Some Applications of Polynomial Optimization in Operations Research and Real-Time Decision Making

Amir Ali Ahmadi*and Anirudha Majumdar†

Abstract

We demonstrate applications of algebraic techniques that optimize and certify polynomial inequalities to problems of interest in the operations research and transportation engineering communities. Three problems are considered: (i) wireless coverage of targeted geographical regions with guaranteed signal quality and minimum transmission power, (ii) computing real-time certificates of collision avoidance for a simple model of an unmanned vehicle (UV) navigating through a cluttered environment, and (iii) designing a nonlinear hovering controller for a quadrotor UV, which has recently been used for load transportation. On our smaller-scale applications, we apply the sum of squares (SOS) relaxation and solve the underlying problems with semidefinite programming. On the larger-scale or real-time applications, we use our recently introduced "SDSOS Optimization" techniques which result in second order cone programs. To the best of our knowledge, this is the first study of real-time applications of sum of squares techniques in optimization and control. No knowledge in dynamics and control is assumed from the reader.

1 Introduction

In this paper we consider applications of *polynomial optimization* in the area of operations research and transportation engineering. While techniques in more established areas of optimization theory such as linear, integer, combinatorial, and dynamic programming have found wide applications in these areas [11, 10, 9, 22], the relatively newer field of polynomial optimization, which has gone through rapid advancements in recent years, may yet prove to reveal many unexplored applications. It is our aim in this paper to bring a few such applications to the attention of the operation research community and to highlight some algorithmic tools based on algebraic techniques that we believe are particularly suited for approaching problems of this sort.

The fundamental problem underlying all of our applications is that of *optimizing over nonnegative polynomials*. This is the task of finding the coefficients $c_\alpha := c_{\alpha_1,\ldots,\alpha_n}$ of some multivariate polynomial $p(x) := p(x_1,\ldots,x_n) = \sum_\alpha c_\alpha x^\alpha$ in order to get $p(x) \geq 0$, either globally (i.e., $\forall x \in \mathbb{R}^n$), or on certain basic semialgebraic sets. A *basic semialgebraic set* is a subset of the Euclidean space defined by a finite number of polynomial (in)equalities. That is a set of the form

$$\mathcal{S} := \{x \in \mathbb{R}^n |\ g_i(x) \geq 0, h_i(x) = 0\},$$

where the functions g_i, h_i are all multivariate polynomials. The polynomial optimization problem (POP) is itself a problem of this form. Indeed, the task of finding the minimum of a polynomial function q on a basic semialgebraic set $\mathcal{S}$ is the same as that of finding the largest constant γ such that $q(x) - \gamma$

*Amir Ali Ahmadi is with the Department of Operations Research and Financial Engineering at Princeton University. a_a_a@princeton.edu, http://aaa.princeton.edu/

†Anirudha Majumdar is with the Department of Electrical Engineering and Computer Science, CSAIL, MIT. anirudha@mit.edu, http://www.mit.edu/~anirudha

is nonnegative on $\mathcal{S}$. There are, however, many other applications of optimization over nonnegative polynomials, some to be seen in this work.

Our paper is organized as follows. In Section 2, we briefly review the concept of sum of squares (sos) decomposition and its relation to semidefinite programming (SDP). This is a popular approach for certifying polynomial nonnegativity. While remarkably powerful, it often faces scalability limitations on larger-scale problems. As a potential remedy, we have recently introduced [5, 4] the concepts of *diagonally dominant and scaled diagonally dominant sum of squares (dsos and sdsos)* decomposition, which instead of SDP result in linear programs (LP) and second order cone programs (SOCP) respectively. These concepts are also presented in Section 2.

In Section 3, we consider the problem of providing guaranteed wireless coverage to certain basic semialgebraic subsets of the Euclidean space with minimum transmission power. The general problem here has been previously considered in the literature but we show that tools from polynomial optimization allow us to handle the problem in broader and arguably more realistic scenarios. Our next two examples are related to transportation problems in operations research. In particular, in Section 5, we consider a simple model of an unmanned aerial vehicle (UAV), which aims to fly through a cluttered environment in a collision free manner. The techniques presented in this section can also be adapted for applications to ground vehicles. We demonstrate how one can choose a control law and at the same time find a *formal certificate*—an independently verifiable proof—that the resulting dynamics will guarantee no collisions with obstacles. We show that using our SOCP techniques, the underlying computational task can be carried out in the order of 20-30 milliseconds, hence making a plausibility claim about a real-time application of this approach. In Section 6, we use the same technical tools to design a stabilizing controller for a quadrotor system, a device that has increasing potential for use in transportation (see Section 6). The designed controller prevents the quadrotor from losing balance when it is subject to environmental disturbance, or an external perturbation. The SDP resulting from this controller design problem is so large that it cannot be solved on our machine (3.4 GHz PC with 4 cores and 16 GB RAM). This is another example demonstrating the promise of our new sdsos machinery for handling problems of large scale.

Our second and third applications include the employment of Lyapunov techniques to convert a problem in dynamics and control to a problem in polynomial optimization. Since we do not want to assume this background from the reader, we present the essentials of these very basic concepts in Section 4. The mathematical background in this section (just like Section 2) is presented at a minimal level to make the paper self-contained, while keeping the focus on the applications and the algorithmic aspects. We end the paper with some brief concluding remarks in Section 7.

2 Algebraic certificates of nonnegativity via convex optimization

The task of optimizing over nonnegative polynomials or even checking nonnegativity of a given polynomial, either globally or on a basic semialgebraic set, is known to be NP-hard [33]. This is true already for checking global nonnegativity of a quartic (degree-4) polynomial, or for checking nonnegativity of a quadratic polynomial on a set defined by linear inequalities. A popular relaxation scheme for this problem is through the machinery of the so-called *sum of squares optimization*.

We say that a polynomial p is a sum of squares (sos), if it can be written as $p = \sum_i q^2$ for some other polynomials q_i. Obviously, such a decomposition is a sufficient (but in general not necessary [19]) condition for (global) nonnegativity of p. The situation where p is only constrained to be nonnegative on a certain basic semialgebraic set[1]

$$\mathcal{S} := \{x \in \mathbb{R}^n |\ g_1(x) \geq 0, \ldots, g_m(x) \geq 0\}$$

[1]In this formulation, we have avoided equality constraints for simplicity. Obviously, there is no loss of generality in doing this as an equality constraint $h(x) = 0$ can be imposed by the pair of inequality constraints $h(x) \geq 0, -h(x) \geq 0$.

can also be handled with the help of appropriate sum of squares multipliers. For example, if we succeed in finding sos polynomials $s_0, s_1, \ldots, s_m$, such that

$$p(x) = s_0(x) + \sum_{i=1}^{m} s_i(x) g_i(x), \tag{1}$$

then we have found a certificate of nonnegativity of p on the set $\mathcal{S}$. Indeed, if we evaluate the above expression at any $x \in \mathcal{S}$, nonnegativity of the polynomials $s_0, s_1 \ldots, s_m$ imply that $p(x) \geq 0$. A Positivstellensatz theorem from real algebraic geometry due to Putinar [40] states that if the set $\mathcal{S}$ satisfies the so-called *Archimedean* property, a property only slightly stronger than compactness[2], then every polynomial positive on $\mathcal{S}$ has a representation of the type (1), for some sos polynomials $s_0, s_1, \ldots, s_m$ of high enough degree (see also [34] for degree bounds). Even with absolutely no qualifications about the set $\mathcal{S}$, there are other Positivstellensatz theorems (e.g., due to Stengle [41]) that certify nonnegativity of a polynomial on a basic semialgebraic set using sos polynomials. These certificates are only slightly more complicated than (1) and involve sos multipliers associated with products among polynomials g_i that define $\mathcal{S}$ [36]. A great reference for the interested reader is the survey paper by Laurent [24].

The computational advantage of a certificate of (global or local) nonnegativity via sum of squares polynomials is that it can be automatically found by semidefinite programming. What establishes the link between sos polynomials and SDP is the following well-known theorem. Recall that a symmetric $n \times n$ matrix A is *positive semidefinite* (psd) if $x^T A x \geq 0, \forall x \in \mathbb{R}^n$, and that semidefinite programming is the problem of optimizing over psd matrices subject to affine inequalities on their entries [43]. We denote the positive semidefiniteness of a matrix A with the standard notation $A \succeq 0$.

Theorem 2.1 (see, e.g., [35],[36])**.** *A multivariate polynomial $p(x)$ in n variables and of degree $2d$ is a sum of squares if and only if there exists a symmetric matrix Q (often called the Gram matrix) such that*

$$\begin{array}{rcl} p(x) & = & z^T Q z, \\ Q & \succeq & 0, \end{array} \tag{2}$$

where z is the vector of monomials of degree up to d

$$z = [1, x_1, x_2, \ldots, x_n, x_1 x_2, \ldots, x_n^d].$$

The search for the matrix Q satisfying a positive semidefiniteness constraint, as well as linear equality constraints coming from (2) is a semidefinite programming problem. The size of the matrix Q in this theorem is

$$\binom{n+d}{d} \times \binom{n+d}{d},$$

which approximately equals $n^d \times n^d$. While this number is polynomial in n for fixed d, it can grow rather quickly even for low degree polynomials. For example, the polynomials that we will be requiring to be sos in our controller design problem for the quadrotor (Section 6) have 16 variables and degree 6 and result in a Gram matrices with about half a million decision variables. A semidefinite constraint of this size is quite expensive—for example, the SDP solvers of SeDuMi [42] and MOSEK [2] fail to solve the quadrotor problem on our machine and quickly run out of memory.

2.1 DSOS and SDSOS Optimization

In order to address the problem of scalability posed by SDP, we have recently introduced [5, 4] alternatives to SOS programming that lead to linear programs (LPs) and second order cone programs (SOCPs).

[2]In particular, if we have as an outer estimate a ball of some radius R in which our set $\mathcal{S}$ lives, then we can add a single quadratic inequality $\sum_i x_i^2 \leq R$ to the description of $\mathcal{S}$ to have it satisfy the Archimedean property without changing the set.

The key insight there is to replace the condition that the Gram matrix Q be positive semidefinite (psd) with stronger sufficient conditions in order to obtain inner approximations to the cone $SOS_{n,d}$ of sos polynomials in n variables and of degree d. In particular, Q will be required to be either *diagonally dominant* (dd) or *scaled diagonally dominant* (sdd). We recall these definitions below.

Definition 2.2. *A symmetric matrix A is* diagonally dominant *(dd) if $a_{ii} \geq \sum_{j \neq i} |a_{ij}|$ for all i.*

We will refer to the set of $n \times n$ dd matrices as DD_n.

Remark 2.1. It is clear from Definition 2.2 that the set DD_n has a polytopic description and can thus be optimized over using LP.

Definition 2.3. *Denote the set of $n \times n$ symmetric matrices as S^n. Let $M_{2\times 2}^{ij} \in S^n$ denote the symmetric matrix with all entries zero except the elements $M_{ii}, M_{ij}, M_{ji}, M_{jj}$. Then, a symmetric matrix A is* scaled diagonally dominant *(sdd) if it can be expressed in the following form:*

$$A = \sum_{i \neq j} M_{2\times 2}^{ij}, \quad \begin{bmatrix} M_{ii} & M_{ij} \\ M_{ji} & M_{jj} \end{bmatrix} \succeq 0.$$

Remark 2.2. The relationship between dd and sdd matrices is made clear in [5]. As we show there, a symmetric matrix A is sdd if and only if there exists a positive diagonal matrix D such that AD (or equivalently, DAD) is diagonally dominant.

The set of $n \times n$ sdd matrices will be denoted by SDD_n. We note that sdd matrices are sometimes referred to as *generalized diagonally dominant* matrices [13].

Theorem 2.4. *The set of matrices SDD_n can be optimized over using second order cone programming.*

Proof. Positive semidefiniteness of the 2×2 matrices in Definition 2.3 is equivalent to the diagonal elements M_{ii}, M_{jj}, along with the determinant $M_{ii}M_{jj} - M_{ij}^2$, being nonnegative. This is a *rotated quadratic cone* constraint and can be imposed using SOCP [6]. □

Remark 2.3. The fact that diagonal dominance is a sufficient condition for positive semidefiniteness follows directly from Gershgorin's circle theorem. The fact that sdd implies psd is immediate from Definition 2.3 since a sdd matrix is a sum of psd matrices. Hence, denoting the set of $n \times n$ symmetric positive semidefinite matrices (psd) as S_n^+, we have from the definitions above that:

$$DD_n \subseteq SDD_n \subseteq S_n^+.$$

We now introduce some naturally motivated cones that are inner approximations of the cone of nonnegative polynomials and that lend themselves to LP and SOCP. In analogy with the representation of sos polynomials in terms of psd matrices (Theorem 2.1), we define the *dsos* and *sdsos* polynomials in terms of dd and sdd matrices respectively.

Definition 2.5 ([5, 4])**.**

- *A polynomial p of degree $2d$ is* diagonally-dominant-sum-of-squares *(dsos) if it admits a representation as $p(x) = z^T(x)Qz(x)$, where $z(x)$ is the standard monomial vector of degree d, and Q is a dd matrix.*

- *A polynomial p of degree $2d$ is* scaled-diagonally-dominant-sum-of-squares *(sdsos) if it admits a representation as $p(x) = z^T(x)Qz(x)$, where $z(x)$ is the standard monomial vector of degree d, and Q is a sdd matrix.*

We denote the set of polynomials in n variables and degree d that are dsos and sdsos by $DSOS_{n,d}$ and $SDSOS_{n,d}$ respectively.

The following inclusion relations are straightforward:

$$DSOS_{n,d} \subseteq SDSOS_{n,d} \subseteq SOS_{n,d}.$$

Theorem 2.6. *The set $DSOS_{n,d}$ is polyhedral and the set $SDSOS_{n,d}$ has a second order cone representation. For any fixed d, optimization over $DSOS_{n,d}$ (resp. $SDSOS_{n,d}$) can be done with linear programming (resp. second order cone programming), of size polynomial in n.*

Proof. This follows directly from Remark 2.1 and Theorem 2.4. The size of these programs is polynomial in n since the size of the Gram matrix is $\binom{n+d}{d} \times \binom{n+d}{d}$, which scales as n^d. □

Remark 2.4. While here we have chosen to define the $DSOS_{n,d}$ and $SDSOS_{n,d}$ cones directly in terms of dd and sdd matrices in order to expose their LP and SOCP characterizations, it is more natural to define them as sos polynomials of a *particular form*. This alternate characterization is provided in [5]. In particular, we have the following equivalent definitions:

- A polynomial p is dsos if it can be written as

$$p = \sum_i \alpha_i m_i^2 + \sum_{i,j} \beta_{ij}^+ (m_i + m_j)^2 + \beta_{ij}^- (m_i - m_j)^2,$$

  for some monomials m_i, m_j and some constants $\alpha_i, \beta_{ij}^+, \beta_{ij}^- \geq 0$.

- A polynomial p is sdsos if it can be written as

$$p = \sum_i \alpha_i m_i^2 + \sum_{i,j} (\beta_i^+ m_i + \gamma_j^+ m_j)^2 + (\beta_i^- m_i - \gamma_j^- m_j)^2,$$

  for some monomials m_i, m_j and some constants $\alpha_i, \beta_i^+, \gamma_j^+, \beta_i^-, \gamma_j^- \geq 0$.

We will refer to optimization problems with a linear objective posed over the cones $DSOS_{n,d}$, $SDSOS_{n,d}$, and $SOS_{n,d}$ as DSOS programs, SDSOS programs, and SOS programs respectively. In general, quality of approximation decreases, while scalability increases, as we go from SOS to SDSOS to DSOS programs. Depending on the size of the application at hand, one may choose one approach over the other. In this paper, we will be using SOS optimization (Section 3) and SDSOS optimization (Sections 5 and 6) in our numerical experiments. The reader is referred to [26, 4, 5] for many numerical examples involving DSOS optimization. We also remark in passing that SDSOS or even DSOS programming enjoy many of the same theoretical (asymptotic) guarantees of SOS programming—results of this nature are proven in [5].

We now proceed to some potential operations research applications of the tools discussed so far.

3 Wireless coverage with minimum transmission

In the problem considered in this section we have a number n of wireless electromagnetic transmitters located at positions $(\bar{x}_i, \bar{y}_i), i = 1, \ldots, n$ on the plane. Each transmitter is an omnidirectional power source, emitting waves in all directions with equal intensity. Due to the laws of electromagnetics, the energy E_i propagated from each jamming device is inversely proportional to the squared distance from the device:

$$E_i(x, y) = \frac{c_i \lambda}{(x - \bar{x}_i)^2 + (y - \bar{y}_i)^2},$$

where λ is some propagation constant, set hereafter to 1 with no loss of generality, and c_i is the transmission rate of device i. The goal is to make sure that certain regions of the plane are guaranteed to receive a given cumulative energy level of at least C units, while minimizing transmission power. These regions can for example be populated urban geographical domains where a wireless service provider would like to guarantee a certain level of signal quality.

The problem we describe is motivated by some interesting and relatively recent work in [16], [15] (see also the thesis [14]), where the motivation is instead to jam the communication network of an adversary with a wireless transmitter. We note, however, that there are a few differences between our setting and that of [16] and [15], the main one being the assumption about the region to be covered. Reference [16] assumes that this region is a set of isolated points (the location of the adversary is known) and this results in a simplified problem. However, more complex objectives are considered by the authors; e.g., the goal is to make the communication graph of the enemy disconnected, or to jam a prescribed fraction of the enemy locations, or to decide which transmitters to turn off. On the opposite end, the work in [15] assumes absolutely nothing about the location of the adversary. As a result, the goal is to cover an entire rectangular region by a prescribed level of jamming power. Our setting, by contrast, allows for the region to be covered to be the union of arbitrary basic semialgebraic sets (see, e.g., Figure 1(a)); this obviously enhances the modeling power. We should also comment that neither our work, nor the works in [16] and [15], satisfactorily address the more difficult problem of optimizing over the location of the transmitters.

A formal summary of our setting is as follows. We are given as input the following quantities: C (required coverage level), γ_i (upper bounds on transmission rates), $(\bar{x}_i, \bar{y}_i), i = 1, \ldots, n$ (location of our transmitters), $\mathcal{B}_j, j = 1, \ldots, m$ (basic semialgebraic sets describing regions to be covered). We assume that the transmitters are outside of the location sets $\mathcal{B}_j$. The goal is to find transmission rates c_i to solve the following optimization problem[3]:

$$\begin{array}{rcll} \text{minimize} & & \sum_{i=1}^n c_i & \\ c_i & \leq & \gamma_i, & \forall i = 1, \ldots, n, \\ E(x,y) := \sum_{i=1}^n \frac{c_i}{(x-\bar{x}_i)^2+(y-\bar{y}_i)^2} & \geq & C, & \forall (x,y) \in \mathcal{B}_j, j = 1, \ldots, m. \end{array} \tag{3}$$

Note that the latter constraints are requiring certain rational functions to be nonnegative on certain basic semialgebraic sets. Upon taking common denominators, we can rewrite these constraints as polynomial inequality constraints:

$$\begin{array}{rl} p(x,y) & := -C \prod_{i=1}^n [(x-\bar{x}_i)^2 + (y-\bar{y}_i)^2] + \sum_{i=1}^n c_i \prod_{k \neq i} [(x-\bar{x}_k)^2 + (y-\bar{y}_k)^2] \geq 0, \\ & \qquad \forall (x,y) \in \mathcal{B}_j, j = 1, \ldots, m. \end{array} \tag{4}$$

Observe that the degree of the polynomial $p(x, y)$ is two times the number of transmitters. Since we are dealing with polynomial inequalities in only two variables, we have no scalability issues restraining us from applying the sos relaxation. Let each set $\mathcal{B}_j$ be defined as

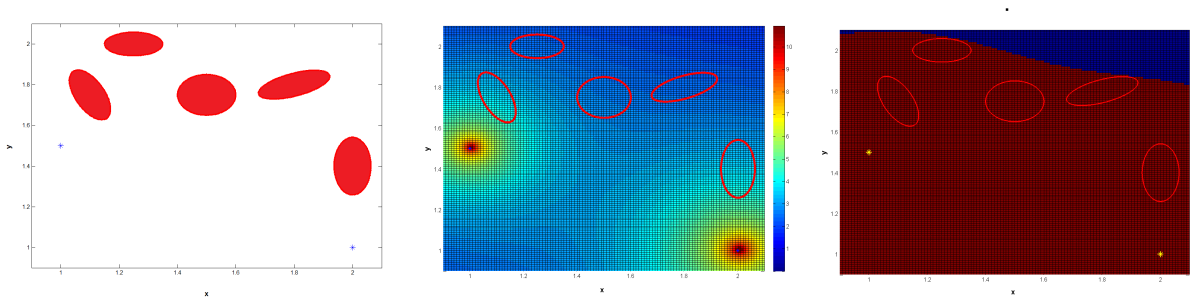
$$\mathcal{B}_j = \{x| \quad g_{j,1}(x,y) \geq 0, \ldots, g_{j,k_j}(x,y) \geq 0\},$$

for some bivariate polynomials $g_{j,1}, \ldots, g_{j,k_j}$. The optimization problem that we will be solving is:

$$\begin{array}{rcll} \text{minimize} & & \sum_{i=1}^n c_i & \\ p & = & \sigma_0 + \sum_{i=1}^{k_j} \sigma_{j,k} g_{j,i}, & j = 1, \ldots, m, \\ & & \sigma_0, \sigma_{j,k} & \text{sos}, \end{array} \tag{5}$$

where p is as in (4) and $\sigma_0, \sigma_{j,k}$ are bivariate polynomials whose degree is upper bounded by some even integer d. Note that the above is a semidefinite programming problem (via Theorem 2.1) with

[3] An alternative reasonable objective is to minimize $\max_i c_i$. This can as easily be handled.



(a) The input to the problem.

(b) Log of the cumulative energy at the SDP solution.

(c) Region that receives a total energy of at least $C = 10$ units (in dark red).

Figure 1: An instance of the wireless coverage problem.

decision variables consisting of the scalars c_i and the coefficients of the polynomials $\sigma_0, \sigma_{j,k}$. It is easy to see that for each value of the degree d, the optimal value of (5) is an upper bound on the optimal value of (3). Moreover, since in our setting each set $\mathcal{B}_i$ satisfies the Archimedean property[4], Putinar's Positivstellensatz tells us that by increasing d, we will be able to solve (3) to global optimality.

Let us now solve a concrete example. Our input data is demonstrated in Figure 1(a). We have two transmitters, located at points $(1, 1.5)$ (called transmitter 1) and $(2, 1)$ (called transmitter 2) on the plane. The area to be covered is given by the five ellipsoidal regions

$$\mathcal{B}_j = \{z := (x, y)^T | \ (z - z_j)^T A_j (z - z_j) \leq \alpha_j\},$$

with $A_1 = \begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix}, A_2 = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}, A_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, A_4 = \begin{bmatrix} 1 & -1 \\ -1 & 3 \end{bmatrix}, A_5 = \begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix}, z_1 = (1.1, 1.75)^T, z_2 = (1.25, 2)^T, z_3 = (1.5, 1.75)^T, z_4 = (1.8, 1.8)^T, z_5 = (2, 1.4)^T, \alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.1, \alpha_5 = 0.2.$

The required energy level on these areas is $C = 10$ and the upper bounds on both transmission rates c_1, c_2 is 11. We first would like to know if by only turning on one of the two transmitters we can meet the required energy level. For the transmitter at the location $(2, 1)$, the optimal value of the SDP in (5) with degree of sos multipliers set to zero (i.e., constant multipliers) is 17.594. In fact, in this case, we know that this upper bound is already exact! This is because in the case of one transmitter, the polynomial p in (4) is quadratic. If a quadratic polynomial is nonnegative on a region defined by another quadratic, this fact is always certified by a constant degree multiplier—this is the celebrated $\mathcal{S}$-lemma; see [37]. Similarly, if we solve the problem for the transmitter located at $(1, 1.5)$, the optimal value of (5) which matches the optimal value of (3) is 11.446. So our task is indeed not achievable with one transmitter only.

With both transmitters on, the SDP in (5) is infeasible for degree-0 sos multipliers (giving an upper bound of infinity). However, when we increase the degree of these multipliers to 2, a solution is returned with $c_1 = 2.561$ and $c_2 = 5.550$ at optimality. By further increasing the degree of our sos multipliers, no improvement in optimal value is observed and we conjecture that the numbers above are already optimal for the original problem (3). Figure 1(b) shows the logarithm of the cumulative energy level $E(x, y)$ at each point in space. (The logarithm is taken to better observe the dispersion of energy.) Figure 1(c) shows all pixels that receive the required energy level of $C = 10$ units. As promised, all five ellipsoids are covered and interestingly the boundary of the region covered touches two of the ellipsoids.

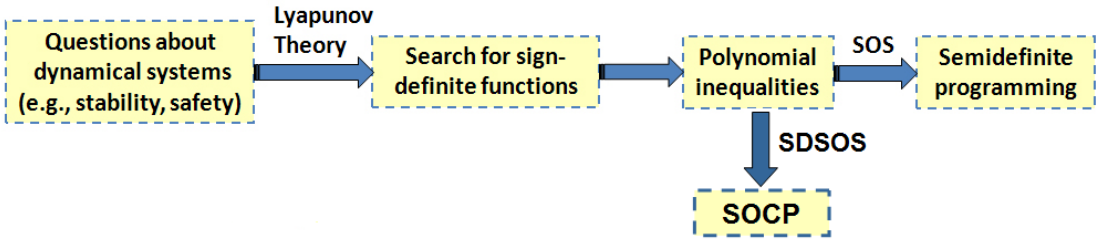


Figure 2: The steps involved in Lyapunov analysis of dynamical systems via convex optimization.

4 Lyapunov theory and optimization

The examples presented in our next two sections involve decision-making about trajectories of dynamical systems. The machinery that allows us to reduce such tasks to problems in optimization is *Lyapunov theory*. As depicted in Figure 2, the general idea is the following: In order to guarantee that trajectories of dynamical systems satisfy certain desired properties, it will be enough to find certain scalar valued functions that satisfy certain inequalities. These functions will be parameterized as polynomials and DSOS/SDSOS/SOS relaxation techniques will be used to find their unknown coefficients in such a way that the desired inequalities are automatically satisfied. For our two applications, we explain next what these inequalities actually are.

Barrier functions (Section 5). Consider a differential equation $\dot{x} = f(x)$, where $\dot{x}$ denotes the derivative of the state vector x with respect to time and $f : \mathbb{R}^n \to \mathbb{R}^n$ is a polynomial function. Suppose we are given two basic semialgebraic sets $\mathcal{S}_{\text{safe}}$ and $\mathcal{S}_{\text{unsafe}}$ and we want to guarantee that trajectories starting in $\mathcal{S}_{\text{safe}}$ would never end up in $\mathcal{S}_{\text{unsafe}}$. This guarantee can be achieved if we succeed in finding a function $V : \mathbb{R}^n \to \mathbb{R}$, called a *barrier function* [38], [39], [8], with the following three properties:

$$V(x) < 1 \quad \forall x \in \mathcal{S}_{\text{safe}}, \quad V > 1 \quad \forall x \in \mathcal{S}_{\text{unsafe}}, \quad \dot{V}(x) \leq 0 \quad \forall x.$$

The expression $\dot{V}$ denotes the time derivative of V along trajectories. If V is a polynomial, $\dot{V}$ will also be a polynomial given (via the chain rule) by:

$$\dot{V}(x) = \langle \nabla V(x), f(x) \rangle.$$

The three inequalities above imply that it is impossible for a trajectory to go from $\mathcal{S}_{\text{safe}}$ to $\mathcal{S}_{\text{unsafe}}$ since the function V evaluated on this trajectory would need to go from a value less than one to a value more than one, but that cannot happen since the value of V is non-increasing along trajectories.

Stability and region of attraction computation (Section 6). Suppose once again that we have a differential equation $\dot{x} = f(x)$ with origin as an equilibrium point (i.e., satisfying $f(0) = 0$). In numerous applications in control and robotics, one would like to make sure that deviations from an equilibrium point tend back to the equilibrium point. This is the notion of asymptotic stability. A particularly important problem in this area is the so-called "region of attraction (ROA) problem": For what set of initial conditions in $\mathbb{R}^n$ do trajectories flow to the origin? This question can be addressed with Lyapunov theory. In fact, Lyapunov's stability theorem (see, e.g., [23, Chap. 4]) tells us that if we can find a (Lyapunov) function $V : \mathbb{R}^n \to \mathbb{R}$, which together with its gradient ∇V satisfies

$$V(x) > 0 \quad \forall x \neq 0, \qquad \text{and} \qquad \dot{V}(x) = \langle \nabla V(x), f(x) \rangle < 0 \quad \forall x \in \{x |\ V(x) \leq \beta, x \neq 0\}, \tag{6}$$

[4]Indeed each set $\mathcal{B}_i$ is compact and the entire environment can be placed in a ball of some prescribed radius R. This quadratic constraint can be added to the description of each $\mathcal{B}_i$ to satisfy the Archimedean property.

then the sublevel set $\{x|\ V(x) \leq \beta\}$ is part of the region of attraction. Notice again that if f is a polynomial function (an immensely important case in applications [3, Chap. 4]), and if we parameterize V as a polynomial function, then the search for the coefficients of V satisfying the conditions in (6) is an optimization problem over the set of nonnegative polynomials.

5 Real-time Planning with Barrier Functions

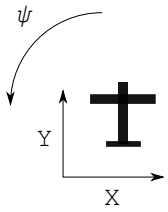


Figure 3: An illustration of the states of the UAV model we consider.

One promising application domain for polynomial optimization in transportation is for real-time planning and control on autonomous vehicles. In this example, we consider such an application for a simple model of an unmanned aerial vehicle (UAV) navigating through a cluttered two dimensional environment. In order to make the navigation task more realistic, we also consider a bounded but uncertain "cross-wind" term in the dynamics. This results in an uncertain differential equation and requires reasoning about *families* of trajectories that the system could end up following, making the problem more challenging. The states and dynamics of the UAV are inspired by the widely-used Dubins car model [18] and are given by:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}, \qquad \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -v \sin\psi + w \\ v \cos\psi \\ u \end{bmatrix}, \tag{7}$$

where x and y are the x and y positions of the UAV in the environment, $v = 1$ m/s is the speed of the airplane, ψ is the yaw angle, u is the control input and w is the "cross-wind" (bounded between $[-0.05, 0.05]$). An illustration of the states of the model are given in Figure 3. We Taylor expand these dynamics to degree 3 to obtain polynomial dynamics in order to use DSOS/SDSOS/SOS programming.

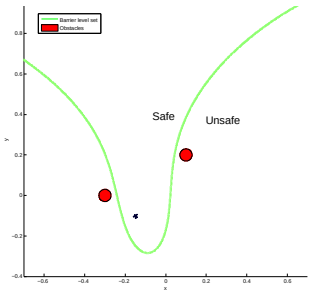
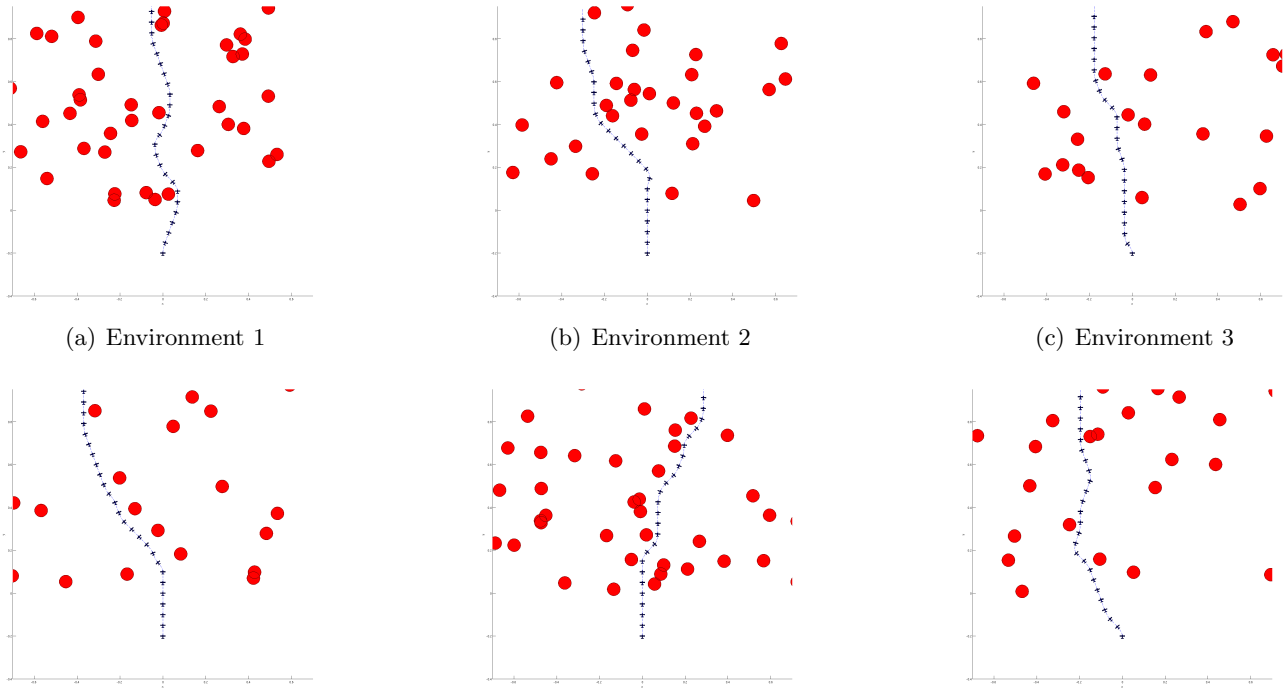


Figure 4: A barrier function computed for a particular initial state and obstacle configuration. The UAV is guaranteed to remain safe when the controller is executed despite the effects of the cross-wind. The green curve is a level set of a degree-4 polynomial found by SDSOS optimization.



(a) Environment 1 (b) Environment 2 (c) Environment 3

(d) Environment 4 (e) Environment 5 (f) Environment 6

Figure 5: UAV successfully navigating through different obstacle environments using the planning algorithm described in Section 5. A video of the navigation can be found at http://youtu.be/J3a6v0tlsD4. This video also shows the barrier certificates (not shown here) as they get updated in real time.

Our goal is to make the UAV navigate through cluttered environments that are *unknown pre-runtime* while avoiding collisions with the obstacles in the environment *despite* the effects of the cross-wind on the vehicle dynamics. In order to achieve this, we pre-compute five control primitives that the UAV can choose from at runtime. These controllers take the form:

$$u_i(\mathbf{x}) = -K(\psi - \psi_{des,i}),\ i = 1, \ldots, 5. \tag{8}$$

These control primitives cause the UAV to control its yaw angle to a particular angle ($\psi_{des,i}$). We choose $K = 50$ and $\psi_{des,1} = 0$ rad, $\psi_{des,2} = -20\pi/180$ rad, $\psi_{des,3} = 20\pi/180$ rad, $\psi_{des,4} = -45\pi/180$ rad, $\psi_{des,5} = 45\pi/180$ rad.

The UAV's task is to choose from these control primitives in order to navigate its way through the environment. After executing a particular chosen primitive for a short interval of time (1/20 seconds in our case), the UAV replans by choosing a control primitive again. Hence, the key decision that our planning algorithm needs to make is the choice of control primitive given a particular configuration of obstacles in its environment. We take our inspiration from [8], which uses *barrier certificates* for verifying the safety of a controller given a particular set of obstacles (but does not consider the case where obstacle positions are not known beforehand and decisions must be made in real time). Similarly, other previous SOS programming approaches [27] to collision avoidance have involved solving SOS programs offline and then using these precomputed results to do planning in real time. In contrast, in our example here, the optimization problems are solved in real-time. We describe our approach below.

At every control iteration, we identify the closest two obstacles in the environment in front of the UAV. We then evaluate each control primitive u_i and check if executing it from our current state will result in the UAV avoiding collision with the obstacles. The first safe controller found is executed. The safety of a controller can be checked by computing barrier functions using the polynomial optimization approaches described in Section 2. Denoting the current state as $\mathbf{x_0} = (x_0, y_0, \psi_0)$ and the obstacle sets

as $X_{obs,1} \subset \mathbb{R}^2$ and $X_{obs,2} \subset \mathbb{R}^2$, we use polynomial optimization to search for a function $V(\mathbf{x})$ of degree 4 that satisfies the following conditions:

$$V(\mathbf{x}_0) = 0, \tag{9}$$

$$V(\mathbf{x}) > 1, \ \forall (x, y) \in X_{obs,i}, \ i = 1, 2, \tag{10}$$

$$\dot{V}(\mathbf{x}, w) = \frac{\partial V}{\partial \mathbf{x}} f(\mathbf{x}, u_i(\mathbf{x}), w) < 0, \ \forall \mathbf{x} \in X, \ \forall w \in [-0.05, 0.05]. \tag{11}$$

Here, X is a "large" set that the system is guaranteed to remain within for the duration of time for which the control primitive is executed. In particular, we choose it to be the unit sphere around the current state. The conditions above imply that the state $\mathbf{x}$ is constrained to evolve within the 1-sublevel set (in fact the 0-sublevel set) of the function $V(\mathbf{x})$ and is thus *guaranteed* to not collide with the obstacles despite the effects of the cross-wind.

Hence, at each control iteration we need to solve a maximum of 5 optimization problems, all of which are independent and can be parallelized. In our example, we use SDSOS programming to compute barrier functions and observe running times of approximately $0.02 - 0.03$ seconds for feasible problems and $0.08 - 0.09$ seconds for infeasible problems (i.e., problems where no barrier function can be found) using the Gurobi SOCP solver [1] (a more thorough running time analysis is presented later). Hence, a real-time implementation of this approach on a hardware platform is plausible. Such a hardware implementation can benefit from already-existing SOCP solvers that are specifically designed to run on embedded systems [29], [28]. In particular, [17] presents an approach for generating stand-alone C code for an SOCP solver that can run very efficiently and with low memory footprint. The use of such real-time SOCP solvers has already been considered for tasks such as landing of spacecraft (e.g., for NASA's Mars exploration project) [12].

A particular example of a barrier function computed for the controller u_1 is shown in Figure 4. The obstacles are shown in red and the initial state of the UAV is also plotted. The 1-level set of the computed barrier is plotted in green and certifies that the initial state is *guaranteed* to remain safe when the controller is executed.

Figure 5 demonstrates the performance of the algorithm described above with SDSOS programming used to compute barrier functions on a number of environments. Each subfigure shows a randomly chosen environment (with obstacle positions chosen from the uniform distribution) with circular obstacles that the UAV has to navigate. The trajectory traversed by the UAV following the described planning algorithm is indicated in these plots and remains collision free in each case. Note that the original (non-Taylor expanded) dynamics are used for the simulations.

We end the discussion of this example by comparing running times and performance of the SDSOS and SOS approaches to this problem. In order to do this, we fix the initial state of the vehicle to be $(0, 0, \psi_0)$ for varying values of ψ_0. For each ψ_0, we randomly sample 100 different environments containing two obstacles each. The obstacles are disks of radius 0.03 m with centers (x_c, y_c) uniformly sampled in the range $x_c \in [-0.2, 0.2]$ m, $y_c \in [0, 0.2]$ m. For each environment, we attempt to find a valid barrier certificate for the first controller in our library (i.e., the one that servos the vehicle to $\psi_{des,1} = 0$). The results are summarized in Table 1 which presents the number of environments (out of 100) for which a barrier certificate was successfully found using SDSOS and SOS programming. As the table illustrates, the number of times SDSOS programming *fails* to find a barrier certificate when SOS programming *succeeds* is quite small.

We also compare running times of the two approaches in Figure 6. We use the Gurobi SOCP solver [1] for the SDSOS problems and SeDuMi [42] as the SDP solver for SOS problems. As the histograms of running times illustrate, the SDSOS approach is significantly faster than the SOS approach. We note that while the MOSEK SOCP/SDP solvers are typically faster, we were unable to make these work on this problem due to numerical issues.

ψ_0	0°	10°	20°	30°	40°
SDSOS	66 %	59 %	70 %	68 %	56 %
SOS	68 %	62 %	70 %	76 %	65 %

Table 1: Comparison of percentage of times a valid barrier certificate was found using SDSOS and SOS programming for randomly sampled obstacle environments and initial yaw angles. (Only the ratio between the two is meaningful here.)

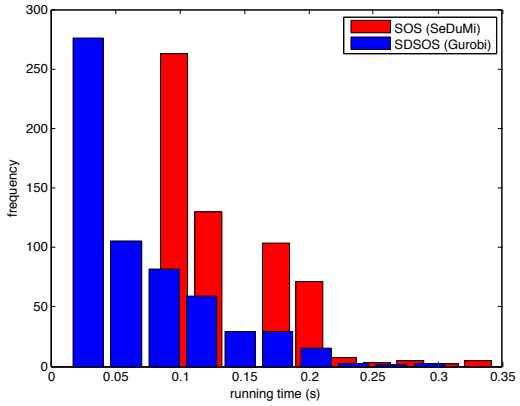


Figure 6: Histograms of running times for SOS and SDSOS approaches on the collision avoidance problem.

6 Nonlinear Control Design for a Quadrotor Model

Quadrotors (see Figure 7) have recently been recognized as a popular platform for academic research in systems theory due to their agile maneuvering capabilities and inexpensive cost [31, 20]. They have also been considered for the task of load transportation, not only in laboratory settings [32][5], but also by the aerospace companies Bell and Boeing and the online retail company Amazon[6]. In this section, we consider the problem of designing a nonlinear stabilizing feedback controller for the quadrotor's hovering configuration, which is relevant to almost all of its applications. In addition to a stabilizing controller, we also obtain a *formal certificate* of stability of the resulting system. This certificate takes the form of an inner approximation of the region of attraction (ROA), i.e., the set of initial conditions the controller is guaranteed to stabilize to the goal position.

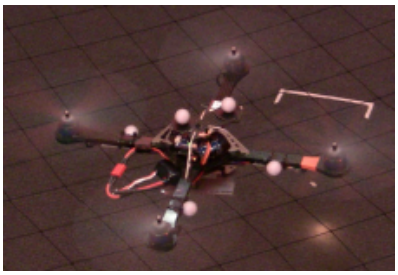


Figure 7: We design a hovering controller for the quadrotor model described in [30]. (Image from [30].)

We use the dynamics model described in [30] for our numerical experiments. The model includes 16

[5]A video corresponding to the paper is available at https://www.youtube.com/watch?v=YBsJwapanWI
[6]https://www.youtube.com/watch?v=Le46ERPMlWU

states:

$$x := [x_1, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r, \omega_1, \omega_2, \omega_3, \omega_4],$$

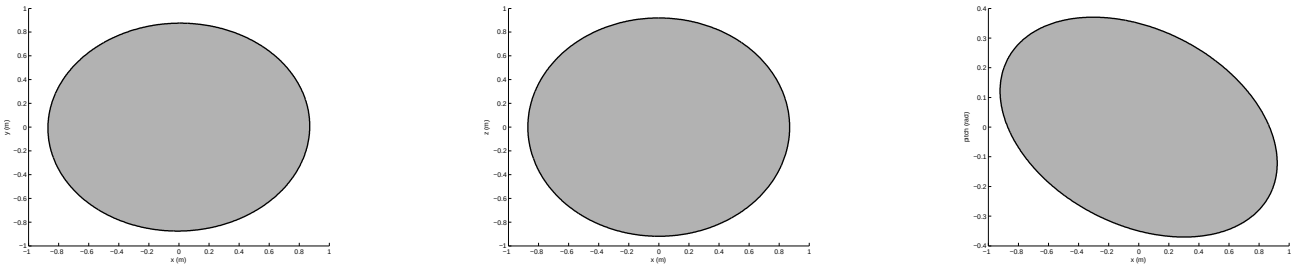
where x_1, y, z are the coordinates of the center of mass of the system, ϕ, θ, ψ are the Euler angles describing its orientation, p, q, r are angular velocities of the quadrotor expressed in the body frame, and $\omega_i, i = 1, ..., 4$, are the angular speed of the rotors. The rotor angular speeds cannot be controlled directly and have nontrivial dynamics. The control inputs of the system are thus the *desired* speed of the rotors (the rotors take some time to catch up to the desired speed).

In the end our system takes the form $\dot{x} = f(x) + g(x)u(x)$ with f and g given and the control u as a decision function. We use the method presented in our earlier work [25] in collaboration with Russ Tedrake to design a hovering controller u for the system. The fixed point corresponding to the hovering configuration has the first twelve states of the system equaling 0 but with non-zero rotor speeds ω_i counteracting the force of gravity. The dynamics of the system are Taylor expanded to degree 3 in order to obtain polynomial dynamics. We search for a degree 2 Lyapunov function $V(x)$ and a degree 3 feedback controller $u(x)$ in order to maximize the size of the region of attraction (ROA) of the resulting closed-loop system (i.e., the differential equation with $u(x)$ plugged in). We use SDSOS programming since the state space is too large for SOS programming to handle, causing our computer to run out of memory. The resulting optimization problem is:

$$\begin{aligned} \max_{\rho, L(x), V(x), u(x)} \quad & \rho \\ \text{s.t.} \quad & V(x) \in SDSOS_{16,2} \\ & -\dot{V}(x) + L(x)(V(x) - \rho) \in SDSOS_{16,6} \\ & L(x) \in SDSOS_{16,4} \\ & \sum_j V(e_j) = 1. \end{aligned} \tag{12}$$

Here, $L(x)$ is a nonnegative multiplier term and e_j is the j-th standard basis vector for the state space $\mathbb{R}^{16}$. From our discussion in Section 4, it is easy to see that the above conditions are sufficient for establishing $B_\rho = \{x \in \mathbb{R}^{16} \mid V(x) \leq \rho\}$ as an inner estimate of the region of attraction for the system. When $x \in B_\rho$, the second constraint implies that $\dot{V}(x) < 0$ (since $L(x)$ is constrained to be nonnegative). The last constraint normalizes $V(x)$ so that maximizing the level set value ρ leads to enlarging the volume of the ROA.

The optimization problem (12) is not convex in general since it involves conditions that are bilinear in the decision variables. However, problems of this nature are common in the SOS programming literature (see e.g. [21]) and are typically solved by iteratively optimizing groups of decision variables. Each step in the iteration is then a SDSOS program. This iterative procedure is described in more detail in [25] and can be initialized with the Lyapunov function from a Linear Quadratic Regulator (LQR) controller [7]. The iterations are terminated when the objective changes by less than 1 percent.



(a) x – y subspace (b) x – z subspace (c) x – pitch subspace

Figure 8: Slices in different subspaces of the hovering ROA of the quadrotor system.

An important observation is that unlike the sets $POS_{n,d}$ and $SOS_{n,d}$, the sets $DSOS_{n,d}$ and $SDSOS_{n,d}$ are not invariant to coordinate transformations, i.e., a polynomial $p(Ax)$ is not necessarily dsos (resp. sdsos) even if $p(x)$ is dsos (resp. sdsos). Thus, performing coordinate transformations on the problem data (e.g., on the state variables of a dynamical system) can sometimes have an important effect. We describe a particular coordinate transformation that is intuitive and straightforward to implement. It can be used for problems involving the search for Lyapunov functions, and can potentially be extended to other problems as well. In particular, given a Lyapunov function $V(x)$ we find an invertible affine transformation that simultaneously diagonalizes the Hessians of $V(x)$ and $-\dot{V}(x)$ evaluated at the origin (this is always possible for two positive definite matrices). The intuition behind the coordinate change is that the functions $V(x)$ and $-\dot{V}(x)$ locally resemble functions of the form x^TDx (with D diagonal), which are dsos polynomials that are "far away" from the boundary of the DSOS (and hence SDSOS) cone. We solve the optimization problem (12) after performing this coordinate transformation. The transformation is then inverted to obtain ROAs in the original coordinate frame.

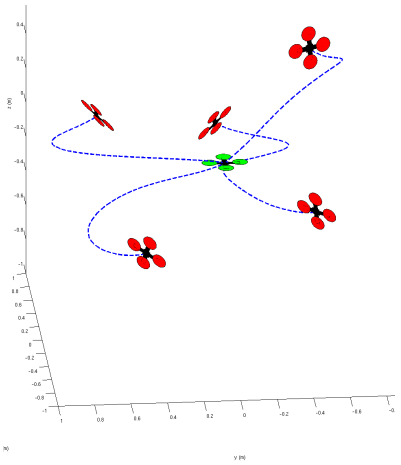


Figure 9: A sampling of five initial conditions that are stabilized by our controller. The goal position is shown in green, the stabilized initial conditions in red, and the intermediate trajectories in blue.

Each iteration of the algorithm employed for solving the optimization problem (12) takes approximately 15 minutes, with convergence occurring between 15 and 20 iterations. Figure 8 shows slices of the computed ROA in multiple subspaces of the state space. As the plot illustrates, we are able to verify stability of the closed loop system for a large set of initial conditions. A qualitative demonstration of the performance of the controller is given in Figure 9. The system is started off from five different initial conditions (shown in red) and our nonlinear hovering controller is applied. The resulting trajectory is shown in blue. In each case the quadrotor is able to stabilize itself to the goal configuration (green).

7 Conclusions

In this paper, we demonstrated three applications of optimization problems over the set of nonnegative polynomials that may be of interest in operations research and transportation engineering. We hope to have conveyed the message that the problem of certifying polynomial inequalities appears in more diverse areas than one might think. There are powerful tools for approaching this problem based on the sum of squares relaxation and semidefinite programming. We believe that our recently introduced techniques of DSOS and SDSOS optimization, which are LP and SOCP-based alternatives to sum of squares programming, can pave the way to new applications of algebraic techniques in optimization—in particular, applications that are large-scale or real-time.

8 Acknowledgements

The authors would like to thank Pablo Parrilo, Russ Tedrake, and the MIT Robot Locomotion Group for many helpful discussions that have contributed greatly to this paper. The authors would also like to acknowledge the use of the software package Drake (https://github.com/RobotLocomotion/drake/wiki) developed by the Robot Locomotion Group for formulating the dynamics in the quadrotor example, along with the SPOTless software developed by Mark Tobenkin, Frank Permenter and Alexandre Megretski for processing the SOS programs in our examples. Finally, we are very grateful for receiving constructive criticism from a referee that led to improvements in this paper.

References

[1] Gurobi optimizer reference manual. *URL: http://www. gurobi. com*, 2012.

[2] *MOSEK reference manual*, 2013. Version 7. Latest version available at `http://www.mosek.com/`.

[3] A. A. Ahmadi. *Algebraic relaxations and hardness results in polynomial optimization and Lyapunov analysis*. PhD thesis, Massachusetts Institute of Technology, September 2011. Available at `http://aaa.princeton.edu/publications`.

[4] A. A. Ahmadi and A. Majumdar. DSOS and SDSOS optimization: LP and SOCP-based alternatives to sum of squares optimization. In *Proceedings of the 48th Annual Conference on Information Sciences and Systems*. Princeton University, 2014.

[5] A. A. Ahmadi and A. Majumdar. DSOS and SDSOS optimization: More tractable alternatives to SOS optimization. *In preparation (http://aaa.princeton.edu/publications)*, 2014.

[6] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical programming*, 95(1):3–51, 2003.

[7] B. D. O. Anderson and J. B. Moore. *Optimal control: linear quadratic methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[8] A. J. Barry, A. Majumdar, and R. Tedrake. Safety verification of reactive controllers for UAV flight in cluttered environments using barrier certificates. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 484–490. IEEE, 2012.

[9] O. Berman, P. Jaillet, and D. Simchi-Levi. Location-routing problems with uncertainty. *Facility location: a survey of applications and methods*, 106:427–452, 1995.

[10] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[11] D. Bertsimas and J. N. Tsitsiklis. Introduction to linear optimization. 1997.

[12] L. Blackmore, B. Acikmese, and D. P. Scharf. Minimum landing error powered descent guidance for Mars landing using convex optimization. *AIAA Journal of Guidance, Control, and Dynamics*, 33, 2010.

[13] E. G. Boman, D. Chen, O. Parekh, and S. Toledo. On factor width and symmetric h-matrices. *Linear algebra and its applications*, 405:239–248, 2005.

[14] C. W. Commander. *Optimization problems in telecommunications with military applications*. PhD thesis, University of Florida, 2007.

[15] C. W. Commander, P. M. Pardalos, V. Ryabchenko, O. Shylo, S. Uryasev, and G. Zrazhevsky. Jamming communication networks under complete uncertainty. *Optimization Letters*, 2(1):53–70, 2008.

[16] C. W. Commander, P. M. Pardalos, V. Ryabchenko, S. Uryasev, and G. Zrazhevsky. The wireless network jamming problem. *Journal of Combinatorial Optimization*, 14(4):481–498, 2007.

[17] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076. IEEE, 2013.

[18] L. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[19] D. Hilbert. Über die Darstellung Definiter Formen als Summe von Formenquadraten. *Math. Ann.*, 32, 1888.

[20] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pages 1–20, 2007.

[21] Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard. Some controls applications of sum of squares programming. In *42nd IEEE Conference on Decision and Control*, volume 5, pages 4676 – 4681, December 2003.

[22] J. K. Karlof. *Integer programming: theory and practice*. CRC Press, 2005.

[23] H. Khalil. *Nonlinear systems*. Prentice Hall, 2002. Third edition.

[24] M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, pages 157–270. Springer, 2009.

[25] A. Majumdar, A. A. Ahmadi, and R. Tedrake. Control design along trajectories with sums of squares programming. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[26] A. Majumdar, A. A. Ahmadi, and R. Tedrake. Control and verification of high-dimensional systems via DSOS and SDSOS optimization. In *Proceedings of the 53rd IEEE Conference on Decision and Control*, 2014.

[27] A. Majumdar and R. Tedrake. Robust online motion planning with regions of finite time invariance. In *Algorithmic Foundations of Robotics X*, pages 543–558. Springer, 2013.

[28] J. Mattingley and S. Boyd. Real-time convex optimization in signal processing. *Signal Processing Magazine, IEEE*, 27(3):50–61, 2010.

[29] J. Mattingley and S. Boyd. CVXGEN: a code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

[30] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *Proceedings of the 12th International Symposium on Experimental Robotics (ISER 2010)*, 2010.

[31] D. Mellinger, N. Michael, M. Shomin, and V. Kumar. Recent advances in quadrotor capabilities. *2011 IEEE International Conference on Robotics and Automation*, May 2011.

[32] D. Mellinger, M. Shomin, N. Michael, and V. Kumar. Cooperative grasping and transport using multiple quadrotors. In *Proceedings of the international symposium on distributed autonomous robotic systems*, 2010.

[33] K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.

[34] J. Nie and M. Schweighofer. On the complexity of Putinar's Positivstellensatz. *Journal of Complexity*, 23(1):135–150, 2007.

[35] P. A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, May 2000.

[36] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2, Ser. B):293–320, 2003.

[37] I. Pólik and T. Terlaky. A survey of the S-lemma. *SIAM Review*, 49(3):371–418, 2007.

[38] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.

[39] S. Prajna, A. Jadbabaie, and G. J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428, 2007.

[40] M. Putinar. Positive polynomials on compact semialgebraic sets. *Indiana University Mathematics Journal*, 42(3):969–984, 1993.

[41] G. Stengle. A Nullstellensatz and a Positivstellensatz in semialgebraic geometry. *Mathematische Annalen*, 207(2):87–97, 1974.

[42] J. Sturm. *SeDuMi version 1.05*, Oct. 2001. Latest version available at `http://sedumi.ie.lehigh.edu/`.

[43] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, Mar. 1996.