Polynomial Constraints for Robustness Analysis of Nonlinear Systems

Neelay Junnarkar[1], Peter Seiler[2], Murat Arcak[1]

Abstract— **This paper presents a framework for abstracting uncertain or non-polynomial components of dynamical systems using polynomial constraints. This enables the application of polynomial-based analysis tools, such as sum-of-squares programming, to a broader class of non-polynomial systems. A numerical method for constructing these constraints is proposed. The relationship between polynomial constraints and existing integral quadratic constraints (IQCs) is investigated, providing transformations of IQCs into polynomial constraints. The effectiveness of polynomial constraints in characterizing nonlinearities is validated via numerical examples to compute inner estimates of the region of attraction for two systems.**

I. INTRODUCTION

A classic problem in nonlinear control is the problem of absolute stability: analyzing stability of a feedback loop of a linear time-invariant (LTI) system and a nonlinearity [1]. The traditional problem considers conditions under which stability can be guaranteed for any memoryless sector-bounded nonlinearity. Integral quadratic constraints (IQCs) are a widely used tool that generalizes this type of analysis to many classes of nonlinearities and uncertainties [2], offering computationally tractable analysis methods. Libraries of IQCs are available for various classes of nonlinearities, such as sector-bounded or slope-restricted nonlinearities, and may be applied depending on the particular nonlinearities of a system. Tight bounds of nonlinearities, and therefore a tight characterization of the nonlinear system, lead to improved performance certificates, such as larger inner estimates of a region of attraction (ROA) or smaller L_2-gain bounds.

IQC analysis has been extended from a nominal linear system to a nominal polynomial system using sum-of-squares (SOS) programming. SOS programming enables optimization with constraints that are sufficient conditions for polynomial nonnegativity, making SOS programming a useful computational tool in the analysis of polynomial dynamical systems [3]. IQCs have been used to characterize the non-polynomial parts of non-polynomial systems, using SOS programming for the polynomial parts [3], [4]. Quadratic constraints have been extended, taking advantage of the capabilities of SOS programming, to non-homogenous quadratics and sector constraints which do not pass through the origin [5]. Changes of variables that lift to higher dimensional spaces have been used to apply SOS techniques to large classes of nonlinearities [6].

Despite their utility, quadratic constraints can result in conservative characterizations of nonlinearities. For example, a sector bound poorly captures the asymptotic behavior of a saturating function such as $\tanh$. We propose a generalization of IQCs to higher-order polynomial constraints, enabling tighter characterizations of nonlinearities. We present definitions of polynomial constraints, a numerical method to directly synthesize a class of polynomial constraints, and establish results on deriving them through polynomial transformation of IQCs. Finally, we demonstrate the utility of polynomial constraints by verifying inner estimates of the regions of attraction for two systems where sector constraints result in conservative ROAs.

Section II introduces polynomial constraints. Section III establishes conditions under which a set can be certified as an inner estimate of the region of attraction, and formulates an SOS-based algorithm for computing this inner estimate. Section IV presents examples of polynomial constraints, methods for constructing them, and results on transformations of constraints. Section V gives numerical examples, applying polynomial constraints to the problem of computing inner estimates of the ROA for two systems and comparing results with estimates computed using sector constraints.

A. Notation

$\mathcal{L}_2^n$, $\mathcal{L}_{2e}^n$, and $\mathcal{L}_\infty^n$ denote signals $[0,\infty) \to \mathbb{R}^n$ that are square integrable, locally square integrable, and bounded, respectively. $\mathbb{R}[x]$ denotes polynomials in x. $\Sigma[x]$ denotes sum of square polynomials in x: $p \in \Sigma[x]$ if there exist $p_1, \ldots, p_k \in \mathbb{R}[x]$ such that $p = \sum_{i=1}^k p_i^2$.

II. POLYNOMIAL CONSTRAINTS

Let $\Psi : \mathcal{L}_2^{n_v \times n_w} \to \mathcal{L}_2^{n_z}$ be an LTI system defined as:

$$\begin{aligned}\dot{x}_\Psi(t) &= A_\Psi x_\Psi(t) + B_{\Psi v} v(t) + B_{\Psi w} w(t), \quad x_\Psi(0) = 0 \\ z(t) &= C_\Psi x_\Psi(t) + D_{\Psi v} v(t) + D_{\Psi w} w(t)\end{aligned}$$

The origin is stable if A_Ψ is Hurwitz. We denote the zero initial-condition response to inputs v and w by $z = \Psi(v, w)$.

An operator $\Delta : \mathcal{L}_{2e}^{n_v} \to \mathcal{L}_{2e}^{n_w}$ satisfies the time-domain hard IQC defined by a stable LTI system Ψ and $M = M^\top \in \mathbb{R}^{n_z \times n_z}$ if

$$\int_0^T z(t)^\top M z(t) dt \geq 0, \quad \forall T \geq 0, v \in \mathcal{L}_{2e}^{n_v}$$

where $z = \Psi(v, \Delta(v))$[2]. Integral polynomial constraints are a natural generalization of IQCs.

[1]Neelay Junnarkar and Murat Arcak are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 USA neelay.junnarkar@berkeley.edu, arcak@berkeley.edu

[2]Peter Seiler is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA pseiler@umich.edu

Definition 1: Let $\Delta : \mathcal{L}_{2e}^{n_v} \to \mathcal{L}_{2e}^{n_w}$ be an operator mapping the signal v to w and let Ψ be a locally input-to-state stable polynomial dynamical system defined as

$$\begin{aligned} \dot{x}_\Psi(t) &= f_\Psi(x_\Psi(t), v(t), w(t)), \quad x_\Psi(0) = 0 \\ z(t) &= g_\Psi(x_\Psi(t), v(t), w(t)) \end{aligned} \tag{1}$$

where $x_\Psi \in \mathbb{R}^{n_\Psi}$, $z \in \mathbb{R}$, and f_Ψ and g_Ψ are polynomials such that $f_\Psi(0,0,0) = 0$. We say Δ satisfies the integral polynomial constraint (IPC) defined by Ψ if

$$\int_0^T z(t)dt \geq 0, \quad \forall T \geq 0, v \in \mathcal{L}_{2e}^{n_v}, w = \Delta(v) \tag{2}$$

Special cases of this include the static polynomial constraint, where $\Psi(v,w)(t) = p(v(t), w(t))$ for some polynomial p, and the pointwise polynomial constraint, where $\Psi(v,w)(t) \geq 0$ for all $t \geq 0$. The most common case we will discuss is the pointwise static polynomial constraint:

$$\Psi(v,w)(t) = p(v(t), w(t)) \geq 0, \quad \forall t \geq 0 \tag{3}$$

Examples of polynomial constraints are given in Section IV.

III. Application to Computing Regions of Attraction

We demonstrate the utility of IPCs by computing inner-estimates of the region of attraction for a non-polynomial system. We bound the non-polynomial parts of the system with IPCs and apply methods for polynomial systems using SOS programming.

Consider the following system where f and g are polynomial, $x \in \mathbb{R}^n$, $\Delta : \mathcal{L}_2^{n_v} \to \mathcal{L}_{2e}^{n_w}$, $f(0,0) = 0$, $g(0) = 0$, and $\Delta(0) = 0$:

$$\begin{aligned} \dot{x}(t) &= f(x(t), w(t)) \\ v(t) &= g(x(t)) \\ w(t) &= \Delta(v)(t) \end{aligned} \tag{4}$$

Theorem 1: Consider the system in (4), suppose Δ satisfies the IPC defined by Ψ, and that Δ has a finite induced $\mathcal{L}_\infty$ gain: there exists $\gamma \geq 0$ such that $\|\Delta(v)\|_\infty \leq \gamma \|v\|_\infty$ for all $v \in \mathcal{L}_\infty$. Define the extended system of (4) and (1) as $\dot{x}_e(t) = f_e(x_e(t), w(t)), v(t) = g_e(x_e(t)), z(t) = h_e(x_e(t), w(t)), w(t) = \Delta(v)(t)$ with $x_e = (x, x_\Psi)$. If there exists a positive definite, continuously differentiable function $V : \mathbb{R}^{n+n_\Psi} \to \mathbb{R}$, a nonnegative $s_\Psi \in \mathbb{R}$, and $\epsilon > 0$ such that $\{x_e | V(x_e) \leq c\}$ is compact and

$$\nabla V(x_e)^\top f_e(x_e, w) + s_\Psi h_e(x_e, w) \leq -\epsilon x_e^\top x_e \tag{5}$$

for all $w \in \mathbb{R}^{n_w}$ and $x_e \in \mathbb{R}^n$ such that $V(x_e) \leq c$, then $\{x_e | V(x_e) \leq c\} \cap \{x_e = (x, x_\Psi) | x_\Psi = 0\}$ is an inner estimate of the ROA of (4).

Proof: By integrating (5) and applying the IPC $\int_0^T z(t)dt \geq 0$, we establish that the sub-level set $\{x_e | V(x_e) \leq c\}$ is forward invariant. Consider a trajectory of the system such that $V(x_e(0)) \leq c$. Forward invariance implies $x_e \in \mathcal{L}_\infty$ by the assumption that the sub-level set is compact. Because g_e is polynomial, it is Lipschitz on the compact set $\{x_e \in \mathbb{R}^n | V(x_e) \leq c\}$, so there exists an $L \geq 0$ such that $\|v(t)\| \leq L\|x(t)\|$ for all $t \geq 0$. This implies $v \in \mathcal{L}_\infty$ because $x_e \in \mathcal{L}_\infty$, and therefore that $w \in \mathcal{L}_\infty$ by the assumption that Δ has a finite induced $\mathcal{L}_\infty$ gain. Because $w \in \mathcal{L}_\infty$, there exists $M \geq 0$ such that $\|w(t)\| \leq M$ for all $t \geq 0$. Then, because f_e is polynomial, it is Lipschitz on the compact set $\{x_e \in \mathbb{R}^n | V(x_e) \leq c\} \times \{w \in \mathbb{R}^{n_w} | \|w\| \leq M\}$. Therefore $\dot{x}_e \in \mathcal{L}_\infty$ by an argument similar as to that for v. Additonally, integrating (5) shows $\epsilon \int_0^T x_e(t)^\top x_e(t) dt \leq V(x_e(0))$, so $x_e \in \mathcal{L}_2$. Therefore, $x_e(t) \to 0$ as $t \to \infty$ because $x_e \in \mathcal{L}_2$ and $\dot{x}_e \in \mathcal{L}_\infty$ by Barbalat's lemma. The intersection of $\{x_e | V(x_e) \leq c\}$ with $\{x_\Psi = 0\}$ is due to the filter initial condition. ■

Remark 1: If Δ satisfies Ψ pointwise, then s_Ψ can be a nonnegative function of x_e and w.

A. Computing Estimates of ROAs using SOS Programming

We formulate the ROA estimation problem as a sequence of SOS programming problems, where nonnegativity constraints are enforced via SOS constraints. For this section, we consider the case where Ψ is a pointwise static polynomial constraint, which enables the inclusion of additional SOS multipliers (see Remark 1). Applying Theorem 1, we aim to solve the following problem, where $\ell_x = \epsilon x^\top x$, $\ell_{xw} = \epsilon(x^\top x + w^\top w)$, and $\epsilon > 0$ is set to $\approx 10^{-6}$:

$$\max_{V, s_c, s_\Psi} \quad \text{volume}(\{x | V(x) \leq 1\}) \tag{6a}$$

$$\text{s.t.} \quad V \in \mathbb{R}[x], \quad V(0) = 0 \tag{6b}$$

$$V - \ell_x \in \Sigma[x] \tag{6c}$$

$$s_c, s_\Psi \in \Sigma[(x,w)] \tag{6d}$$

$$\begin{aligned} &-\nabla V(x)^\top f(x,w) - \ell_{xw} \\ &\quad - s_c(x,w)(-V(x) + 1) \\ &\quad - s_\Psi(x,w) p(g(x,w), w) \in \Sigma[(x,w)] \end{aligned} \tag{6e}$$

To address the bilinearity between decision variables (the term $s_c(x,w)V(x)$), we split this into an alternation between an "expansion step", where V is fixed and the remaining variables are solved for, and a "reshaping step", where V is a decision variable. Similar alternation algorithms have been used in [4], [7], [8].

The SOS program in the expansion step is as follows, where bilinearity in s_c and c is addressed through bisection.

$$s_c^*, c^* = \arg\max_{c \in \mathbb{R}, s_c, s_\Psi \in \Sigma[(x,w)]} \{c \mid \text{s.t. (6e)}\} \tag{7}$$

The SOS program in the reshaping step is as follows, where s_c^* and c^* are the outputs of the expansion step and V^* is the previous V:

$$\max_{V, s_\Psi, s_e} \quad 0 \tag{8a}$$

$$\text{s.t.} \quad \text{(6b), (6c), (6e) with } s_c^* \tag{8b}$$

$$s_\Psi, s_e \in \Sigma[(x,w)] \tag{8c}$$

$$\begin{aligned} &1 - V(x) \\ &\quad - s_e(x,w)(c^* - V^*(x)) \in \Sigma[(x,w)] \end{aligned} \tag{8d}$$

For numerical stability, we find it useful to allow c to be in a small range of c^*, e.g. $[0.9c^*, 0.99c^*]$, and bisect over this range to maximize c. This allows the new certified inner estimate of the ROA to be a superset of a sub-level set of the previous estimate, while changing shape.

Algorithm 1 ROA Estimation via SOS

```
Require: $V_0(x)$                              ▷ Initial Lyapunov function
1: $V(x) \leftarrow V_0(x)$
2: while not converged do
3:     Expansion Step: Maximize $c$ for fixed $V(x)$
4:     $s_c^*, c^* \leftarrow$ solve expand step, (7), via bisection
5:     Reshape Step: Find new $V(x)$
6:     $V^* \leftarrow V$
7:     Bisect over $c \in [0.9c^*, 0.99c^*]$ to maximize $c$:
8:         $V \leftarrow$ solve reshape step, (8)
9: return $\{x \mid V(x) \leq c\}$                ▷ The estimated ROA
```

We use a hyperparameter n_V for the degree of V and n_{total} for the degree of each of the polynomials in the constraints in each SOS program. The degrees of each s-certificate are determined from n_{total}, n_V, and the degrees of any fixed polynomials such as p. Our implementation[1] uses the SOSTOOLS [9] package in Matlab.

IV. Constructing Polynomial Constraints

To build intuition, we first provide a few examples of polynomial constraints, plotted in Figure 1, where we construct them by inspection and by using polynomial approximations.

For the function $\tanh(x)$, a soft saturation, any local sector bound will be conservative due to the local linear nature and asymptotic saturated nature of the function. With polynomial constraints, we can consider cubic functions in y that better approximate $\tanh$, constructing the polynomial constraint $p(x,y) = (x - y^3 - y)(\frac{1}{6}y^3 + y - x)$. A deadzone-style nonlinearity such as $x - \tanh(x)$ might instead consider a cubic in x, and in general we may rotate polynomial constraints through linear combinations of x and y. As another example, consider $e^x - x - 1$, an exponential with the affine term removed, which we can locally bound with quadratics in x, constructing $p(x,y) = (y - 0.3x^2)(0.7x^2 - y)$

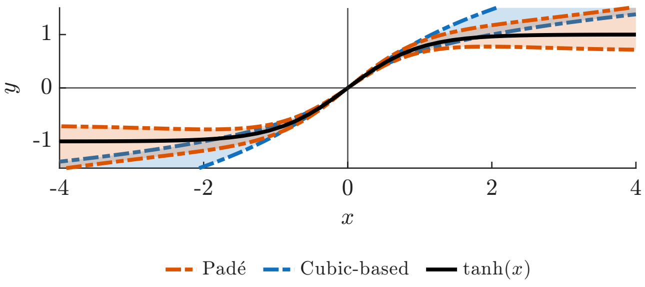
A simple method to construct a constraint $p(x,y) \geq 0$ for many functions $f(x)$ is to use the Taylor series expansion of a function: construct the polynomial constraint

$$p(x,y) = (\epsilon_1 x^k - (y - T_n(x)))(\epsilon_2 x^k + (y - T_n(x)))$$

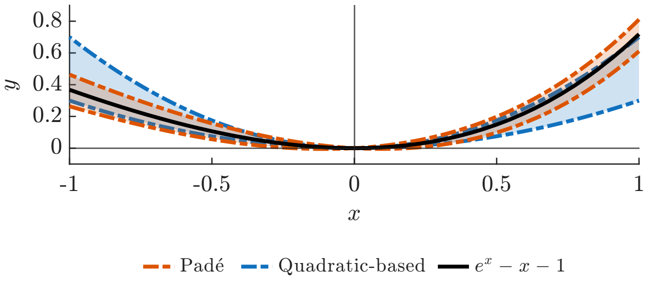
where $\epsilon_1, \epsilon_2 > 0$, $k \in \mathbb{N}$, and $T_n(x)$ is the Taylor series truncated to be degree n. The appropriateness of this technique depends on f; it does not provide good bounds on $\tanh$, but provides much better bounds on $\tanh^{-1}$, which may be converted into bounds on $\tanh$ by swapping x and y. A similar technique, typically resulting in tighter bounds for the same overall degree polynomial constraint, is to use a Padé approximant $N(x)/D(x)$ instead of the Taylor series:

$$\begin{aligned} p(x,y) =& (\epsilon_1 x^k D(x) - (yD(x) - N(x))) \\ &\cdot (\epsilon_2 x^k D(x) + (yD(x) - N(x))) \end{aligned} \tag{9}$$

[1]https://github.com/neelayjunnarkar/polynomial-constraints/



(a) Constraints on $\tanh(x)$



(b) Constraints on $e^x - x - 1$

Fig. 1: Simple local polynomial constraints on two nonlinearities. Shaded regions indicate nonnegativity. The quadratic-based constraint is a 4th degree polynomial and the cubic-based and Padé approximant-based constraints are 6th degree. The Padé approximant-based constraints are both computed using (9) with $\epsilon_1 = \epsilon_2 = 0.1$, $k = 1$, and the $[3/2]$ Padé approximant.

A. Numerical Synthesis

Although simple to construct, hand-crafted and approximation-based polynomials lack the tuning flexibility required to incorporate knowledge of the dynamical system into the constraints: there may exist a constraint that is tighter over the specific local region we are analyzing, or a constraint that is tighter in just the lower or upper bound, only one of which may be of importance given the dynamical system. To address this, we introduce the following numerical method to construct polynomial constraints, which gives flexibility in tuning the constraint. This optimization program, which we solve with Matlab's `fmincon`, may be initialized with constraints constructed via the above-mentioned methods.

We numerically solve for polynomial constraints by solving the optimization program

$$\min_{p \in \mathbb{R}[x]} \quad f_0(p) \tag{10a}$$

$$\text{s.t.} \quad p(x_i, \Delta(x_i)) \geq 0, \qquad i = 1, \ldots, N \tag{10b}$$

$$\|\text{coefficients}(p)\| = 1 \tag{10c}$$

where the objective function $f_0(p)$, described below, is defined to encourage tight constraints, (10b) ensures Δ satisfies the polynomial constraint p on a constraint evaluation set $\{x_i\}_{i=1}^{n_c}$, and (10c) constrains the scale of the polynomial constraint to ensure the coefficients aren't simply taken to 0. This last constraint leverages that polynomial constraints are invariant under positive scalings, and scaling a polynomial is equivalent to scaling its coefficients.

As a surrogate for tight transformations, we define the objective f_0 to evaluate p on a set of test points $\{x_{t,i}, y_{t,i}\}_{i=1}^{n_t}$, penalize points which evaluate nonnegative, and reward points which evaluate negative:

$$\begin{aligned} f(x,y) &= w_o(x,y)\tanh\left(s \cdot p(x,y)\right) \\ f_0(p) &= \sum_{i=1}^{n_t} f(x_{t,i}, y_{t,i}) \end{aligned} \tag{11}$$

The parameter $s > 0$ is a tunable parameter to match the scale of $\tanh$ to the values of $p(x,y)$ in the test set, and w_o is a weighting function: for example, $w_o(x,y) = 1 + e^{-x^2}$ for tighter constraints near the origin. We construct a collection of test points with the following process: first, sample points $\{x_{t,i}\}_{i=1}^{n_{tx}}$ in the domain of Δ; second, for each $x_{t,i}$, sample points $\{y_{t,i,j}\}_{j=1}^{n_{ty}}$ in a neighborhood (of configurable shape) of $\Delta(x_{t,i})$; and finally, construct test points $\bigcup_{i=1}^{n_{tx}}\{(x_{t,i}, y_{t,i,j}) \mid j = 1, \ldots, n_{ty}\}$.

While (10) ensures construction of a polynomial p such that Δ satisfies p and penalizes nonnegative $p(x,y)$ near the graph of Δ, it does not constraint or penalize points far from the graph. Thus, constraints synthesized with this program often require an additional polynomial constraint that y lie in an interval near the origin, with bounds depending on bounds on Δ in the region being verified. Additionally, because $p(x, \Delta(x)) \geq 0$ is only ensured at a finite list of evaluation points, we verify the constraint p after synthesis by using root finders to search for roots of $p(x, \Delta(x))$ and ensure there are no roots in the desired local region.

B. Transformations of Polynomial Constraints

In this section we present some results on ensuring the synthesized polynomial constraints are not unnecessarily conservative, and a method for constructing IPCs that leverages the existing body of IQCs.

Theorem 2: Consider Δ and $\tilde{\Delta}$, both operators from $\mathcal{L}_{2e}^{n_v}$ to $\mathcal{L}_{2e}^{n_w}$. Suppose there exists a polynomial dynamical system $H : \mathcal{L}_2^{n_v} \times \mathcal{L}_2^{n_w} \to \mathcal{L}_2^{n_v} \times \mathcal{L}_2^{n_w}$ such that the restriction of H to the graph of Δ, $h = H|_{\text{graph } \Delta}$, is an invertible operator from the graph of Δ to the graph of $\tilde{\Delta}$. Then $\tilde{\Delta}$ satisfies the IPC defined by Ψ if and only if Δ satisfies the IPC defined by $\Psi \circ h$.

Proof: If $\tilde{\Delta}$ satisfies Ψ, $T \geq 0$, and $v \in \mathcal{L}_2^{n_v}$, then $\int_0^T (\Psi \circ h)(v, \Delta(v))(t)dt = \int_0^T \Psi(\tilde{v}, \tilde{\Delta}(\tilde{v}))(t)dt \geq 0$, so Δ satisfies $\Psi \circ h$. If Δ satisfies $\Psi \circ h$, $T \geq 0$, and $\tilde{v} \in \mathcal{L}_2^{n_v}$, then $\int_0^T \Psi(\tilde{v}, \tilde{\Delta}(\tilde{v}))(t)dt = \int_0^T (\Psi \circ h \circ h^{-1})(\tilde{v}, \tilde{\Delta}(\tilde{v}))(t)dt = \int_0^T (\Psi \circ h)(v, \Delta(v))(t)dt \geq 0$, so $\tilde{\Delta}$ satisfies Ψ. ■

Note that H need not be invertible and h^{-1} need not be polynomial. To construct transformations h, the following properties are useful.

Lemma 1: Let Δ, $\tilde{\Delta}$, and h be defined as in Theorem 2. Let h_1 and h_2 be the component functions of h so that $h(v, \Delta(v)) = (h_1(v, \Delta(v)), h_2(v, \Delta(v))$. Then h is invertible if and only if h_1 is invertible.

Proof: Define $g : \tilde{v} \mapsto (\tilde{v}, \tilde{\Delta}(\tilde{v}))$. Note that g is invertible and g^{-1} is the map projecting onto the first element. We can define h in terms of h_1 by $h = g \circ h_1$ and h_1 in terms of h by $h_1 = g^{-1} \circ h$. If h is invertible, then $h_1^{-1} = h^{-1} \circ g$. If h_1 is invertible, then $h^{-1} = h_1^{-1} \circ g^{-1}$. ■

Corollary 1: $\tilde{\Delta} = h_2 \circ h_1^{-1}$.

Corollary 2: h is invertible if and only if $v \mapsto h_1(v, \Delta(v))$ is invertible.

The above results on equivalence of IPCs enables the construction of an IPC on one operator by polynomial transformation of an existing IPC on another operator. However, showing two operators are related by a polynomial transformation may in general be difficult. We instead propose starting with an operator Δ for which an IPC is desired, and then constructing h so that the operator $h_2 \circ h_1^{-1}$, which we define to be $\tilde{\Delta}$, satisfies a specified class of IQCs. For example, constructing h so that $h_2 \circ h_1^{-1}$ is sector-bounded or slope-restricted by imposing constraints on the coefficients of h and therefore satisfying sector constraints and Zames-Falb multipliers respectively. The class of IQCs on $h_2 \circ h_1^{-1}$ is then transformed into a class of IPCs on Δ. This procedure yields IPCs parameterized by the IQC multipliers, and enables the use of polynomial transformations to create less conservative constraints. Conservativeness can be lowered further by parameterizing the class of transformations such that $h_2 \circ h_1^{-1}$ satisfies the desired IQC instead of picking a single transformation h.

As an illustration, we demonstrate a transformation between $\tanh$ and a function which satisfies the sector bound $[-1, 1]$ tightly, and use this to construct a constraint on $\tanh$, recovering the familiar $[0, 1]$ sector bound.

Example 1: Let Δ be $\tanh$ and consider the linear transformation $H \in \mathbb{R}^{2\times 2}$. A sufficient condition for invertibility of h is invertibility of H. Now we use the above procedure to construct a linear transformation H such that $\tilde{\Delta}$ satisfies the sector bound $[-1, 1]$. A quadratic constraint for this bound is $M = \operatorname{diag}(1, -1)$. Let H be as follows.

$$H = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Then Δ satisfies $H^\top M H$ if and only if $(ax + b\tanh(x))^2 - (cx + d\tanh(x))^2 \geq 0$ for all x. To select an H that constructs a tight bound, we use the slope restrictions on $\tanh$ to show $(ad - bc)/(b^2 - d^2) = 1/2$. One solution for this is $a = 1, b = -1/2, c = 1, d = -3/2$. This corresponds to the following H and polynomial constraint on $\tanh$:

$$H = \begin{bmatrix} 1 & -\frac{1}{2} \\ 1 & -\frac{3}{2} \end{bmatrix}, \quad H^\top M H = \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix}.$$

This is the usual sector bound in $[0, 1]$ as expected.

Applying a simple case of this transformation method to (10), we search for h which transforms a sector constraint into a polynomial constraint on Δ. Specifically, we search for an invertible transformation h that maps the graph of Δ to a function that satisfies the quadratic constraint defined by $\left[\begin{smallmatrix} 0 & 0.5 \\ 0.5 & -1 \end{smallmatrix}\right]$. This implies that Δ satisfies $h(x, \Delta(x))^\top M h(x, \Delta(x)) \geq 0$ for all x, which simplifies to $h_2(x, \Delta(x))\big(h_1(x, \Delta(x)) - h_2(x, \Delta(x))\big) \geq 0$ for all x. We define $p(x,y) = h_2(x,y)\big(h_1(x,y) - h_2(x,y)\big)$. A sufficient condition for invertibility of h is that the derivative of $x \mapsto h_1(x, \Delta(x))$ be strictly increasing. Thus, we can numerically solve for a transformation h by modifying (10) to enforce that $p = h_2 \cdot (h_1 - h_2)$ and add the constraint $h_1'(x_i, \Delta(x_i)) > 0$.

V. Examples Computing Regions of Attraction

A. Triple Integrator

Consider the following triple-integrator system with a saturation on the control input, and a state-feedback controller designed using LQR with weights $Q = I$ and $R = 1$.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= \tanh(u) = \tanh\left(\underbrace{\begin{bmatrix} -1 & -2.4142 & -2.4142 \end{bmatrix}}_{K} x\right) \end{aligned} \quad (12)$$

We take Δ to be the difference between the dynamics and the linearization: $w = \Delta(v) = \tanh(v) - v$ where $v = Kx$. This gives a model in the form of (4):

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= Kx + \Delta(Kx) \\ \Delta(v) &= \tanh(v) - v. \end{aligned} \quad (13)$$

We use the method from Section IV with weighting function $w_o(v, w) = 1 + \exp(-w^2)$ to construct a 6th degree local polynomial constraint on Δ. The test set of points is designed to penalize high $|w|$ because the linearization is stable. This constraint, p_{num}, is valid for $v \in [-4, 4]$ with the additional constraint that $w \in [-3, 3]$. We find benefit in using p_{num} in combination with a $[3/2]$ Padé approximant-based constraint, $p_{\text{padé}}$, with $k = 1$, $\epsilon_1 = 0.01$, and $\epsilon_2 = 0.03$, which is valid for the same v as p_{num}. This constraint is generally looser for $|w| > |\tanh(v) - v|$ but tighter otherwise, and is incorporated into the SOS programs with its own s-certificate. Using a list of constraints is similar to searching over constraints in the conic hull of the list. The coefficients of these polynomials are available in the associated code repository. We construct local sector constraints for Δ and find the largest ROA with the sector bound valid for $v \in [-2.1, 2.1]$. The sector constraint is $p_{\text{sector}}(v, w) = -w(w + 0.5379v)$. The constraints p_{sector}, p_{num}, and $p_{\text{padé}}$ are plotted in Figure 2.

We then use Algorithm 1 to synthesize ROAs. We add additional constraints to the SOS programs to ensure that the verified sub-level sets $V(x) \leq c$ are subsets of the region where the constraints are valid: $(1 + s_d(x, w))q(x, w) - s_n(x, w)(c - V(x)), s_d(x, w), s_n(x, w) \in \Sigma[(x, w)]$ where $q(x, w) \geq 0$ represents the region where the constraints are valid. To compute the ROA estimate using the sector constraint, we compute 50 iterations of Algorithm 1 with $n_V = 2, n_{\text{total}} = 6$, and an initial Lyapunov function from the certificate of the LQR controller. To compute the ROA estimate using the polynomial constraints, we initialize with the sector-based estimate after 30 iterations, and take 20 additional iterations of $n_V = 2, n_{\text{total}} = 6$.

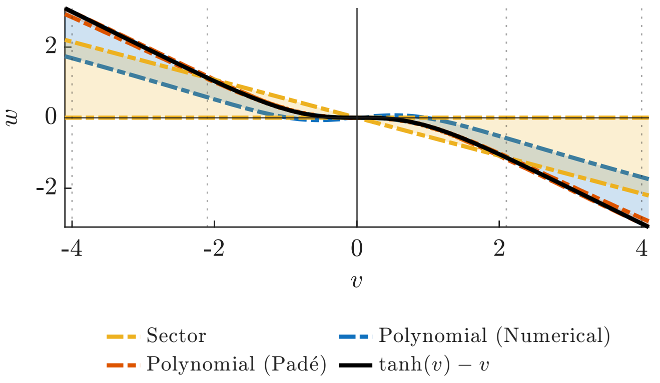


Fig. 2: Triple Integrator: Local constraints on $\tanh(v) - v$. Shaded regions indicate nonnegativity. The numerically-constructed polynomial constraint requires the additional constraint that $w \in [-3, 3]$ for the constraint to be valid for $v \in [-4, 4]$ to exclude regions where $|w| \gg |v|$. The Padé approximation-based constraint is valid for $v \in [-4, 4]$, and the sector constraint is valid for $[-2.1, 2.1]$. The polynomial constraints give tighter bounds on $\tanh(v) - v$ than does the sector constraint, while also being valid for a wider region.

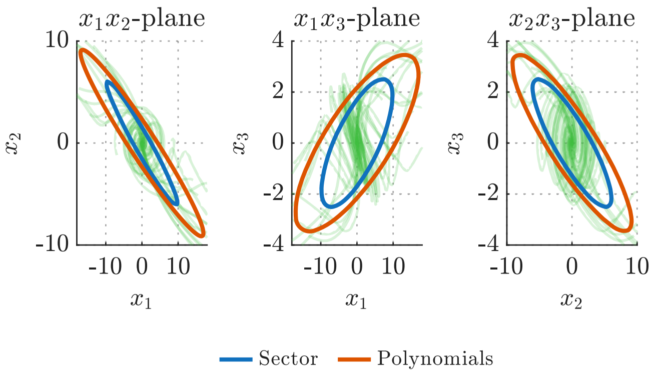


Fig. 3: Triple Integrator: Regions of attraction, synthesized using different constraints on Δ, projected onto planes. The ROA computed using the polynomial constraints has a larger volume. Trajectories converging to the origin plotted in green.

Figure 3 shows the projections of the resulting regions of attraction onto the x_1x_2, x_1x_3, x_2x_3 planes and Table I compares the volumes of the ROAs and the computation times (for the polynomial constraints, the time to compute 30 iterations with p_{sector} is included). All programs were run on an Intel i5-6600K processor. The ROA computed using the polynomial constraints is 3.38 times larger.

TABLE I: Triple Integrator

Constraint Type	**Volume (Relative)**	**Computation Time (s)**
Sector	153.70 (1.00x)	265
Polynomials	519.71 (3.38x)	309

B. System with Exponential

Consider the following system that involves an exponential, which is non-polynomial and asymmetric.

$$\begin{aligned} \dot{x}_1 &= -x_1 - x_2 + e^{x_1} - 1 \\ \dot{x}_2 &= x_1 - x_2 \end{aligned} \quad (14)$$

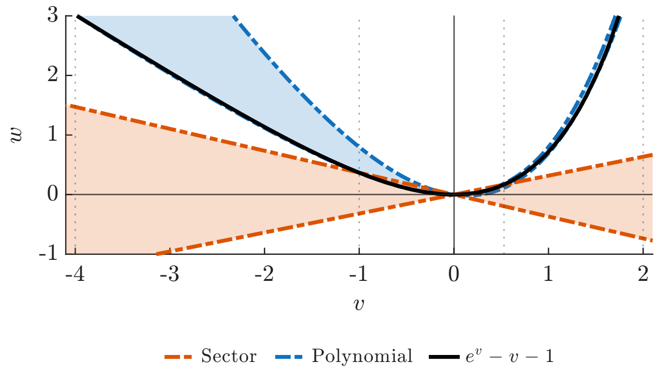


Fig. 4: System with Exponential: Local constraints on $e^v - v - 1$. Shaded regions indicate nonnegativity. The sector constraint is valid for $v \in [-1, 0.53]$. The polynomial constraint is valid for $v \in [-4, 2]$ with the additional constraint that $w \in [0, 3.1]$ to exclude regions where $|w| \gg |v|$. The polynomial constraints give tighter bounds than does the sector constraint, in particular in the upper bound where $v > 0$, which is important for stability analysis of (14).

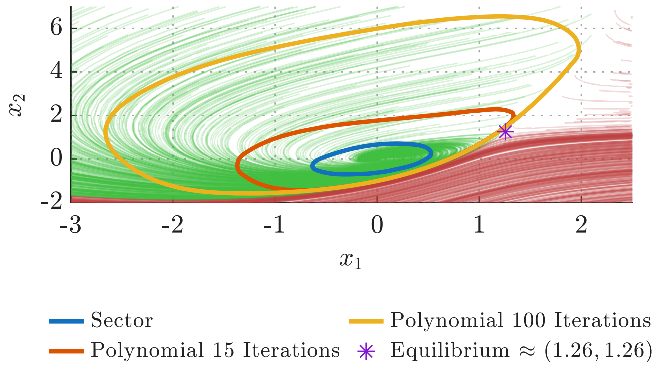


Fig. 5: System with Exponential: Inner estimates of region of attraction of origin, computed using two different characterizations of Δ. The green lines denote trajectories converging to the origin and the red lines denote trajectories diverging from the origin. The ROAs computed using the polynomial constraint have larger areas, with additional computation leading to larger ROAs.

In addition to the origin, this system has an equilibrium at approximately $(1.26, 1.26)$. We examine the region of attraction of the origin. As before, we take Δ to be the difference between the dynamics and the linearization, so $\Delta(x_1) = e^{x_1} - x_1 - 1$, and we analyze the ROA of the origin of the following system.

$$\begin{aligned} \dot{x}_1 &= -x_2 + \Delta(x_1) \\ \dot{x}_2 &= x_1 - x_2 \\ \Delta(x_1) &= e^{x_1} - x_1 - 1 \end{aligned} \tag{15}$$

Using the method from Section IV and the objective weighting function $w_o(v, w) = 1$ if $v \geq 0$ and $w_o(v, w) = 0.5$ if $v < 0$, we construct a 6th degree local polynomial constraint on $w = \Delta(v)$, which is valid for $v \in [-4, 2]$ with the additional constraint that $w \in [0, 3.1]$. The weighting function is designed to be tight where $v > 0$. Coefficients are available in the associated code repository. We compare with the local sector condition $p_{\text{sector}}(v, w) = (0.318v - w)(0.368v + w)$, which is valid for $v \in [-1, 0.53]$. The upper-bound of 0.53 on v was the largest bound we found for which the ROA synthesis problem was feasible. The two constraints are visualized in Figure 4.

We synthesize ROAs using Algorithm 1, with an initial quadratic Lyapunov function constructed by solving the Lyapunov equation for the linearized system with $Q = I$. When using the sector constraint, we compute 10 iterations with $n_V = 2, n_{\text{total}} = 6$ followed by 5 iterations with $n_V = 6, n_{\text{total}} = 10$. When using the sector constraint, we find no benefit in further iterations. When using the polynomial constraint, we compute 5 iterations of $n_V = 2, n_{\text{total}} = 6$, followed by 5 iterations of $n_V = 4, n_{\text{total}} = 8$, and 90 iterations of $n_V = 6, n_{\text{total}} = 10$. We compare the ROA computed using the polynomial constraint after both 15 total iterations and after 100 total iterations with the ROA computed using the sector constraint.

Figure 5 plots the ROAs and Table II compares the volumes and computation times. The ROAs computed using the polynomial constraint are significantly larger than the one computed using the sector constraint. In addition, they closely follow the boundary of the region of attraction, unlike the ROA computed using the sector constraint.

TABLE II: System with Exponential

Constraint Type	**Volume (Relative)**	**Computation Time (s)**
Sector	1.16 (1.00x)	67
Polynomial 15 Iterations	5.99 (5.16x)	190
Polynomial, 100 Iterations	25.6 (16.0x)	1593

References

[1] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control* (Interdisciplinary Applied Mathematics). New York, NY: Springer, 1999, vol. 10.

[2] A. Megretski and A. Rantzer, “System Analysis Via Integral Quadratic Constraints,” *IEEE Transactions on Automatic Control*, vol. 42, no. 6, pp. 819–830, Jun. 1997.

[3] A. Papachristodoulou and S. Prajna, “A tutorial on sum of squares techniques for systems analysis,” in *Proceedings of the American Control Conference.*, Jun. 2005, 2686–2700 vol. 4.

[4] A. Iannelli, P. Seiler, and A. Marcos, “Estimating the region of attraction of uncertain systems with integral quadratic constraints,” in *2018 IEEE Conference on Decision and Control*, Miami Beach, FL: IEEE, Dec. 2018, pp. 3922–3927.

[5] M. Newton and A. Papachristodoulou, “Neural network verification using polynomial optimisation,” in *60th IEEE Conference on Decision and Control*, ISSN: 2576-2370, Dec. 2021, pp. 5092–5097.

[6] A. Papachristodoulou and S. Prajna, “Analysis of non-polynomial systems using the sum of squares decomposition,” in *Positive Polynomials in Control*, D. Henrion and A. Garulli, Eds., vol. 312, Series Title: Lecture Notes in Control and Information Sciences, Berlin, Heidelberg: Springer Berlin Heidelberg, Aug. 4, 2005, pp. 23–43.

[7] U. Topcu, A. Packard, P. Seiler, and T. Wheeler, “Stability region analysis using simulations and sum-of-squares programming,” in *American Control Conference*, 2007, pp. 6009–6014.

[8] G. Chou and R. Tedrake, “Synthesizing stable reduced-order visuomotor policies for nonlinear systems via sums-of-squares optimization,” in *62nd IEEE Conference on Decision and Control*, ISSN: 2576-2370, Dec. 2023, pp. 624–631.

[9] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. A. Parrilo, *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, 2013.