Linear Relaxations of Polynomial Positivity for Polynomial Lyapunov Function Synthesis.

Mohamed Amin Ben Sassi†, Sriram Sankaranarayanan†, Xin Chen* and Erika Ábrahám*.

† Department of Computer Science, University of Colorado, Boulder, CO, USA.
* Department of Computer Science, RWTH Aachen University, Aachen, Germany.

In this paper, we examine linear programming (LP) based relaxations for synthesizing polynomial Lyapunov functions to prove the stability of polynomial ODEs. A common approach to Lyapunov function synthesis starts from a desired parametric polynomial form of the polynomial Lyapunov function. Subsequently, we encode the positive-definiteness of the function, and the negative-definiteness of its derivative over the domain of interest. Therefore, the key primitives for this encoding include: (a) proving that a given polynomial is positive definite over a domain of interest, and (b) encoding the positive definiteness of a given parametric polynomial, as a constraint on the unknown parameters. We first examine two classes of relaxations for proving polynomial positivity: relaxations by sum-of-squares (SOS) programs, against relaxations that produce linear programs. We compare both types of relaxations by examining the class of polynomials that can be shown to be positive in each case.

Next, we present a progression of increasingly more powerful LP relaxations based on expressing the given polynomial in its Bernstein form, as a linear combination of Bernstein polynomials. The well-known bounds on Bernstein polynomials over the unit box help us formulate increasingly precise LP relaxations that help us establish the positive definiteness of a polynomial over a bounded domain. Subsequently, we show how these LP relaxations can be used to search for Lyapunov functions for polynomial ODEs by formulating LP instances. We compare our approaches to synthesizing Lyapunov functions with approaches based on SOS programming relaxations. The approaches are evaluated on a suite of benchmark examples drawn from the literature and automatically synthesized benchmarks. Our evaluation clearly demonstrates the promise of LP relaxations, especially for finding polynomial local Lyapunov functions that prove that the system is asymptotically stable over a given bounded region containing the equilibrium. In particular, the LP approach is shown to be as fast as the SOS programming approach, but less prone to numerical problems.

Keywords: Positive Polynomials, Sum-Of-Squares, Bernstein Polynomials, Interval Arithmetic, Handelman Representations, Stability, Lyapunov Functions

1. Introduction

In this paper, we consider the problem of automatically synthesizing Lyapunov functions to prove the asymptotic (or Lyapunov) stability of a system of differential equations over a compact set \( K \). The problem of discovering stability proofs for closed loop systems in the form of Lyapunov functions, is an important step in the formal verification of closed loop control systems [44]. Furthermore, extensions of Lyapunov functions such as control Lyapunov functions can be used to design controllers, and input-to-state stability (ISS) Lyapunov functions are used to verify the stability of inter-connected systems in a piecewise fashion.

In this paper, we focus on the synthesis of polynomial Lyapunov functions for proving the stability of autonomous non-linear systems with polynomial dynamics using linear programming (LP) relaxations. At its core, this requires us to find a positive definite polynomial whose Lie derivatives are negative.
definite. Therefore, the problem of finding a Lyapunov function depends intimately on techniques for finding positive-definite polynomials over the domain $K$ of interest. Proving that a multivariate polynomial is positive definite over an interval is co-NP hard, and therefore considered to be a hard problem [12]. Many relaxations to this problem have been studied, wherein a relaxed procedure can either conclude that the polynomial is positive definite with certainty, or fail with no conclusions. We examine two main flavors of relaxation based on what are informally termed “Handelman representations” and “Schmüdgen/Putinar representations” [8]. The former approach leads to LP relaxation, whereas the latter to relaxations based on semi-definite programming (SDP) [31, 43]. We compare the two classes of approaches, and demonstrate examples that can be established by one class but not the other.

In this paper, we extend Handelman representations using the idea of Bernstein polynomials from approximation theory [2, 10, 28]. Bernstein polynomials are a special basis of polynomials that have many rich properties, especially over the interval $[0, 1]$. For instance, tight bounds on the values of these polynomials over the interval $[0, 1]$ are known. We show a series of three LP relaxations, each more precise than the previous, that exploit these bounds in the framework of a reformulation linearization approach [41, 42]. We show that the Bernstein polynomial approach is, in general, incomparable to the Putinar representation approach by presenting examples that can be solved by one class but not by the other class of techniques.

Next, we adapt the relaxations for finding Lyapunov functions. The key difference is that to find a Lyapunov function, we search for a parametric polynomial $V(x, c)$ for unknown coefficients $c$, which is positive definite and whose derivative is negative definite over the region of interest. A straightforward approach leads to a bilinear program, that can be dualized a as multi-parametric program. We apply the basic requirements for a Lyapunov function, to cast the multi-parametric program back into a LP, without any loss in precision.

Finally, we have implemented the approach and describe our results on a suite of examples. We also compare our work with a SOS programming relaxation using the SOSTOOLS package [33]. We find that the LP relaxations succeed in finding Lyapunov functions for all cases, while the Putinar representation fails in many benchmarks due to numerical (conditioning) issues while solving the SDP. Overall, the LP relaxations are shown to present a promising approach for synthesizing Lyapunov functions over a bounded region $K$.

1.0.1 Organization Section 2 presents some preliminary notions of Lyapunov functions, representations of positive polynomials including Handelman, Schmüdgen and Putinar representations. We then present the basic framework for synthesizing Lyapunov functions by formulating a parametric polynomial that represents the desired function. Section 3 presents the basic properties of Bernstein polynomials and the three LP relaxations for proving polynomial positivity. It then compares the relaxations with each other, and with the Putinar representation approach. Next, we describe the synthesis of Lyapunov functions in Section 4. Section 5 presents the numerical results. Appendix A describes our benchmarks in detail and presents the Lyapunov functions synthesized by our approach on these, to aid the reviewers with their evaluation of our results. In particular, the appendix will not form a part of the final draft of this paper.

1.1 Related Work

In this section, we restrict our discussion to those works that are closely related to the overall problem of finding Lyapunov functions for non-linear systems. Related approaches for proving polynomial positivity are discussed in depth, at the appropriate places in the paper.
Much research has focused on the topic of stability analysis for non-linear systems, which continues to be a challenging problem. The sum-of-squares (SOS) relaxation approach is quite popular, and has been explored by many authors including [21, 29, 45, 48]. Papachristadoulou and Prajna were among the first to use SOS relaxations for finding polynomial Lyapunov functions [29]. The core idea is to express the polynomial and its negative Lie derivative as sum-of-square polynomials for global stability analysis, or use a suitable representation such as Putinar representation for finding Lyapunov functions over a bounded region. Their approach is implemented in the SOSTOOLS package [33]. Extensions have addressed the problem of controller synthesis [21], finding region of stability [45]; and using a combination of numerical simulations with SOS programming to estimate the region of stability [48]. A related set of approaches directly relax the positivity of the Lyapunov form and the negativity of its derivative using Linear Matrix Inequalities (LMIs) [3, 4, 20, 47]. Algebraic methods based for example on Gröbner basis [11], or on constructive semi-algebraic systems techniques [39, 40] exists.

On the other hand, the use of LP relaxation has not received as much attention. Johansen presents [22], an approach based on linear and quadratic programming was presented. This approach needs a so called linear parametrization form to reduce the stability conditions to an infinite number of linear inequalities, which are reduced to a finite number by discretizing the state space. As a consequence, the number of linear inequalities characterizing the Lyapunov functions grows exponentially with both the dimension of the state space and the required accuracy. Another approach using linear programming was presented by Hafstein [14, 15]. This approach searches for a piecewise affine Lyapunov function, and requires a triangulation of the state space. Our approach can also benefit from a sub-division of the state-space to increase accuracy. However, we focus on deriving polynomial Lyapunov functions. The use of Bernstein polynomial properties to formulate relaxations is a unique contribution here. Ratschan and She use interval arithmetic relaxations with branch-and-bound to discover Lyapunov like functions to prove a notion of region stability of polynomial systems [35]. This is extended in our previous work to find LP relaxations using the notion of Handelman representations [36]. In practice, the interval arithmetic approach is known to be quite coarse for proving polynomial positivity, especially for intervals that contain 0. Therefore, Ratschan and She restrict themselves to region stability by excluding a small interval containing the equilibrium from their region of interest. Furthermore, the coarseness of interval relaxation is remedied by resorting to branch-and-bound over the domain. A detailed comparison between interval and Handelman approach is provided in our previous work [36], wherein we conclude that both approaches have complementary strengths. A combined approach is thus formulated. In this paper, we start from such a combined approach and generalize it further through Bernstein polynomials. We use non-trivial properties of Bernstein polynomials that cannot be proven through interval analysis or Handelman representations, to further improve the quality of these relaxations. Section 3.3 provides detailed comparisons between the various approaches presented in this paper with the approaches based on interval arithmetic, Handelman representations and SOS programming relaxations.

2. Preliminaries

In this section, we recall the definition of Lyapunov functions and discuss procedures for synthesizing them. Subsequently, we examine two techniques for proving the positivity of polynomials: so-called Handelman representation technique that produces linear programming (LP) relaxations and a Putinar representation technique that produces semi-definite programming (SDP) relaxations. We extend these to recall algorithmic schemes for synthesizing Lyapunov functions, wherein we treat constraints that arise from the positivity of polynomials parameterized by unknown coefficients.
DEFINITION 2.1 (Positive Semi-Definite Functions) A function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is positive semi-definite over a domain \( U \subseteq \mathbb{R}^n \) iff
\[
(\forall x \in U) \ f(x) \geq 0.
\]
Furthermore, \( f \) is positive definite iff \( f \) is positive semi-definite, and additionally, (a) \( f(x) > 0 \) for all \( x \in U \setminus \{0\} \), and (b) \( f(0) = 0 \).

2.1 Lyapunov Functions

We now recall the key concepts of stability and Lyapunov functions. Let \( \mathcal{X} \) be a continuous system over a state-space \( \mathcal{X} \subseteq \mathbb{R}^n \) specified by a system of ODEs
\[
\frac{dx}{dt} = f(x), \ x \in \mathcal{X}.
\]

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

DEFINITION 2.2 (Lyapunov and Asymptotic Stability) A system is Lyapunov stable over an open region \( U \) around the equilibrium \( x^* \), if for every neighborhood \( N \subseteq U \) of \( x^* \) there is a neighborhood \( M \subset N \) such that
\[
(\forall x(0) \in M) \ (\forall t \geq 0) \ x(t) \in N.
\]
A system is asymptotically stable if it is Lyapunov stable and all trajectories starting from \( U \) approach \( x^* \) 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 \( x^* = 0 \). The definitions below are based on the terminology used by Meiss ([26]).

DEFINITION 2.3 A continuous and differentiable function \( V(x) \) is a weak Lyapunov function over a region \( U \subseteq \mathcal{X} \) iff the following conditions hold:

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

Additionally, \( V \) is a strong Lyapunov function if \( -\frac{dV}{dt} \) 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.

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

2.2 Proving Polynomial Positivity

At the heart of Lyapunov function synthesis, we face the challenge of establishing that a given function \( V(x) \) is positive (negative) definite over \( U \). The problem of deciding whether a given polynomial \( V(x) \) is positive definite is NP-hard [12]. A precise solution requires a decision procedure over the theory of reals. ([5, 46]). To wit, we check the validity of the formula:
\[
(\forall x \in U) \ V(x) \geq 0
\]
using tools such as QEPCAD ([6]) and REDLOG ([9]). This process is exact, but intractable for all but the smallest of systems and low degree polynomials for \( V \). Therefore, we seek stricter versions of positive semi-definiteness that yield a more tractable system of constraints.
We examine relaxations to the problem of establishing that a given polynomial is positive semi-definite over a region \( K \subseteq \mathbb{R}^n \). In the literature, we can distinguish two kind of techniques for establishing that a given polynomial is positive semi-definite \cite{32}. Here, we call them Handelman representations and Schmüdgen/Putinar representations.

### 2.2.1 Handelman Representations

The first approach writes the given polynomial \( p \) as a conic combination of products of the constraints defining \( K \). This idea was first examined by Bernstein for proving the positivity of univariate polynomials over the unit interval \([0, 1]\) \cite{2}. Furthermore, Hausdorff \cite{19} extend it to the interval \([-1, 1]\).

**Theorem 2.1** (Bernstein and Hausdorff). A polynomial \( p(x) \) is strictly positive over \([-1, 1]\) iff

\[
p(x) = \sum_{i=0}^{d} \lambda_i (1-x)^i (1+x)^{d-i},
\]

for a sufficiently large degree \( d \) and non-negative constants \( \lambda_i \geq 0 \).

This approach is generalized to multivariate polynomials over \( x : (x_1, \ldots, x_n) \) and general semi-algebraic sets \( K \subseteq \mathbb{R}^n \) rather than the unit interval. Let \( K \) be defined as a semi-algebraic set:

\[
K = (p_1(x) \geq 0 \land \cdots \land p_m(x) \geq 0)
\]

for multivariate polynomials \( p_1, \ldots, p_m \). A power-product over the set of polynomials \( P : \{p_1, \ldots, p_m\} \) is a polynomial of the form \( f : p_1^{n_1} p_2^{n_2} \cdots p_m^{n_m} \). The degree of the power-product is given by \( (n_1, \ldots, n_m) \).

We say that \( (n_1, \ldots, n_m) \leq D \) if \( n_j \leq D \) for each \( j \in [1, m] \). Let \( PP(P, D) \) represent all power products from the set \( P \) bounded by degree \( D \).

**Theorem 2.2** (Conic Combination of Power Products). If a polynomial \( p \) can be written as a conic combination of power-products of \( P : \{p_1, \ldots, p_m\} \), i.e.,

\[
p(x) \equiv \sum_{f \in PP(P, D)} \lambda_f f, \text{ s.t. } (\forall f \in PP(P, D)) \lambda_f \geq 0,
\]

then the polynomial \( p \) is non-negative over \( K \):

\[
(\forall x \in \mathbb{R}^n) \; x \in K \Rightarrow p(x) \geq 0.
\]

The proof is quite simple. The conic combination of power-products in \( PP(P, D) \) as shown in Eq. (2.2), is said to be a Handelman representation for a polynomial \( p \) \cite{8}. However, the converse of Theorem 2.2 does not hold, in general. Therefore, polynomials that are positive semi-definite over \( K \) need not necessarily have a Handelman representation.

**Example 2.3** Consider the first orthant in \( \mathbb{R}^2 \) given by \( K_1 : (x_1 \geq 0 \land x_2 \geq 0) \) and the polynomial \( p : x_1^2 - 2x_1x_2 + x_2^2 \). It is easily seen that \( p \) cannot be written as a conic combination of power products over \( x_1, x_2 \), no matter what the degree limit \( D \) is chosen to be.

An important question is when the converse of Theorem 2.2 holds. One important case for a compact, polyhedron \( K \) defined as \( K : \bigwedge_{j=1}^{m} (a_j x - b_j) \geq 0 \) is given by Handelman \cite{?}. Let \( P \) denote the set \( \{f_1, \ldots, f_m\} \), and \( PP(P, D) \) denote the power products of degree up to \( D \), as before.
THEOREM 2.4 (Handelman) If \( p \) is strictly positive over a compact polyhedron \( K \) then there exists a degree bound \( D > 0 \) such that
\[
p \equiv \sum_{f \in \mathcal{P}(P,D)} \lambda_f f, \text{ for } \lambda_f \geq 0.
\]

EXAMPLE 2.5 Consider a polynomial \( p(x_1, x_2) = -2x_1^3 + 6x_1^2x_2 + 7x_1^2 - 6x_1x_2^2 + 2x_2^2 + 7x_2^2 - 9 \) over the set \( K : (x_1 - x_2 - 3 \geq 0 \land x_2 - x_1 - 1 \geq 0) \). We can establish the positivity of \( p \) over \( K \) through its Handelman representation:
\[
p \equiv 2f_1^2f_2 + 3f_1f_2
\]

The problem of checking if a polynomial \( p \) is positive semi-definite over a set \( K \) is therefore tackled as follows:

1. Choose a degree limit \( D \) and construct all terms in \( \mathcal{P}(P,D) \), where \( P = \{p_1, \ldots, p_m\} \) are the polynomials defining \( K \).
2. Express \( p \equiv \sum_{f \in \mathcal{P}(P,D)} \lambda_f f \) for unknown multipliers \( \lambda_f \geq 0 \).
3. Equate coefficients on both sides to obtain a set of linear inequality constraints involving \( \lambda_f \).
4. Use a Linear Programming (LP) solver to solve these constraints. If feasible, the result yields a proof that \( p \) is positive semi-definite over \( K \).

We note that the procedure fails if \( p \) is not positive-definite over \( K \), or \( p \) does not have a Handelman representation over \( K \). Nevertheless, it provides an useful LP relaxation for polynomial positivity.

2.2.2 Sum-Of-Squares Decomposition

Another important approach to proving positivity is through the well-known sum-of-squares (SOS) decomposition.

DEFINITION 2.4 A polynomial \( p(x) \) is a sum-of-squares (SOS) iff there exists polynomials \( p_1, \ldots, p_k \) over \( x \) such that \( p \) can be written as
\[
p \equiv p_1^2 + \ldots + p_k^2
\]

It is easy to show that any SOS polynomial is positive semi-definite over \( \mathbb{R}^n \). On the other hand, not every positive semi-definite polynomial is SOS (the so-called Motzkin polynomial provides a counter-example) \[30\].

2.2.3 Schmüdgen Representation

Whereas SOS polynomials are positive semidefinite over \( \mathbb{R}^n \), we often seek if \( p \) is positive semi-definite over a semi-algebraic set \( K : (p_1 \geq 0 \land \cdots \land p_m \geq 0) \).

We define the pre-order generated by a set \( P = \{p_1, \ldots, p_m\} \) of polynomials as the set
\[
R(P) = \{ p_1^{e_1}p_2^{e_2}\cdots p_m^{e_m} \mid (e_1, \ldots, e_m) \in \{0,1\}^m \}.
\]

It is easy to see that if for some given \( x \), \( p_i(x) \geq 0 \) for all \( i \in [1,m] \), then every element of \( r \in R(P) \), we have \( r(x) \geq 0 \). In fact, the following result follows easily:
THEOREM 2.6 If a polynomial \( p \) can be expressed as \( \text{SOS polynomial combination} \) of elements in \( R(P) \),
\[
p \equiv \sum_{r \in R(P)} q_r r
\]
for SOS polynomials \( q_r \), \( (2.4) \)
then \( p \) is positive semi-definite over \( K \).

Decomposing a polynomial \( p \) according to \( (2.4) \) will be called the Schm"udgen representation of \( p \). The terminology is inspired by the following result due to Schm"udgen \([38]\):

THEOREM 2.7 (Schm"udgen Positivstellensatz) If \( K \) is compact then every polynomial \( p(x) \) that is strictly positive over \( K \) has a Schm"udgen representation of the form given in \( (2.4) \).

While Schm"udgen representations are powerful, and in fact, subsume the Handelman representation approach, or even the Bernstein polynomial relaxations to be presented in Section 3, the computational cost of using them is prohibitive. Using the form \( (2.4) \) requires finding \( 2^m \) SOS polynomials. In our applications, \( K \) typically represents the unit rectangle \([-1, 1]^n\), which makes the size of a Schm"udgen representation exponential in the size of the variables.

2.2.4 Putinar Representation
The Putinar representation approach provides a less expensive alternative. Once again, let \( K = (p_1 \geq 0 \wedge \cdots \wedge p_m \geq 0) \) be a set of interest.

THEOREM 2.8 If a polynomial \( p \) can be expressed as
\[
p \equiv q_0 + q_1 p_1 + \cdots + q_m p_m
\]
for SOS polynomials \( q_0, \ldots, q_m \), then \( p \) is positive semi-definite over \( K \).

Decomposing a polynomial \( p \) according to Equation \( (2.5) \) is said to provide a Putinar representation for \( p \). The converse of Theorem 2.8 was proved by Putinar \([34]\).

THEOREM 2.9 (Putinar) Let \( K = (p_1 \geq 0 \wedge \cdots \wedge p_m \geq 0) \) be a compact set, and suppose there exists a polynomial \( p_0 \) of the form \( p_0 = r_0 + \sum_{i=1}^{m} r_i p_i \) where \( r_0, \ldots, r_m \) are all SOS, and the set \( \hat{K} = \{ x | p_0(x) \geq 0 \} \) is also compact.

It follows that every polynomial \( p(x) \) that is strictly positive on \( K \) has a Putinar representation:
\[
p \equiv q_0 + \sum_{j=1}^{m} q_j p_j
\]
for SOS polynomials \( q_0, \ldots, q_m \).

A Putinar representation of \( p \) for a set \( P = \{ p_1, \ldots, p_m \} \) involves expressing \( p \equiv q_0 + \sum_{j=1}^{m} q_j p_j \) for a SOS polynomial \( q_j \). Searching whether a polynomial \( p \) is positive semi-definite over \( K : \wedge_{p_j \in P} p_j \geq 0 \) involves searching for a Putinar representation.

\[
\text{find } q_0, \ldots, q_m \text{ s.t. } p \equiv q_0 + \sum_{j=1}^{m} q_j p_j, \text{ } q_0, \ldots, q_m \text{ are SOS}
\]

The key steps involve parameterizing \( q_0, \ldots, q_m \) in terms of polynomials of bounded degree \( D \) over a set of unknown coefficients \( c \), and then solving the resulting problem through a relaxation to semi-definite programming, originally proposed by Shor and further developed by Parillo \([30, 43]\). The resulting optimization problem is called a Sum-of-Squares programming problem (SOS).
Comparing Handelman vs. Putinar Representations

Comparing the presentations of Handelman vs. Putinar representations, the tradeoffs look quite obvious. Whereas Handelman representations produce linear programs, that can be solved using exact arithmetic, Putinar representations produce sum-of-squares programs that are solved numerically by relaxation to semi-definite programs. In fact, numerical issues sometimes arise, and have been noted in our previous work [36]. On the other hand, it also seems that Handelman representations may be weaker than Putinar representations. Consider the example below:

**Proposition 2.1** The polynomial \( p(x) : x^2 \) does not have a Handelman representation inside the interval \([-1,1]\).

*Proof.* This is easy to show. Suppose we were able to express \( x^2 \) as a (non-trivial) conic combination of power products of the form

\[
x^2 = \sum_{j=1}^{m} \lambda_j (1-x)^{m_j} (x+1)^{m_j}, \lambda_j > 0
\]

We note that at \( x = 0 \), the LHS is zero whereas the RHS is strictly positive. This implies that \( \lambda_j = 0 \) for \( j = 1, \ldots, m \). Thus, the RHS is identically zero, yielding a contradiction. \( \square \)

On the other hand, the polynomial \( x^2 \) is SOS, and thus trivially shown to be positive over \([-1,1]\) (if not over \( \mathbb{R} \)) by a Putinar representation.

However, consider the set \( K = \{ x \geq 0, y \geq 0 \} \) and the polynomial \( p(x,y) : xy \). It is easy to show that \( p(x,y) \geq 0 \) over \( K \) by means of an Handelman representation. On the other hand, \( p(x,y) \) does not to have a Putinar representation of the form ??.

**Proposition 2.2** The polynomial \( p(x,y) : xy \) does not have a representation of the form

\[
xy = q_0 + q_1 x + q_2 y, \text{ where } q_0, q_1, q_2 \text{ are SOS}.
\]

This is a corollary to Proposition 3.3 (page 16).

2.3 Synthesis of Lyapunov Function

We now summarize the standard approach to synthesizing Lyapunov functions using Handelman or Putinar representations. The Handelman approach reduces the synthesis to solving a set of linear programs, and was presented in our previous work [36]. The Putinar representation approach uses SOS programming, and was presented by Papachristodoulou et al. [29]. This approach is implemented in a package SOSTOOLS that provides a user-friendly interface for posing SOS programming problems and solving them by relaxing to a semi-definite program [33].

Let \( U \subseteq \mathbb{R}^n \) be a compact set and \( \mathcal{S} \) be a system defined by the ODE \( \frac{dx}{dt} = f(x) \). We assume that the origin is the equilibrium of \( \mathcal{S} \), i.e., \( f(0) = 0, 0 \in \text{interior}(U) \), and wish to prove local asymptotic (or Lyapunov) stability of \( \mathcal{S} \) for the region \( U \).

Therefore, we seek a Lyapunov function of the form \( V(x,c) \) is a polynomial over \( x \) whose coefficients are polynomials over \( c \). Let \( V' \) denote the Lie derivative of \( V \), i.e., \( V'(x,c) = (\nabla_x V) \cdot f \). We define the set \( C \) as follows:

\[
C = \{ c \mid V(x,c) \text{ is positive definite for } x \in U \}.
\]

Also, let \( \hat{C} \) represent the set:

\[
\hat{C} = \{ c \mid V'(x,c) \text{ is negative definite for } x \in U \}.
\]
We replace negative definiteness for negative semi-definiteness if Lyapunov stability, rather than asymptotic stability is of interest. The overall procedure for synthesizing Lyapunov functions proceeds as follows:

1. Fix a template form $V(x, c)$ with parameters $c$.
2. Compute constraints $\psi[c]$ that characterize the set $C$ in Equation (\ref{eq:psi}).
3. Compute constraints $\hat{\psi}[c]$ that characterizes the set $\hat{C}$ from Equation (\ref{eq:hatpsi}).
4. Compute a value $c \in C \cap \hat{C}$ by solving the constraints $\psi \land \hat{\psi}$. The resulting function $V_c(x)$ is a Lyapunov function.

The main problem, therefore, is to characterize a set $C$ for the unknown parameters $c$, so $V_c(x)$ is positive definite over $U$ for all $c \in C$. Thus, the process of searching for Lyapunov functions of a given form devolves into the problem of finding a system of constraints for the sets $C, \hat{C}$.

Handelman representations and Putinar representations provide us two approaches to encoding the positive definiteness of $V$ and negative definiteness of $V'$ to characterize the sets $C, \hat{C}$.

**Handelman Representations:** We now briefly summarize our previous work that uses Handelman representations for Lyapunov function synthesis \[36\].

Let us assume that the set $U$ is written as a semi-algebraic set:

$$ U : \bigwedge_{j=1}^{m} p_j(x) \geq 0 $$

Let $P = \{ p_1, \ldots, p_m \}$ represent these constraints. Given a degree limit $D$, we construct the set $PP(P,D)$ of all power-products of the form $\prod_{j=1}^{m} p_j^{n_j}$ wherein $0 \leq n_j \leq D$.

We encode positive semi-definiteness of a form $V(x, c)$ by writing it as

$$ V(x, c) \equiv \sum_{f \in PP(P,D)} \lambda_f f \text{ wherein } \lambda_f \geq 0. $$

Positive definiteness is encoded using a standard trick presented by Papachristodoulou et al. \[29\]. Briefly, the idea is to write $V = \hat{V} + \sum_{j=1}^{n} \epsilon_j x_j^2$ for $\hat{V}(x, c)$, an unknown positive semi-definite function and a fixed positive definite contribution given by setting $\epsilon, p$. This idea is used in all our examples wherein positive definiteness is to be encoded rather than positive semi-definiteness.

We eliminate $x$ by equating the coefficients of monomials on both sides and obtain a set of linear constraints $\psi[c, \lambda]$ involving $c$ and $\lambda$. The set $C$ is characterized as a polyhedron obtained by the projection

$$ C : \{ c \mid \exists \lambda \geq 0 : \psi(c, \lambda) \}. $$

In practice, we do not project $\lambda$, but instead retain $\psi$ as a set of constraints involving both $c, \lambda$. Similarly, we consider the Lie derivative $V'(c, x)$ and obtain constraints $\psi(c, \mu)$ for a different set of multipliers $\mu$.

The overall problem reduces to finding a value of $c$ that satisfies the constraints

$$ \psi(c, \lambda) \land \hat{\psi}(c, \mu), $$

for some $\lambda, \mu \geq 0$. This is achieved by solving a set of linear programs.
Putinar Representations: Papachristadoulou and Prajna present the Putinar representations approach to synthesizing Lyapunov functions. Once again, we consider a semi-algebraic set $U$, as before. We fix a form $V(c, x)$ for the Lyapunov and write

$V \equiv q_0 + \sum_{j=1}^{m} q_j p_j$.

for SOS polynomials $q_0, \ldots, q_m$. The approach fixes the degree of each $q_j$ and uses SOS programming to encode the positivity. The result is a system of constraints over the parameters $c$ for $V$ and the unknowns $\lambda$ that characterize the SOS multipliers $q_j$. The same approach encodes the negative semi-definiteness of $V'$ over $U$. The combined result is a semi-definite program that jointly solves for the positive definiteness of $V$ and the negative definiteness of $V'$. A solution is recovered by solving the feasibility problem for an SDP to yield the values for $c$ that yield a Lyapunov certificate for stability.

2.4 Simplified Encoding

Thus far, our approaches have encoded both the positive definiteness of $V(x, c)$ and the negative definiteness of $V'(x, c)$ to yield a combined linear or semi-definite program that can be used to synthesize the Lyapunov function. In this section, we propose a simplified approach that simply focuses on encoding the negative definiteness of $V'(x, c)$, extracting a solution $c$ and checking that the result $V_c(x)$ is in fact positive definite.

1. Choose a form $V(x, c)$.
2. Encode negative definiteness of $V'(x, c)$ (the Lie derivative) over $U$. In particular, we do not encode the positive definiteness of $V(x, c)$.
3. Compute a solution for $c$ and check that the solution is, in fact, positive definite over $U$.

The approach is motivated by the following result from Vannelli and Vidyasagar (Page 72, Lemma 3) [49].

**Theorem 2.10** If $\mathcal{S}$ is an asymptotically stable system on $U$, $V(x)$ is a continuous function over $U$ with $V(0) = 0$, and $V'$ is negative definite, then $V$ is positive definite in some neighborhood of $0$.

**Proof.** Assume, for the sake of contradiction, that every neighborhood $N$ of $0$ has a point $x_0 \neq 0$ such that $V(x_0) \leq 0$. We will now show that the system is unstable by proving that the trajectory starting at $x_0$ cannot converge asymptotically to $0$. Let $t \in [0, T)$ represent a time interval for which $x(t) \in N \setminus \{0\}$. If $x(t) \in N \setminus \{0\}$ forever, then we set $T = \infty$. Consider any finite, or infinite sequence of time instances $t_0 = 0 < t_1 < t_2 \ldots < T$. We observe that $0 \geq V(x(t_0)) > V(x(t_1)) > \cdots$, since

$$V(x(t_i)) = V(x(t_{i-1})) + \int_{t_{i-1}}^{t_i} V'(x(s))ds.$$

By the continuity of $V$, and the fact that $V(0) = 0$, we conclude that the trajectory $x(t)$ cannot converge asymptotically to $0$. In other words, the system is not asymptotically stable. This directly contradicts our original claim. □
3. Linear Programming relaxations based on Bernstein polynomials

In this section, we recall the use of Bernstein polynomials for establishing bounds on polynomials in intervals. Given a multi-variate polynomial \( p \), proving that \( p \) is positive semi-definite in \( K \) is equivalent to showing that the optimal value of the following optimization problem is non-negative:

\[
\begin{align*}
\text{minimize} & \quad p(x) \\
\text{s.t} & \quad x \in K.
\end{align*}
\]  

(3.1)

Whereas (3.1) is hard to solve, we will construct a linear programming (LP) relaxation, whose optimal value is guaranteed to be a lower bound on \( p^* \). If the bound is tight enough, then we can prove the positivity of polynomial \( p \) on \( K \).

In general, the Handelman representation approach presented in Section 2.2 can be used to construct a linear programming relaxation [36]. In this section, we will use Bernstein polynomials for the special unit box \( K = [0, 1]^n \). Bernstein polynomials extend the Handelman approach, and will be shown to be strictly more powerful when \( K \) is the unit box. In our application examples, \( K \) is often a hyper-rectangle but not necessarily the unit box. We use an affine transformation to transform \( p \) and \( K \) back to the unit box, so that the Bernstein polynomial approach can be used.

3.1 Overview of Bernstein polynomials

Bernstein polynomials were first proposed by Bernstein as a constructive proof of Weierstrass approximation theorem [Bernstein]. Bernstein polynomials are useful in many engineering design applications for approximating geometric shapes [10]. They form a basis for approximating polynomials over a compact interval, and have nice properties that will be exploited to relax the optimization (3.1) to a linear program. Here, we should mention that a relaxation using Bernstein polynomials was provided in the context of reachability analysis for polynomial dynamical systems [7] and improved in [37]. The novelty in this work is not only the adaptation of these relaxations in the context of polynomial Lyapunov function synthesis but also a new tighter relaxation will be introduced by exploiting the induction relation between Bernstein polynomials. More details on Bernstein polynomials are available elsewhere [28].

We first examine Bernstein polynomials for the univariate case and then extend them to multivariate polynomials.

**Definition 3.1 (Univariate Bernstein Polynomials)** Given an index \( i \in \{0, \ldots, m\} \), the \( i^{th} \) univariate Bernstein polynomial of degree \( m \) over \([0, 1]\) is given by the following expression:

\[
\beta_{i,m}(x) = \binom{m}{i} x^i (1-x)^{m-i}, \quad i \in \{0, \ldots, m\}.
\]  

(3.2)

In the Bernstein polynomial basis, a univariate polynomial \( p(x) : \sum_{j=0}^{m} p_j x^j \) of degree \( m \), can be written as:

\[
p(x) = \sum_{i=0}^{m} b_{i,m} \beta_{i,m}(x)
\]

where for all \( i = 0, \ldots, m \):

\[
b_{i,m} = \sum_{j=0}^{i} \binom{i}{j} \binom{m}{j} p_j.
\]  

(3.3)
The coefficients $b_{i,m}$ are called the Bernstein coefficients of the polynomial $p$.

 Bernstein polynomials have many interesting properties. We summarize the most relevant ones for our applications, below:

**Lemma 3.1** For all $x \in [0,1]$, and for all $m \in \mathbb{N}$, the Bernstein polynomials $\{\beta_{0,m}, \ldots, \beta_{m,m}\}$ have the following properties:

1. Unit partition: $\sum_{i=0}^{m} \beta_{i,m}(x) = 1$.
2. Bounds: $0 \leq \beta_{i,m}(x) \leq \beta_{i,m}(\frac{x}{m})$, for all $i = 0, \ldots, m$.
3. Induction property: $\beta_{i-1,m}(x) = \frac{m-i+1}{m} \beta_{i,m}(x) + \frac{i-1}{m} \beta_{i-1,m}(x)$, for all $i = 0, \ldots, m - 1$.

Using the unit partition and positivity of Bernstein polynomials, the following bounds result holds:

**Corollary 3.1** On the interval $[0,1]$, a polynomial $p$ with Bernstein coefficients $b_{0,m}, \ldots, b_{m,m}$, the following inequality holds:

$$\min_{i=0,\ldots,m} b_{i,m} \leq p(x) \leq \max_{i=0,\ldots,m} b_{i,m}.$$  \hspace{1cm} (3.4)

We generalize the previous notions to the case of multivariate polynomials i.e $p(x) = p(x_1,\ldots,x_n)$ where $x = (x_1,\ldots,x_n) \in U = [0,1]^n$. For multi-indices, $I = (i_1,\ldots,i_n) \in \mathbb{N}^n$, $J = (j_1,\ldots,j_n) \in \mathbb{N}^n$, we fix the following notation:

- $I \leq J \iff i_l \leq j_l$, for all $l = 1,\ldots,n$.
- $I = \left( \frac{I}{J}, \ldots, \frac{i_n}{j_n} \right)$ and $\left( \frac{I}{J} \right) = \left( \begin{array}{c} i_1 \\ j_1 \\ \vdots \\ i_n \\ j_n \end{array} \right)$.
- $I_k = (i_1,\ldots,i_{r-1},i_r+k,i_{r+1},\ldots,i_n)$ where $r \in \{1,\ldots,n\}$ and $k \in \mathbb{Z}$.

Let us fix our maximal degree $\delta = (\delta_1,\ldots,\delta_n) \in \mathbb{N}^n$ for a multi-variate polynomial $p$ ($\delta_l$ is the maximal degree of $x_l$ for all $l = 1,\ldots,n$). Then the multi-variate polynomial $p$ will have the following form:

$$p(x) = \sum_{I \leq \delta} p_I x^I$$ where $p_I \in \mathbb{R}$, $\forall I \leq \delta$.

Multivariate Bernstein polynomials are given by products of the univariate polynomials:

$$B_{I,\delta}(x) = \beta_{i_1,\delta_1}(x_1)\ldots\beta_{i_n,\delta_n}(x_n)$$ where $\beta_{i_l,\delta_l}(x_l) = \left( \frac{\delta_l}{i_l} \right) x_l^{i_l} (1-x_l)^{\delta_l-i_l}$. \hspace{1cm} (3.5)

Thanks to the previous notations, these polynomials can also be written as follows:

$$B_{I,\delta}(x) = \left( \begin{array}{c} \delta \\ I \end{array} \right) x^I (1-x)^{\delta - I}.$$ \hspace{1cm} (3.6)

Now, we can have the general expression of a multi-variate polynomial in the Bernstein basis:

$$p(x) = \sum_{I \leq \delta} b_{I,\delta} B_{I,\delta}(x),$$
where Bernstein coefficients \( b_{I,\delta} \) are given as follows:

\[
b_{I,\delta} = \sum_{J \subseteq I} \binom{i_1}{j_1} \cdots \binom{i_n}{j_n} b_J = \sum_{J \subseteq I} \binom{I}{J} \delta^J p_J.
\]  

Therefore, the generalization of Lemma 3.1 will lead to the following properties:

**Lemma 3.2** For all \( x = (x_1, \ldots, x_n) \in U \) we have the following properties:

1. Unit partition: \( \sum_{I \subseteq \delta} B_{I,\delta}(x) = 1 \).
2. Bounded polynomials: \( 0 \leq B_{I,\delta}(x) \leq B_{I,\delta}(\frac{1}{\delta}) \), for all \( I \subseteq \delta \).
3. Induction relation: \( B_{I,\delta-1} = \frac{\delta - i}{\delta} B_{I,\delta} + \frac{i + 1}{\delta} B_{I-1,\delta} \), for all \( r = 1, \ldots, n \), and all \( I \subseteq \delta, r \).

Finally, in the case of general rectangle \( K = [x_1, x_1] \times \cdots \times [x_n, x_n] \) it suffices to make a change of variables \( x_j = x_j + z_j(\bar{x}_j - x_j) \) for all \( j = 1, \ldots, n \) to obtain new variables \( z = (z_1, \ldots, z_n) \in U \).

### 3.2 Bernstein relaxations

We assume that \( K \) is a bounded rectangle. Without loss of generality, we can assume that \( K = [0,1]^n \) since we can be reduced to the unit box by a simple affine transformation. Using the previous properties we are going to construct three LP relaxations to problem (3.1).

**Reformulation Linearization Technique (RLT)** We first recall a simple approach to relaxing polynomial optimization problems to linear programs [41, 42]. We then carry out these relaxations for Bernstein polynomials, and show how the properties in Lemma 3.2 can be incorporated into the relaxation schemes. Recall, once again, the optimization problem (3.1).

\[
\begin{align*}
\text{minimize} \quad & p(x) \\
\text{s.t.} \quad & x \in K.
\end{align*}
\]

For simplicity, let us assume \( K = [0,1]^n \) be the unit rectangle. \( K \) is represented by the constraints \( \bigwedge_{j=1}^n (x_j \geq 0 \quad \land \quad (1 - x_j) \geq 0 \). The standard RLT approach consists of writing \( p(x) = \sum_I p_I x^I \) as a linear form \( p(x) : \sum_I p_I y_I \) for fresh variables \( y_I \) that are place holders for the monomials \( x^I \). Next, we write down as many facts about \( x^I \) over \( K \) as possible. The basic approach now considers all possible power products of the form \( \pi_{J,\delta} : x^J (1 - x)^{\delta - J} \) for all \( J \subseteq \delta \), where \( \delta \) is a given degree bound. Expanding \( \pi_{J,\delta} : \sum_{I \subseteq \delta} a_{I,J} x^I \), we write the linear inequality constraint,

\[
\sum_{I \subseteq J} a_{I,J} y_I \geq 0
\]

The overall LP relaxation is obtained as

\[
\begin{align*}
\text{minimize} \quad & \sum_I p_I y_I \\
\text{s.t.} \quad & \sum_{I \subseteq J} a_{I,J} y_I \geq 0, \quad \text{for each } J \subseteq \delta
\end{align*}
\]  

(3.8)
Additionally, it is possible to augment this LP by adding inequalities of the form \( \ell_I \leq y_I \leq u_I \) through the interval evaluation of \( x^I \) over the set \( K \).

**Proposition 3.1** For any polynomial \( p \), the optimal value computed by the LP \((\ast\ast)\) is a lower bound on that of the polynomial program \((\ast)\).

**RLT using Bernstein Polynomials** The success of the RLT approach depends heavily on writing “facts” involving the variables \( y_I \) that substitute for \( x^I \). We now present the core idea of using Bernstein polynomial expansions and the richer bounds that are known for these polynomials from Lemma 3.2 to improve upon the basic RLT approach.

First, we write \( p(x) \) as a sum of Bernstein polynomials of degree \( \delta \).

\[
p(x) = \sum_{I \leq \delta} b_I,\delta B_I,\delta(x),
\]

wherein \( b_I,\delta \) are calculated using the formula in equation \((\ast\ast)\). Let us introduce a fresh variable \( z_I,\delta \) as a place holder for \( B_I,\delta(x) \). Lemma 3.2 now gives us a set of linear inequalities that hold between these variables \( z_I,\delta \). Therefore, we obtain three LP relaxations of increasing precision using Bernstein polynomials. Once again, let \((b_I,\delta)_{I \leq \delta}\) denote Bernstein coefficients of \( p \) with respect to a maximal degree \( \delta \). We formulate three LP relaxations, each providing a better approximation for the feasible region of the original problem \((\ast)\).

The first relaxation uses the fact that \( B_I,\delta(x) \geq 0 \) for all \( x \in [0,1]^n \) and that \( \sum_{I \leq \delta} B_I,\delta = 1 \).

\[
p_{\delta}^{(1)} = \min_{I \leq \delta} b_I,\delta = \minimize \sum_{I \leq \delta} b_I,\delta z_I,\delta
\]

s.t \( z_I,\delta \in \mathbb{R}, \quad I \leq \delta, \quad z_I,\delta \geq 0, \quad I \leq \delta, \quad \sum_{I \leq \delta} z_I,\delta = 1 \), \hspace{1cm} (3.9)

From corollary [3.1] it is easy to see that \( p_{\delta}^{(1)} \) is equal to the minimal Bernstein coefficient for \( p \). The optimization is superfluous here, but will be useful subsequently.

Next, we incorporate sharper bounds for \( B_I,\delta(x) \) for each \( I \) for \( x \in K \).

\[
p_{\delta}^{(2)} = \minimize \sum_{I \leq \delta} b_I,\delta z_I,\delta
\]

s.t \( z_I,\delta \in \mathbb{R}, \quad I \leq \delta, \quad 0 \leq z_I,\delta \leq B_I,\delta(\delta), \quad I \leq \delta, \quad \sum_{I \leq \delta} z_I,\delta = 1 \), \hspace{1cm} (3.10)

Finally, the recurrence relation between the polynomials are expressed as equations in the LP relaxation to constrain the relaxation even further.
Linear Relaxation of Polynomial Positivity

\[ p^{(3)}_\delta = \text{minimize} \sum_{I \leq \delta} b_{I,\delta} z_{I,\delta} \]

\[ \text{s.t} \]

\[ z_{I,\delta} \in \mathbb{R}, \ I \leq \delta, \]

\[ z_{I,\delta'} \in \mathbb{R}, \ J \leq \delta', \ \delta' < \delta, \]

\[ 0 \leq z_{I,\delta} \leq B_{I,\delta}(\frac{\delta}{\delta}), \ I \leq \delta, \]

\[ 0 \leq z_{J,\delta'} \leq B_{J,\delta'}(\frac{\delta'}{\delta}), \ J \leq \delta', \ \delta' < \delta, \]

\[ \sum_{I \leq \delta} z_{I,\delta} = 1, \]

\[ \sum_{J \leq \delta'} z_{J,\delta'} = 1, \ \delta' < \delta, \]

\[ z_{J,\delta'} = \frac{\delta - \delta'}{\delta} z_{J,\delta'} + \frac{\delta + 1}{\delta} z_{J,\delta',1}, \ J \leq \delta', \ \delta' < \delta. \]

(3.11)

It is easy to show from the properties of Bernstein polynomials that each of these relaxations provides a lower bound on the original polynomial optimization problem.

**Proposition 3.2** \( p^{(1)}_\delta \leq p^{(2)}_\delta \leq p^{(3)}_\delta \leq p^* \) where \( p^* \) is the optimal value of (??).

**Proof.** By writing \( p \) in its Bernstein basis, the problem (??) will be the following:

\[ \text{minimize} \sum_{I \leq \delta} b_{I,\delta} B_{I,\delta}(y) \]

\[ \text{s.t} \]

\[ y \in [0,1]^n. \]

The relaxations (??) and (??) are obtained by replacing \( B_{I,\delta}(y) \) with the decision variables \( z_I \) and using the first and second property of Lemma 3.2. The third one (??) is obtained by adding new variables for Bernstein polynomials of lower degree and exploiting the induction property of Lemma 3.2. \( \square \)

**Remark 3.1** The choice of the appropriate relaxation is a tradeoff between complexity and precision. In fact, the third relaxation (??) which gives the more precise result, can be very expensive especially when the number of variable or/and their degrees grows up. Here we should mention that this relaxation can be used for a fixed level. More precisely, instead of exploiting all the constraints arising for the degrees \( \delta' < \delta \) we can only restrict ourselves for a first level of degrees which means that we consider only \( \delta' \) such that \( \delta' = \delta - 1 \) for some fixed directions \( r \).

3.3 **Comparison with other relaxations**

We are going to compare the introduced Bernstein relaxations with other existing relaxations. First we will show the benefit of using Bernstein relaxations instead of the approach given in [36] (combining Handelman representation and interval arithmetics techniques) where the both approaches are based on Linear programming then we briefly compare with the SOS approach based on SDP programming.

3.4 **Comparison with Handelman and Interval Representations**

Our earlier work [36], uses RLT with a combination of Handelman representation augmented by interval arithmetic constraints to prove polynomial positivity, as a primitive for Lyapunov function synthesis. As long as the domain \( K \) of interest is a hyper-rectangle, the relaxations provided by Bernstein polynomials will provide results that are guaranteed to be at least as good, if not strictly better. For simplicity, let us fix \( K \) as the unit box \([0,1]^n\) and compare the two relaxations.
A first remark will be that the Handelman representation contains polynomials with degree less or equal to the fixed degree \( \delta \), whereas our Bernstein polynomials are all of degree equal to \( \delta \). This does not affect the optimal value of the relaxation, as noted by Sherali and Tuncbilek [41]. Therefore, no gain of precision can be made using the Handelman relaxation thanks to the additional polynomials of degree less than \( \delta \).

The major advantage of using Bernstein polynomials is that, in addition to positivity, the three non-trivial properties of Lemma 3.2 are known. We illustrate two examples where these properties make a difference.

We first demonstrate that the second relaxation (\( \text{(??)} \)) is strictly more powerful than the relaxation in (\( \text{(??)} \)).

**Example 3.1** Consider the simple univariate polynomial below:

\[
p(x) = 4x^2 - 4x + 1 \geq 0 \quad \text{on} \quad [0, 1].
\]

(3.12)

For this example both Handelman and interval arithmetics fail to prove positivity and a decomposition is required. In fact, a simple transformation \( y \mapsto 2x - 1 \), maps \( p(x) \) to \( \tilde{p}(y) = y^2 \), and the interval \( \tilde{K} : y \in [-1, 1] \). The very same situation noted in proposition 2.1 results.

For this example, we find that the relaxation (\( \text{(??)} \)) with a degree 2 computes exact optimal value \( p_\delta^{(2)} = p^* = 0 \), proving positivity of \( p \) over \([-1, 1]\).

Now, we demonstrate that the third relaxation (\( \text{(??)} \)) is strictly more powerful through an example.

**Example 3.2** We will consider the following bivariate polynomial:

\[
p(x) = x^2 + y^2 \quad \text{on} \quad [-1, 1]^2.
\]

(3.13)

For a degree \( \delta = (2, 2) \), the optimal value of (\( \text{(??)} \)) is \( p_\delta^{(1)} = -2 \), which does not establish positivity of \( p \) on \([-1, 1]\). If we use the second linear program (\( \text{(??)} \)), the optimal value will be improved and we find \( p_\delta^{(2)} = -0.5 \), but still not sufficient to prove positivity of \( p \) on \([-1, 1]\). Now, when we use the third linear program (\( \text{(??)} \)), we obtain the exact optimal value \( p_\delta^{(3)} = 0 \) and ensure the positivity of \( p \) over \([-1, 1]^2\).

3.5 *Comparison with Putinar Representations/SOS Programming*

It is well known that not every positive (semi-) definite polynomial can be expressed through a SOS decomposition. Nevertheless, the SOS decomposition has been seen to be quite powerful in practice. However, problems arise when a polynomial \( p(x) \) is to be proven positive inside a set \( K \). In such situations, the Putinar representation described in Section 2.2.4 is applied even when the conditions for Theorem 2.9 do not apply. As noted earlier, the application of this approach to Lyapunov function synthesis has been studied by Papachristadoulou and Prajna [29] and a related approach using LMIs by Tibken [47].

In such a situation, we can show that Handelman representations can be useful in showing positivity where Putinar representations can fail. Consider the set \( K : [0, 1] \times [0, 1] \) and the bivariate polynomial \( p(x, y) = xy \).

**Proposition 3.3** There do not exist SOS polynomials \( q_0, q_1, q_2, q_3, q_4 \) such that

\[
xy \equiv q_0 + q_1x + q_2y + q_3(1 - x) + q_4(1 - y).
\]
In other words, the polynomial $xy$ does not have a Putinar representation over the unit box $K : [0,1] \times \{0,1\}$.

**Proof.** Suppose, for contradiction, there exist SOS polynomials $q_0, q_1, q_2, q_3, q_4$ such that

$$xy \equiv q_0 + q_1 x + q_2 y + q_3 (1-x) + q_4 (1-y).$$

(3.14)

We establish a contradiction by considering the lowest degree terms of the polynomials $q_0, \ldots, q_4$. We use the notation $\text{COEFF}(q,x^iy^j)$ to refer to the coefficient of the monomial $x^iy^j$ in the polynomial $q(x,y)$.

First, plugging in $x = 0, y = 0$, we observe that $\text{COEFF}(q_0,1) = \text{COEFF}(q_3,1) = \text{COEFF}(q_4,1) = 0$. In other words, the constant coefficients of $q_0, q_3, q_4$ are zero.

Since $q_0, q_3, q_4$ are psd, if they have zero constant terms then they do not have linear terms involving $x$ or $y$.

$$\text{COEFF}(q_j, x) = \text{COEFF}(q_j, y) = \text{COEFF}(q_j, 1) = 0, \ j \in \{0,3,4\}.$$ 

Therefore, the constant terms of $q_1, q_2$ are zero as well:

$$\text{COEFF}(q_1, 1) = \text{COEFF}(q_2, 1) = 0.$$ 

Otherwise, the RHS will have non-zero terms involving $x, y$ but the LHS has no such terms. Once again, from the positivity of $q_1, q_2$, we have

$$\text{COEFF}(q_j, x) = \text{COEFF}(q_j, y) = 0, \ j \in \{1,2\}$$

Having established that no constant or linear terms can exist for $q_0, \ldots, q_4$, we turn our attention to the quadratic terms $x^2, y^2, xy$. Consider the coefficients of $x^2$ on both sides of Eq. (3.14):

$$\text{COEFF}(q_0, x^2) + \text{COEFF}(q_3, x^2) + \text{COEFF}(q_4, x^2) = 0, \text{COEFF}(q_0, y^2) + \text{COEFF}(q_3, y^2) + \text{COEFF}(q_4, y^2) = 0.$$ 

Since $q_i$ are psd and lack constant/linear terms, we can show that

$$\text{COEFF}(q_j, x^2) \geq 0, \text{COEFF}(q_j, y^2) \geq 0, \ j \in \{0, \ldots, 4\}$$

Therefore, we conclude that

$$\text{COEFF}(q_j, x^2) = \text{COEFF}(q_j, y^2) = 0, \ j \in \{0,3,4\}.$$ 

Finally, we compare $xy$ terms on both sides of Eq. (3.14) to obtain:

$$\text{COEFF}(q_0, xy) + \text{COEFF}(q_3, xy) + \text{COEFF}(q_4, xy) = 1.$$ 

Therefore, we have $\text{COEFF}(q_j, xy) > 0$ for some $j \in \{0,3,4\}$, while $\text{COEFF}(q_j, x^2) = \text{COEFF}(q_j, y^2) = 0$. We now contradict the assumption that $q_j$ is psd. Based on what we have shown thus far, we can write

$$q_j(x,y) = cxy + \text{third or higher order terms, where } c > 0.$$ 

Let us fix $x = \varepsilon, y = -\varepsilon$ for some $\varepsilon > 0$.

$$q_j(\varepsilon, -\varepsilon) = -c_0 \varepsilon^2 + o(\varepsilon^3)$$
Therefore, we conclude for $\varepsilon$ small enough, $q_j(\varepsilon, -\varepsilon) < 0$, thus contradicting the positivity assumption for $q_j$ for some $j \in \{0, 3, 4\}$.

As a result, we conclude the existence of situations where linear relaxations following Handelman or Bernstein representations can provide proofs of positivity, where Putinar representations can fail.

Furthermore, the SOS relaxation to SDP 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. On the other hand, LP solvers can use exact arithmetic in spite of the high cost of doing so, to obtain results that hold up to verification. Examples of numerical issues in SOS programming for Lyapunov function synthesis are noted in our previous work [36], and will not be reproduced here. Consequently, much work has focused on the problem of finding rational feasible points for sum-of-squares to generate polynomial positivity proofs that in exact arithmetic [17, 27, 31]. Recently, a self-validated SDP solver VSDP has been proposed by Lange et al. [18]. However, its application to SOS optimization has not been investigated.

4. Synthesis of polynomial Lyapunov functions

Given an ODE in the form: $\frac{dx}{dt} = f(x)$ with equilibrium $x^* = 0$, we wish to find a Lyapunov function $V(x)$ over a given rectangular domain $R_x$ containing 0.

Note 4.1 (Positive Semi-definite vs. Positive Definite) As presented in Section 2.3 our approach fixes a polynomial template $V_c(x) = V(x, c)$ for the target Lyapunov function, and computes its Lie derivative form $V'(x, c)$. It then searches for coefficients $c$ such that $V(x, c)$ is positive definite over $R_x$ and $V'$ is negative definite. We recall the standard approach to encoding positive definiteness, following Papachristodoulou & Prajna [29], by writing $V = U + x^t \Lambda x$ for a positive semi-definite function $U$ and a diagonal matrix $\Lambda$ with small but fixed positive diagonal entries. Therefore, we will focus on encoding positive or negative semi-definiteness and use this approach to extend to positive/negative definiteness.

Note 4.2 (Simplified Encoding) Following the approach described in Section 2.4, we will focus on encoding the negative definiteness of $V'(x, c)$ over the given domain $R_x$, temporarily ignoring the positivity of $V$. Once a suitable $c$ is found, we simply check of $V_c(x)$ is positive definite over $R_x$. Failing this, we simply choose a point $y \in R_x$ where $V$ fails to be positive and simply repeat our procedure by adding an additional constraint that $V(y, c) > 0$.

We will now demonstrate how the three LP relaxations described in ?? extend to search for Lyapunov functions, wherein

(a) The polynomial of interest is $V(x, c)$ parameterized by unknowns $c$,

(b) The interval of interest is a general box $\prod_{j=1}^n [\ell_j, u_j]$ rather than $[0, 1]^n$,

(c) We wish to encode the positive semi-definiteness of $-V'(x, c)$ rather than $V$ itself (following the technique in ??).

4.0.1 Encoding Positivity of Parametric Polynomial We first consider the problem of extending the LP relaxation to find values of parameters $c$, such that, a parametric polynomial $V(x, c)$ is positive semi-definite over the interval $[0, 1]^n$. 
Recall, that given a known polynomial \( p(x) \), our first step was to write down \( p(x) \) using its Bernstein expansion as \( p(x) : \sum_{l \leq \delta} b_l B_l, \delta \). Thus, the overall form of \( \text{?????} \) can be written as

\[
\min \sum_{l \leq \delta} b_l c_l \text{ s.t. } A z \leq b.
\]

However, the Bernstein coefficients for \( V(x, c) \) are not known in advance. Let \( m \) denote a vector of monomials \( x^I \) for \( I \leq \delta \). The polynomial \( V(x, c) \) can be written as \( c^t m \). Furthermore, each polynomial \( x^I \) itself has a Bernstein expansion:

\[
x^I : \sum_{J \leq \delta} b_{IJ} B_{IJ}, \delta.
\]

Consider a matrix \( B \), wherein, each row corresponds to a monomial \( x^I \), and each column to a Bernstein polynomial \( B_{IJ} \). The coefficient corresponding to row \( I \) and column \( J \) is \( b_{IJ} \), the Bernstein coefficient for \( x^I \) corresponding to \( B_{IJ} \). Therefore, we use \( B \) to convert polynomials from monomial to the Bernstein basis.

\[
V(x, c) : c^t m = c^t B z, \text{ wherein } z \text{ represents the Bernstein polynomials.}
\]

Therefore, the LP relaxations \( \text{?????} \) have the following form:

\[
\min c^t B z \text{ s.t. } A z \leq b \tag{4.1}
\]

Equation (4.1) is, in fact, a bilinear program which can be reformulated using its dual to a multiparametric linear optimization problem[23, 24]. However, the direct resolution of a multi parametric program is very expensive since it requires to find exponentially many critical regions, and for each region we have to find our optimal value which will be an affine function depending on the parameter vector \( c \). Therefore, rather than solve the optimization problem (4.1), we simply seek values of \( c \) such that

\[
\text{find } c \text{ s.t. } (\forall z) A z \leq b \Rightarrow c^t B z \geq 0. \tag{4.2}
\]

We now use Farkas lemma, a well known result in linear programming, to dualize \( \text{?????} \).

**Lemma 4.1** \( c \) is a solution to the problem in \( \text{?????} \) if and only if there exist multipliers \( \lambda \geq 0 \) such that

\[
A' \lambda = -B' c, \text{ b' } \lambda \leq 0, \text{ and } \lambda \geq 0
\]

As a result, we now have a procedure to reduce the search for a Lyapunov function as the feasibility problem for a set of linear constraints. It now remains to address: (a) the transformation from a given domain \( R_x \) to the domain \( [0, 1]^n \) for applying the Bernstein polynomial based LP relaxations, and (b) encode negative definiteness of the derivative \( V'(x, c) \).

### 4.0.2 Transforming Co-ordinates

Let \( R_x : \prod_{j=1}^n [\ell_j, u_j] \) be the domain of interest. We consider the change-of-basis transformation from \( x \in R_x \) to a new set of variables \( y \in [0, 1]^n \)

\[
x_j \mapsto \ell_j + y_j(u_j - \ell_j)
\]
Let \( \mathbf{m} \) denote the original monomial basis over \( \mathbf{x} \) consisting of monomials \( \mathbf{x}^I \) for \( I \leq \delta \). Corresponding to this, we define \( \hat{\mathbf{m}} \) as the monomial basis over \( \mathbf{y} \) consisting of monomials \( \mathbf{y}^J \) for \( I \leq \delta \). It is easy to see that any monomial \( \mathbf{x}^I \) can be written as a polynomial involving monomials \( \mathbf{y}^J \) of degree \( J \) at most \( I \). Therefore, \[
\mathbf{m} \equiv T \hat{\mathbf{m}} \quad \text{wherein} \]
each row of \( T \) corresponds to a monomial \( \mathbf{x}^I \) and each column to a monomial \( \mathbf{y}^J \). Each row therefore lists the coefficients of the monomial \( \mathbf{x}^I \) as a function over \( \mathbf{y} \).

Therefore, \( \mathbf{c}' \mathbf{m} \equiv \mathbf{c}' T \hat{\mathbf{m}} \). Rather than encoding the positivity of the original polynomial \( V(\mathbf{x}, \mathbf{c}) : \mathbf{c}' \mathbf{m} \) over \( R_\varepsilon \), we encode that of \( (T' \mathbf{c})' \hat{\mathbf{m}} \) over \([0, 1]^n\).

### 4.0.3 Lie derivatives

Finally, Lyapunov function synthesis requires us to encode the negative definiteness of \( V'(\mathbf{x}, \mathbf{c}) \) rather than \( V \). Once again, this requires us to consider the coefficients of the form \( V'(\mathbf{x}, \mathbf{c}) : \mathbf{c}' \mathbf{m} \) as a linear transformation applied over \( \mathbf{c} \).

Since the RHS of the ODE is polynomial, we consider the Lie derivative of each monomial \( \mathbf{x}^I \) as a polynomial \( p_I \). Let \( \mathcal{D} \) represent the matrix wherein each row of \( \mathcal{D} \) represents the monomial \( \mathbf{x}^I \) and the contents of the row are the coefficients of the Lie derivative of \( \mathbf{x}^I \).

Therefore, applying Lie derivative to \( V(\mathbf{x}, \mathbf{c}) : \mathbf{c}' \mathbf{m} \), we obtain
\[
V'(\mathbf{x}, \mathbf{c}) : \mathbf{c}' \mathcal{D} \mathbf{m}'.
\]
Here the vector \( \mathbf{m}' \) represents the set of monomials involved in the Lie derivative.

### 4.0.4 Overall Encoding

To summarize, we are asked to find a value of \( \mathbf{c} \) such that the Lie derivative of the polynomial \( V(\mathbf{x}, \mathbf{c}) \) is non-negative over \( R_\varepsilon \). Let \( \mathcal{D} \) represent the matrix form of the Lie derivatives on the monomial basis \( \mathbf{m} \). \( T \) represent the transformation of the monomials from \( R_\varepsilon \) to \([0, 1]^n\), and finally \( \mathcal{B} \) represent the transformation to Bernstein form. The overall optimization involves finding \( \mathbf{c} \) such that
\[
\text{find } \mathbf{c} \text{ s.t. } (\forall \mathbf{z}) \mathbf{A} \mathbf{z} \leq \mathbf{b} \Rightarrow (\mathcal{B} \times T' \times \mathcal{D}' \mathbf{c})' \mathbf{z} \leq 0 \tag{4.3}
\]
As a result, applying Farkas lemma transforms this into solving the feasibility problem below:
\[
\text{find } \mathbf{c} \text{ s.t. } (\exists \lambda) \mathbf{A}' \lambda = \mathcal{B} T' \mathcal{D}' \mathbf{c}, \quad \mathbf{b}' \lambda \leq 0, \text{ and } \lambda \geq 0. \tag{4.4}
\]
LP feasibility

We note that \( \mathbf{c} = 0 \) is seemingly a trivial solution to the feasibility problem in \( \tag{4.3} \). But, this does not yield a Lyapunov function. To address this, we recall that our goal is to encode the negative definiteness and not the negative semi-definiteness of the derivative. On the other hand, \( \tag{4.4} \) encodes the negative semi-definiteness.

As mentioned earlier, we ensure that \( U : V'(\mathbf{x}, \mathbf{c}) - \mathbf{x}' \mathbf{A} \mathbf{x} \) is negative semidefinite using \( \tag{4.3} \) rather than \( V' \) itself. The matrix \( \Lambda \) is a diagonal matrix whose diagonal entries are all set to a small value \( \varepsilon > 0 \), chosen by the user. We choose \( \varepsilon = 0.1 \) for most of our experiments.

**Remark 4.1** The problem posed in \( \tag{4.3} \) can be simplified considerably for the LP relaxation \( \tag{4.4} \). In the absence of further bounds about the Bernstein polynomials, the smallest Bernstein coefficient is a lower bound on the minimum value of a polynomial. Therefore, the constraints in \( \tag{4.3} \) can be simplified as
\[
\text{find } \mathbf{c} \text{ s.t. } \mathcal{B} \cdot \mathbf{c} \geq 0 \tag{4.5}
\]
Effectively the form above imposes that all the Bernstein coefficients of $V(x, c)$ are non-negative. This implicitly eliminates the multipliers $\lambda$ from the LP relaxation.

**Remark 4.2** Infeasibility of \(??\) means that our search failed to find a Lyapunov function. This can indicate many problems, including (a) the system is not stable, (b) the system is stable but no polynomial Lyapunov function exists [1], (c) the system is stable with a polynomial Lyapunov but it is not provable using the relaxation that we have chosen to arrive at our LP.

### 4.1 Higher relaxation degree

Our linear relaxations are based on a fixed degree for the Bernstein polynomial expansion. By default, this degree called $\delta$ is fixed to some chosen value at the beginning of the algorithm. However, if the technique fails to find a Lyapunov function, we may improve precision by increasing the degree $\delta$. The following convergence result motivates the possible improvement in the lower bounds of the LP relaxation by increasing the degree bound $\delta$:

**Theorem 4.1** Let $p$ be a multivariate polynomial of degree $\delta = (\delta_1, \ldots, \delta_n)$ and let $b_{I, \delta} = b_I$ be its Bernstein coefficients with respect to the unit box $[0, 1]^n$:

$$\left| b_{I, \delta} - p \left( \frac{I}{\delta} \right) \right| = O \left( \frac{1}{\delta_1} + \cdots + \frac{1}{\delta_n} \right) \text{ for all } I \leq \delta. \tag{4.6}$$

As a consequence, when the optimal value of our linear or bilinear program is negative, we can just increase the degree of the relaxation allowing the relaxation to be more precise and then increasing the possibility to find our Lyapunov function.

### 4.2 Branch and bound decomposition

A second, more widely used approach to improving the relaxation, is to perform a branch and bound decomposition. The essential idea consists on verifying the so called vertex condition [13] for the given hyper-rectangle which guarantees that the LP relaxation coincides with the optimal value. Informally, this condition requires that no local minima for a polynomial $p$ exist in the interior of the rectangle. If it doesn’t hold we will simply divide our rectangle and keep doing it until reaching the global minimum and getting exact bounds in each sub box. In our case, we have two main differences:

1. We do not have a fixed polynomial, but a parametric polynomial $V(x, c)$.
2. The global minimum for a Lyapunov function $V$ is known in advance as the equilibrium $0$. Likewise, the negation of its derivative also has $0$ for a global minimum.

For these reasons, our branch-and-bound approach focuses on decomposing the given region $R_x$, so that the equilibrium $0$ lies in the boundaries of our cells rather than the relative interior, in an attempt to satisfy the vertex condition. So if a Lyapunov function is not found, we simply choose a variable $x_j$ and consider two cells $R^{(1)}_x : R_x \cap \{ x_j \leq 0 \}$ and $R^{(2)}_x : R_x \cap \{ x_j \geq 0 \}$. The cells may be recursively subdivided if necessary. In the limit, this approach creates $2^n$ cells, and can be expensive for systems with more than 10s of variables. The computational complexity can be mitigated by examining a few cells in the decomposition and trying to find a Lyapunov candidate based on the examined cells. We can then check if the Lyapunov candidates are indeed Lyapunov functions by considering the other cells. This approach can, in the worst case, examine every cell in the decomposition. However, if a good empirical
strategy for selecting the cells can be found, the approach can save much effort involved in encoding the LP relaxations for an exponential number of cells.

5. Numerical results

In this section, we present an evaluation of various linear programming relaxations using Bernstein polynomials \( \mathbb{B} \), extended using the technique for encoding the positivity of a parametric form, presented in Section 4.

5.1 Implementation

Our approaches are implemented as a MATLAB\textsuperscript{tm} toolbox for synthesizing Lyapunov function. Apart from a description of the system to be analyzed, the inputs include the maximum degree \( \delta : (\delta_1, \ldots, \delta_n) \) for the Bernstein expansion in each variable, the region of interest (fixed to \([-1, 1]^n\) for all of our evaluation), and the number of subdivisions along each dimension. Furthermore, our toolbox implements three LP relaxations, each adding more constraints over the previous. The first relaxation is based on \( \mathbb{B} \) simply uses the non-negativity and the unit summation properties of Bernstein polynomials. The second LP relaxation is based on \( \mathbb{B} \), adds upper bounds to the Bernstein polynomials and finally, the third approach \( \mathbb{B} \) adds the recurrence relations between the Bernstein polynomials. Each approach is used in the Lyapunov search by encoding the dual form \( \mathbb{B} \).

5.2 Numerical Examples

We first compare and contrast the three LP relaxations here over some benchmark examples from our previous work [36]. In doing so, we compare the results we obtain for each benchmark with those obtained by using the \texttt{findlyap} function in SOSTOOLS [33] and the Lyapunov functions obtained in our previous work. For each example, we wish to prove asymptotic stability over \( \mathbb{R}^n = [-1, 1]^n \). We will report for each program the Lyapunov function, the number of boxes in our decomposition, and two computational times.

\texttt{Setup} is the needed time to compute the data for the linear program. This includes:

1. The time needed to compute the matrix \( \mathbb{B} \) (for all three relaxations),
2. Computing bounds on the Bernstein polynomials (for second and third relaxations), and
3. Time needed to compute recurrences for each Bernstein polynomial (for the third relaxation)

In fact, much of the computation of \( \mathbb{B} \) and the bounds on it are independent of the actual problem instance. They can be performed once, and cached for a given number \( n \) of variables and given degree bounds \( \delta \), instead of recomputing them separately for each problem.

\texttt{LpTime} is the computational time associated with solving the linear programming relaxation using the \texttt{linprog} function provided by MATLAB\textsuperscript{tm}. Also, we should mention that since all the LPs are feasibility problems, the objective function is set to be the maximization of the sum of the coefficients.

\textbf{Example 5.1} Consider the system over \((x, y)\):

\[
\frac{dx}{dt} = -x^3 + y, \quad \frac{dy}{dt} = -x - y.
\] (5.1)
Table 1. Table showing Lyapunov functions computed by each of the three LP relaxations on the three systems considered in Example 5.1. The column Relaxation indicates which of the three LP relaxations was used, the Lyapunov function for each approach, the number of Boxes in the subdivision and the computational times split into computing the matrices and linear programming data, and the actual time needed to solve the LP. All timings are in seconds.

| System | Relaxation | Lyapunov | # Boxes | SETUP | LPTIME |
|--------|------------|----------|---------|--------|--------|
| (??)   | LP1        | 4.5807x^2 + 4.5807xy + 2.2906y^2 | 2       | 0.06   | 0.36   |
|        | LP2        | 5x^2 + 4.9995xy + 2.5002y^2   | 2       | 0.09   | 0.34   |
|        | LP3        | 5x^2 + 5xy + 2.5y^2           | 2       | 0.15   | 0.37   |
| (??)   | LP1        | 4.3039x^2 + 4.3039y^2         | 4       | 0.13   | 0.38   |
|        | LP2        | 4.9998x^2 + 5y^2             | 4       | 0.18   | 0.41   |
|        | LP3        | 5x^2 + 5y^2                  | 4       | 0.37   | 0.77   |
| (??)   | LP1        | 4.6809x^2 + 4.9547y^2        | 4       | 0.16   | 0.36   |
|        | LP2        | 4.9998x^2 + 5y^2             | 4       | 0.18   | 0.40   |
|        | LP3        | 5x^2 + 5y^2                  | 4       | 0.33   | 0.43   |

The Handelman relaxation technique in our previous work [36] finds the Lyapunov function \( x^2 + y^2 \) taking less than 0.1 seconds, whereas SOS discovers \( 1.2118x^2 + 1.6099 \times 10^{-3}xy + 1.212y^2 \), requiring 0.4 seconds. The three relaxations each using a subdivision of \([-1, 1]^2\) discover the function \( x^2 + xy + \frac{1}{2}y^2 \) (with a multiplicative factor, and modulo small perturbations due to floating point error). Interestingly, the system is globally asymptotically stable, and the Lyapunov function discovered by our approach is valid globally.

Next, we consider the system:

\[
\frac{dx}{dt} = -x^3 - y^2, \quad \frac{dy}{dt} = xy - y^3.
\]

(5.2)

The Handelman relaxation approach [36] finds a 4 degree Lyapunov function \( x^4 + 2x^2y^2 + y^4 \), requiring less than 0.1 seconds, whereas the SOS approach produces \( 0.62788x^4 + 0.052373x^3 + 0.65378x^2y^2 + 1.1368x^2 - 0.18536xy^2 + 0.60694y^4 + 1.1368y^2 \) after deleting terms with coefficients less than \( 10^{-7} \). The SOS approach requires roughly 0.4 seconds for this example. Our approach discovers degree two Lyapunov function \( x^2 + y^2 \) that is also globally stable.

Finally, we consider the system:

\[
\frac{dx}{dt} = -x - 1.5x^2y^3, \quad \frac{dy}{dt} = -y^3 + 0.5x^2y^2.
\]

(5.3)

The approach in [36] proves asymptotic stability over \([-1, 1]^2\) through the function \( 0.2x^2 + y^2 \), requiring 0.4 seconds, whereas the SOS approach finds \( 2.4229x^2 + 4.4868y^2 \) requiring a running time of 8.8 seconds.

The specific Lyapunov functions found for systems (????), the running times and number of subdivisions needed are summarized in Table 1.

5.2.1 Synthetic Benchmarks We now consider a second class of synthetic benchmarks that were generated using a special problem generator, constructed for generating challenging examples of locally
Table 2. Performance of our approach on the synthesized benchmarks. The column $n$: number of variables, $d_{\text{max}}$: maximum degree of the vector field, SUCC? indicates whether the approach succeeded in finding a Lyapunov function, 3: succeeded with Lyapunov, NP: numerical problem, MO: out-of-memory, $d_L$: degree of Lyapunov function, $d_Q$: degree of SOS multipliers, SETUP: setup time, $T_{\text{SDP}}$: SDP Solver time, Rel. Typ.: Relaxation Type, #Box: number of boxes in decomposition, $T_{\text{LP}}$: LP solver time. All times are reported in seconds.

| ID | $n$ | $d_{\text{max}}$ | Putinar (SOS) | Bernstein (our approach) |
|----|-----|------------------|---------------|--------------------------|
|    |     |                  | SUCC? $d_L$ $d_Q$ SETUP $T_{\text{SDP}}$ | Rel. Typ. SUCC? # Box SETUP $T_{\text{LP}}$ |
| 1  | 2   | 3                | ✓ 2 2 0.35 0.9 | LP1 ✓ 4 0.17 0.43 |
|    |     |                  |               | LP2 ✓ 4 0.20 0.42 |
|    |     |                  |               | LP3 ✓ 2 0.17 0.38 |
| 2  | 2   | 3                | ✓ 2 2 0.3 0.67 | LP1 ✓ 4 0.17 0.42 |
|    |     |                  |               | LP2 ✓ 4 0.19 0.38 |
|    |     |                  |               | LP3 ✓ 2 0.17 0.35 |
| 3  | 2   | 3                | ✓ 2 2 0.33 0.61 | LP1 ✓ 4 0.17 0.37 |
|    |     |                  |               | LP2 ✓ 4 0.18 0.35 |
|    |     |                  |               | LP3 ✓ 2 0.16 0.35 |
| 4  | 2   | 3                | ✓ 2 2 0.3 0.97 | LP1 ✓ 4 0.17 0.37 |
|    |     |                  |               | LP2 ✓ 4 0.21 0.39 |
|    |     |                  |               | LP3 ✓ 2 0.17 0.35 |
| 5  | 3   | 3                | ✓ 2 2 0.86 1.12 | LP1 ✓ 8 0.81 0.47 |
|    |     |                  |               | LP2 ✓ 8 0.97 0.61 |
|    |     |                  |               | LP3 ✓ 4 1.24 0.70 |
| 6  | 3   | 5                | ✓(NP) 2 2 0.81 2.3 | LP1 ✓ 8 7.15 6.4 |
|    |     |                  |               | LP2 ✓ 8 7.83 17.17 |
|    |     |                  | ✓ 2 4 39.5 4.2 | LP3 ✓(NP) 8 17.4 102.3 |
| 7  | 3   | 5                | ✓(NP) 2 2 0.8 2.2 | LP1 ✓ 8 6.50 5.2 |
|    |     |                  |               | LP2 ✓ 8 7.42 5.7 |
|    |     |                  | ✓(NP) 2 4 40.5 4.6 | LP3 ✓ 8 13.2 26.8 |

stable polynomial vector fields of varying degrees and number of variables to evaluate the various techniques presented here. Our overall idea is to fix two homogeneous polynomials $V_1(x)$ and $V_2(x)$ that are positive definite over a region of interest, chosen to be $K: [-1,1]^n$ for our examples.

Subsequently, for each choice of $V_1, V_2$, we attempt to find a system $\frac{dx}{dt} = F(x)$ such that the Lie derivative of $V_1$ is $-V_2$, and with an equilibrium at 0.

$$(\nabla V_1) \cdot F = -V_2, \text{ and } F(0) = 0.$$  \hspace{1cm} (5.4)

Naturally, any such system using the vector field $F$ is guaranteed to be asymptotically stable due to the existence of $V_1, V_2$. To synthesize a benchmark that is guaranteed to have asymptotic stability, we need to find a suitable $F$ within a given degree bound. To this end, we parameterize our system $F$ by a set of parametric polynomials and attempt to find parameters that satisfy $(5.4)$. It is easy to show that our approach leads to a set of linear equations on the parameters defining the entries in $F$ and solving
these equations yields a suitable system \( F \). The difficulty here lies in choosing appropriate \( V_1, V_2 \) so that the system \( F \) can be found. In our experience, if \( V_1, V_2 \) are chosen arbitrarily, the likelihood of finding a function \( F \) that satisfies \( \mathbb{P} \) seems quite small. Furthermore, since \( F \) involves \( n \) polynomials, the technique yields prohibitively large equations for \( n \geq 6 \). Our approach to synthesize benchmarks is based on carefully controlling the choice of \( V_1, V_2 \) and repeated trial-and-error, until feasible system of equations were discovered, to synthesize a benchmark. Having synthesized our benchmark, we destroy the functions \( V_1, V_2 \) used to generate it and simply present the system \( F \) to our implementation, as well as for SOS program.

The key to finding benchmarks lies in the generation of the polynomials \( V_1, V_2 \). We generated \( V_1 \) as one of two simple forms: (a) \( V_1 : \mathbf{x}'A_1\mathbf{x} \), or (b) \( V_1 : \mathbf{m}'A_2\mathbf{m} \), wherein \( \mathbf{m} \) is a vector of squares of the system variables of the form \([x_1^2, x_2^2, \ldots, x_n^2] \), and \( A_1, A_2 \) are diagonal matrices with non-negative diagonal entries chosen at random.

The polynomial \( V_2 \) is chosen to be a positive definite polynomial over \([-1, 1]^n\). The key idea here is to generate \( V_2 \) that is guaranteed to be positive definite over \([-1, 1]^n\) by writing

\[
V_2(\mathbf{x}) : \mathbf{x}'A\mathbf{x} + \sum_j q_j \prod_{i=1}^n (1 + x_i)^{p_{ji}}(1 - x_i)^{q_{ji}},
\]

essentially as a Schmüdgen representation involving the polynomials \((1 - x_i), (1 + x_i)\) for \( i \in [1, n] \) and sum-of-squares polynomials \( q_j \) obtained by squaring and adding randomly generated polynomials together.

**Remark 5.1** Even though our approach synthesizes an ODE \( \frac{d\mathbf{x}}{dt} = F(\mathbf{x}) \) that by design has a Lyapunov function \( V(\mathbf{x}) \), we note that the resulting system may (and often does) admit many other Lyapunov functions with a possibility of a larger domain of attraction towards the equilibrium 0.

In many cases, the process of trial and error is required to find pairs \( V_1, V_2 \) that yield a feasible vector field. Using this process, 15 different benchmarks were synthesized with 5 each of degrees 2, 3, and 4, respectively. Appendix A reports the ODEs for these benchmarks and the Lyapunov functions synthesized by our technique.

### 5.2.2 Results

Tables 2 and 3 compare the performance of the three LP relaxations implemented in our prototype with an implementation Putinar (SOS), based on Putinar representation of the Lyapunov function and the negation of its derivative, built using SOSTOOLS. Here we should mention that, in order to reduce the complexity of the \('LP3'\) relaxation, we reduce ourselves to a first level of lower degrees (see Remark 5.1). For each of the 15 benchmarks, we run both tools under different setups. The Putinar (SOS) approach is run with varying degrees of the Lyapunov function \( d_L \) and degrees of the SOS multipliers \( d_Q \). We attempted three sets \((d_L, d_Q) = (2, 2), (2, 4), (4, 4)\) in succession, stopping as soon as a Lyapunov function is found without a failure. To experiment with our approach and enable a full comparison, we attempt all the three relaxations for all the benchmarks.

We note that the LP relaxation approach is generally successful in discovering Lyapunov functions. One of the three relaxations is able to find a Lyapunov function in all the benchmarks. In 7 out of 15 cases, all three LP relaxations succeed. On the other hand, the Putinar (SOS) approach succeeds in 9 out of the 15 attempts, with *numerical problems* (NP) being the most common failure mode. These may arise due to many reasons, but commonly due to the Hessian matrix becoming ill-conditioned during the calculation of a Newton step. For benchmarks 12-15, the polynomials involved become so large, that the Putinar (SOS) approach runs out of memory during the problem setup. Our approach also suffers
from the same set of problems, but to a noticeably lesser extent. For instance, 9 out of the 45 linear
programs failed due to numerical problems. On the other hand, 10 out of the 28 SDPs terminate with a
numerical problem.

On most of the smaller benchmarks, all approaches have comparable timings. In general, the third
relaxation (LP3) is the most expensive, often more expensive than the other two LP relaxations or the
SOS (Putinar) approach. Likewise, when the degree of the SOS multipliers $d_0$ is increased from 2 to
4, we witness a corresponding 40x factor increase in the time taken to setup the SDP, with a smaller
increase in the time taken to solve the SDP. For the larger examples, the LP relaxation requires more
time, but is generally successful in finding an answer.

6. Conclusion
To conclude, we have examined three different LP relaxations for synthesizing polynomial Lyapunov
functions for polynomial systems. We compare these approaches to the standard approaches using
Schmüdgen and Putinar representations that are used in SOS programming relaxations of the problem.
In theory, the Schmüdgen representation approach subsumes the three LP relaxations. In practice, how-
ever, we are forced to use the Putinar representation. We show that the Putinar representation can prove
some polynomials positive semi-definite that our approaches fail to. On the other hand, the reverse is
also true: we demonstrate a polynomial that is easily shown to be positive semi-definite on the interval
$[-1,1]^n$ through LP relaxations. However, the same fact cannot be demonstrated by a Putinar represen-
tation approach. We then compare both approaches over a set of numerical benchmarks. We find that
the LP relaxations succeed in finding Lyapunov functions for all cases, while the Putinar representa-
tion fails in many benchmarks due to numerical (conditioning) issues while solving the SDP. As future
work, we wish to extend our approach to a larger class of Lyapunov functions. We also are looking into
the problem of analyzing systems with non-polynomial dynamics and the synthesis of non-polynomial
Lyapunov functions.

References

[1] Amir Ali Ahmadi, Miroslav Krstic, and Pablo A. Parrilo. A globally asymptotically stable poly-
nomial vector field with no polynomial Lyapunov function. In CDC-ECE, pages 7579–7580, 2011.

[2] Sergei Natanovich Bernstein. Démonstration du théorème de weierstrass fondée sur le
calcul des probabilités. Communications de la Société Mathématique de Kharkov 2, (1):1–2.

[3] Sergei Natanovich Bernstein. On the representation of positif polynomials. Soobshch Kharkov
marem ob-va, 2(14):227–228, 1915.

[4] G. Chesi. Estimating the domain of attraction via union of continuous families of lyapunov esti-
mates. Systems and Control letters, 56(4):326–333, 2005.

[5] G. Chesi, A. Garulli, A. Tesi, and A. Vicino. Lmi-based computation of optimal quadratic lyapunov
functions for odd polynomial systems. International Journal of Robust and Nonlinear Control, 15
(1):35–49, 2005.

[6] G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposi-
tion. In H.Brakhage, editor, Automata Theory and Formal Languages, volume 33 of Lecture Notes in
Computer Science, pages 134–183. Springer, 1975.
[6] George E. Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, Sep 1991.

[7] T. Dang and D. Salinas. Image computation for polynomial dynamical systems using the Bernstein expansion. In *CAV’09*, volume 5643 of *LNCS*, pages 219–232. Springer, 2009.

[8] Ruchira Datta. Computing handelman representations. In *Mathematical Theory of Networks and Systems*, 2002. Cf. [math.berkeley.edu/~datta/MTNSHandelman.ps](http://math.berkeley.edu/~datta/MTNSHandelman.ps).

[9] Andreas Dolzmann and Thomas Sturm. REDLOG: Computer algebra meets computer logic. *ACM SIGSAM Bull.*, 31(2):2–9, June 1997.

[10] Rida T. Farouki. The bernstein polynomial basis: A centennial retrospective. *Comput. Aided Geom. Des.*, 29(6):379–419, August 2012.

[11] K. Forsman. Construction of lyapunov functions using grbner bases. In *In Proc. of the 30th Conf. on Decision and Control*, pages 798–799. IEEE, 1991.

[12] Michael R. Garey and David S. Johnson. *Computers and Intractability: A guide to the theory of NP-Completeness*. W.H.Freeman, 1979.

[13] J. Garloff. The bernstein algorithm. *Reliable Computing*, 2:154–168, 1993.

[14] S. Hafstein. *Stability Analysis of Nonlinear Systems with Linear Programming*. PhD thesis, Gerhard-Mercator-University Duisburg, 2002.

[15] S. Hafstein. Revised cpa method to compute lyapunov functions for nonlinear systems. *Journal of Mathematical Analysis and Applications*, 4(20):610–640, 2014.

[16] D. Handelmann. Representing polynomials by positive linear function on comapct convex poyhe-dra. *Pacific J.Math. Ann*, 132:35–62, 1988.

[17] John Harrison. Verifying nonlinear real formulas via sums of squares. In Klaus Schneider and Jens Brandt, editors, *Proc. Intl. Conf. on Theorem Proving in Higher Order Logics*, volume 4732 of *Lecture Notes in Computer Science*, pages 102–118. Springer-Verlag, 2007.

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

[19] F. Hausdorff. Summationsmethoden und momentfolgen i. *Math. Zeit*, 9:74–109, 1921.

[20] D. Henrion and J.B. Lasserre. Convergent relaxations of polynomial matrix inequalities and static output feedback. *IEEE Transactions on Automatic Control*, 51(42):192–202, 2006.

[21] Z. W. Jarvis-Wloszek. *Lyapunov Based Analysis and Controller Synthesis for Polynomial Systems using Sum-of-Squares Optimization*. PhD thesis, UC Berkeley, 2003.

[22] A. Johansen. Computation of lyapunov functions for smooth nonlinear systems using convex optimization. *Automatica*, 36(11):1617–1626, 2000.

[23] C.N. Jones, M. Baric, and M. Morari. Multiparametric Linear Programming with Applications to Control. *European Journal of Control*, 13(2-3):152–170, March 2007. URL [http://control.ee.ethz.ch/index.cgi?page=publications;action=details;id=2699](http://control.ee.ethz.ch/index.cgi?page=publications;action=details;id=2699).
REFERENCES

[24] M. Kvasnica, P. Grieder, M. Baotic, and M. Morari. Multi-Parametric Toolbox (MPT). In HSCC (Hybrid Systems: Computation and Control), pages 448–462, March 2004. URL http://control.ee.ethz.ch/index.cgi?page=publications;action=details;id=53

[25] Q. Lin and J. Rokne. Interval approximation of higher order to the ranges of functions. Computers Math. Applic, 31(7):101–109, 1996.

[26] James D. Meiss. Differential Dynamical Systems. SIAM, 2007.

[27] David Monniaux and Pierre Corbineau. On the generation of positivstellensatz witnesses in degenerate cases. In ITP, volume 6898 of Lecture Notes in Computer Science, pages 249–264. Springer, 2011.

[28] César Muñoz and Anthony Narkawicz. Formalization of a representation of Bernstein polynomials and applications to global optimization. Journal of Automated Reasoning, 51(2):151–196, August 2013. doi: 10.1007/s10817-012-9256-3. URL http://dx.doi.org/10.1007/s10817-012-9256-3

[29] Antonis Papachristodoulou and Stephen Prajna. On the construction of lyapunov functions using the sum of squares decomposition. In IEEE CDC, pages 3482–3487. IEEE Press, 2002.

[30] Pablo A Parillo. Semidefinite programming relaxation for semialgebraic problems. Mathematical Programming Ser. B, 96(2):293–320, 2003.

[31] André Platzer, Jan-David Quesel, and Philipp Rümmer. Real world verification. In Proceedings of Intl. Conf. on Automated Deduction, pages 485–501. Springer, 2009.

[32] Victoria Powers and Bruce Reznick. Polynomials that are positive on an interval. Trans. Amer. Maths. Soc, 352:4677–4692, 2000.

[33] Stephen Prajna, Antonis Papachristodoulou, Peter Seiler, and Pablo A. Parrilo. SOSTOOLS: Sum of squares optimization toolbox for MATLAB, 2004.

[34] M. Putinar. Positive polynomials on compact semi-algebraic sets. Indiana Univ. Math, 42:969–984, 1993.

[35] Stefan Ratschan and Zhikun She. 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, 2010.

[36] Sriram Sankaranarayanan, Xin Chen, and Erika Ábrahám. Lyapunov function synthesis using handelman representations. IFAC conference on Nonlinear Control Systems, 2013.

[37] M.A. Ben Sassi, R. Testylier, T. Dang, and A. Girard. Reachability analysis for polynomial system using linear programming relaxations. In ATVA’2012, pages 137–151, 2012.

[38] K. Schmüdgen. The k-moment problem for compact semi-algebraic sets. Math. Ann, 289:203–206, 1991.

[39] Zhikun She, Bican Xib, Rong Xibao, and Zhiming Zhenga. A semi-algebraic approach for asymptotic stability analysis. Nonlinear Analysis: Hybrid Systems, 3(4):588–596, 2009.
REFERENCES

[40] Zhikun She, Haoyang Li, Bai Xue, Zhiming Zhenga, and Bican Xiab. Discovering polynomial lyapunov functions for continuous dynamical systems. *Journal of Symbolic Computation, 58*: 41–63, 2013.

[41] H.D. Sherali and C.H. Tuncbilek. A global optimization algorithm for polynomial programming using a reformulation-linearization technique. *Journal of Global Optimization, 2*: 101–112, 1991.

[42] H.D. Sherali and C.H. Tuncbilek. New reformulation-linearization/convexification relaxations for univariate and multivariate polynomial programming problems. *Operation Research Letters, 21*: 1–9, 1997.

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

[44] Paulo Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[45] W. Tan and A. Packard. Stability region analysis using sum of squares programming. In *Proc. ACC*, 2007.

[46] Alfred Tarski. A decision method for elementary algebra and geometry. Technical report, Univ. of California Press, Berkeley, 1951.

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

[48] Ufuk Topcu, Andrew Packard, Peter Seiler, and Timothy Wheeler. Stability region analysis using simulations and sum-of-squares programming. In *Proc. ACC*, pages 6009–6014. IEEE Press, 2007.

[49] A. Vannelli and M. Vidyasagar. Maximal lyapunov functions and domains of attraction for autonomous nonlinear systems. *Automatica, 21*(1):69–80, 1985. ISSN 0005-1098. doi: http://dx.doi.org/10.1016/0005-1098(85)90099-8.

A. Description of Synthesized Benchmarks

In this section, we describe each of the 15 benchmarks and present the results of our implementation.

**Benchmark #1:** Consider the two variable polynomial ODE:

\[
\frac{dx}{dt} = -12.5x + 2.5x^2 + 2.5y^2 + 10x^2y + 2.5y^3.
\]

\[
\frac{dy}{dt} = -y - y^2.
\]

The second relaxation finds the Lyapunov function and derivative shown below:

**Lyapunov function:**

\[2.02x^2 + 5y^2.\]
Lyapunov derivative function:

\[-50.5x^2 - 10y^2 + 10.1x^3 + 10.1xy^2 - 10y^3 + 40.4x^3y + 10.1xy^3.\]

**Benchmark #2:** Consider the two variable polynomial ODE:

\[
\frac{dx}{dt} = 6.933333x^3 + 4.566667x^2 - 21.5x.
\]

\[
\frac{dy}{dt} = 6.933333x^3 + 0.4x^2y + 2.066667x^2 + xy^2 + 0.6xy - 9x - y^2 - y.
\]

The first relaxation finds the Lyapunov function and derivative shown below:

Lyapunov function:

\[4.9183x^2 - 3.3198xy + 4.1497y^2.\]

Lyapunov derivative function:

\[-181.6089x^2 - 8.2995y^2 + 38.0596x^3 + 8.2995xy^2 - 8.2995y^3.
\]

\[+45.1833x^4 + 33.1978x^3y + 8.2995xy^3.\]

**Benchmark #3:** Consider the two variable polynomial ODE:

\[
\frac{dx}{dt} = -1.5x - x^2 + 0.5xy + 0.5y^2 - 2x^3 + xy.
\]

\[
\frac{dy}{dt} = -0.5y.
\]

The first relaxation finds the Lyapunov function and derivative shown below:

Lyapunov function:

\[4.9693x^2 + 4.8581y^2.\]

Lyapunov derivative function:

\[-14.908x^2 - 4.8581y^2 - 9.9386x^3 + 4.9693xy^2 + 4.9693xy^2 - 19.8773x^4 + 9.9386x^3y.\]

**Benchmark #4:** Consider the two variable polynomial ODE:

\[
\frac{dx}{dt} = -2x^3 - 0.5xy - 0.5x.
\]

\[
\frac{dy}{dt} = 0.25xy^2 - 0.125xy + 0.25y^2 - 0.4125y.
\]

The first relaxation finds the Lyapunov function and derivative shown below:
Lyapunov function:

\[ 4.9663x^2 + 4.8552y^2. \]

Lyapunov derivative function:

\[ -4.9663x^2 - 4.0056y^2 - 4.9663x^2y - 1.2138xy^2 + 2.4276y^3 - 19.8653x^4 + 2.4276xy^3. \]

**Benchmark #5:** Consider the three variable polynomial ODE:

\[
\frac{dx}{dt} = -2x^3 - 0.5xy - 0.5x - z^3 - z^2. \\
\frac{dy}{dt} = 0.25xy^2 - 0.125xy + 0.25y^2 - 0.4125y. \\
\frac{dz}{dt} = -z^2 - z.
\]

The first relaxation finds the Lyapunov function and derivative shown below:

Lyapunov function:

\[ 4.9295x^2 + 4.9513y^2 + 4.9848z^2. \]

Lyapunov derivative function:

\[ -4.9295x^2 - 4.0848y^2 - 9.9696z^2 - 4.9295x^2y - 1.2378xy^2 - 9.859xz^2 + 2.4756y^3 - 9.9696z^3 - 19.7179x^4 + 2.4756xy^3 - 9.859xz^3. \]

**Benchmark #6:** Consider the three variable polynomial ODE:

\[
\frac{dx}{dt} = -0.5x^3y + 0.5x^3z^2 - 3x^3 + y^5 + yz^4 - z^4. \\
\frac{dy}{dt} = 0.25y^2 - 0.25y. \\
\frac{dz}{dt} = yz^4 + z^4 - 2z^3.
\]

The third relaxation finds the Lyapunov function and derivative shown below:

Lyapunov function:

\[ 4.1212x^4 - 0.0292x^3y + 4.9077x^3y^2 + 3.5749x^3y^2 + 3.5749x^2y^2 + 4.9755x^2z^2 + 4.9863x^2 - 1.5913xy^3 + 1.5914xy^2 + 4.9939y^4 + 0.0598y^3z - 1.1362y^3 + 4.9812y^2z^2 - 0.0597y^2z + 4.9950y^2z + 0.0198yz^3 + 4.9864z^4 + 4.9926z^2. \]
Lyapunov derivative function:

\[-2.4975y^2 - 0.79571xy^2 + 3.3496y^3 + 0.029844yz^2 - 29.9178x^4 - 1.7881xy^2 + 1.9892xy^3 - 5.8461y^4 - 0.07469y^3z^2 - 2.4885y^2z^2 - 0.0049472yz^3 - 19.9701z^4 - 44.1669x^5 - 4.96964xy^4 - 4.78155x^3yz^2 + 0.013348xz^3 + 1.7874x^2y^3 + 0.013982x^2z^3 - 11.1666xy^4 + 4.9552xz^4 + 4.994yz^5 + 0.0452y^4 + 2.4906y^3z^2 + 0.12432y^2z^3 - 0.033711y^5 + 9.9792z^5 - 49.4547x^6 - 7.0989x^5y - 0.057848xz^5 - 21.4464x^4y^2 - 24.8666xz^4 + 3.9781x^3yz^3 - 0.011402xz^3yz^2 - 14.7231x^2y^4 - 34.6321x^2z^4 + 9.9782xy^4 + 0.013982xy^4z^3 + 9.9597xyz^4 + 0.01905xz^5 - 1.5914y^6 - 0.11959y^3z^3 - 21.576y^2z^4 + 9.8832yz^5 - 39.8832z^6 - 8.2424x^6y + 0.043775x^5y^2 + 7.3615x^5z^2 - 3.575x^4yz^2 - 4.9783xz^4y^2 - 15.6893x^3y^4 + 0.79344x^3yz^2 + 16.4807x^3z^4 + 14.8106x^2yz^5 - 0.019283xz^2y^4 + 14.8042x^2yz^4 + 9.9317xz^2z^5 - 7.1553xy^6 - 0.01503xy^5z - 9.951xy^4z^2 - 7.155xy^2z^4 + 0.0148xyz^5 - 9.958xz^6 + 3.1827y^7 + 3.1827x^3yz^4 + 9.9793y^2z^5 + 0.053588yz^6 + 19.9477x^7 + 8.2424x^6z^2 - 0.043771x^5yz^2 + 3.5749x^4yz^2 + 4.975x^4z^2 + 16.485x^3yz^3 - 0.79563x^3yz^5 + 16.49363x^3yz^3 - 0.08754x^2yz^6 + 0.019x^2yz^5 + 0.087017x^2yz^5 + 0.01905xz^5 + 19.9703x^2yz^5 + 7.1497x^2yz^7 + 9.951xy^5z^2 + 7.15xy^3z^4 + 0.101xy^2z^5 + 9.944xyz^6 + 1.5913y^8 - 1.5315y^4z^4 + 9.9627xyz^3z^5 + 0.063907y^2z^6 + 19.9431yz^7.\]

**Benchmark #7:** Consider the three variable polynomial ODE:

\[
\begin{align*}
\frac{dx}{dt} &= -0.53y^3 - 0.5x^3z^2 - x^3 + y^4z^2 - yz^3 + yz^2 - y^3 - z^2 \\
\frac{dy}{dt} &= 0.5^2z^2 - 0.5y^2 - 2y \\
\frac{dz}{dt} &= -yz^2 + yz + z^2 - z
\end{align*}
\]

The first relaxation finds the Lyapunov function and derivative shown below:

Lyapunov function:

\[
1.8371x^5 + 0.1146x^4y + 0.1431x^3z^4 + 4.9587xyz^4 - 2.0557x^3y^2 - 0.0143x^3yz - 0.4698x^3yz^3 + 3.2944x^3z^2 + 4.0441x^3 - 1.2295x^2y^3 + 4.9548x^2yz^2 + 3.2610x^2yz^2 + 0.6981x^2z^3 + 4.9648x^2z^2 + 4.9588x^2z^2 + 1.9598xy^4 + 0.9480xy^3 + 1.0295xy^2z + 0.6737xy^2z + 2.3539xyz + 2.1976x^2y^4 + 0.2212xyz + 3.3047z^2 - 0.3773y^5 + 0.1262y^4z + 4.9884y^4 - 1.7272yz^2 - 4.5919y^3z + 0.7677y^2z^3 + 4.9842y^2z^2 + 4.9898y^2z + 1.8746yz^4 + 0.4655yz^3 + 0.9830yz^2 + 0.8032z^5 + 4.9823z^4 - 0.6791z^3 + 4.9962z^2
\]

Lyapunov derivative function:

\[
-10.087xz^2 + 24.0416yz^2 + 16.1044yz - 3.9716x^4 - 66.1853x^2z^2 + 15.7876x^2z + 1.3715xz^3 - 19.9593y^4 + 71.2314y^2z^2 - 12.204yz^3 + 48.9619z^4 - 12.1323x^5 - 4.986x^4y - 0.42935x^4z - 77.376xz^3z^2.
\]
Consider the three variable polynomial ODE:

\[
\begin{align*}
\frac{dx}{dt} &= -0.5x^3 + 0.5x^2z^2 - x^3 + y^4z + y^4 - y^3z + 3yz^2 + z^3 - 3z^2 \\
\frac{dy}{dt} &= y^4z - y^4 - 2y^3 - z^3 + 3z^2 \\
\frac{dz}{dt} &= z^2 - 3z
\end{align*}
\]

The first relaxation finds the Lyapunov function and derivative shown below:

**Lyapunov function:**

\[
1.8371x^5 + 0.1146x^4y + 0.1431x^4z + 4.9587x^4 - 2.0557x^3y^2 \\
-0.4698x^3y + 3.2944x^3z^2 + 4.0441x^3 - 1.2295x^2y^3 + 4.9584x^2y^2 \\
+ 3.2610x^2yz^2 + 0.6981x^2z^3 + 4.9648x^2z^2 + 4.9858x^2 + 1.9598xy^4
\]
Consider the three variable polynomial ODE:

\[ \begin{align*}
\frac{dx}{dt} &= 0.05 x^2 y + 0.05 x^2 - 0.05 x^2 - 0.05 x^2 + 0.125 x^3 - 0.125 y^3 \\
&+ 0.125 y^2 - 0.125 z^2 + 0.2 y^3 + 0.2 y^4 - 0.2 z^5 - 0.2 z^4, \\
\frac{dy}{dt} &= 0.125 y^2 - 0.125 y^2 + 0.125 y^2 - 0.125 y + 0.2 z^5 + 0.2 z^5 \\
\frac{dz}{dt} &= -0.2 z^2 - 0.1 z
\end{align*} \]

**Benchmark #9:** Consider the three variable polynomial ODE:

\[ \begin{align*}
\frac{dx}{dt} &= 0.05 x^2 y + 0.05 x^2 - 0.05 x^2 - 0.05 x^2 + 0.125 x^3 - 0.125 y^3 \\
&+ 0.125 y^2 - 0.125 z^2 + 0.2 y^3 + 0.2 y^4 - 0.2 z^5 - 0.2 z^4, \\
\frac{dy}{dt} &= 0.125 y^2 - 0.125 y^2 + 0.125 y^2 - 0.125 y + 0.2 z^5 + 0.2 z^5 \\
\frac{dz}{dt} &= -0.2 z^2 - 0.1 z
\end{align*} \]
The second relaxation finds the Lyapunov function and derivative shown below:

**Lyapunov function:**
\[ 2.7500 x^2 + 2.7500 y^2 + 5.0000 z^2 \]

**Lyapunov derivative function:**
\[ -0.275x^2 - 0.6875y^2 - z^2 - 0.275x^3 + 0.275x^2y - 0.275x^2z - 0.6875xy^2 - 0.6875y^2z + 0.6875y^3z + 0.275x^3yz + 0.6875xy^2z - 1.1xz^4 + 1.1y^4z - 1.1xz^5 + 1.1xyz^4 \]

**Benchmark #10:** Consider the three variable polynomial ODE:
\[
\begin{align*}
\frac{dx}{dt} &= -0.01x + 1.666667xz^2 - 1.111111x^2y + 0.555556x^2z^2 - 0.555556z^2 \\
\frac{dz}{dt} &= -5y^2 + 5yz - 7.5z - 5y^3 + 5y^2 \\
\frac{dy}{dt} &= 2y^2 - 2y
\end{align*}
\]

The third relaxation finds the Lyapunov function shown below:

**Lyapunov function:**
\[ 1.5308 x^2 + 4.9266 z^2 + 4.9819 y^2 \]

**Lyapunov derivative function:**
\[ -0.030616x^2 - 73.8988z^2 - 19.9274y^2 - 1.7009xz^2 - 3.4017xy^2 + 49.2659z^2y + 49.2659zy^2 + 3.4017xz^2y + 3.4017xyz^2 - 3.4017x^2z^2 + 3.4017xyz^2 + 5.1026x^2z^2y^2 - 3.4017x^2y^2z^2 \]

**Benchmark #11:** Consider the four variable polynomial ODE:
\[
\begin{align*}
\frac{dx}{dt} &= -18xyw - 13xy - 18xw - 37.5x - 16z^3 + 4z^2y - 31.5z^2w - 6.5z^2 + 32zyw + 48zy - 16z^2w - 36zw \\
&\quad + 8y^3 + 36y^2w + 28y^2 + 68yw + 16y - 14w^2 \\
\frac{dz}{dt} &= -16z^2 + 24zy - 31.5zw - 27.5z - 32y^2 + 32yw + 16y - 16w^2 - 28w \\
\frac{dy}{dt} &= -36y^2w - 52y^2 - 36yw - 112y + 64w \\
\frac{dw}{dt} &= -4w.
\end{align*}
\]
The first relaxation finds the Lyapunov function and derivative shown below:

Lyapunov function:
\[ 1.6209y^2 + 1.3650yw + 4.8875w^2 \]

Lyapunov derivative function:
\[ -363.0877y^2 - 303.641w^2 - 168.5765y^3 - 187.6862yw^2 - 49.1398yw^2 - 116.7068y^3w - 49.1397y^2w \]

**Benchmark #12:** Consider the four variable polynomial ODE:

\[
\frac{dx}{dt} = 28x^3 - 28x^2z - 28x^2y + 9.5x^2 + 3xz^2 + 28xzy - xzw + 21xz + 14x^2 + 2xyw - 1.5x + 10.5xw - 60.5x - 6z^2y \\
- 15.5z^2w + 19.5z^2 - 22.5zy^2 - 2zyw - 18zy + 9zw + 9z + 12.5x^3 - 8y^2w + 8y^2 + 1yw^2 - 8yw + 41y + 12.5w^2 + 6w \\
\frac{dz}{dt} = 2z^3 + 4z^2 + 8.5z^2 + 4zw^2 + 4zw^2 + 5.75zw^2 - 7.25zy + 8.5zw^2 - 11zw - 42.5z + 9y^2w + 17.75y^2 + 22.5yw^2 \\
+ 12.5yw - 23y + 2.25w^3 + 11.25w^2 - 7w \\
\frac{dy}{dt} = -21y^2 - 12yw - 129y - 45w^3 - 101w^2 - 62w \\
\frac{dw}{dt} = -13.5w^2 - 27w.
\]

The second relaxation finds the weak Lyapunov function shown below:

Lyapunov function:
\[ 1.5759y^2 - 1.2527yw + 5.0000w^2 \]

Lyapunov derivative function:
\[ -406.592y^2 - 192.3343w^2 - 66.1895y^3 - 11.5165y^2w - 286.3966yw^2 - 8.4804w^3 - 141.8345yw^3 + 56. \]

In those examples all the approaches fail to find a Lyapunov function:
Benchmark #13: Consider the four variable polynomial ODE:

\[
\frac{dx}{dt} = -1.510417x^5 + 8x^4yw + 8.5x^4y - 8x^4w - 12.208333x^4 - 12x^3zyw \\
- 9.75x^3zy - 6x^3zw + 2x^3y^2 + 22x^3y + 4x^3yw + 6.5x^3y + 2.5x^3w^2 - 47x^3y - 60.875x^3 \\
- 8x^2z^3w + 2x^2z^2yw - 16.875x^2z^2y + 8x^2z^2w^2 - 13x^2z^2w - 8x^2zy^2w \\
- 7.5x^2zy^2 + 2x^2zy^2w + 37x^2zyw - 3.75x^2zy - 4x^2zyw^2 - 14.75x^2zw^2 - 46.5x^2zw - 8x^2y^3w \\
- 7.5x^2y^3 + 4x^2y^2w + 1x^2y^2 + 16x^2yw^2 + 6.5x^2yw^2 - 2x^2yw + 2x^2y - 12x^2w^3 \\
+ 6.5x^2w^3 - 7x^2w + 11.75x^2y - 6.4375x^2y + 16x^2w^2 + 25.5x^2w \\
+ 4x^2yw^2 + 12.25x^2y^2 - 2x^2y^2w + 26.5x^2yw + 1.125x^2y^2 - 1x^2w^3 \\
- 60.875x^2w^2 - 47.75x^2w + 44x^2yw^2 + 54.25x^2y - 24x^2y^2w - 83x^2yw^2 - 55.5x^2y^2 \\
+ 49.25x^2yw^2 + 29x^2yw - 13x^2y + 42xw^3 - 20.75xzw^2 + 32.5xzw - 1.5xy^4 + 16x^3w^2 \\
+ 9xy^4 - 0.5xy^3 - 29.5xy^2w - 43xy^2w - 45.5xy^2 + 16xyw^2 + 15xyw + 24xy \\
- 6zw^3 - 64.5xw^2 + 58.5xw + 41.83333x + 4.5z^3w + 6.5z^3yw - 12.71875z^3y + 12z^4w^2 \\
- 7.25z^3w - 6z^3y^2 - 8.375z^3y^2 - 9z^3yw^2 - 15.75z^3yw - 22.4375z^3y - 9z^3w^3 \\
- 50.4375z^2w^3 - 69.875z^2w^3 + 14z^3y^2w + 11.625z^3y^2 - 2z^3y^2w^2 - 56.5z^3y^2w \\
- 34.75z^3y^2 + 54.625z^2yw^2 - 33.5z^3yw + 11z^2y + 61z^2w^3 + 14.625z^2w^3 - 0.25z^2w \\
- 8z^2w - 20.75zy^4 + 8zy^3w^2 + 18.5zy^3w + 22.75zy^3 - 31.75zy^3w^2 - 50.5zy^3w \\
- 17.75zy^3 - 12zy^3 + 1.25zy^3w + 71.5zyw + 8zy - 1zw^4 + 6zw^3 - 143.25zw^2 - 1.75zw \\
+ 16yw^2 - 18yw - 16yw^2 - 56yw - 46yw + 5yw^2 + 4yw^2 + 22yw + 8yw^2 \\
+ 1yw^2 - 49yw^2 + 137yw^2 + 19yw^2 + 2yw^2 - 25yw^2 - 9yw^3 + 31yw^2 - 55yw + 12yw \\
- 2w^3 - 23.5uw^4 - 11uw^3 - 31.5uw^2 - 11uw \\

\frac{dz}{dt} = -3.020833x^5 + 15.46875z^3 + 10.583333z^3 - 21.0625z^3 + 18.25z^3z \\
- 31.75z^3 + 18.875x^3z + 9.75x^3z^2 - 51.625x^3z + 8.5x^3 - 10.25xz^3 - 7.5xz^3 \\
+ 12.75xz^2 + 14.25xz - 40.666667x + 3.5z^3 - 8z^3 - 51z^3 - 5z^3 - 4.5z^3 \\

\frac{dy}{dt} = 3.25z^3w - 6.359375z^3 + 13z^4yw + 15.3125z^4y - 4.5z^4w^2 + 0.125z^4w \\
- 44.71875z^4 - 9z^4yw - 10.1875z^4y - 1z^3yw - 20.25z^3yw + 20.875z^3y \\
+ 42.3125z^3w - 12.75z^3w - 3.5z^3 - 6.375z^3y + 4z^3y^2w - 6.75z^3yw \\
- 17.125z^3y^2 + 27.625z^2yw - 9.25z^2yw - 72.375z^2y - 6z^2w^3 - 19z^2w^3 - 23.25z^2w \\
- 3.5z^2 + 8zy^4 + 9zy^3w - 8zy^3w - 2z^3y^2w - 23zy^3 + 9zy^3w^2 - 42zy^3w + 38zy^2 \\
+ 4zy^4 + 11zy^3 - 57.5zyw - 164.5zyw + 45.5zy + 7zw^4 - 12.75zw^4 - 44.5zw^4 + 74.75zw^2 \\
- 23.5zw^2 - 22z + 4zw^2 + 4zw^2 + 6yw^2 + 16yw^2 - 16yw^2 - 85yw + 8yw^2 \\
+ 3yw^2 + 11yw^2 + 2zw^2 + 2yw^2 + 6yw^2 - 52yw^2 + 17yw^2 - 25yw^2 - 8w^3 - 16yw^4 - 3w^3 - 14.5w^2 + 9w \\

\frac{dw}{dt} = -2e^6 + 6e^6w + 5.375e^6 - 4.5e^6w^2 - 22.21875e^4w - 74.9375e^4 \\
+ 34.5e^3w^2 + 22.3125e^3w - 15.125e^3 - 5e^3w^3 + 33.5e^3w^2 - 128.125e^2w \\
- 40.875e^2 - 21.75e^2w + 13.5e^2w + 13.75zw + 4.5e - 12w^4 - 22w^3 - 12.5w^2 - 4w
**Benchmark #14:** Consider the four variable polynomial ODE:

\[
\frac{dx}{dt} = 7.145833x^3 - 20x^4y - 2.4166667x^4 - 10x^3zy + 16x^2zw + 20x^3y^2 - 18x^3yw - 28x^3y
- 10x^3w^2 - 12x^3w - 77.5416667x^3 + 3.5x^2z^2yw - 25x^2z^2y + 2.5x^2z^2w
- 20x^2zy^2w - 30x^2zy^2 + 12x^2zyw^2 - 9x^2zy + 11x^2zy - 12x^2zw^3 - 21x^2zw^2 + 15x^2zw
+ 28x^2yw^2 - 28x^2yw + 28x^2yw^2 + 26x^2yw^2 - 26x^2yw^2 + 18x^2yw^2 + 14x^2yw^2 + 2x^2yw^2
- 40x^2yw - 7x^2y - 2x^2w^3 - 8x^2w^2 - 17x^2w - 42x^2 + 13.75xz^3y + 5.5xz^2y - 24xz^3w^2
- 2.75xz^3w + 4xz^3yw + 9xz^2yw^2 - 2xz^2yw + 9.5xz^2y - 6xz^2w^3
- 10.5xz^2w^2 + 31.5xz^2w + 2xz^3y^2 + 19xz^3y - 24xz^3y^2 - 12xz^3w + 2xz^3y + 31xz^3w
+ 43xz^3w + 15xz^3w + 23.5xz^3w - 12xz^2w^3 - 40.5xz^2w^2 - 40.5xz^2w^2 - 92x - 12z^3w + 10.375z^4yw
+ 4.25z^4y - 12z^3w^3 + 3.125z^3w^3 + 12z^2y^3w + 0.5z^2y^2 + 4z^2yw^2 + 16z^2yw + 23.75z^2y
- 15z^3w^3 - 30.25z^3w^2 + 25.75z^3w + 12z^3w + 39.5z^2y^3 - 32z^2y^3w^2
+ 16z^2yw^2 + 63z^2yw + 29.5z^2yw^2 - 40.5z^2yw + 47.75z^2y - 36.5z^2y^3 - 27.5z^2w^2
- 88.25z^2w^2 - 3zy^2w + 7zy^2w - 26zy^2w^2 + 48zy^2w + 26.5zy^2w^2 + 35.5z^2w^2
+ 26.75z^2w^2 - 51z^2y^2 + 14zyw^2 - 2zyw - 5zyw - 62zyw - 50.5z^2y - 19.5zw^4 - 29zw^4
- 53.25zw^4 - 37zw + 2.5yw^4 - 2yw^4 - 8.5yw^4 + 24yw^3 - 72.5yw^3 - 88yw^3 + 3yw^3 - 18yw^3
- 94.5yw^3 - 126yw^3 - 8yw^3 + 14yw^3 + 5yw^3 - 10yw^3 - 46.5yw^2 + 84yw + 12y
- 14y^5 + 2yw^4 + 16w^5 + 29.5w^5 - 12w
\]

\[
\frac{dy}{dt} = 4.291667x^3 + 5.4375x^4z - 20.83333x^4 - 11.125x^3z + 12.75x^3z - 102.08333x^3 - 10.25x^2z^2 - 0.5x^2z^2 - 2.625x^2z
- 88x^2 - 24.5xz^4 - 21xz^3 - 70xz - 57x + 5z^5 + 7z^4 - 112.5z^3 - 31z^2 - 90z
\]

\[
\frac{dz}{dt} = 5.1875z^5w - 7.875z^5 + 5z^4yw + 10.25z^4y + 20z^4w - 4.5z^4 - 2.625z^4
+ 5z^3y^3w + 9.75z^3y^2 - 16z^3yw^2 - 12z^3yw + 9.5z^3y + 14.75z^3w^3 - 56z^3w^2
- 85.25z^3w - 33.125z^3w + 3z^3yw^2 + 31z^3yw^2 + 3z^3yw^2 + 8z^3yw^2 + 7.25zw^2
+ 3.5zw^3 - 14.25z^2yw^2 - 34.625z^2yw - 74z^2yw + 7z^2w^4 + 4zw^3 + 24.5zw^2
- 20zw^3 - 35.75zw^2 + 10zw^4 + 11.25zw^2 + 22zw^3 + 28.75zw^2 + 12zw^3
+ 4.25zw^2 + 44zw^2 + 10.5zw^2 + 37zw^3 + 24.75zw^2 + 83zw^2 - 83zw^2 - 83zw^2 + 7zw^5
+ 41zw^4 + 30zw^4 + 47.75zw^2 - 12zw^2 - 18zw^2 + 10zw^5 - 20zw^4 + 17.5zw^4
+ 12.5zw^4 + 7zw^2w + 8zw^2 - 100zw^2 + 10zw^2 - 103zw^2 - 14zw^2w - 4zw^2 - 42zw^3
- 70zw^3 - 15zw^5 - 36zw - 8zw^4 - 20zw^3 + 8.5zw^2 - 12w
\]

\[
\frac{dw}{dt} = -6z^6 - 6z^6w + 13.5625z^2 - 7.5zw^2 - 15.125z^4w + 3.375zw^2 - 6.25z^2w^2
- 22.75z^2w + 9.875z^2 - 9.75z^2w^3 - 38.5z^2w^2 - 74.125zw^2 - 91zw^2
- 13zw^3 + 8zw^2 + 21.75zw + 30zw - 13.5zw^4 - 39.5zw^3 - 7w^2 - 43w
\]
The second relaxation finds the Lyapunov function shown below:
Benchmark #15: Consider the four variable polynomial ODE:

\[
\frac{dx}{dt} = 8x^5 + 84x^4zw - 2x^4z - 204.5x^3yw - 4x^3w^2 + 42x^3w + 29x^2 + 28x^2zw + 18.375x^2z^2 - 53.25x^2yw
+ 13x^2zw - 25.5xz^2w + 8.75x^2z - 19x^2w - 56.5x^2w - 70x^3 - 1.187500x^3z^2 + 1.1875x^3zw
- 7.5x^2z^2w^2 + 43.25x^2zw + 30.25x^2z - 44.75x^2zy^2w + 15x^2zy^2w - 244.5x^2zy^w - 28x^2zw^3 - 177x^2zw^2
+ 300x^2zw - 48.5x^2z + 31.5x^2y^2w - 46x^2y^2w^2 + 374.5x^2y^2w - 12x^2y^2w^3 + 275x^2yw^2 - 596.5x^2yw
+ 2x^2w^3 + 112x^2w^3 + 88.5x^2w^2 + 128x^2w - 41.5x^2 + 9.656250x^4 + 0.6875x^3yw - 3.75x^3w^2
+ 74.625x^3zw + 28.562500x^3z - 15.375x^3yw - 7.5x^3yw^2 - 159.25x^3zw^2 - 26x^3zw + 126x^2w^2
- 93.75xz^2 - 29.75xy^3w - 25xy^2w^2 + 210.25xy^2w + 36xy^3w + 169.5xy^2w - 420.25xy^w + 18xzw^2 - 57xz^3
- 52.25xz^2w - 266xzw + 17.25xz - 11xw^3 - 108xw^3 - 135xw^3 + 42xw - 76x + 14.015625x^3 + 0.343750x^4yw - 1.875^4w^2
+ 31.312500c^4w + 33.734375c^4w - 7.687500c^2yw - 3.75c^3w^2 - 62.625c^3yw - 33c^3w - 142.468750c^3 + 16.875c^2yw^3
- 16.5c^2yw^3 + 109.125c^2yw^3 + 22c^2yw^3 + 96.25c^2yw^3 - 204.625c^2yw^3 + 9c^2w^4 - 25.5c^2w^3
- 37.625c^2w^3 - 147.5c^2w^3 - 2.25c^2w^3 - 117.75c^2w^3 + 117.75c^2w^3 - 73.25c^2w^3 - 72c^2w^3
- 264.5y^2w^2 + 538.25yz^2w - 18yz^3 - 24yzw^3 + 97.25yzw^3 + 188yzw - 28zw^4 + 95zw^3
+ 104zw^3 - 51zw - 164.875000c + 246yw^3 - 238yw^3 + 210yw^3 + 173yw^3 + 3
+ 564y^3w^2 - 1049.5yw^3 + 40y^2w^4 + 65yw^3 - 134.5y^2w^2 - 440.5yw^2 - 34.5yw^5
- 37.5yw^4 - 285yw^3 - 248yw^2 + 243yw + 17w^3 + 40w^3 + 13w^3 + 23w^2 - 42w
\]

\[
\frac{dz}{dt} = -7x^2yw - 8x^2yw + 85x^2w - 16x^2yw^2 - 71x^2yw^2 - 28x^2yw^2 - 66x^2yw^2 + 70x^2yw + 4x^2w^4 + 140x^2w^3
+ 84x^2w^3 + 278x^2w + 15.468750c^4 +
18.093750c^4 - 130.937500c^3 + 82^2 - 183.75z
\]

\[
\frac{dy}{dt} = 203x^3w + 8x^3 + 30.75x^3z + 3.5x^3y - 59.5x^4d + 11.25x^3z^3 - 39.5x^3zy - 38.25x^3z^2 - 33.5x^3y^2 + 4x^3y^2
+ 38x^3yw^2 + 331x^3yw + 59.5x^3y - 268x^3w^2 + 722.5x^3w + 38x^3 + 44.5x^3z^2 + 21x^3z^2y - 1.125x^3z^2
+ 7x^2zy^3 + 19.75x^2zy - 5x^2z + 14.5x^2y^3 - 52x^2y - 57x^2 + 3.906250x^2c + 8.125000x^2c + 14.8125
x^3^2 + 26.625x^2yw^2 + 35.625x^2yw^2 - 0.25x^2c + 4.25x^2y^3 + 17.25x^2y^2 - 2.5x^2c^2 - 65.75xz
- 243.5xz^2w + 2.5xy^3w - 283xy^3w - 140.5xy^3w - 21.5y^3w - 144y^3w^3
- 549xy^3w^2 + 1021.5xz^2w - 6x^zw^6 - 40xy^w^4 - 40xy^w^3 + 17.65xzw^2 + 438xzw^2 + 37.5xzw^2
+ 35xzw^2 + 36xzw^2 + 296xzw^2 + 315xzw^2 - 212xzw - 68x - 9.921875c^4 - 0.343750c^4yw^5
+ 32.375z^2y + 1.875000c^4w^2 - 3.312500c^4w^2 - 10.546875c^4 + 15.75x^3c^2 - 4.1875c^3y
- 96.40625c^3 - 44.875000c^2z^2 + 31.5x^3y^3 + 33.5z^3y^2w - 53.125000
z^2y^2w + 2.625zy^2w^2 - 22zy^3w^2 - 152.25zy^2w^2 + 226.625zy^2w^2 - 45.875000c^2y - 9c^2w^4
- 30.5zw^3 + 39.625z^2w + 75.5zw^2 + 24.25zw^3 - 4.75zy^3 - 8.75zy^2 - 21.5
zy - 119.625 - 240yw^3 + 182w^3 - 184y^w - 15.5y^4 - 117y^3w^3 - 548y^3w^3 + 1027.5y^3w^3
+ 6.5y^3 - 40y^2w^2 - 93yw^2 + 208.5yw^2 + 527.5yw^2 + 16yw^3 + 34yw^5 +
105yw^6 + 299yw^6 + 336yw^6 - 169yw^6 - 93.5yw^6 + 52yw^6 - 916yw^6 + 165yw^6 - 73w^2 + 74w
\]

\[
\frac{dw}{dt} = 246y^3 - 182y^3w + 168.5y^3 + 145yw^2 + 548yw^4 - 1026yw^4 + 40yw^3
+ 116yw^2 - 191.5yw^3 - 523.5yw^3 - 34.5yw^4 - 105.5yw^4 - 324.5yw^2
- 366yw^2 + 125yw^2 + 34yw^4 - 18.5yw^3 - 191.5yw^2 + 28yw^2 - 128yw - 29w - 74w^2 + 3.1w^2 - 41w
\]
The second relaxation finds the Lyapunov function shown below:
Table 3. Performance of our approach on the synthesized benchmarks (continued). Note that MO: out-of-memory termination, TO: time-out. All times are reported in seconds.

| ID | $n$ | $d_{\text{max}}$ | Putinar (SOS) | Bernstein (our approach) |
|----|-----|-----------------|---------------|-------------------------|
|    |     |                 | SUCC? | $d_L$ | $d_Q$ | SETUP | $t_{\text{SDP}}$ | Rel. Typ. | SUCC? | # Box | SETUP | $t_{LP}$ |
| 8  | 3   | 5               | X(NP) | 2     | 2     | 0.8   | 1.7 | LP1 | ✓   | 8   | 10.63 | 10.9  |
|    |     |                 | X(NP) | 2     | 4     | 40.9  | 7.9 | LP2 | ✓   | 8   | 11.91 | 30.97 |
|    |     |                 | X(NP) | 4     | 4     | 40.1  | 5.5 | LP3 | X(NP) | 8   | 22.38 | 130.77 |
| 9  | 3   | 2               | X(NP) | 2     | 2     | 0.9   | 4.1 | LP1 | X(NP) | 8   | 1.99  | 0.61  |
|    |     |                 | X(NP) | 2     | 4     | 42.2  | 3.7 | LP2 | ✓   | 8   | 2.06  | 0.92  |
|    |     |                 | ✓   | 4     | 4     | 41.9  | 3.1 | LP3 | ✓   | 8   | 3.04  | 3.81  |
| 10 | 3   | 5               | X(NP) | 2     | 2     | 0.8   | 2   | LP1 | X(NP) | 8   | 3.48  | 3.19  |
|    |     |                 | ✓   | 2     | 4     | 42.0  | 4.0 | LP2 | ✓   | 8   | 1.23  | 1.88  |
|    |     |                 | ✓   | 2     | 4     | 42.0  | 4.0 | LP3 | ✓   | 8   | 1.56  | 0.60  |
| 11 | 4   | 3               | ✓   | 2     | 2     | 3.7   | 3.1 | LP1 | ✓   | 16  | 3.58  | 3.25  |
|    |     |                 | ✓   | 2     | 2     | 3.7   | 3.1 | LP2 | ✓   | 16  | 4.34  | 17.27 |
|    |     |                 | ✓   | 2     | 2     | 3.7   | 3.1 | LP3 | X   | 16  | 9.05  |       |
| 12 | 4   | 3               | X(NP) | 2     | 2     | 3.7   | 2.1 | LP1 | ✓   | 16  | 5.16  | 16.85 |
|    |     |                 | X(MO) | 2     | 4     | >600  |     | LP2 | ✓   | 16  | 6.38  | 12.86 |
|    |     |                 | X(MO) | 2     | 4     | >600  |     | LP3 | X(NP) | 16  | 22.23 | 224.23 |
| 13 | 4   | 6               | X(NP) | 2     | 2     | 4     | 3.1 | LP1 | X(NP) | 16  | 41.36 | 627.25 |
|    |     |                 | X(MO) | 2     | 4     | >600  |     | LP2 | X(NP) | 16  | 43.38 | 988.31 |
|    |     |                 | X(MO) | 2     | 4     | >600  |     | LP3 | X(TO) | 16  | >1200 |       |
| 14 | 4   | 6               | X(NP) | 2     | 2     | 3.8   | 3.6 | LP1 | X(NP) | 16  | 37.45 | 339.86 |
|    |     |                 | X(MO) | 2     | 4     | >600  |     | LP2 | X(NP) | 16  | 41.93 | 1049.53 |
|    |     |                 | X(MO) | 2     | 4     | >600  |     | LP3 | X(TO) | 16  | >1200 |       |
| 15 | 4   | 6               | X(NP) | 2     | 2     | 3.8   | 3.9 | LP1 | X(NP) | 16  | 38.55 | 368.48 |
|    |     |                 | X(MO) | 2     | 4     | >600  |     | LP2 | X(NP) | 16  | 45.32 | 888.33 |
|    |     |                 | X(MO) | 2     | 4     | >600  |     | LP3 | X(TO) | 16  | >1200 |       |
