Lyapunov Function Synthesis using Handelman Representations.

Sriram Sankaranarayanan * **Xin Chen** ** **Erika Ábrahám** **

* *University of Colorado, Boulder, CO, USA (e-mail: first.lastname@colorado.edu).*
** *RWTH Aachen University, Aachen, Germany (e-mail: {xin.chen,abraham}@cs.rwth-aachen.de)*

Abstract: We investigate linear programming relaxations to synthesize Lyapunov functions that establish the stability of a given system over a bounded region. In particular, we attempt to discover functions that are more readily useful inside symbolic verification tools for proving the correctness of control systems. Our approach searches for a Lyapunov function, given a parametric form with unknown coefficients, by constructing a system of linear inequality constraints over the unknown parameters. We examine two complementary ideas for the linear programming relaxation, including interval evaluation of the polynomial form and "Handelman representations" for positive polynomials over polyhedral sets. Our approach is implemented as part of a branch-and-relax scheme for discovering Lyapunov functions. We evaluate our approach using a prototype implementation, comparing it with techniques based on Sum-of-Squares (SOS) programming. A comparison with SOSTOOLS is carried out over a set of benchmarks gathered from the related work. The evaluation suggests that our approach using Simplex is generally fast, and discovers Lyapunov functions that are simpler and easy to check. They are suitable for use inside symbolic formal verification tools for reasoning about continuous systems.

Keywords: Stability, Lyapunov Functions, Linear Programming, Interval Arithmetic, Sum Of Squares.

1. INTRODUCTION

In this paper, we examine linear programming relaxations for discovering Lyapunov functions for polynomial systems to prove asymptotic stability over a given region U. The search for Lyapunov functions typically involves the use of a template form $V(\boldsymbol{x}, \boldsymbol{c})$ over the state variables $\boldsymbol{x}$ and parameters $\boldsymbol{c}$. The Lyapunov conditions are encoded as constraints on the unknown parameters $\boldsymbol{c}$, and solved to obtain Lyapunov functions. Essential primitives include (a) deciding whether a given polynomial $p(\boldsymbol{x})$ is positive (semi-) definite over U and (b) deriving constraints on the parameters $\boldsymbol{c}$ that ensure that $V(\boldsymbol{x}, \boldsymbol{c})$ is non-negative over U. In general, there are many approaches to check and encode polynomial positivity. These include exact approaches based on quantifier elimination over the reals (Collins [1975], Collins and Hong [1991], Dolzmann and Sturm [1997], Tarski [1951]), Sum-Of-Squares programming (Shor [1987] and Parrilo [2000]) and linear programming relaxations using interval evaluation (Ratschan and She [2010]), and the Handelman representations (Handelman [1988]), considered here.

The Sum-Of-Squares (SOS) programming approach allows us to decide if a polynomial is positive (semi-) definite by expressing it as a sum of squares of polynomials. This is achieved by formulating and solving a semi-definite program (SDP). The approach extends naturally to parameterized polynomial forms, and can be used to carry out Lyapunov function synthesis (Papachristodoulou and Prajna [2002], Tibken [2000], Topcu et al. [2007]). The implementation of these ideas in SOSTOOLS (Prajna et al. [2004]) has spurred further progress in this direction. However, SOS programming primarily relies on numerical interior-point SDP solvers that use floating point arithmetic (Boyd and Vandenberghe [2004]). As a result, a Lyapunov function discovered by the approach may often be invalid due to numerical errors. Further heuristics are often needed to post-process the results to ensure their validity. Thus, in the context of finding *proofs* of stability that are useful for formal verification, techniques based on SOS can be problematic.

Our approach uses a linear programming relaxation of the problem that can be solved exactly using algorithms such as Simplex (Chvátal [1983]). The relaxation uses combination of interval evaluation of polynomials, and so-called *Handelman representations* to ensure the non-negativity of a polynomial form over a region. Interval arithmetic techniques (Moore et al. [2009]) evaluate polynomials over intervals to provide guaranteed bounds on their range. Ratschan & She (Ratschan and She [2010]) use interval arithmetic to achieve a Linear Programming formulation of the conditions for a Lyapunov-like function that establishes asymptotic convergence to a set of states. However, the interval evaluation used is quite conservative, and often fails to derive Lyapunov functions. This is remedied by applying interval evaluation approaches in a *branch-and-relax* scheme that repeatedly subdivides the state-space. In this paper, we consider Handelman representations to complement interval evaluation. Our approach draws inspiration from a well known (but not quite well used) idea (Datta [2002], Parrilo and Sturmfels [2001]) originating from a result by (Handelman [1988]) that characterizes positive polynomials over compact polyhedra. We compare the power of interval evaluation and Handelman representations, providing evidence that both approaches have complementary strengths. Their combination can produce

a linear programming relaxation that is more suitable for Lyapunov function synthesis.

We have implemented our approach using a OCaml-based front-end to compute constraints and a linear programming solver GLPK to find solutions. We compare our approach with a Lyapunov function search implemented inside the SOSTOOLS package (Prajna et al. [2004]). On a suite of benchmarks drawn from the related work, we find that our approach is fast, and can discover Lyapunov proofs that are *reliable*: they can be formally verified inside theorem provers, and in turn used to prove various correctness properties of control systems.

2. PRELIMINARIES

We recall some basic notions including Lyapunov functions, positive (semi-) definite polynomials and techniques to search for positive definite polynomials under some constraints on their coefficients.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is *positive semi-definite* over a domain $U \subseteq \mathbb{R}^n$ iff $(\forall\ \boldsymbol{x} \in U)\ f(\boldsymbol{x}) \geq 0$. Furthermore, a positive semi-definite function f is positive definite iff $f(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in U \setminus \{0\}$, and $f(0) = 0$.

2.1 Lyapunov Functions

Throughout this paper, we study continuous dynamical systems $\mathcal{S}$ over a state-space $\mathcal{X} \subseteq \mathbb{R}^n$ specified by coupled time invariant Ordinary Differential Equations (ODEs):

$$\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}),\ \boldsymbol{x} \in \mathcal{X}\,.$$

We assume that f is Lipschitz continuous over $\boldsymbol{x}$. An equilibrium of the system $\boldsymbol{x}^* \in \mathcal{X}$ satisfies $f(\boldsymbol{x}^*) = 0$.

A system is *Lyapunov stable* over an open region U around the equilibrium $\boldsymbol{x}^*$, if for every neighbourhood $N \subseteq U$ of $\boldsymbol{x}^*$ there is a neighbourhood $M \subset N$ such that $(\forall\ \boldsymbol{x}(0) \in M)\ (\forall\ t \geq 0)\ \boldsymbol{x}(t) \in N$. A system is *asymptotically* stable if it is Lyapunov stable and all trajectories starting from U approach $\boldsymbol{x}^*$ as $t \to \infty$. Lyapunov functions are useful in proving that a system is stable in a region around the equilibrium. Without loss of generality, we assume that $\boldsymbol{x}^* = \mathbf{0}$. The definitions below are based on the terminology used by (Meiss [2007]).

Definition 2.1. A continuous and differentiable function $V(\boldsymbol{x})$ is a *weak Lyapunov function* over a region $U \subseteq \mathcal{X}$ iff the following conditions hold:

(1) $V(\boldsymbol{x})$ is positive definite over U, i.e, $V(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in U \setminus \{\mathbf{0}\}$ and $V(\mathbf{0}) = 0$.
(2) $\frac{dV}{dt} = (\nabla V \cdot f) \leq 0$ for all $\boldsymbol{x} \in U$.

Additionally, V is a *strong Lyapunov function* if $\left(-\frac{dV}{dt}\right)$ is positive definite.

Weak Lyapunov functions are used to prove that a system is Lyapunov stable in a region U whereas a strong Lyapunov function proves asymptotic stability. The approaches presented in this paper can be used to search for weak as well as strong Lyapunov functions.

2.2 Searching for Lyapunov Functions

Stability is an important property of control systems. Techniques for discovering Lyapunov functions to certify the stability of a closed loop control system model are quite useful in control systems design.

One commonly used class of techniques assumes a *template* form of the function $V(\boldsymbol{x}, \boldsymbol{c})$ where $\boldsymbol{c} = (c_1, \ldots, c_k)$ is a set of *unknown parameters*. Our goal is to search for values of $c_i \in \mathbb{R}$ that yield a Lyapunov function $V_c(\boldsymbol{x})$ upon substitution.

Let $U \subseteq \mathcal{X}$ be a *region of interest* over which a Lyapunov function is sought to prove that all trajectories starting from U are stable around the origin. We assume that U is bounded and the origin belongs to the interior of U. To find a Lyapunov function, we encode the conditions in Def. 2.1 to yield constraints over the unknown parameters $\boldsymbol{c}$. Solving these constraints yields us values of $\boldsymbol{c}$ that can be used to formulate a suitable Lyapunov function. However, if no value of $\boldsymbol{c}$ can be found, we conclude either (a) there exists $\boldsymbol{x}_0 \in U$ such that the trajectory with $\boldsymbol{x}(0) = \boldsymbol{x}_0$ is unstable, or (b) the template form and techniques used to formulate the constraints are inadequate.

At the heart of Lyapunov function synthesis, we face the challenge of establishing that a given function $V(\boldsymbol{x})$ is positive (negative) definite over U. In turn, we extend this to templates $V(\boldsymbol{x}, \boldsymbol{c})$, wherein we wish to characterize a set C for the unknown parameters $\boldsymbol{c}$, so $V_c(\boldsymbol{x})$ is positive definite over U for all $\boldsymbol{c} \in C$. Thus, the process of searching for Lyapunov functions of a given form devolves into the problem of finding a system of constraints for the set C.

We assume that the form $V(\boldsymbol{x}, \boldsymbol{c})$ is a polynomial over $\boldsymbol{x}$ whose coefficients are polynomials over $\boldsymbol{c}$. We define the set C as follows:

$$C = \{\boldsymbol{c} \mid V(\boldsymbol{x}, \boldsymbol{c}) \text{ is positive definite for } \boldsymbol{x} \in U\}\,. \quad (1)$$

The overall procedure for synthesizing Lyapunov functions proceeds as follows:

(1) Fix a template form $V(\boldsymbol{x}, \boldsymbol{c})$ with parameters $\boldsymbol{c}$.
(2) Compute constraints $\psi[\boldsymbol{c}]$ that characterize the set C in Equation (1).
(3) Consider the template form $V'(\boldsymbol{x}, \boldsymbol{c}) = \nabla_{\boldsymbol{x}} V \cdot f(\boldsymbol{x})$. We compute constraints $\psi'[\boldsymbol{c}]$ that characterizes the set C' such that $V'(\boldsymbol{x}, \boldsymbol{c})$ is negative (semi-) definite.
(4) Compute a value $\boldsymbol{c} \in C \cap C'$ by solving the constraints $\psi \wedge \psi'$. The resulting function $V_c(\boldsymbol{x})$ is a Lyapunov function.

The problem of deciding whether a given polynomial $V(\boldsymbol{x})$ is positive definite is NP-hard. The problem is even harder for encoding constraints to characterize the sets C, C'. A precise solution requires quantifier elimination over reals (Collins [1975], Tarski [1951]). To wit, we eliminate the universal quantifiers from the formula: $(\forall\ \boldsymbol{x} \in U)\ V(\boldsymbol{x}, \boldsymbol{c})$ using tools such as QEPCAD (Collins and Hong [1991]) and REDLOG (Dolzmann and Sturm [1997]). This process is *exact*, but intractable for all but the smallest of systems and low degree polynomials for V. Furthermore, the resulting constraints $\psi \wedge \psi'$ that characterize $C \cap C'$ are semi-algebraic. Finding a solution $\boldsymbol{c}$ for $\psi \wedge \psi'$ is therefore a hard problem.

Therefore, we seek stricter versions of positive semi-definiteness that yield a more tractable system of constraints. We now examine three such approaches, starting from the SOS decomposition, which is considered to be the most promising approach.

2.3 Sum-of-Squares (SOS) Programming

SOS decomposition shows that a given polynomial $V(\boldsymbol{x})$ is positive semi-definite by expressing it as the sum of squares: $V(\boldsymbol{x}) = V_0(\boldsymbol{x})^2 + V_1(\boldsymbol{x})^2 + \cdots + V_k(\boldsymbol{x})^2$ for polynomials $V_0,\ V_1,\ \ldots,\ V_k$. Likewise, given a form $V(\boldsymbol{x}, \boldsymbol{c})$, we find constraints over $\boldsymbol{c}$ that enable such an SOS decomposition. In order to find a SOS representation of a polynomial, a *semi-definite programming* (SDP) relaxation was first proposed by (Shor [1987]) and further developed by (Parillo [2003]). For a given form $V(\boldsymbol{x}, \boldsymbol{c})$, we derive a SDP over $\boldsymbol{c}$ whose solutions are guaranteed to yield SOS decomposition. In turn, efficient SDP solvers based on interior point methods can be used to carry out the search for a feasible solution. The SOSTOOLS package in Matlab(tm) provides one such implementation (Prajna et al. [2004]). The use of SOS to perform stability analysis for polynomial ODEs was explored by (Parrilo [2000]), (Tibken [2000]), (Papachristodoulou and Prajna [2002]), and more recently using simulations (Topcu et al. [2007]).

It is well known that not every positive (semi-) definite polynomial can be expressed through a SOS decomposition. Nevertheless, the SOS decomposition is quite powerful in practice. In many systems, it is quite natural to find SOS proofs of stability. However, the SOS approach relies on numerical interior point solvers to find a feasible point. From the point of view of a *guaranteed method*, such an approach can be problematic. We illustrate this using a simple example.

Example 2.1. Consider the ODE shown in (Papachristodoulou and Prajna [2002])):

$$\frac{dx_1}{dt} = -x_1^2 - 4x_2^3 - 6x_3x_4 \quad \frac{dx_2}{dt} = -x_1 - x_2 + x_5^3$$
$$\frac{dx_3}{dt} = x_1x_4 - x_3 + x_4x_6 \quad \frac{dx_4}{dt} = x_1x_3 + x_3x_6 - x_4^3$$
$$\frac{dx_5}{dt} = -2x_2^3 - x_5 + x_6 \quad \frac{dx_6}{dt} = -3x_3x_4 - x_5^3 - x_6$$

Using the `findlyap` function that is available as part of SOSTOOLS (Prajna et al. [2004]), we obtain a large polynomial partially shown below:

$$2.8579 \times 10^{-6}x_1^4 - 1.1589 \times 10^{-5}x_1^3x_2 - 8.0897 \times 10^{-11}x_1^3x_3 +$$
$$1.3015 \times 10^{-9}x_1^3x_4 + 9.0654 \times 10^{-6}x_1^3x_5 - 4.2947 \times 10^{-8}x_1^3x_6 +$$
$$1.592 \times 10^{-8}x_1^3 + 1.841 \times 10^{-5}x_1^2x_2^2 + 2.5404 \times 10^{-10}x_1^2x_2x_3$$
$$+ \cdots +$$
$$-2.3256 \times 10^{-8}x_5^2x_6 + 0.99383x_5^2 + 2.36 \times 10^{-7}x_5x_6^3 +$$
$$4.4262 \times 10^{-8}x_5x_6^2 - 0.27104x_5x_6 + 0.00024964x_6^4 +$$
$$1.1805 \times 10^{-7}x_6^3 + 2.5742x_6^2$$

On the other hand, the Lyapunov function $x_1^2 + 2x_2^4 + 3x_3^2 + 3x_4^2 + x_5^4 + 2x_6^2$ found by the linear programming approach in this paper is quite easy to verify. It is possible to use SOSTOOLS to discover such functions, following some heuristics proposed by (Papachristodoulou and Prajna [2002]). However, in the general case, the problem of generating reliable Lyapunov candidates is of great interest to applications in formally verifying properties of control systems.

A partial solution consists of using a *rational vector recovery* procedure that searches for a nearby rational solution so that the conditions for Lyapunov functions are satisfied precisely. This is aided by the fact that SOS programming approach provides a positive definite polynomial as well as a numerical decomposition as a sum of squares (Harrison [2007], Platzer et al. [2009]). The difficulty of such an approach to Lyapunov function synthesis is two-fold. First of all, the search has to maintain the positive definiteness of $V(x)$ and the negative definiteness of its derivative $V'(x)$, simultaneously. This is further complicated when a *positivstellensatz* proof involving many SOS polynomial multipliers is used to find a function over a bounded region of the space. Secondly, the number of floating point coefficients can often be too large to permit an efficient search, as witnessed in Example 2.1.

Yet another approach involves using interval arithmetic techniques to bound the solution of SDPs and provide guaranteed certificates of optimality and infeasibility (Härter et al. [2012]). However, to our knowledge, the VSDP approach remains to be integrated and evaluated inside a SOS programming solver.

2.4 Interval Methods

On the other end of the spectrum, interval arithmetic techniques remain a simple approach to proving positive (semi-) definiteness over an interval domain U. Let $U = [\ell_1, u_1] \times [\ell_2, u_2] \times \cdots \times [\ell_n, u_n]$ be the product of intervals wherein $-\infty < \ell_i < u_i < \infty$ for each $i \in [1, n]$. We evaluate the function $V(\boldsymbol{x})$ over the interval U using standard interval arithmetic approaches (Moore et al. [2009]). If the lower bound of V over the interval U is positive (non-negative), we conclude that V must be positive (semi-) definite over U. Interval methods can be extended to unbounded domains, as well. Finally, approaches such as Bernstein polynomial expansions provide interesting ways of finding bounds for polynomials inside an interval (see Stahl [1995] for a survey of these methods).

However, interval methods have severe limitations due to the problem of lost dependencies between the various terms in $V(\boldsymbol{x})$. To illustrate this, consider $x \in [-1, 1]$. We wish to prove that $x^2 - 2x + 1 \geq 0$. Interval evaluation of the LHS polynomial yields the conservative bound $[-1, 4]$ for $x^2 - 2x + 1$ that fails to prove the required assertion.

Formally, the interval evaluation approach for encoding the positivity of the polynomial form $V(\boldsymbol{x}, \boldsymbol{c}) = \sum \boldsymbol{c_\alpha} \boldsymbol{x^\alpha}$ over the interval I involves the following steps:

(1) Let $M = \{\boldsymbol{x^{\alpha_1}}, \ldots, \boldsymbol{x^{\alpha_k}}\}$ be the monomials in V.
(2) For each monomial in $\boldsymbol{x^\alpha} \in M$, we compute lower and upper bounds ℓ_α and u_α, respectively over the interval I. We obtain polynomials $f_\alpha : \boldsymbol{x^\alpha} - \ell_\alpha$ and $g_\alpha :\ u_\alpha - \boldsymbol{x^\alpha}$ from these bounds.
(3) To ensure non-negativity of V over I, we write $V(\boldsymbol{x}, \boldsymbol{c}) = \sum_\alpha \lambda_\alpha f_\alpha + \sum_\alpha \gamma_\alpha g_\alpha$ for unknown non-negative multipliers $\lambda_\alpha, \gamma_\alpha \geq 0$. Equating the coefficients of various monomials on both sides, we obtain a system of linear inequality constraints $\psi[\boldsymbol{c}, \boldsymbol{\lambda}]$.
(4) A series of calls to a linear programming solver with a suitable objective function can systematically discover if a non-trivial solution $\boldsymbol{c} \neq 0$ to ψ exists. The polynomial V is obtained by substituting the value of $\boldsymbol{c}$.

However, since interval evaluation is often conservative, it is frequently applied with a branch-and-bound approach that repeatedly subdivides the domain of interest and evaluates the function on each subdomain. Techniques such as *interval constraint propagation* allow us to subdivide efficiently without necessarily examining a large number of subdivisions (see Benhamou and Granvilliers [2006] for a survey). Finally, tools such as iSAT implement interval constraint propagation techniques to solve non linear constraints over intervals (Fränzle et al. [2007]). The work of (Ratschan and She [2010]) uses these ideas to find Lyapunov functions for polynomial systems.

Their approach derives linear programs to encode the positive definiteness of V and the negative definiteness of V'. The conservative nature of interval evaluation loses dependencies between the various monomials in $V(\boldsymbol{x}, \boldsymbol{c})$, requiring an expensive branch-and-relax procedure.

2.5 Linear Combination of Basis Polynomials

Consider polynomials $p_1, \ldots, p_k$ over $\boldsymbol{x} \in \mathbb{R}^n$. We wish to encode the entailment

$$(p_1(\boldsymbol{x}) \geq 0 \ \wedge \ \cdots \ \wedge \ p_k(\boldsymbol{x}) \geq 0) \ \models \ p(\boldsymbol{x}) \geq 0\,.$$

A linear programming relaxation using the "S-procedure" (or the Lagrangian relaxation) simply checks if multipliers $\lambda_1, \ldots, \lambda_k \geq 0$ can be found so that $p = \sum_{j=1}^{k} \lambda_j p_j$. Comparing both sides term by term yields a linear programming relaxation. Such a relaxation is complete (or lossless) for few special cases: (a) the polynomials p_j and p are affine (Farkas Lemma), or (b) $k \leq 1$ and p_1, p are positive definite quadratic forms.

In this paper, we wish to encode the non-negativity of the polynomial form $V(\boldsymbol{c}, \boldsymbol{x})$ over an interval $\boldsymbol{x} \in I$. The interval itself can be represented through constraints $\bigwedge_{j=1}^{n}(x - \ell_j \geq 0) \ \wedge \ (u_j - x \geq 0)$ wherein ℓ_j, u_j are the upper and lower bounds respectively for x_j. However, it is easy to see that unless $V(\boldsymbol{c}, \boldsymbol{x})$ is affine in $\boldsymbol{x}$, using the S-procedure directly is bound to fail. Therefore, we construct a *basis set* $B(I)$ of polynomials that are known to be positive (or non-negative) over I. Constructing the basis set can be performed in many ways. Interval evaluation approach of Ratschan and She (*ibid.*) can be viewed as an application of S-Procedure by constructing the basis set by (a) selecting the set of all monomials of the form $\boldsymbol{x}^{\boldsymbol{\alpha}}$ with $|\boldsymbol{\alpha}|_1 \leq d$, (b) evaluating bounds $[\ell_{\boldsymbol{\alpha}}, u_{\boldsymbol{\alpha}}]$ on each such monomial in the set, and (c) using the polynomials $\boldsymbol{x}^{\boldsymbol{\alpha}} - \ell_{\boldsymbol{\alpha}}$ and $u_{\boldsymbol{\alpha}} - \boldsymbol{x}^{\boldsymbol{\alpha}}$.

In this work, we consider other ways in which the S-Procedure can be applied to construct a set of basis polynomials.

3. HANDELMAN REPRESENTATIONS

Handelman representations arise from a theorem by Handelman (Datta [2002], Handelman [1988], Parrilo and Sturmfels [2001]) that characterizes positive polynomials over convex polyhedra. In this paper, we use these representations to encode polynomial non-negativity over intervals by means of a linear program (LP). Let K be a polyhedron defined as a conjunction of linear inequality constraints $K : \bigwedge_{j=1}^{m} f_j : \ (\boldsymbol{a_j}\boldsymbol{x} - b_j) \geq 0$ and p be a polynomial over $\boldsymbol{x}$. We observe that if p can be written as a positive linear combination of products of the constraints defining K, i.e, $p = \sum_{i=1}^{N} \lambda_i f_1^{n_{i,1}} f_2^{n_{i,2}} \cdots f_m^{n_{i,m}}$, then it is non-negative over K. Handelman's result shows that the converse is also true for strictly positive polynomials:

Theorem 3.1. (Handelman). If p is strictly positive over K and K is compact, then p can be written as a positive linear combination of the inequalities defining K:

$$p = \sum_{k=1}^{n} \lambda_k \Pi_{j=1}^{m} f_j^{n_{k,j}} \text{ for } \lambda_k > 0 \text{ and } n_{k,j} \in \mathbb{N}\,.$$

Handelman representations directly yield an LP relaxation for checking if a polynomial p is positive semi-definite over a polyhedron K defined by inequalities $\bigwedge_{j=1}^{m} f_j \geq 0$, where f_j is an affine expression $\boldsymbol{a_j}\boldsymbol{x} + b_j$.

(1) Fix a degree limit $D > 0$.
(2) Generate all possible products of degree upto D of the form $p_{\boldsymbol{\alpha}} : f_1^{\alpha_1} \cdots f_m^{\alpha_m}$ where $\alpha_i \in \mathbb{N}$ and $\sum_{j=1}^{m} \alpha_j \leq D$.
(3) Compute an LP relaxation ψ using the basis set $B = \{p_{\boldsymbol{\alpha}} \mid |\boldsymbol{\alpha}|_1 \leq D\}$.

If the LP ψ is infeasible then we may (a) subdivide K into polyhedra $K_1, \ldots, K_p$ such that $K = \bigcup_{j=1}^{p} K_j$, and prove that p is non-negative over each of the subdivisions; and/or (b) increase the degree bound D for the representations.

Handelman representations allow us to encode the positive semi-definiteness of a template form $V(\boldsymbol{x}, \boldsymbol{c})$ as a linear program that involves $\boldsymbol{c}$ and a set of multiplier variables $\boldsymbol{\lambda}$. We solve this LP and simply ignore the values of $\boldsymbol{\lambda}$.

3.1 Comparison with Intervals

We now compare the Handelman representation with the interval evaluation approach. A simple comparison reveals that both approaches have drawbacks that can be addressed, in part, by a combined joint approach.

Claim 1. There exists an interval I and polynomials f_1, f_2 that are non-negative over I such that f_1's non-negativity can be proved using a Handelman representation but not by an interval evaluation. Likewise, f_2's non-negativity can be proved using interval evaluation but not through a Handelman representation.

Consider the interval $x \in [-1, 1]$. The polynomial $f_1 : \ x^2 - 3x + 2$ is positive semi-definite. However, interval evaluation cannot show this since the computed bound $f_1 \in [-1, 6]$ is too conservative. On the other hand, we obtain a Handelman representation $f_1 = (1 - x)^2 + (1 - x)$.

On the flip side, interval evaluation can prove that $f_2 : \ x^2$ is non-negative over $[-1, 1]$. However, f_2 does not have a Handelman representation over $[-1, 1]$. If such a representation were to exist, then

$$f_2 = \sum \lambda_j (1 - x)^{m_j} (x + 1)^{m_j} \ \text{ for } \lambda_j > 0\,.$$

At $x = 0$, we have $f_2(0) = 0$ but the RHS remains positive. Therefore, no Handelman representation exists. On the other hand, Theorem 3.1 assures us that $x^2 + \epsilon$ must have a Handelman representation for arbitrarily small $\epsilon > 0$. The table below shows the smallest ϵ for which a Handelman representation can be found for $x^2 + \epsilon$ as a function of the degree bound D.

D	2	3	4	5	6	7
ϵ	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{7}$

It follows from Theorem 3.1 that as $D \to \infty$, we have $\epsilon \to 0$. However, $\epsilon > 0$ for all D.

The example suggests that the two approaches can complement each other. Let $V(\boldsymbol{x}, \boldsymbol{c})$ be a given polynomial form whose positivity is to be encoded over a given interval $I : [\ell_1, u_1] \times \cdots [\ell_n, u_n]$. Let D be a given degree limit.

(1) We build a *basis set* $B = B_{intvl} \cup B_{handel}$ of polynomials.
(2) The set B_{intvl} consists of all *monomial bounds* of the form $\boldsymbol{x}^{\boldsymbol{\alpha}} - \ell_\alpha$ and $u_\alpha - \boldsymbol{x}^{\boldsymbol{\alpha}}$ for each monomial $\boldsymbol{x}^\alpha$ in V and its interval bounds $[\ell_\alpha, u_\alpha]$.

(3) The set B_{handel} consists of all power products $\Pi f_j^{\alpha_j}$, where f_j denotes the expressions $x_j - \ell_j$ and $u_j - x_j$ derived from the interval I
(4) We encode the equality $V = \sum_{q \in B} \lambda_q q$ for non-negative multipliers λ_q. Equating the coefficients yields a system of linear inequalities $\psi[\boldsymbol{c}, \boldsymbol{\lambda}]$.

3.2 Lyapunov Function Synthesis

Given an ODE $\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x})$ with equilibrium $\boldsymbol{x}^* = 0$, we wish to find a Lyapunov function $V(\boldsymbol{x})$ over a given interval I containing 0. This proves that the interval I is a subset of the region of attraction of I.

Our approach chooses a polynomial template $V(\boldsymbol{x}, \boldsymbol{c})$ for the target Lyapunov function and computes its derivative $V'(\boldsymbol{x}, \boldsymbol{c})$. We encode the positive definiteness of V and the negative semi-definiteness of V' using the LP relaxation procedure described above. This yields constraints ψ involving $\boldsymbol{c}$ and unknowns $\boldsymbol{\lambda}$. If a solution $\boldsymbol{c} \neq 0$ is obtained, we can construct a Lyapunov function. Our approach for encoding positive definiteness uses a trick proposed by (Papachristodoulou and Prajna [2002]), wherein we write $V = U + \boldsymbol{x}^t \Lambda \boldsymbol{x}$ for a positive semi-definite function U and a suitable positive diagonal matrix Λ.

Often, if the interval I is large, then the degree D required for finding a suitable Lyapunov function may be large. In this situation, we simply divide I into finitely many sub-intervals $I_1, \ldots, I_N$. We perform the encoding for each sub-interval and conjoin the resulting constraints to yield a single linear program. We then search for a non-zero feasible solution to this LP by setting the objective functions suitably.

4. IMPLEMENTATION AND EVALUATION

In this section, we describe our prototype implementation and evaluate it using various examples drawn from the literature on synthesizing Lyapunov functions.

Implementation: Our implementation uses a front-end that reads in a description of a polynomial ODE, an interval containing the equilibrium, the maximum degree of the Lyapunov template and a bound on the number of subdivisions along each dimension. It then constructs a generic polynomial template of the given degree $V(\boldsymbol{x}, \boldsymbol{c})$. The positive definiteness of V and negative (semi-) definiteness of V' are encoded using Handelman representation combined with interval evaluation, with a specified degree limit D. Finally, the linear inequality constraints involving $\boldsymbol{c}$ and a set of multiplier variables $\boldsymbol{\lambda}$ are output to a constraint file that is then processed by the floating point LP solver GLPK running the Simplex algorithm. The front end itself uses exact rational arithmetic implemented in the package GMP. Our implementation can be configured to use any LP solver, preferably one that can provide exact answers. However, in our experience, the exact arithmetic implementation of the polynomial manipulation in the front end is more crucial to the overall reliability of the procedure. The benchmarks we have attempted for this paper yield Lyapunov functions that are quite easy to verify.

We now evaluate our approach over a suite of examples taken from the related work on Lyapunov function synthesis. For each of the examples below, our approach chooses a degree d for the polynomial $V(\boldsymbol{x}, \boldsymbol{c})$, a degree D for the desired Handelman representation and a bound b on the number of subdivisions along each dimension. Appropriate values of d, D and b are discovered by trial and error starting from initial guesses for d that are given by the degree of the dynamics. The guess is increased subsequently. The values of D and b are initialized to 1 and increased until our approach discovers a Lyapunov function. A more rigorous heuristic for exploring the space of parameters that control our approach remains to be explored as part of our future work. Table 1 presents the details on the degree bounds, size of the LP and running time for each benchmark instance. We note that all but two instances were entirely solved using interval evaluation and without requiring branch-and-relax or Handelman representations. However, in two of the instances considered, interval evaluation by itself could not find a Lyapunov function. The use of Handelman representation enabled us to find Lyapunov functions. Running times were under a second for all but one example.

Table 1. Performance of our implementation on the examples presented in this paper. Each benchmark is indexed by its example number, d: degree bound for Lyapunov, D: degree bound for Handelman rep., b: # of subdivisions along each dimension, $|\boldsymbol{c}|$: # of template parameters, $|\boldsymbol{\lambda}|$: Number of multiplier variables, # Cons: number of constraints and Time (seconds). All experiments were run on a Macbook air laptop running MAC OSX mountain lion with 8GB RAM.

Ex.	d	D	b	$\lvert\boldsymbol{c}\rvert$	$\lvert\boldsymbol{\lambda}\rvert$	# Cons	Time
4.1	2	0	1	28	5	15	< 0.1
4.2	4	0	1	59	14	42	< 0.1
4.3	2	5	2	1664	5	224	0.4
4.4	4	0	1	268	69	223	0.9
4.5	3	0	1	60	19	44	0.8
4.6	3	2	1	780	209	583	39
2.1	4	0	1	14	57	37	< 0.1

Example 4.1. Consider the planar system $\frac{dx}{dt} = -x^3 + y$, $\frac{dy}{dt} = -x - y$. We wish to prove stability over the interval $[-100, 100] \times [-100, 100]$. Starting from a generic template of degree 2 involving x, y, our approach finds the Lyapunov function $x^2 + y^2$. It is easy to check that this function actually proves global asymptotic stability.

SOSTOOLS using the `findlyap` function yields the function $1.2118x^2 + 1.6099 \times 10^{-5}xy + 1.212y^2$. As such, this is not a Lyapunov function. Even if the xy term is discarded, the slight discrepancy between the coefficients of x^2 and y^2 terms causes the derivative $-2.4236x^4 - 0.00040002xy - 2.424y^2$ to fail to be negative semi-definite. The time taken by SOS was roughly 0.4 seconds.

Example 4.2. Consider the system $\frac{dx}{dt} = -x^3 - y^2$, $\frac{dy}{dt} = xy - y^3$. Our approach finds the degree 4 Lyapunov function $x^4 + 2x^2y^2 + y^4$ over the region $[-100, 100] \times [-100, 100]$. This function turns out to be a global Lyapunov function. In contrast, SOSTOOLS yields the function

$$\begin{array}{l} 0.62788x^4 + 3.3661 \times 10^{-8}x^3y + 0.052373x^3 + 0.65378x^2y^2 + \\ 1.4856 \times 10^{-8}x^2y + 1.1368x^2 + 2.596 \times 10^{-8}xy^3 - 0.18536xy^2 \\ -3.1327 \times 10^{-12}xy + 0.60694y^4 + 4.5162 \times 10^{-7}y^3 + 1.1368y^2 \end{array} .$$

Once again, SOS required 0.4 seconds to compute this result.

Example 4.3. Consider the ODE $\frac{dx}{dt} = -x - 1.5x^2y^3$, $\frac{dy}{dt} = -y^3 + 0.5x^2y^2$. Our approach derives the function $0.2x^2 + y^2$ to prove asymptotic stability over the interval $[-1, 1] \times [-1, 1]$.

SOSTOOLS produces the function $2.4229x^2 + 4.4868y^2$ over the same interval with a running time of 8.8 seconds.

Example 4.4. Consider the system

$$\begin{aligned}\frac{dx_1}{dt} &= -x_1 + x_2^3 - 3x_3x_4\\ \frac{dx_2}{dt} &= -x_1 - x_2^3\\ \frac{dx_3}{dt} &= x_1x_4 - x_3\\ \frac{dx_4}{dt} &= x_1x_3 - x_4^3\end{aligned}$$

Our approach yields the Lyapunov function $2x_1^2 + x_2^4 + 6x_3^2 + 6x_4^2$ over the region $[-1, 1]^4$. However, the function suffices to demonstrate global asymptotic stability, as well.

Example 4.5. Consider the uncertain linear system

$$\frac{dx_1}{dt} = x_2 \text{ and } \frac{dx_2}{dt} = -(2+\mu)x_1 - x_2\,.$$

Our approach finds a Lyapunov function $(2+\mu)x_1^2 + x_2^2$ when run over the range $x_1, x_2 \in [-2, 2]^2$ and $\mu \in [-1.99, 5]$ by splitting μ into two subranges: $\mu \in [-1.99, 0]$ and $\mu \in [0, 5]$.

Example 4.6. Consider the uncertain system

$$\frac{dx}{dt} = -(1+\mu_1)x + (4+\mu_2)y \text{ and } \frac{dy}{dt} = -(1+\mu_3)x - \mu_4 y^3\,,$$

with constant uncertain parameters $\mu_1, \mu_2, \mu_3, \mu_4 \in [0, 100]^4$. Our approach discovers the Lyapunov function $(1+\mu_3)x^2 + (4+\mu_2)y^2$ for $(x, y) \in [-2, 2]^2$. Using the synthesized Lyapunov function, we conclude global asymptotic stability for $\mu_1 > -1, \mu_2 > -4, \mu_3 > -1, \mu_4 > 0$. Our implementation can be sped up considerably by treating $\mu_1, \ldots, \mu_4$ as parameters rather than as state variables with derivatives set to 0.

5. CONCLUSION

To conclude, we have examined linear programming relaxations for encoding polynomial positivity. Our approach combines interval evaluation with Handelman representations. We note that for many instances, our approach is successful in discovering Lyapunov functions with rational coefficients that are easily verified. Future work will focus entirely on improving the interval evaluation technique by using ideas from Bernstein polynomials. The use of Taylor model arithmetic to handle non polynomial dynamics is another important area of future investigations.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for their helpful comments. Sriram Sankaranarayanan gratefully acknowledges support from the US National Science Foundation (NSF) under NSF CAREER award # CNS 0953941.

REFERENCES

Benhamou, F. and Granvilliers, L. (2006). Continuous and interval constraints. In *Handbook of Constraint Programming*, 571–590. Elsevier.

Boyd, S. and Vandenberghe, S. (2004). *Convex Optimization*. Cambridge University Press.

Chvátal, V. (1983). *Linear Programming*. Freeman.

Collins, G. (1975). Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In H.Brakhage (ed.), *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, 134–183. Springer.

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

Datta, R. (2002). Computing handelman representations. In *Mathematical Theory of Networks and Systems*. Cf. `math.berkeley.edu/~datta/MTNSHandelman.ps`.

Dolzmann, A. and Sturm, T. (1997). REDLOG: Computer algebra meets computer logic. *ACM SIGSAM Bull.*, 31(2), 2–9.

Fränzle, M., Herde, C., Ratschan, S., Schubert, T., and Teige, T. (2007). Efficient solving of large non-linear arithmetic constraint systems with complex Boolean structure. *JSAT—Journal on Satisfiability, Boolean Modeling and Computation, Special Issue on SAT/CP Integration*, 1, 209–236.

Handelman, D. (1988). Representing polynomials by positive linear functions on compact convex polyhedra. *Pacific J. Math*, 132(1), 35–62.

Harrison, J. (2007). Verifying nonlinear real formulas via sums of squares. In K. Schneider and J. Brandt (eds.), *Proc. Intl. Conf. on Theorem Proving in Higher Order Logics*, volume 4732 of *Lecture Notes in Computer Science*, 102–118. Springer-Verlag.

Härter, V., Jansson, C., and Lange, M. (2012). VSDP: A matlab toolbox for verified semidefinte-quadratic-linear programming. Cf. `http://www.ti3.tuhh.de/jansson/vsdp/`.

Meiss, J.D. (2007). *Differential Dynamical Systems*. SIAM.

Moore, R., Kearfott, R.B., and Cloud, M. (2009). *Introduction to Interval Analysis*. SIAM.

Papachristodoulou, A. and Prajna, S. (2002). On the construction of lyapunov functions using the sum of squares decomposition. In *IEEE CDC*, 3482–3487. IEEE Press.

Parillo, P.A. (2003). Semidefinite programming relaxation for semialgebraic problems. *Mathematical Programming Ser. B*, 96(2), 293–320.

Parrilo, P. (2000). *Structured Semidefinite Programs and Semialgebraid Geometry Methods in Robustness and Optimization*. Ph.D. thesis, California Institute of Technology.

Parrilo, P.A. and Sturmfels, B. (2001). Minimizing Polynomial Functions. *ArXiv Mathematics e-prints*.

Platzer, A., Quesel, J.D., and Rümmer, P. (2009). Real world verification. In *Proceedings of Intl. Conf. on Automated Deduction*, 485–501. Springer.

Prajna, S., Papachristodoulou, A., Seiler, P., and Parrilo, P.A. (2004). *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*.

Ratschan, S. and She, Z. (2010). Providing a basin of attraction to a target region of polynomial systems by computation of lyapunov-like functions. *SIAM J. Control and Optimization*, 48(7), 4377–4394.

Shor, N. (1987). Class of global minimum bounds on polynomial functions. *Cybernetics*, 23(6), 731–734. Originally in Russian: Kibernetika (6), 1987, 9–11.

Stahl, V. (1995). *Interval Methods for Bounding the Range of Polynomials and Solving Systems of Nonlinear Equations*. Ph.D. thesis, Johannes Kepler Universität Linz, Austria.

Tarski, A. (1951). A decision method for elementary algebra and geometry. Technical report, Univ. of California Press, Berkeley.

Tibken, B. (2000). Estimation of the domain of attraction for polynomial systems via lmis. In *IEEE CDC*, volume 4, 3860–3864 vol.4. IEEE Press.

Topcu, U., Packard, A., Seiler, P., and Wheeler, T. (2007). Stability region analysis using simulations and sum-of-squares programming. In *Proc. ACC*, 6009–6014. IEEE Press.