Region of Attraction Estimation for the GTM Design Model

Peter Seiler, Gary Balas, and Andrew Packard

peter.j.seiler@gmail.com, balas@musyn.com

September 29, 2009

1 Overview

Region of attraction analysis is applied to NASA's Generic Transport Model (GTM), a remote-controlled 5.5 percent scale commercial aircraft. A four-state polynomial approximation of the GTM longitudinal dynamics is developed and used for the region of attraction analysis. The polynomial model is used to investigated the stability properties of this nonlinear model around an operating point by estimating of its region of attraction.

In this report we first describe the computational method for estimating the region of attraction of a polynomial system. This method relies on connections between sum-of-squares polynomials, semidefinite matrices, and convex optimization. We briefly describe how to approximate the GTM dynamics with a polynomial model. Finally, the region of attraction of the polynomial GTM model is estimated around a trim point. The region of attraction analysis described in this report is still an active area of research. We anticipate future versions of ROME including this capability for off-line analysis of the aircraft flight control system.

2 Region of Attraction Estimation

Linear analysis is a local analysis which is only valid near the operating point. Many systems, including the GTM aircraft, are more accurately described by nonlinear dynamical systems. For linear systems asymptotic stability of an equilibrium point is a global property. In other words, if the equilibrium point is asymptotically stable then the state trajectory will converge back to the equilibrium when starting from any initial condition. A key difference with nonlinear systems is that equilibrium points may only be locally asymptotically stable. Khalil [1] and Vidyasagar [2] provide good introductory discussions of this issue. The region-of-attraction (ROA) of an asymptotically stable equilibrium point is the set of initial conditions whose state trajectories converge back to the equilibrium [1]. If the ROA is small, then a disturbance can easily drive the system out of the ROA and the system will then fail to come back to the stable equilibrium point. Thus the size of the ROA is a measure of the stability properties of a nonlinear system around an equilibrium point. This provides the motivation to estimate the region-of-attraction (ROA) for an equilibrium point of a nonlinear system. In this section we describe our technical approach to estimating the ROA.

Consider autonomous nonlinear dynamical systems of the form:

$$\dot{x}(t) = f(x(t)),\ x(0) = x_0 \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state vector at time t. We assume that $x = 0$ is a locally asymptotically stable equilibrium point. Formally, the ROA is defined as:

$$R_0 = \left\{ x_0 \in \mathbb{R}^n :\ \text{If } x(0) = x_0 \text{ then } \lim_{t\to\infty} x(t) = 0 \right\} \tag{2}$$

Computing the exact ROA for nonlinear dynamical systems is, in general, a very difficult problem. There has been significant research devoted to estimating invariant subsets of the ROA [3–11]. Our approach is to restrict the search to ellipsoidal approximations of the ROA. The ellipsoid is specified by $\{x_0^T N x_0 \leq \beta\}$

where $N = N^T > 0$ is a user-specified matrix which determines the shape of the ellipsoid. Given N, the problem is to find the largest ellipsoid contained in the ROA:

$$\begin{aligned} \beta^* = &\max \beta \\ &\text{subject to: } \{x_0^T N x_0 \leq \beta\} \subset R_0 \end{aligned} \tag{3}$$

Determining the best ellipsoidal approximation to the ROA is still a challenging computational problem. Instead we will attempt to solve for upper and lower bounds satisfying $\underline{\beta} \leq \beta^* \leq \bar{\beta}$. If these upper and lower bounds are close then we have approximately solved the best ellipsoidal approximation problem given in Equation 3.

The upper bounds are computed via a search for initial conditions leading to divergent trajectories. If $\lim_{t\to\infty} x(t) = +\infty$ when starting from $x(0) = x_{0,div}$ then $x_{0,div} \notin R_0$. If we define $\bar{\beta}_{div} := x_{0,div}^T N x_{0,div}$ then $\{x_0^T N x_0 \leq \bar{\beta}_{div}\} \not\subset R_0$. Thus $\bar{\beta}_{div}$ is a true upper bound on β^* and $\{x_0^T N x_0 \leq \bar{\beta}_{div}\}$ is an outer approximation of the best ellipsoidal approximation to the ROA. An exhaustive Monte Carlo search is used to find the tightest possible upper bound on β^*. Specifically, initial conditions are randomly chosen starting on the boundary of a large ellipsoid: Choose x_0 satisfying $x_0^T N x_0 = g$ where g is sufficiently large that $g \gg \beta^*$. If a divergent trajectory is found, then an upper bound $\bar{\beta}_{div}$ on β^* is computed by finding the minimum value of p along the divergent trajectory. The state vector x_{div} that achieves $p(x_{div}) = \bar{\beta}_{div}$ is stored and g is decreased by a small factor, e.g. $g = 0.995\bar{\beta}_{div}$. The search continues until a maximum number of simulations is reached. $\bar{\beta}_{MC}$ will denote the smallest upper bound computed with this Monte Carlo search.

The lower bounds are computed using Lyapunov functions and recent results connecting sums-of-squares polynomials to semidefinite programming. To compute these bounds we need to further assume that the vector field $f(x)$ in the system dynamics (Equation 1) is a polynomial function. The computational algorithm is briefly described in the following paragraphes. Additional details are provided in literature references [12–19]. Lemma 1 is the main Lyapunov theorem used to compute lower bounds on β^*. This specific lemma is proved by Tan [16] but very similar results are given in textbooks, e.g. by Vidyasagar [2].

Lemma 1 *If there exists a continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ such that:*

- $V(0) = 0$ *and* $V(x) > 0$ *for all* $x \neq 0$
- $\Omega_\gamma := \{x \in \mathbb{R}^n \ : \ V(x) \leq \gamma\}$ *is bounded.*
- $\Omega_\gamma \subset \{x \in \mathbb{R}^n \ : \ \nabla V(x) f(x) < 0\}$

then for all $x \in \Omega_\gamma$, the solution of Equation 1 exists, satisfies $x(t) \in \Omega_\gamma$ for all $t \geq 0$, and $\Omega_\gamma \subset R_0$.

A function V satisfying the conditions in Lemma 1 is a Lyapunov function and Ω_γ provides an estimate of the region of attraction. Any subset of Ω_γ is also inside the ROA. In principle we can compute a lower bound on β^* by solving the maximization:

$$\begin{aligned} \underline{\beta} := &\max \beta \\ &\text{subject to: } \{x_0^T N x_0 \leq \beta\} \subset \Omega_\gamma \end{aligned} \tag{4}$$

Our computational algorithm replaces the set containment constraint with a sufficient condition involving non-negative functions:

$$\begin{aligned} \underline{\beta} := &\max \beta \\ &\text{subject to: } s(x) \geq 0 \ \forall x \\ &\qquad -(\beta - x^T N x) s(x) + (\gamma - V(x)) \geq 0 \ \forall x \end{aligned} \tag{5}$$

The function $s(x)$, known as a "multiplier" function, is a decision variable of the optimization, i.e. it will be found as part of the optimization. It is straight-forward to show that the two non-negativity conditions in Optimization 5 are a sufficient condition for the set containment condition in Optimization 4. If both $V(x)$ and $s(x)$ are restricted to be polynomials then both constraints involve the non-negativity of polynomial functions. A sufficient condition for a generic multi-variate polynomial $p(x)$ to be non-negative is the existence of polynomials $\{g_1, \ldots, g_n\}$ such that $p = g_1^2 + \cdots + g_n^2$. A polynomial which can be decomposed in this

way is appropriately called a sum-of-squares (SOS). Finally, if we replace the non-negativity conditions in Optimization 5 with SOS constraints, then we arrive at an SOS optimization problem:

$$\begin{aligned} \underline{\beta} := \max \beta & \\ \text{subject to: } & s(x) \text{ is } SOS \\ & -(\beta - x^T N x)s(x) + (\gamma - V(x)) \text{ is } SOS \end{aligned} \tag{6}$$

It is sufficient to note that there are connections between SOS polynomials and semidefinite matrices. Moreover, optimization problems involving SOS constraints can be converted and solved as a semidefinite programming optimization. There is freely available software to set up and solve these problems [20–22]. The connections between SOS polynomials, semidefinite matrices and optimization are described in more detail in Appendix A.

The choice of the Lyapunov function which satisfies the conditions of Lemma 1 has a significant impact on the quality of the lower bound, $\underline{\beta}$. The simplest method is to find $P > 0$ which solves the Lyapunov equation $A^T P + PA = -I$. $A := \left.\frac{\partial f}{\partial x}\right|_{x=0}$ is the linearization of the dynamics about the origin. $V_{LIN}(x) := x^T P x$ is a quadratic Lyapunov function since $x = 0$ is assumed to be a locally asymptotically stable equilibrium point. Thus we can solve for the largest value of γ satisfying the set containment condition in Lemma 1: $\Omega_\gamma \subset \{x \in \mathbb{R}^n \ : \ \nabla V_{LIN}(x) f(x) < 0\}$. This problem can also be turned into an SOS optimization with "multiplier" functions as decision variables. $\underline{\beta}_{LIN}$ will denote the lower bound obtained using the quadratic Lyapunov function obtained from linearized analysis.

Unfortunately, $\underline{\beta}_{LIN}$ is typically orders of magnitude smaller than the upper bound $\bar{\beta}_{MC}$. Several methods to compute better Lyapunov functions exist, including V-s iterations [12–15], bilinear optimization [16], and use of simulation data [17–19]. We briefly describe the V-s iteration starting from V_{LIN}. In the first step of the iteration, the multiplier functions and $\underline{\beta}_{LIN}$ are computed. Then the multiplier functions are held fixed and the Lyapunov function candidate becomes the decision variable. The SOS constraints of this new problem are those which arise from the two set containment conditions: $\Omega_\gamma \subset \{x \in \mathbb{R}^n \ : \ \nabla V_{LIN}(x) f(x) < 0\}$ and $\{x_0^T N x_0 \leq \beta\} \subset \Omega_\gamma$. In the next iteration, the multiplier functions are again decision variables and a lower bound is computed using the new Lyapunov function computed in the previous iteration. The V-s iteration continues as long as the lower bound continues to increase. In this iteration, Lyapunov functions are allowed to be of higher polynomial degree. Increasing the degree of the Lyapunov function will improve the lower bound at the expense of computational complexity. The computational time grows very rapidly with the degree of the Lyapunov function and so degree 4 candidates are about the maximum which can be used for problems like the GTM analysis. $\underline{\beta}_2$ and $\underline{\beta}_4$ will denote the best lower bounds computed with the V-s iteration for quadratic and quartic Lyapunov functions. Nonlinear analysis code based on SOS optimization (including demos of the V-S iteration) can be downloaded from:

`http://jagger.me.berkeley.edu/software/acc09/`

3 GTM Polynomial Modeling

A four-state nonlinear model of the longitudinal aircraft dynamics is given by:

$$\dot{U} = \frac{1}{m}\left(F_x \cos\alpha + F_z \sin\alpha\right) \tag{7}$$

$$\dot{\alpha} = \frac{1}{mU}\left(-F_x \sin\alpha + F_z \cos\alpha\right) + q \tag{8}$$

$$\dot{q} = \frac{M_y}{I_{yy}} \tag{9}$$

$$\dot{\theta} = q \tag{10}$$

where U is the equivalent airspeed (ft/sec), α is the angle of attack (rad), q is the pitch rate (rad/sec), and θ is the pitch angle (rad). The forces and moments are given by:

$$\begin{aligned} F_x &= \bar{q} S_{ref} C_x(\alpha, q, \delta_e) + 2T_X(\delta_{th}) - mg\sin\theta \\ F_z &= \bar{q} S_{ref} C_z(\alpha, q, \delta_e) + 2T_Z(\delta_{th}) + mg\cos\theta \\ M_y &= \bar{q} S_{ref} \bar{c} C_m(\alpha, q, \delta_e) + 2T_X(\delta_{th}) \Delta Z_{ENG} \end{aligned}$$

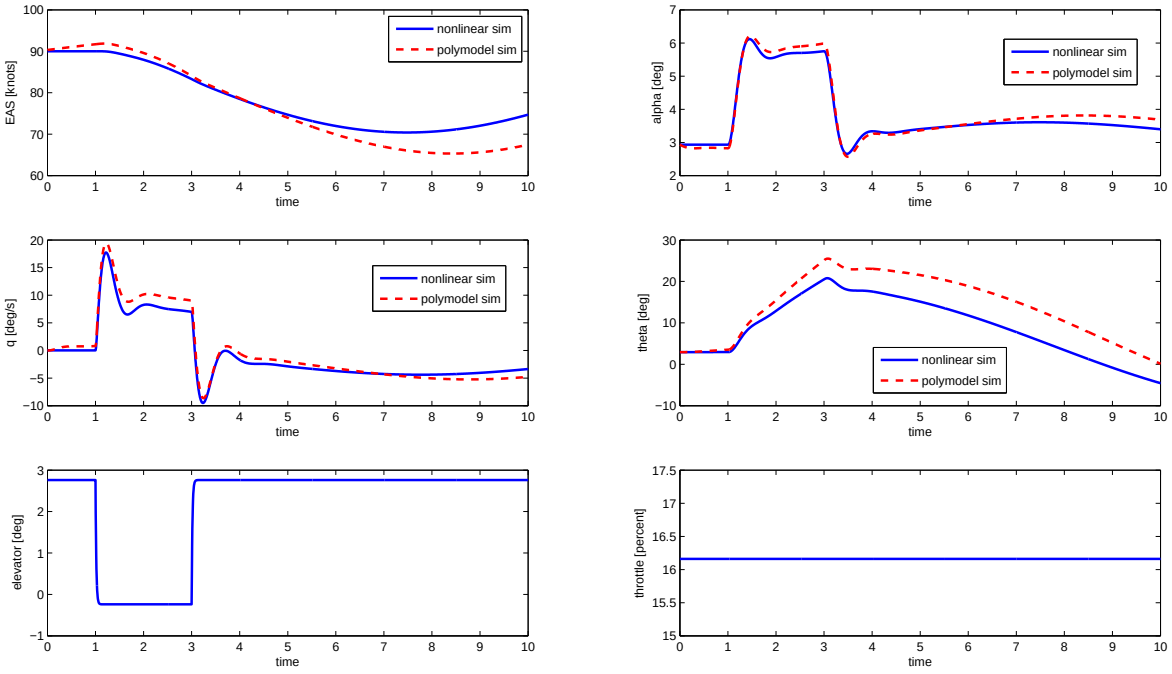

Figure 1: Comparison of Polynomial and Full Nonlinear Simulation Results

where δ_e is the elevator deflection (rad) and δ_{th} is the normalized thrust. T_X and T_Z are the engine thrust along the body longitudinal and vertical axes. These are computed by transforming the engine thrust $T(\delta_{th})$ from engine to body coordinates. The GTM has two engines and this accounts for the factor of two in the engine thrust terms (T_X, and T_Z). The engine thrust (T) and the aerodynamic coefficients (C_x, C_z, and C_m) are provided as look-up tables. Least squares approximation is used to approximate these look-up tables by polynomial functions. Moreover, least squares is used to create polynomial approximations of trigonometric functions and $\frac{1}{U}$. $\frac{1}{U}$ appears in both the aerodynamic coefficient models and in the state dynamic equations. The polynomial approximation of $\frac{1}{U}$ was only used in the state equations. A reasonable approximation was obtained by using a constant equivalent airspeed for the $\frac{1}{U}$ dependence in the aero coefficient models.

Appendix B provides the Matlab code to construct the four-state, fifth-order polynomial model of the GTM longitudinal dynamics. All model constants are provided in this appendix. The code requires the multipoly toolbox for maninpulating polynomial objects which can be obtained from

`http://jagger.me.berkeley.edu/software/acc09/`.

The polynomial model is simulated with an elevator pulse input. The model is trimmed at straight/level flight with an equivalent airspeed of 90 knots (=151.9 ft/sec). A two-second elevator pulse was applied to the GTM longitudinal axis starting at t=1 second of the simulation. Figure 1 compares the polynomial model response with the results from the full, 4-state nonlinear GTM model (with table look-ups). The results from the polynomial model provide a good approximation of those from the full nonlinear model.

4 GTM ROA Estimation

The region of attraction of the open-loop GTM polynomial model is estimated about the trim condition EAS = 150 ft/sec and steady/level flight (flight path angle = 0 deg). The state and input for this trim condition

are:

$$\begin{aligned} U &= 150 \text{ ft/sec} \\ \alpha &= 0.0458 \text{ rads} = 2.62 \text{ degs} \\ q &= 0 \text{ rads/sec} = 0 \text{ degs/sec} \\ \theta &= 0.0458 \text{ rads} = 2.62 \text{ degs} \\ \delta_{thr} &= 0.0859 \text{ (Normalized)} \\ \delta_e &= 0.0463 \text{ rads} = 2.66 \text{ degs} \end{aligned}$$

To improve the numerical conditioning, the states are shifted and scaled using the scaling matrix D: $z := D^{-1}(x - x_{trim})$ where:

$$D := diag(50\text{ft/sec}, 20\text{deg}, 50\text{deg/sec}, 20\text{deg})$$

The shape function is defined to be $p(z) := z^T z$. This is an ellipsoid in the original coordinates with radii along each coordinate given by the entries of D. All results were generated on a 2.66 GHz Intel Core 2 Quad processor.

First, a Monte Carlo search is run to find the smallest value of $p(z)$ resulting in a divergent trajectory. After 10000 simulations, the smallest value of $p(z)$ was $\bar{\beta}_{MC} = 3.13$. $\bar{\beta}_{MC}$ is an upper bound on the largest ellipsoid in the region of attraction. It took 1721 sec $\approx$ 29min to run these 10000 simulations. The initial condition leading to a divergent trajectory was:

$$\begin{aligned} U &= 186.12 \text{ ft/sec} \\ \alpha &= -0.41 \text{ rads} = -23.48 \text{ degs} \\ q &= 0.008 \text{ rads/sec} = 0.45 \text{ degs/sec} \\ \theta &= 0.38 \text{ rads} = 21.66 \text{ degs} \end{aligned}$$

The divergent trajectory starting from this initial condition is shown in Figure 2. The equivalent goes to zero around $t = 6$ sec and so the polynomial model is not valid for the entire simulation.

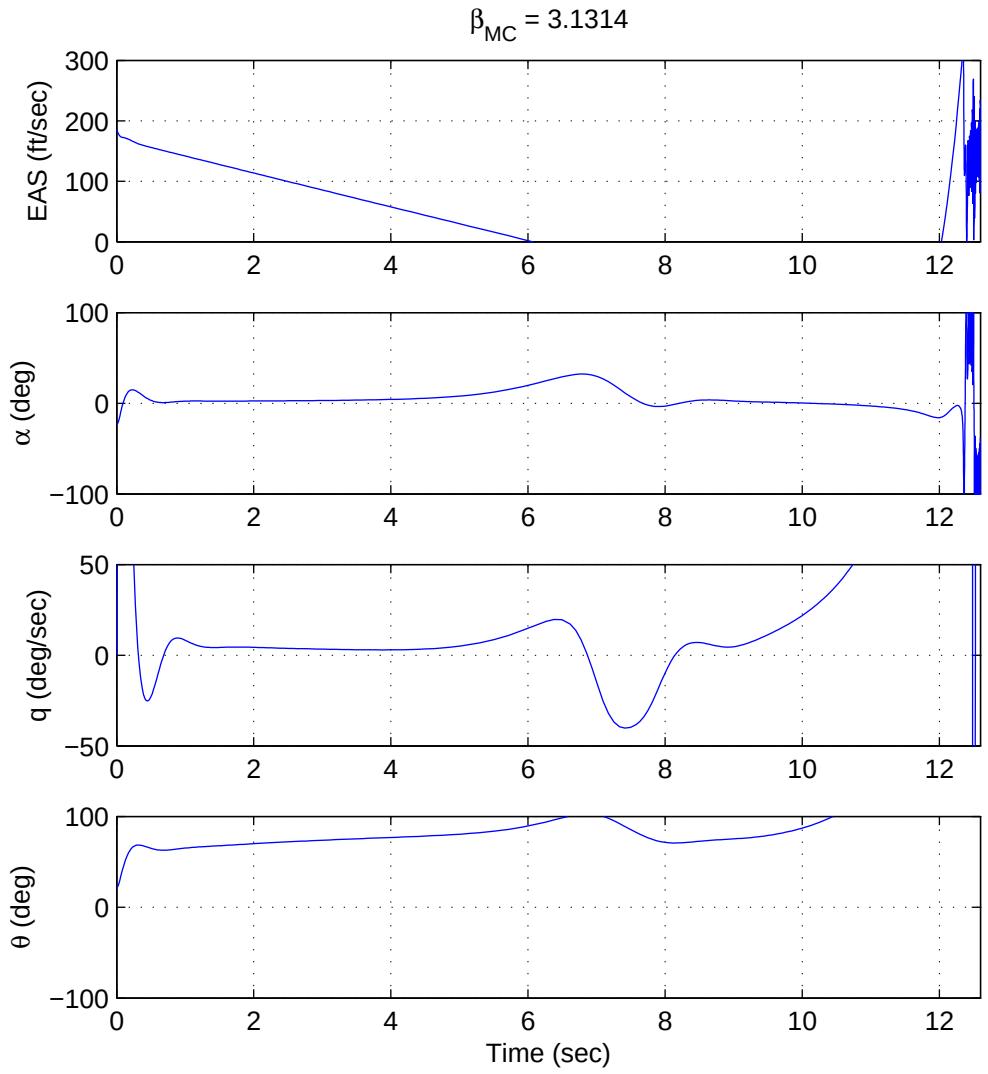

Figure 2: Divergent Trajectory from Monte Carlo Simulations

Next, SOS optimization is used to find the largest value of $p(z)$ that is provably in the region of attraction. The provable region using the quadratic Lyapunov function from linear analysis was only $\underline{\beta}_{LIN} = 1.89 \times 10^{-5}$. This took 20.9 sec. Running thirty steps of the V-s iteration with quadratic Lyapunov functions increased the provable region to $\underline{\beta}_2 = 0.21$. This took 362.9 sec. Running seventy steps of the V-s iteration with quartic Lyapunov functions increased the provable region to $\underline{\beta}_4 = 1.86$. This took 7074.8 sec $\approx$ 117.9 min. The progress of the V-s iteration with quadratic and quartic Lyapunov functions is shown in Figure 3 (black curves). This figure also shows the progress of Monte Carlo search for divergent trajectory as a function of the number of simulations (red curve). β_{MC} decreased rapidly in the first 500 simulations and then made only small progress thereafter. No progress in the optimization is made after 5,2223 simulations.

The inner/outer bounds on the region of attraction can be visualized by plotting slices of the ellipsoid $p(z) = \beta$. Figure 4 shows a slice in the EAS/α plane. All curves are drawn in the original coordinates with respect to the trim condition. The red curve is a slice of the ellipsoid $p(z) = \beta_{MC}$. The region of attraction cannot contain all of this ellipsoid since we found a divergent trajectory on the surface of this ellipsoid. The black and blue curves are slices of the ellipsoids $p(z) = \underline{\beta}_2$ and $p(z) = \underline{\beta}_4$, respectively. Every initial condition of these ellipsoids was proven, via the SOS methods, to be in the region of attraction. Increasing the degree of the Lyapunov function enables the SOS optimization to prove a larger estimate of the region of attraction. It is possible that a degree 6 Lyapunov function could prove an even larger region but this would require significantly more computation. Figure 5 shows another slice in the θ/α plane.

These ROA analysis results indicate that in the EAS/α and θ/α planes the open-loop aircraft will return to the trim solution: $U = 150$ ft/sec, $\alpha = 2.62^\circ$, $q = 0^\circ/sec$, and $\theta = 2.62^\circ$ for initial perturbations in an ellipse defined by U between 80 and 220 ft/sec, θ and α perturbations between -24° and 30°. Similarly analysis can be performed on the closed-loop system to determine the ROA of individual trim points based on the flight control system. Trim points can be investigated using ROA to identify the best trim point to return to in the presence of model uncertainty or system faults.

Nonlinear region-of-attraction and disturbance rejection analysis are an excellent complement to the robust analysis tools developed and implemented within the ROME framework. We for see many opportunities to improve and accelerate validation of flight control systems both off-line and on-line given the recent advances in nonlinear analysis. We look forward to the use of ROME on the AirStar platform in the coming months.

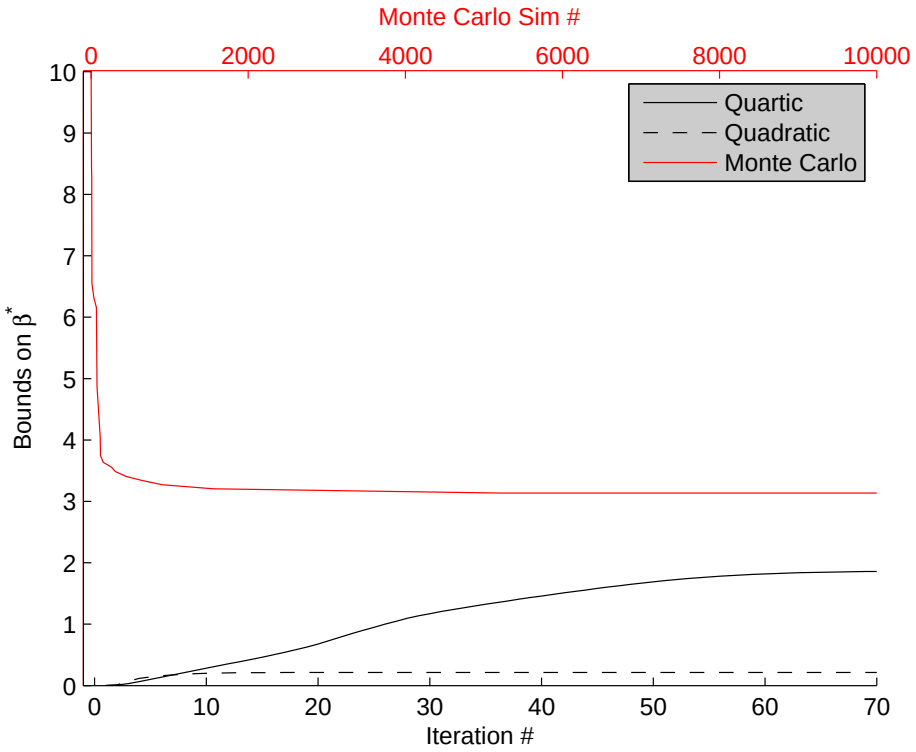

Figure 3: Upper and Lower bounds on ROA Estimate

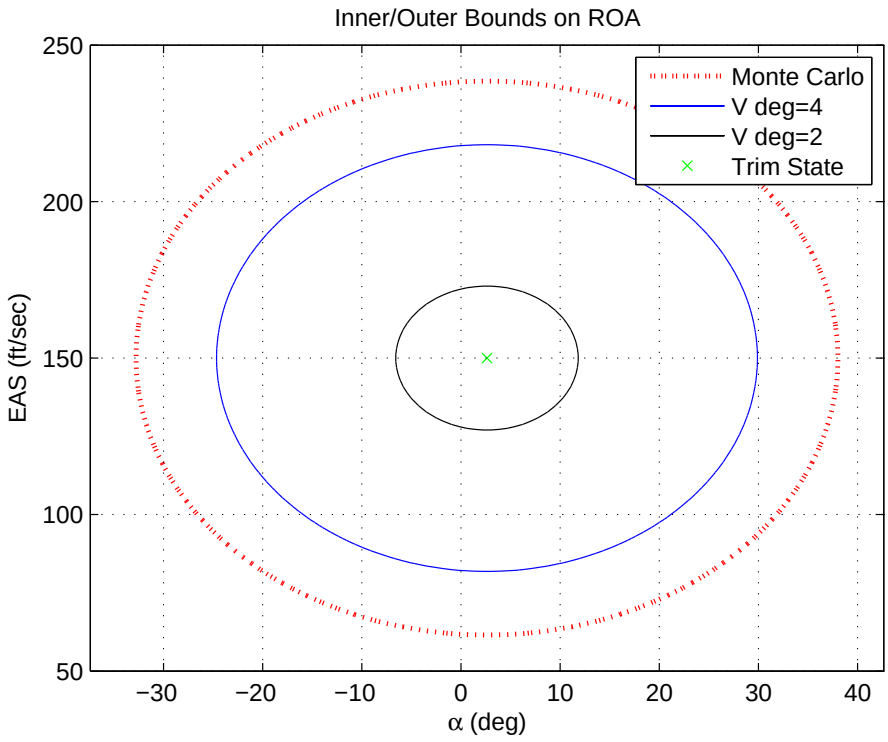

Figure 4: Inner/Outer bounds on ROA Estimate: Slice in EAS/α plane

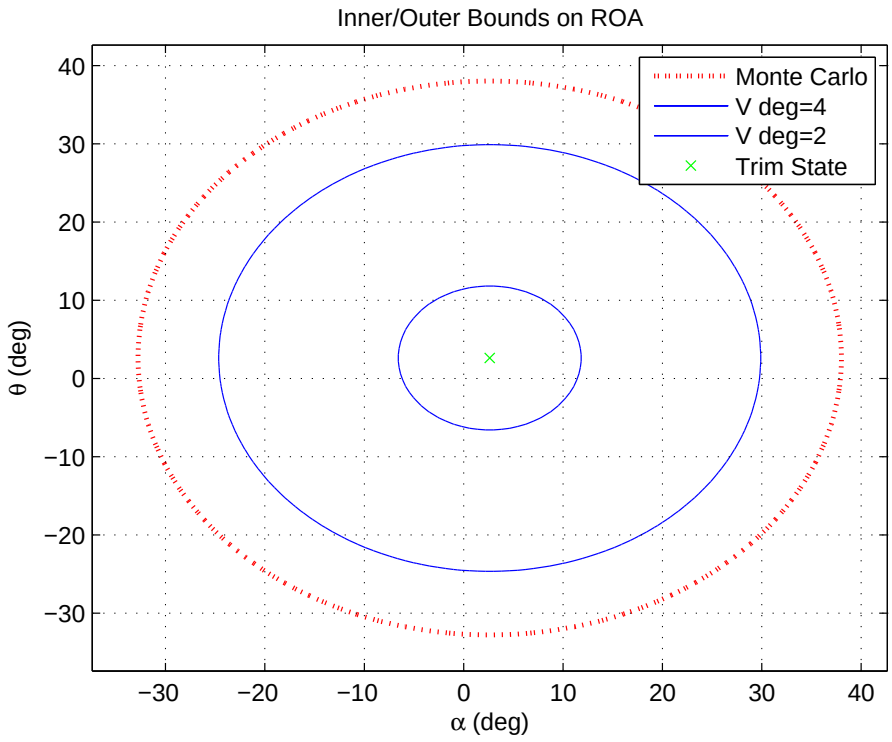

Figure 5: Inner/Outer bounds on ROA Estimate: Slice in θ/α plane

A Sum of Squares Optimization

In this appendix we provide a brief review of computational methods for sum-of-squares polynomial optimizations. Briefly, a polynomial p is a *sum of squares* (SOS) if there exist polynomials $\{f_i\}_{i=1}^m$ such that $p = \sum_{i=1}^m f_i^2$. Sum-of-squares programs are optimization problems involving sum-of-squares polynomial constraints. As discussed in Section 2, the region of attraction estimation problem can be posed as an SOS optimization. Many other nonlinear analysis problems can also be posed within this optimization framework. The computational solutions to these problems rely on connections between semidefinite matrices and SOS polynomials [3, 23, 24]. In this appendix we first present notation and background material. Next we discuss the connections between semidefinite matrices and SOS polynomials. Finally we discuss software available to solve SOS optimization problems.

A.1 Background

A.1.1 Polynomial Notation

$\mathbb{R}[x]$ denotes the set of all polynomials in variables $\{x_1, \ldots, x_n\}$ with real coefficients. $\mathbb{N}$ denotes the set of nonnegative integers, $\{0, 1, \ldots\}$, and $\mathbb{N}^n$ is the set of n-dimensional vectors with entries in $\mathbb{N}$. For $\alpha \in \mathbb{N}^n$, a monomial in variables $\{x_1, \ldots, x_n\}$ is given by $x^\alpha \doteq x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$. The degree of a monomial is defined as $\deg x^\alpha \doteq \sum_{i=1}^n \alpha_i$. In this notation a polynomial in $\mathbb{R}[x]$ is simply a finite linear combination of monomials:

$$p \doteq \sum_{\alpha \in \mathcal{A}} c_\alpha x^\alpha = \sum_{\alpha \in \mathcal{A}} c_\alpha x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$$

where $c_\alpha \in \mathbb{R}$ and $\mathcal{A}$ is a finite collection of vectors in $\mathbb{N}^n$. Using the definition of deg for a monomial, the degree of p is defined as $\deg p \doteq \max_{\alpha \in \mathcal{A}} \ [\deg x^\alpha]$.

A polynomial p is a *sum of squares* (SOS) if there exist polynomials $\{f_i\}_{i=1}^m$ such that $p = \sum_{i=1}^m f_i^2$. The set of SOS polynomials is a subset of $\mathbb{R}[x]$ and is denoted as $\Sigma[x]$. We note that if p is a sum of squares then $p(x) \geq 0 \ \forall x \in \mathbb{R}^n$. Thus $p \in \Sigma[x]$ is a sufficient condition for a polynomial to be globally non-negative. The converse is not true, i.e. non-negative polynomials are not necessarily SOS polynomials. This is related to one of the problems posed by Hilbert in 1900 [25].

A.1.2 Semidefinite Programming

This brief review of semidefinite programming (SDP) is based on a survey by Vandenberghe and Boyd [26] and a monograph by Boyd, et al. [27]. A symmetric matrix $F \in \mathbb{R}^{n \times n}$ is positive semidefinite if $x^T F x \geq 0$ for all $x \in \mathbb{R}^n$. Positive semidefinite matrices are denoted by $F \succeq 0$. A semidefinite program is an optimization problem of the following form:

$$\begin{array}{ll} \min_\lambda & c^T \lambda \\ \text{subject to:} & F_0 + \sum_{k=1}^r \lambda_k F_k \succeq 0 \end{array} \tag{11}$$

The symmetric matrices $F_0, \ldots, F_r \in \mathbb{R}^{n \times n}$ and the vector $c \in \mathbb{R}^r$ are given data. The vector $\lambda \in \mathbb{R}^r$ is the decision variable and the constraint, $F_0 + \sum_{k=1}^r \lambda_k F_k \succeq 0$, is called a linear matrix inequality. We refer to Equation 11 as the primal problem. The dual associated with this primal problem is:

$$\begin{array}{ll} \max_Z & -\mathbf{Tr}\,[F_0 Z] \\ \text{subject to:} & \mathbf{Tr}\,[F_k Z] = c_k \quad k = 1, \ldots, r \\ & Z \succeq 0 \end{array} \tag{12}$$

where $Z = Z^T \in \mathbb{R}^{n \times n}$ is the decision variable for the dual problem. $\mathbf{Tr}\,[\cdot]$ denotes the trace of a matrix. This dual problem can be recast in the form of Equation 11 and thus it is also a semidefinite program. While the primal and dual forms may look restrictive, these formulations are quite versatile and SDPs find applications in many problems of interest. Moreover, SDPs are convex and quality software exists to solve these problems. In particular, SeDuMi [22, 28] is a freely available MATLAB toolbox that simultaneously solves the primal and/or dual forms of a semidefinite program.

In some cases, our only goal is to find a decision variable that satisfies the linear matrix inequality constraint. These are semidefinite programming feasibility problems. The following is an example:

$$\text{Find } \lambda_1, \ldots, \lambda_r \in \mathbb{R} \text{ such that } F_0 + \sum_{k=1}^r \lambda_k F_k \succeq 0 \tag{13}$$

A.2 Connections Between SOS Polynomials and Semidefinite Matrices

Theorem 1 below gives a concrete statement of the connection between sums of squares and positive semidefinite matrices. We require two facts that follow from [29] (refer to Theorem 1 and its preceding Lemma):

1. If p is a sum of squares then p must have even degree.
2. If p is degree $2d$ ($d \in \mathbb{N}$) and $p = \sum_{i=1}^{m} f_i^2$ then $\deg f_i \leq d \ \forall i$.

Next, we define z as the column vector of all monomials in variables $\{x_1, \ldots, x_n\}$ of degree $\leq d$: [1]

$$z \doteq \begin{bmatrix} 1, & x_1, & x_2, & \ldots, & x_n, & x_1^2, & x_1x_2, & \ldots, & x_n^2, & \ldots, & x_n^d \end{bmatrix}^T \tag{14}$$

There are $\binom{k+n-1}{k}$ monomials in n variables of degree k. Thus z is a column vector of length $l_z \doteq \sum_{k=0}^{d} \binom{k+n-1}{k} = \binom{n+d}{d}$. If f is a polynomial in n variables with degree $\leq d$, then f is a finite linear combination of monomials of degree $\leq d$. Consequently, there exists $a \in \mathbb{R}^{l_z}$ such that $f = a^T z$. The proof of the following theorem, introduced as a "Gram Matrix" method by Choi, Lam, and Reznick [30], is included for completeness. This result can be found more recently in [31].

Theorem 1 *Suppose $p \in \mathbb{R}[x]$ is a polynomial of degree $2d$ and z is the $l_z \times 1$ vector of monomials defined in Equation 14. Then p is a SOS if and only if there exists a symmetric matrix $Q \in \mathbb{R}^{l_z \times l_z}$ such that $Q \succeq 0$ and $p = z^T Q z$.*

Proof:
($\Rightarrow$) If p is a SOS, then there exists polynomials $\{f_i\}_{i=1}^m$ such that $p = \sum_{i=1}^m f_i^2$. As noted above, $\deg f_i \leq d$ for all i. Thus, for each f_i there exists a vector, $a_i \in \mathbb{R}^{l_z}$, such that $f_i = a_i^T z$. Define the matrix, $A \in \mathbb{R}^{l_z \times m}$, whose i^{th} column is a_i and define $Q \doteq AA^T \succeq 0$. Then $p = z^T Q z$.
($\Leftarrow$) Assume there exists $Q = Q^T \in \mathbb{R}^{l_z \times l_z}$ such that $Q \succeq 0$ and $p = z^T Q z$. Define $m \doteq rank(Q)$. There exists a matrix $A \in \mathbb{R}^{l_z \times m}$ such that $Q = AA^T$. Let a_i denote the i^{th} column of A and define the polynomials $f_i \doteq z^T a_i$. By definition of f_i, $p = z^T(AA^T)z = \sum_{i=1}^m f_i^2$. ■

A.3 Software for SOS Optimizations

A sum-of-squares program is an optimization problem with a linear cost and SOS constraints on the decision variables [20]:

$$\begin{aligned} &\min_{u \in \mathbb{R}^n} c_1 u_1 + \cdots + c_n u_n \\ &\text{subject to:} \\ &a_{k,0}(x) + a_{k,1}(x)u_1 + \cdots + a_{k,n}(x)u_n \in \Sigma[x] \ k = 1, \ldots N_s \end{aligned} \tag{15}$$

The polynomials $\{a_{k,j}\}$ are given as part of the optimization data and $u \in \mathbb{R}^n$ are decision variables. This formulation appears far removed from dynamical systems but we'll show in Section **??** that nonlinear analysis problems can be posed within this optimization framework.

Theorem 1 provides the bridge to convert an SOS program into a semidefinite-programming problem. For example, the constraint $a_{k,0}(x) + a_{k,1}(x)u_1 + \cdots + a_{k,n}(x)u_n \in \Sigma[x]$ can be equivalently written as:

$$a_{k,0}(x) + a_{k,1}(x)u_1 + \cdots + a_{k,n}(x)u_n = z^T Q z \tag{16}$$

$$Q \succeq 0 \tag{17}$$

Q is a new matrix of decision variables that is introduced when we convert an SOS constraint to an LMI constraint. Equating the coefficients of $z^T Q z$ and $a_{k,0}(x) + a_{k,1}(x)u_1 + \cdots + a_{k,n}(x)u_n$ imposes linear equality constraints on the decision variables u and Q. Thus, Equation 16 can be rewritten as a set of linear equality constraints on the decision variables. All SOS constraints in Equation 15 can be replaced in this fashion with

[1] Any ordering of the monomials can be used to form z. In Equation 14, x^α precedes x^β in the definition of z if:

$$\deg x^\alpha < \deg x^\beta \quad \text{or} \quad \deg x^\alpha = \deg x^\beta \text{ and the first nonzero entry of } \alpha - \beta \text{ is } > 0$$

linear equality constraints and LMI constraints. As a result, the SOS program in Equation 15 can be written in the SDP dual form (Equation 12).

While this may appear cumbersome, there is software available to perform the conversion. For example, SOSTOOLS [20] and Yalmip [21] are freely available MATLAB toolboxes for solving SOS optimizations. These packages allow the user to specify the polynomial constraints using a symbolic toolbox. Then they convert the SOS optimization into an SDP which is solved with SeDuMi [22, 28] or another freely available SDP solver. Finally these toolboxes convert the solution of the SDP back to a polynomial solution. A drawback is that the size of the resulting SDP grows rapidly if the SOS optimization involves polynomials with many variables and/or high degree. While various techniques can be used to exploit the problem structure [32], this computational growth is a generic trend in SOS optimizations. We have developed some methods which use simulation to ease this computational growth [17, 18, 33].

B GTM Polynomial Model

```
%-----------------------------------------------------------------------
% gtmpolymod.m
%
% This file creates a 4-state, 5th order polynomial model for
% the GTM logitudinal dynamics.  The states / inputs are
%    x = [ Equivalent Air Speed, eas (ft/sec);
%          Angle of Attack, alp (rad);
%          Pitch Rate, q (rad/sec);
%          Pitch Euler Angle, theta (rad)]
%
%    u = [ Normalized Throttle Position, dthr (unitless, 0 to 1);
%          Elevator Deflection, de (rad)]
%-----------------------------------------------------------------------

% ----- Create poly vars for states and inputs
pvar de dthr eas alp q theta;

% ----- Constants
g         = 32.17405;       % Gravitational accel [ft/sec^2]
rho0      = 0.002375;       % Air density [ slugs/ft^3 ]
d2r       = pi/180;         % Degrees to Radians [ rad/deg ]
r2d       = 180/pi;         % Radians to Degrees [ deg/rad ]
Sref      = 5.9018;         % Wing reference area [ ft^2]
mass      = 1.5416;         % Mass [slugs]
cbar      = 0.9153;         % Mean chord [ft]
Iyy       = 4.2540;         % Princ. moment of inertia [lf-ft^2]

% ----- Polynomial Approximations

% Trig approximations
sinth = theta - theta^3/6;
costh = 1 - theta^2/2;

sinalp = alp - alp^3/6;
cosalp = 1 - alp^2/2;

% Second-order least squares fit of 1/eas
Umin = 100;
Umax = 200;

Npts = 100;
Urange = linspace(Umin,Umax,Npts);
Urange = Urange(:);
AA=[ones(Npts,1) Urange Urange.^2];
bb=[1./Urange];
cc=AA\bb;
eas_inv = cc(1)+cc(2)*eas+cc(3)*eas^2;


% ----- Engine Model
engl_ang = [0    2.1460    1.6830];
engr_ang = [0    2.1460    -1.6830];
deltaLEng= [0.4498   -1.1833    0.2976];
```

```
% direction cosines vector (DCV) to transform thrust axis to
% airplane body axis. note only L eng. thrust is used, R eng is
% assumed to be at equal setting as the left.
DCV_LEng = [cosd(engl_ang(2))*cosd(engl_ang(3));...
           -sind(engl_ang(3));...
            sind(engl_ang(2))*cosd(engl_ang(3))];
DCV_REng = [cosd(engr_ang(2))*cosd(engr_ang(3));...
           -sind(engr_ang(3));...
            sind(engr_ang(2))*cosd(engr_ang(3))];

% Throttle / Thrust relation
b2=8.871;
b1=9.151;
b0=1.04;
Tdthr = b0+b1*dthr+b2*dthr^2;

% Engine Thrust
Tx = DCV_LEng(1)*Tdthr;
Tz = DCV_LEng(3)*Tdthr;

% ----- Aero coef models
% Assume constant eas in the denominator of Czq and Cmq
qbar    = 0.5*rho0*eas^2;
eas_nom = 152;

a31 = -1.0667e-005*r2d^3;
a21 = -3.9326e-004*r2d^2;
a11 = 0.0129*r2d;
a01 = -0.0414;
Cxa = a31*alp^3+a21*alp^2+a11*alp+a01;

Cxde = 0;
Cxq = 0;
Cx = Cxa + Cxde*de + Cxq*q;

a32 = -4.6405e-006*r2d^3;
a22 = 0.0029*r2d^2;
a12 = -0.1101*r2d;
a02 = -0.0045;
Cza = a32*alp^3+a22*alp^2+a12*alp+a02;

Czde = -0.0083*r2d;
Czq = -24*(0.5*cbar/eas_nom);
Cz = Cza + Czde*de + Czq*q;

a33 = -1.0904e-004*r2d^3;
a23 = 0.0022*r2d^2;
a13 = -0.0348*r2d;
a03 = 0.1553;
Cma = a33*alp^3+a23*alp^2+a13*alp+a03;

Cmde=-0.032*r2d;
Cmq = -45*(0.5*cbar/eas_nom);
Cm = Cma + Cmde*de + Cmq*q;

% ----- Forces / Moments
```

```
neng = 2;
Fx = qbar*Sref*Cx + neng*Tx - mass*g*sinth;
Fz = qbar*Sref*Cz + neng*Tz + mass*g*costh;
My = qbar*Sref*cbar*Cm + neng*Tx*deltaLEng(3);

% ----- State derivatives
x = [eas; alp; q; theta];
u = [dthr; de];

xdot(1) = 1/mass * (Fx*cosalp + Fz*sinalp);
xdot(2) = 1/mass * eas_inv * ( -Fx*sinalp + Fz*cosalp) + q;
xdot(3) = My/Iyy;
xdot(4) = q;
xdot = xdot(:);

% ----- Remove terms with deg >5
deg = 0:5;
xdot = cleanpoly(xdot,[],deg);
```

References

[1] H. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002.

[2] M. Vidyasagar, *Nonlinear Systems Analysis*, 2nd ed. Prentice Hall, 1993.

[3] P. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," Ph.D. dissertation, California Institute of Technology, 2000.

[4] A. Vannelli and M. Vidyasagar, "Maximal lyapunov functions and domains of attraction for autonomous nonlinear systems," *Automatica*, vol. 21, no. 1, pp. 69–80, 1985.

[5] B. Tibken and Y. Fan, "Computing the domain of attraction for polynomial systems via BMI optimization methods," in *Proceedings of the American Control Conference*, 2006, pp. 117–122.

[6] B. Tibken, "Estimation of the domain of attraction for polynomial systems via LMIs," in *Proceedings of the IEEE Conference on Decision and Control*, 2000, pp. 3860–3864.

[7] J. Hauser and M. Lai, "Estimating quadratic stability domains by nonsmooth optimization," in *Proceedings of the American Control Conference*, 1992, pp. 571–576.

[8] O. Hachicho and B. Tibken, "Estimating domains of attraction of a class of nonlinear dynamical systems with LMI methods based on the theory of moments," in *Proceedings of the IEEE Conference on Decision and Control*, 2002, pp. 3150–3155.

[9] R. Genesio, M. Tartaglia, and A. Vicino, "On the estimation of asymptotic stability regions: State of the art and new proposals," *IEEE Transactions on Automatic Control*, vol. 30, no. 8, pp. 747–755, 1985.

[10] E. Davison and E. Kurak, "A computational method for determining quadratic Lyapunov functions for nonlinear systems," *Automatica*, vol. 7, pp. 627–636, 1971.

[11] H.-D. Chiang and J. Thorp, "Stability regions of nonlinear dynamical systems: A constructive methodology," *IEEE Transactions on Automatic Control*, vol. 34, no. 12, pp. 1229–1241, 1989.

[12] Z. Jarvis-Wloszek, "Lyapunov based analysis and controller synthesis for polynomial systems using sum-of-squares optimization," Ph.D. dissertation, University of California, Berkeley, 2003.

[13] Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard, "Some controls applications of sum of squares programming," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, vol. 5, 2003, pp. 4676–4681.

[14] W. Tan and A. Packard, "Searching for control Lyapunov functions using sums of squares programming," in *42nd Annual Allerton Conference on Communications, Control and Computing*, 2004, pp. 210–219.

[15] Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard, *Positive Polynomials in Control*, ser. Lecture Notes in Control and Information Sciences. Springer-Verlag, 2005, vol. 312, ch. Controls Applications of Sum of Squares Programming, pp. 3–22.

[16] W. Tan, "Nonlinear control analysis and synthesis using sum-of-squares programming," Ph.D. dissertation, University of California, Berkeley, 2006.

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

[18] U. Topcu, A. Packard, and P. Seiler, "Local stability analysis using simulations and sum-of-squares programming," *Automatica*, vol. 44, no. 10, pp. 2669–2675, 2008.

[19] U. Topcu, "Quantitative local analysis of nonlinear systems," Ph.D. dissertation, University of California, Berkeley, 2008.

[20] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo, *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, Available from `http://www.cds.caltech.edu/sostools` and `http://www.mit.edu/~parrilo/sostools`, 2004.

[21] J. Lofberg, "Yalmip : A toolbox for modeling and optimization in MATLAB," in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. [Online]. Available: http://control.ee.ethz.ch/ joloef/yalmip.php

[22] J. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, pp. 625–653, 1999.

[23] P. Parrilo, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming Ser. B*, 2003.

[24] J. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 796–817, 2001.

[25] B. Reznick, "Some concrete aspects of Hilberts 17th problem," *Contemporary Mathematics*, vol. 253, no. 251-272, 2000.

[26] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.

[27] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, ser. Studies in Applied Mathematics. SIAM, 1994, vol. 15.

[28] J. Sturm, "SeDuMi version 1.05," http://fewcal.kub.nl/sturm/software/sedumi.html, 2001.

[29] B. Reznick, "Extremal PSD forms with few terms," *Duke Mathematical Journal*, vol. 45, no. 2, pp. 363–374, 1978.

[30] M. Choi, T. Lam, and B. Reznick, "Sums of squares of real polynomials," *Proceedings of Symposia in Pure Mathematics*, vol. 58, no. 2, pp. 103–126, 1995.

[31] V. Powers and T. Wörmann, "An algorithm for sums of squares of real polynomials," *Journal of Pure and Applied Algebra*, vol. 127, pp. 99–104, 1998.

[32] K. Gatermann and P. Parrilo, "Symmetry groups, semidefinite programs, and sums of squares," *Journal of Pure and Applied Algebra*, vol. 192, pp. 95–128, 2004.

[33] W. Tan, U. Topcu, P. Seiler, G. Balas, and A. Packard, "Simulation-aided reachability and local gain analysis for nonlinear dynamical systems," in *Proceedings of the IEEE Conference on Decision and Control*, 2008, pp. 4097–4102.