox/spotless, 2015.

[77] O. Toker and H. Özbay. On the np hardness of solving bilinear matrix inequalities and simultaneous stabilization with static output feedback. In *American control conference : Seattle, Washington*, 1995.

[78] Ufuk Topcu, Andrew Packard, and R.M. Murray. Compositional stability analysis based on dual decomposition. *Proceedings of the 48h IEEE Conference on Decision and Control (CDC)*, pages 1175–1180, 2009.

[79] Ufuk Topcu, Andrew Packard, and Peter J. Seiler. Local stability analysis using simulations and sum-of-squares programming. *Automatica*, 2008.

[80] Jos van der Westhuizen and Joan Lasenby. The unreasonable effectiveness of the forget gate. *CoRR*, abs/1804.04849, 2018.

[81] Jan Camiel Willems. Lyapunov functions for diagonally dominant systems. *Automatica*, 12(5):519–523, 1976.

[82] Eric Wong and J. Zico Kolter. Provable Defenses Against Adversarial Examples via the Convex Outer Adversarial Polytope. 2017.

[83] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292, 2018.

[84] V. A. Yakubovich. Nonconvex optimization problem: The infinite-horizon linear-quadratic control problem with quadratic constraints. *Systems and Control Letters*, 19(1):13–22, 1992.