Cubic interpolation to certify nonnegativity

Mitchell Tong Harris Pablo A. Parrilo

December 15, 2023

Abstract

We propose a method for semi-infinite optimization problems that replaces function nonnegativity constraints by nonnegative constraints of the functions' cubic interpolants. The cubic nonnegativity constraint are second order cone constraints. The second order cone problem is more efficient to solve than semidefinite programs required by sum of squares implementations of a high degree polynomial approximation. Common local methods that rely on linear interpolation of the domain require finer discretizations than is needed when using cubic approximation. We present theoretical guarantees for our method, theoretical comparisons to the linear programming competitor, and applications demonstrating the empirical benefits of using cubic interpolation.

1 Introduction

Chebyshev's eponymous polynomials arose when he searched, in 1853, for a that minimizes

$$\sup_{-1\le x\le 1}\left|x^n-\sum_{i=0}^{n-1}a_ix^i\right|, \tag{1}$$

which would give an approximation to x^n by a lower degree polynomial [10, §1]. The objective in (1) is the *supremum norm* of the approximation error. In general, minimizing the supremum norm of a function is difficult, but it is useful especially in this context of finding the best uniform approximation of a function. In the case of approximating an arbitrary function, the problem is called Chebyshev Approximation and is discussed further in Section 4.2.

Chebyshev Approximation is an example of a *semi-infinite program* (SIP), which is the class of optimization problems with finitely many decision variables but infinitely many constraints. The name SIP was first coined in 1962 [11], and the theory behind and techniques for SIPs have continued to attract much attention due to their wide applicability, including applications to approximation theory, economics, and optimal control [22, 35, 49]. An exhaustive bibliography on SIPs has been collected by López and Still [34].

Many SIPs can be written as optimization problems in which a function is constrained to be nonnegative on some domain. Each value of x in the domain provides one of an infinite collection of constraints. In this paper, we propose a new method to approximate SIPs that can be written as

$$\begin{aligned}\min_{a\in\mathbb{R}^N}\quad & c^Ta\\ \text{s.t.}\quad & f(a;x)\ge 0 \text{ for all } x\in[-1,1]\end{aligned} \tag{General-SIP}$$

for some $c\in\mathbb{R}^N$ and a function $f(a;x)$ that is linear in a and for any fixed a, $f(a;x)$ is continuous for $x\in[-1,1]$ and smooth for $x\in(-1,1)$. The techniques we present easily generalize to several constraints of the same form and to intervals other than $[-1,1]$. Later, we will take derivatives of $f(a;x)$, and by this we always mean with respect to x. Often we will write $f(a)$ to implicitly mean $f(a;x)$.

The main tool is a novel, efficient characterization of nonnegativity, which is a combination of simple ideas from approximation theory and convex optimization. We combine three ideas:

1. Approximation theory: A cubic interpolant is a good approximation for a function. An example of how much better it may be than a linear interpolant, even with far fewer grid points, is given in Figure 1.

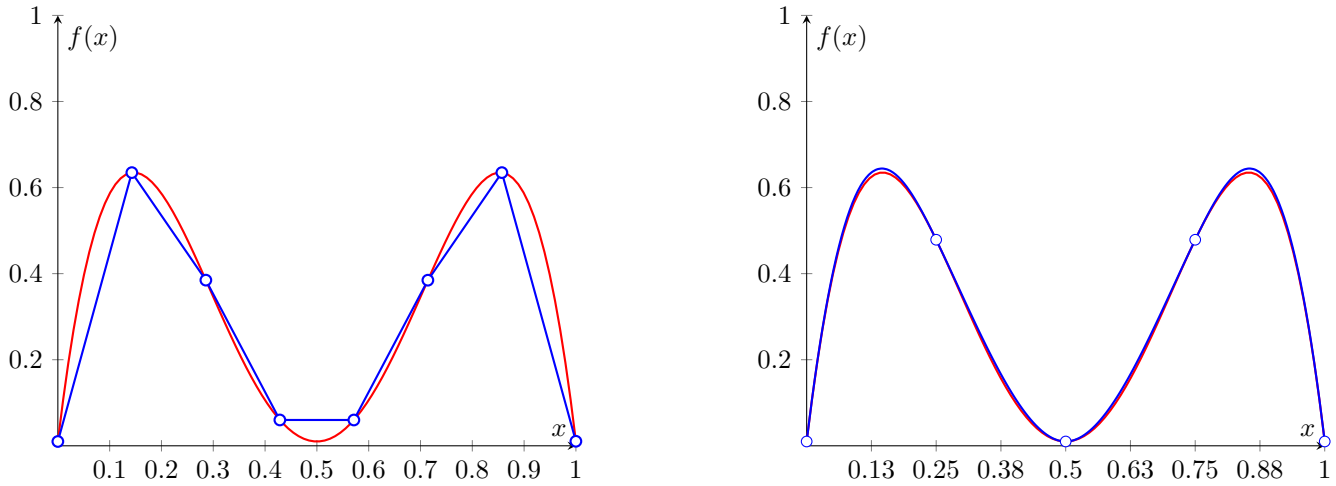


Figure 1: Comparison of cubic interpolation and linear interpolation of the function $f(x) = 10x(1-x)(1-2x)^2 + 0.01$. Even with fewer grid points, the approximation accuracy of cubic interpolation is better than linear interpolation. To guarantee $f(x) \geq 0$, we only need to show that on each subinterval the interpolant is always bigger than the maximum possible approximation error.

2. Convex optimization: Constraining a lower bound of a cubic polynomial on an interval is an inexpensive second order cone constraint.

3. Triangle inequality: If the cubic interpolant always exceeds the maximum possible approximation error, then the function must always be nonnegative.

These ideas have been independently used as described in Section 1.1, but taken together they provide a powerful alternative to other SIP techniques.

Further contributions of this work include theoretical guarantees for the method and comparisons, both theoretical and numerical, to a similar technique based on linear programming (LP). Additionally, we can easily certify tighter bounds for particular cases of rational approximation of $|x|$ than originally proven by [41]. Finally, as opposed to other methods based on splines, we demonstrate how we can find approximations in terms of bump functions, which are not polynomials. A number of other applications are presented in Section 4 for which we use the publicly-released software implementation described in Section 5.

1.1 Related work

Decades of work on SIPs have produced several techniques, which can generally be divided into two main categories: methods that iteratively refine infeasible solutions and methods that search over larger and larger subsets of the feasible set. The method we propose is most similar to the second class, but we mention them both for completeness.

1.1.1 Infeasible methods

For this class of methods, described in [26, §7], one iterates over infeasible solutions until eventually one can prove feasibility. In each step, a program with a finite number of constraints is solved. One example is an *exchange method* [12, 29] where one or more violated constraints may be found by solving a secondary optimization problem, and then the larger program with a finite number of constraints is solved again. This reduces to a cutting plane algorithm if exactly one constraint is added and none are removed. Another example is *grid refinement* [25, 48]. In the simplest case, one starts with a grid of the domain and solves the with those finite number of constraints. Doubling the resolution of the grid gives another, finite problem albeit with many more constraints. The difficulty in designing and implementing such methods lies in deciding which points in the refined grid are necessary so that the number of constraints does not become unwieldy.

Finally, *local reduction* methods are designed to add constraints not by finding a violated constraint or using a pre-picked grid, but by a middle ground: if $f(a;x) \geq 0$ is the constraint for decision variable a and x in some compact set, then one can take the current iterate a^i and use all the local minima t_j of $f(a^i;x)$ as additional discretization points of the semi-infinite constraint.

In practice, the authors of [26] suggest a combination of these techniques because the strategies suffer from several issues. First, local convergence is usually not guaranteed, and so generally no complexity analysis is given. In the case of exchange methods, for instance, the number of constraints may increase a lot on any iteration. Second, both exchange and local reduction methods require solving an inner optimization problem whose complexity depends on properties of the semi-infinite constraints. Finally, a problem affecting convergence of any infeasible method is that it is unknown how many constraints are active at optimality. If there are an infinite number, then it's possible that as the solutions are converging to the true feasible solution, the number of constraints maintained in each finite optimization problem becomes intractable. In addition to the algorithmic issues, a drawback of any method where each iterate is infeasible is that no information about the true solution is attained until the method terminates with a feasible solution.

1.1.2 Inner approximations to feasible set

Often there exist reasonably large subsets of the feasible set over which optimization is efficient. Optimizing over an inner approximation of the feasible set guarantees that the solution is also feasible for the original SIP, therefore every inner approximation provides a bound for the original SIP. We discuss two methods based on how they provide an inner approximation.

Bounded variations between knot points Whereas grid refinement methods only use the information at a finite set of points, this information can be combined with the derivatives of the constraint function. The second derivative can be used to bound the variation of the function between knot points (first proven by [50] and used by e.g. [4,37]). If the function value at the knot points exceeds the possible variation bound between knot points, then the function must be nonnegative on the whole interval. Such constraints can be encoded in a linear program (LP). Note that this method can be used even when the constraints are not polynomials. Furthermore, there is no significant incurred cost when increasing the degrees of constraints. The method we propose, which is closely related to variation bounding, scales just as efficiently in the degree of the constraints.

Throughout the paper, we develop our proposal in parallel to explaining the details of the simple idea of bounded variations. Such an exposition clarifies the purpose of each ingredient and also highlights the differences with our method. Even though the descriptions of the methods are similar, the theoretical guarantees are asymptotically different, and our proposal outperforms the baseline method on every numerical example.

Sum of squares There is a long line of research on SIPs with polynomial nonnegativity constraints, going back to [40, 53], based on the idea that a sum of squares (SOS) of polynomials is nonnegative. A similar characterization of nonnegativity based on sums of squares (e.g. [54, p. 4]) also exists for the case of polynomial nonnegativity on an interval. The work of Lasserre [32] and Parrilo [44] show how to use SOS characterizations to efficiently optimize (via semidefinite programs (SDPs) [55]) with nonnegativity constraints. Unfortunately, the size of the SDP increases with the degree of the polynomials constrained to be nonnegative. Additionally, by their very nature, this technique only works for polynomials.

1.1.3 Optimizing over splines

Interpolation is a powerful and widely exploited tool in numerical analysis [14, 27, 47] that comes in several flavors. Two options for interpolation include constructing a high degree polynomial that matches either the values or both the values and derivatives of the function – Lagrange or Hermite interpolants, respectively. Often, high degree polynomial interpolants suffer from large oscillations [7, §3.5]. Nonetheless, these interpolants have been successfully applied to numerical applications including global optimization [18] and solving differential equations [52].

Piecewise low degree polynomials, called splines, are a third option for interpolation [15]. Perhaps the first appearance was the use of Bézier curves [6] for computer aided design in the automobile industry [21]. Since then, computation with splines has been the subject of research in its own right [17, 51], and these tools have become well-established in computer aided geometric design [20, 38].

Splines have been useful in a wide variety of fields. Some applications include stability analysis [30], digital signal processing [28], turbulent flows [46], image processing [24], and the list goes on. Recent work on splines has exploited their simple structure in the context of optimization. For example, constraining that a low degree spline is nonnegative is inexpensive to enforce and can encode a wide variety of problems, as explored in Papp's PhD thesis [42] and other work [43, 56]. In all of these applications, using splines in place of high degree polynomials has been attractive because of computational challenges (both algorithmic scaling and numerical stability) associated with the latter. We build on this technique to handle constraints on arbitrary functions, which we approximate by piecewise cubic interpolants. We not only use splines to approximate functions as has been done, but our framework supports simultaneous estimation of the function in other ways, such as by high degree polynomials.

2 Certifying nonnegativity constraints with function interpolants

For this whole section, assume a is a fixed constant. We therefore write $g(x)$ in place of $f(a;x)$. In this section we develop a method to certify that

$$g(x) \geq 0 \text{ for all } x \in [-1,1]. \tag{2}$$

This constraint is the characteristic property of (General-SIP), and to approximate solutions to the SIP, we will later optimize over these certificates in Section 3.

The following lemma shows that an auxiliary function p and a number δ can serve as a *certificate* of nonnegativity. Before saying how to find such a certificate, we note that if we can find p and δ satisfying two inequalities, then that proves (2).

Lemma 1. *If there exists a function $p(x)$ such that for all $x \in [-1,1]$*

$$|p(x) - g(x)| \leq \delta \qquad \text{and} \qquad p(x) - \delta \geq 0, \tag{3}$$

then (2) holds.

Proof. The first inequality implies that $p(x) - g(x) \leq \delta$, so then $g(x) \geq p(x) - \delta \geq 0$. □

In the rest of this section, we describe two methods of constructing p in such a way that guarantees the first inequality in (3) and for which there is a simple method of testing the second inequality. Both schemes require a discretization of the interval $[-1,1]$. We call the discretization points *knot points*. For simplicity, throughout this work we assume uniform knot point spacing and let h be the spacing.

2.1 Linear interpolants

Let $-1 = x_0 < x_1 < \cdots < x_K = 1$. Let $g : \mathbb{R} \to \mathbb{R}$ be continuous on $[x_0, x_K]$. For each $0 \leq i < N$, let p_i be the unique linear polynomial such that $p_i(x_i) = g(x_i)$ and $p_i(x_{i+1}) = g(x_{i+1})$. The *linear interpolant* of g on the knot points $x_0, x_1, \ldots, x_K$ is the piecewise linear function such that if $x \in [x_i, x_{i+1}]$, then $p(x) = p_i(x)$.

Using linear interpolants as p in Lemma 1 is a well-known idea. We mention it here to compare with our method in Section 2.2. To guarantee (3) we need two ingredients: a bound on the interpolation error and a way to certify nonnegativity. The interpolation error is given by the following lemma.

Lemma 2 (Theorem 1 of [50])**.** *Suppose g is twice-differentiable and $|g^{(2)}(x)| \leq \mu_2$ for every $x \in [-1,1]$. Let p be the linear interpolant of g on $[-1,1]$ with knot spacing h. Then*

$$|g(x) - p(x)| \leq \frac{h^2 \mu_2}{8} \tag{4}$$

for every $x \in [-1,1]$.

The following tool is how to check nonnegativity of a piecewise linear function.

Lemma 3. *Let $p(x)$ be piecewise linear on an interval with knot points $x_1, \ldots, x_K$. If*

$$p(x_i) \geq 0 \text{ for all } i = 1, \ldots, K, \tag{5}$$

then $p(x) \geq 0$ on the whole interval.

Proof. This follows because the minimum of a linear function on a subinterval occurs at a knot point. □

Therefore, given just $g(x_0), g(x_1), \ldots, g(x_K)$ and a bound μ_2 on $|g^{(2)}(x)|$ over the interval, one can efficiently certify nonnegativity of g on the interval by checking

$$\min_i g(x_i) \geq \frac{h^2 \mu_2}{8}, \tag{6}$$

because then the linear interpolant p of g satisfies the conditions of Lemma 1.

2.2 Cubic interpolants

Let $g : \mathbb{R} \to \mathbb{R}$ be continous on $[x_0, x_K]$ and differentiable on (x_0, x_K). For each $0 \leq i < K$, let p_i be the unique cubic polynomial for which

$$p_i(x_i) = f(x_i), \quad p_i(x_{i+1}) = f(x_{i+1}), \quad p_i'(x_i) = f'(x_i), \quad p_i'(x_{i+1}) = f'(x_{i+1}). \tag{7}$$

The *cubic Hermite interpolant* of f on the knot points $x_0, x_1, \ldots, x_K$ is the piecewise polynomial function such that if $x \in [x_i, x_{i+1}]$, then $p(x) = p_i(x)$. We write the map from functions to interpolants as $p = \text{CubicInterpolant}(f)$, where the knot points have been suppressed. The definition directly implies that the cubic Hermite interpolant always has continuous first derivative. Throughout this work we will use *cubic interpolant* and *cubic Hermite interpolant* interchangeably.[1]

We specify how to certify the conditions of (3) for a cubic interpolant p. Again, we specify the two ingredients necessary: a bound on the interpolation error and a way to efficiently certify nonnegativity. The error is given by the following lemma.

Lemma 4. *Suppose g is four-times differentiable and $|g^{(4)}(x)| \leq \mu_4$ for every $x \in [-1, 1]$. Let $p = \text{CubicInterpolant}(g)$ with knot spacing h. Then*

$$|g(x) - p(x)| \leq \frac{h^4 \mu_4}{384} \tag{8}$$

for every $x \in [-1, 1]$.

Proof. On the subinterval $[x_i, x_{i+1}]$, our setting is the case of $k = 0$ and $m = 2$ in Theorem 9 of [13]. This gives

$$|g(x) - p(x)| \leq \frac{\mu_4}{4!}((x - x_i)(x_{i+1} - x))^2. \tag{9}$$

The maximum of $((x - x_i)(x_{i+1} - x))^2$ occurs when $x = x_i + \frac{h}{2}$, giving a maximum value of $\frac{h^4}{16}$. □

Next, we discuss how to certify nonnegativity of a cubic interpolant. The crucial tool used to check nonnegativity is the second order cone, which can be defined as follows.

Definition 1. *The three-dimensional* rotated second order cone $\mathcal{Q} \subset \mathbb{R}^3$ *is the set*

$$\mathcal{Q} = \{(x_0, x_1; x_2) \in \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R} \,|\, 2x_0 x_1 \geq x_2^2\}. \tag{10}$$

[1] While this work focuses on Hermite interpolants, it is possible to reformulate everything in terms of natural splines [7]. In that case, the derivatives of the function are not needed. There is a linear transformation taking the function values to the coefficients of the cubic polynomials, and there is an error guarantee of the same order. We chose to work with Hermite interpolants because in our applications the derivatives are easy to get and the constant factor approximation bound is slightly better. All the results still translate to the natural spline case.

This is a closed, convex cone, over which optimization is efficient. See [2] for more background on this set. While higher order interpolants will have less approximation error, the analogous constraints do not reduce to an efficient second order cone constraint. See Section 1.4 of [33] for a comparison of second order cone complexity to the corresponding semidefinite program, which is the general constraint needed in the higher degree case.

A second order cone problem (SOCP) is sufficient to certify $p - \delta \geq 0$ on the interval because of the following equivalent condition.

Lemma 5. *A cubic polynomial p is nonnegative on an interval $[x_0, x_1]$ with length h if and only if there exist $a, b \in \mathbb{R}$ such that*

$$(2p(x_0),\ 3p(x_1) - hp'(x_1) - a;\ b) \in \mathcal{Q} \qquad \text{and} \qquad (3p(x_0) + hp'(x_0) - b,\ 2p(x_1);\ a) \in \mathcal{Q}. \tag{11}$$

This is proven in Appendix B. Therefore, given just $g(x_0), \ldots, g(x_N)$, $g'(x_0), \ldots, g'(x_N)$, and a bound μ_4 on $|g^{(4)}(x)|$, we can certify nonnegativity of g on $[-1, 1]$ by letting $\delta = C_0 \mu_4 h^4$ and checking if there exist $a_i, b_i \in \mathbb{R}$ such that

$$\begin{aligned} &\big(2(g(x_i) - \delta),\quad 3(g(x_{i+1}) - \delta) - hg'(x_{i+1}) - a_i;\quad b_i\big) \in \mathcal{Q} \qquad \text{and} \\ &\big(2(g(x_{i+1}) - \delta),\quad 3(g(x_i) - \delta) + hg'(x_i) - b_i;\quad a_i\big) \in \mathcal{Q} \end{aligned} \tag{12}$$

for every $0 \leq i < N$ because then the cubic interpolant p of g satisfies the conditions of Lemma 1.

3 Optimization with new constraints

In this section we use the main idea from Section 2 to write down approximations to (General-SIP), present guarantees for the approximations (Theorems 1, 2, 3, and 4), and compare those guarantees in Section 3.3. We conclude in Section 3.4 by commenting on how to get derivative bounds in practice.

Inspired by Lemma 1, the approximations to (General-SIP) take the following form:

$$\begin{aligned} \min_{a \in \mathbb{R}^N} \quad & c^T a \\ \text{s.t.} \quad & |p(x) - f(a; x)| \leq \delta \text{ for all } x \in [-1, 1] \\ & p(x) - \delta \geq 0 \text{ for all } x \in [-1, 1] \end{aligned} \tag{General-Approx}$$

Depending on the form of p, we get the different nonnegativity tests described in Sections 2.1 (LP) and 2.2 (SOCP).

3.1 Linear interpolants for constraints

The linear programming (LP) approach to approximate (General-SIP) is to substitute the conditions in Section 2.1 into (General-Approx) to get

$$\begin{aligned} \min_{a \in \mathbb{R}^N} \quad & c^T a \\ \text{s.t.} \quad & \mu_2 \geq |f^{(2)}(a; x)| && \forall x \in [-1, 1] \\ & \frac{1}{8} h^2 \mu_2 \leq \delta \\ & f(a; x_i) - \delta \geq 0 && \forall i \end{aligned} \tag{General-LP}$$

where the knot points of interpolation are $\{x_i\}$. The derivative of f is taken with respect to x. Here δ represents an upper bound on the error of the linear interpolation. Roughly the idea is to use a bound on the linear interpolant to imply a bound on $f(a)$.

It is reasonable to call this an LP because all of the constraints are linear. A missing piece to fully specify the LP is the bound on the derivative. We discuss LP-representable options for incorporating bounds on the derivatives in Section 3.4.

See Appendix A for proofs on the following guarantees of (General-LP).

Theorem 1 (Soundness of (General-LP))**.** *Let a be feasible for* (General-LP)*. Then a is feasible for* (General-SIP)*, and so $c^T a$ is an upper bound for* (General-SIP)*.*

Theorem 2 (Completeness of (General-LP))**.** *Suppose that there exists a feasible a^* for* (General-SIP) *and $\epsilon \geq 0$ such that for every $x \in [-1, 1]$, $f(a^*; x) \geq \epsilon$. Suppose μ_2 satisfies $|f^{(2)}(a^*; x)| \leq \mu_2$ for every $x \in [-1, 1]$. If the knot points are chosen such that h satisfies*

$$\epsilon \geq \frac{1}{8}\mu_2 h^2, \tag{13}$$

then a^* is feasible for (General-LP)*.*

3.2 Cubic interpolants for constraints

The second-order cone programming (SOCP) approach to approximate (General-SIP) is to substitute the conditions in Section 2.2 into (General-Approx) to get

$$\begin{aligned} \min_{a \in \mathbb{R}^N} \quad & c^T a \\ \text{s.t.} \quad & p(x) = \text{CubicInterpolant}(f(a; x)) \\ & \mu_4 \geq |f^{(4)}(a; x)| && \forall x \in [-1, 1] \\ & C_0 h^4 \mu_4 \leq \delta \\ & p(x) - \delta \geq 0 && \forall x \in [-1, 1]. \end{aligned} \tag{General-SOCP}$$

The derivative of f is taken with respect to x. Here δ represents an upper bound on the error of the cubic interpolation. Roughly the idea is to use a bound on the cubic interpolant to imply a bound on $f(a)$.

It is reasonable to call this an SOCP because one can encode these constraints with (12). A missing piece to fully specify the SOCP is the bound on the derivative. See Section 3.4 for LP-representable options for incorporating bounds on the derivatives.

See Appendix B for the proofs of these guarantees about (General-SOCP).

Theorem 3 (Soundness of (General-SOCP))**.** *Let a be feasible for* (General-SOCP)*. Then a is feasible for* (General-SIP)*, and so $c^T a$ is an upper bound for* (General-SIP)*.*

Theorem 4 (Completeness of (General-SOCP))**.** *Suppose that there exists a feasible a^* for* (General-SIP) *and $\epsilon \geq 0$ such that for every $x \in [-1, 1]$, $f(a^*; x) \geq 2\epsilon$. Suppose μ_4 satisfies $|f^{(4)}(a^*; x)| \leq \mu_4$ for every $x \in [-1, 1]$. If the knot points are chosen such that h satisfies*

$$\epsilon \geq \frac{\mu_4 h^4}{384}, \tag{14}$$

then a^* is also feasible for (General-SOCP)*.*

3.3 Theoretical comparison of programs resulting from each interpolant type

Theorems 1 and 3 imply that both programs will give valid upper bounds for (General-SIP). A particular case of Theorems 2 and 4 is the following corollary.

Corollary 1. *If a minimizer for* (General-SIP) *is strictly feasible, then* (General-LP) *and* (General-SOCP) *will have the same minimum as* (General-SIP) *when the knot spacing is chosen sufficiently small.*

Proof. When a^* is strictly feasible, that means ϵ can be strictly positive. Then (13) and (14) prescribe a positive number for h so that the minimizer of (General-SIP) is feasible for each respective program. □

We now compare which method can reach this bound with fewer knot points. Let h^L and h^S be the knot separation for the LP and SOCP respectively. Suppose that there is a strictly feasible a^* such that there exists $\epsilon > 0$ such that $f(a^*; x) \geq 2\epsilon$ for $x \in [-1, 1]$. Let μ_2 and μ_4 be such that for all $x \in [-1, 1]$,

$$|f^{(2)}(a^*; x)| \leq \mu_2 \text{ and } |f^{(4)}(a^*; x)| \leq \mu_4. \tag{15}$$

The results from Theorems 2 and 4 imply that if

$$h^L \leq \left(\frac{4\epsilon}{\mu_2}\right)^{\frac{1}{2}} = \mathcal{O}(\epsilon^{\frac{1}{2}}) \text{ and } h^S \leq \left(\frac{\epsilon}{C_0\mu_4}\right)^{\frac{1}{4}} = \mathcal{O}(\epsilon^{\frac{1}{4}}), \tag{16}$$

then the a^* is feasible for the respective programs. If a^* is optimal for (General-SIP), then the linear program and second order cone program give the optimal value. As ϵ decreases, the number of knot points for SOCP does not increase as rapidly as for LP. The constant factor depends on the behaviour of the derivatives of $f(a^*)$. In Sections 4.1 and 4.2 we empirically see how many knot points are necessary for both methods in practice.

3.4 Derivative bounding

Both the linear and second order cone programs rely on a bound on a derivative (μ_2 or μ_4) of the function being interpolated. The difficulty is that the true derivatives themselves vary with the decision variable a. For some values of a, a better bound exists than for other values of a. One way to address this issue is to fix μ_2 or μ_4 as a constant. Then, solve the optimization problem, and finally check if the derivative bound is satisfied for the optimizer. If so, the result was a valid approximation.

There is another way to address this, which we discuss next. If there is a simple map h from parameters a to bounds for μ_2 or μ_4, then including the bound as a function of a as a constraint in the optimization problem (i.e. $\mu_2 \geq h(a)$) may be advantageous. This second method is the one we use in Section 4.

In the case when the function whose derivatives we must bound is a polynomial, there is a convenient and natural such bound on the derivatives. Here, the map h discussed in the previous paragraph would be linear in the decision variables a. To construct the map, we write the polynomial in the Bernstein basis. The Bernstein polynomials [21] of degree d are a basis of degree d polynomials that are nonnegative and sum to 1. Because the intervals in our applications are on $[-1, 1]$, we scale the standard Bernstein basis to

$$B_i^d(x) = \frac{1}{2^d}\binom{d}{i}(1+x)^{d-i}(1-x)^i. \tag{17}$$

According to the following well-known proposition (e.g. [8]), we can use the coefficients in the Bernstein basis as bounds on the polynomial.

Proposition 1. *Let $p = \sum_{i=0}^d p_i B_i^d(x)$ for real numbers p_i, $i = 0, \ldots, d$. Then $|p(x)| \leq \max_i(|p_i|)$ on $[-1, 1]$.*

Therefore, a bound on the derivative of the polynomial we interpolate is given by the maximum coefficient in the Bernstein basis. We can then explicitly include the constraints that either μ_2 or μ_4 exceeds p_i and $-p_i$ for every $0 \leq i \leq d$ when p is the respective *derivative* of the polynomial that must be nonnegative.

4 Applications

In this section, specific instances of the approximation scheme of Section 3 are considered. The actual implementation of these applications was performed with the software presented in Section 5. In Sections 4.1 and 4.2 we numerically compare the second order cone approach to the linear programming method to find sphere packing bounds and polynomial approximations. In Section 4.3, we present the results of the second order cone solution alongside other sophisticated approximation methods.

4.1 Spherical code bounds

Objective Use the fewest number of knot points possible to prove the optimal sphere packing bounds.

Application background The first application is to determine bounds on the size of a spherical code. Fix a constant $\alpha < 1$. A code is a set $X(\alpha) \subseteq \mathbb{R}^d$ of unit vectors such that for all $x, y \in X$, $\langle x, y\rangle \in [-1, \alpha]$. We want to bound the largest possible $|X(\alpha)|$. When $\alpha = \frac{1}{2}$, the largest value of $|X(\alpha)|$ is the kissing number in dimension d.

The bound we use is in terms of a scaled version of Gegenbauer polynomials. Fix a dimension $d \geq 2$. Then $G_n(x)$ of degree n is defined recursively as $G_0(x) = 1$, $G_1(x) = dx$, and

$$\lambda_{n+1} G_{n+1}(x) = xG_n(x) - (1 - \lambda_{n-1})G_{n-1}(x), \tag{18}$$

where $\lambda_n = \frac{n}{d+2n-2}$. These are related to the standard definition of Gegenbauer polynomials (e.g. [54, §4.7]) by the equality that $G_n(x)$ is the nth Gegenbauer polynomial of weight $\frac{d-2}{2}$ scaled by $\frac{d+2\cdot n-2}{d-2}$. Delsarte's spherical code bound is stated in terms of the these scaled Gegenbauer polynomials.

Theorem 5 (Delsarte spherical code bound [19])**.** *Let $f = \sum_{n\geq 0} g_n G_n$ be such that $g_n \geq 0$ for all n, $g_0 = 1$, and $f(x) \leq 0$ for all $x \in [-1, \alpha]$. Then $|X(\alpha)| \leq f(1)$.*

Application SIP The optimal bound is found by solving the SIP

$$\begin{aligned} \min_{\{g_n\}\geq 0, g_0=1} \quad & \sum g_n G_n(1) \\ \text{s.t.} \quad & \sum g_n G_n \leq 0 \text{ on } [-1, \alpha]. \end{aligned} \tag{Delsarte-SIP}$$

The optimal value to (Delsarte-SIP) is the best such bound attainable by applying Theorem 5. On the other hand, any $\{g_n\}$ that is feasible provides an upper bound by Theorem 5. This is a special case of (General-SIP) where $f(g; x) = -\sum g_n G_n(x)$ is a high degree polynomial and c is the vector of values of $G_n(1)$. We only need to add $g_n \geq 0$ as additional linear inequalities.

For this application, we describe and analyze the approximation methods proposed in Section 3.2 and the competing method from Section 3.1. Unlike a sum of squares method, neither of these methods have any problems with using very large maximum n. For both the LP and SOCP methods, we start by dividing $[-1, \alpha]$ into intervals $I_i = [x_i, x_{i+1}]$ such that h is the width of every I_i.

Application LP The LP bound is given by

$$\begin{aligned} \min_{g_n\geq 0, g_0=1} \quad & \sum g_n G_n(1) \\ \text{s.t.} \quad & \mu_2 \geq |(\sum g_n G_n)^{(2)}(x)| && \forall x \in [-1, \alpha] \\ & \frac{1}{8} h^2 \mu_2 \leq \delta \\ & -\sum g_n G_n(x_i) - \delta \geq 0 && \forall x \in [-1, \alpha] \end{aligned} \tag{Delsarte-LP}$$

Application SOCP The SOCP bound is given by

$$\begin{aligned} \min_{g_n\geq 0, g_0=1} \quad & \sum g_n G_n(1) \\ \text{s.t.} \quad & p = \text{CubicInterpolant}(\sum g_n G_n) \\ & \mu_4 \geq |\sum (g_n G_n)^{(4)}(x)| && \forall x \in [-1, \alpha] \\ & C_0 h^4 \mu_4 \leq \delta \\ & -p - \delta \geq 0 && \forall x \in [-1, \alpha]. \end{aligned} \tag{Delsarte-SOCP}$$

Empirical comparison The concrete application allows us to compare the values suggested by the analysis in Section 3.3 that suggests the SOCP provides a better bound with fewer knot points than the LP. We directly compare the values in (16) empirically for this application by computing the kissing numbers in

dimensions $d = 2, 3, 4, 8, 24$, via (Delsarte-LP) and (Delsarte-SOCP). For both problems, the derivatives are bounded by computing $(\sum g_n G_n(x))^{(r)}$ in the Bernstein basis and use the method mentioned in Section 3.4. For this problem, we bounded the derivative on $[-1, 1]$ instead of $[-1, \alpha]$, which would have been possible too, but requires more precomputation for every value of α. For the results below, we fixed $\alpha = \frac{1}{2}$. In Table 1 we report the number of knot points for an evenly spaced grid between $[-1, \frac{1}{2}]$ required to be used in (Delsarte-LP) and (Delsarte-SOCP) so that the floor function of their optimal value is the floor function of the optimal value of (Delsarte-SIP) in that dimension.[2] This makes sense because the cardinality we are bounding must be an integer. We also report the minimum time in ms (as reported by Julia's [5] BenchmarkTools.jl package) required to set up and solve the respective programs with the package described in Section 5. The Delsarte bound is 13 and 25 in dimensions 3 and 4, respectively, and it equals the optimal kissing numbers in dimensions 2, 8, and 24 [45]. We use polynomials of maximum degree at most 20. An order of magnitude fewer knot points is required for the second order cone version in all dimensions. The solve times show that the speed of an LP solver does not make up for the larger problem size. The minimum number of knot points required was computed by a binary search over number of knot points whose objective value provides a bound that is the best known achievable with (Delsarte-SIP).

d	SOCP #	LP #	SOCP time (ms)	LP time (ms)
2	3	11	3.61	3.87
3	5	33	5.67	9.99
4	14	108	13.0	28.9
8	22	661	18.9	180
24	512	>100,000	452	N/A

Table 1: The 2nd and 3rd columns are the number of knot points required to get the best possible Delsarte bound of kissing number in dimension d. The last two columns are the fastest wall times we recorded (ms) to set up and solve the program with [36] and [3] on MacBook Air M1. The LP with $d = 24$ was not correct even with 100,000 knot points, so no time is available.

4.2 Chebyshev Approximation

Objective Guarantee uniform approximation errors with the fewest number of knot points.

Application background Let $\{\Phi_n\}$ be a collection of N polynomials. Consider the problem of finding an approximation $g(x) = \sum a_n \Phi_n(x)$ of a given continuous function $f(x)$ over the interval $[-1, 1]$. A standard solution is to discretize the interval and compute the a_n via least squares. Instead, we can minimize the Chebyshev, or supremum, norm $||f(x) - g(x)||_\infty$ defined as $\sup_{x\in[-1,1]} |f(x) - g(x)|$.

Application SIP Let a be the vector whose nth entry is a_n. Then the approximation problem can be cast as the semi-infinite programming problem

$$\begin{aligned} \min_{a\in\mathbb{R}^N} \quad & t \\ \text{s.t.} \quad & f(x) - \sum a_n \Phi_n(x) \le t \quad \forall x \in [-1, 1] \\ & \sum a_n \Phi_n(x) - f(x) \le t \quad \forall x \in [-1, 1], \end{aligned} \tag{Chebyshev-SIP}$$

where the objective value serves as an upper bound on the deviation between f and g. Even though there are two minor differences between this program and (General-SIP), we can still approximate this program with the tools we developed for (General-SIP). The first minor difference is that we have two semi-infinite constraints of the form of (2). We just need to apply the main idea from Section 2 twice. Second, the decision variables are (a, t), but $f(a, t; x)$ does not depend on t and the objective vector only depends on t.

[2] Note that even the semi-infinite Delsarte bound is not tight in general.

Application LP We let the knot points of $[-1,1]$ be $\{x_i\}$. The method of Section 3.1 gives

$$\begin{aligned}
\min_{a\in\mathbb{R}^N} \quad & t \\
\text{s.t.} \quad & \mu_2 \geq |(f - \sum a_n\Phi_n)^{(2)}(x)| && \forall x \in [-1,1] \\
& \delta \geq \frac{1}{8}h^2\mu_2 \\
& f(x_i) - \sum a_n\Phi_n(x_i) - t \leq -\delta && \forall i \\
& -f(x_i) + \sum a_n\Phi_n(x_i) - t \leq -\delta && \forall i.
\end{aligned} \tag{Chebyshev-LP}$$

Application SOCP The method of Section 3.2 gives

$$\begin{aligned}
\min_{a\in\mathbb{R}^N} \quad & t \\
\text{s.t.} \quad & \mu_4 \geq |(f - \sum a_n\Phi_n)^{(4)}(x)| \quad \forall x \in [-1,1] \\
& C_0h^4\mu_4 \leq \delta \\
& \begin{cases} p_1 = \text{CubicInterpolant}(t - f + \sum a_n\Phi_n) \\ p_1 - \delta \geq 0 \quad \forall x \in [-1,1] \end{cases} \quad (\implies \quad f - \sum a_n\Phi_n \leq t \quad \forall x \in [-1,1]) \\
& \begin{cases} p_2 = \text{CubicInterpolant}(t + f - \sum a_n\Phi_n) \\ p_2 - \delta \geq 0 \quad \forall x \in I \end{cases} \quad (\implies \quad \sum a_n\Phi_n - f \leq t \quad \forall x \in I).
\end{aligned} \tag{Chebyshev-SOCP}$$

The braces group together two constraints that are sufficient conditions for the parenthesized constraints. The optimal value t^* of (Chebyshev-SOCP) is an upper bound for (Chebyshev-SIP). The optimal location $\{a_n^*\}$ of (Chebyshev-SOCP) may not be optimal for (Chebyshev-SIP), but we are guaranteed that $\sup_{x\in[-1,1]} |f(x) - \sum a_n^*\Phi_n(x)| \leq t^*$.

Both approximation methods need a bound on the derivatives of $f - \sum a_n\Phi_n$. A simple bound can be found from

$$|(f - \sum a_n\Phi_n)^{(r)}(x)| \leq |f^{(r)}(x)| + |(\sum a_n\Phi_n)^{(r)}(x)|, \tag{19}$$

where the supremum of the first term can be precomputed. For the second, we use the method described in Section 3.4. If it is a polynomial, we use the Bernstein method described in that section. Taking μ_2 or μ_4 larger than this final sum gives a valid upper bound on the fourth derivative.

This application demonstrates how the cubic interpolants can be used to prove nonnegativity of a function that is not necessarily a polynomial. While in the sphere packing applications we dealt only with high degree polynomials, the function f in this application need not be a polynomial. The function $t - f + \sum a_n\Phi_n(x)$ that we prove is nonnegative is therefore also not a polynomial. We use this method in the subsections that follow.

4.2.1 Approximations of $\text{erf}(x)$

We use the aforementioned reduction to find a polynomial approximation of the error function,

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\,\mathrm{d}t \tag{20}$$

on the interval $[-1,1]$. We note that $\frac{d}{dx}\text{erf}(x) = \frac{2}{\sqrt{\pi}}e^{-x^2}$. Repeatedly taking derivatives, $\frac{d^4}{dx^4}\text{erf}(x) = -\frac{8}{\sqrt{\pi}}e^{-x^2}x(2x^2-3)$. Offline computation of one more derivative and then using a numerical root finder to find extreme points, we find 4.41 is an upper bound for the fourth derivative and .968 is an upper bound for the second derivative. We use a triangle inequality bound to convert these into bounds on the derivatives of $f - \sum a_n\Phi_n$ for any given a_n.

Chebyshev polynomials First, let $\{\Phi_n\}_{n=0}^{N-1}$ be the collection of nth Chebyshev polynomials of the first kind. We compare the errors guaranteed in a Chebyshev approximation by both the SOCP method and the LP method described in Section 3.1. For various number of knot points, we approximate the optimal t in (Chebyshev-SIP) with $N = 14$. The results from each approximation are plotted in Figure 2. With the same number of knot points, the SOCP approximation has a better error guarantee. It does not make sense to extend the plot beyond 64 knot points because 10^{-8} was the chosen tolerance of the solver we used.

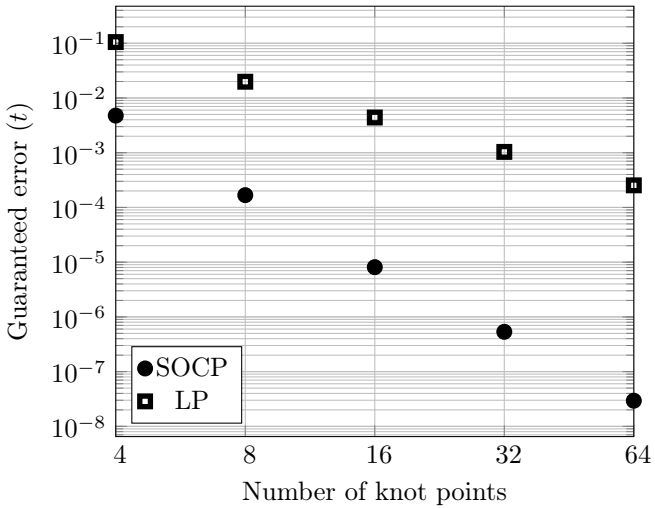


Figure 2: The error guaranteed of the approximation of $\text{erf}(x)$ by Chebyshev poylnomials using the SOCP method and the LP method with various numbers of knot points.

Gaussian functions Second, let $\{\Phi_n\}_{n=0}^{N-1}$ be the N Gaussian functions with different means

$$\Phi_n = e^{-(x-t_n)^2}, \tag{21}$$

where $t_n = -1 + 2\frac{n}{N-1}$. Because the quantity is not a polynomial, we do not estimate the derivatives with the Bernstein method described in Section 3.4. Instead we use a simple bound of the absolute values of the coefficients times the maximum derivative of a Gaussian function (which can be computed offline).

We again solve (Chebyshev-SIP) now with the new set of Φ_n. We compare the errors guaranteed by the LP and SOCP approximations. For this experiment, we used $N = 10$. The result is plotted in Figure 3. Again, the error guarantee given by the SOCP decreases more rapidly than for the LP. The conclusion of the second experiment is that this method is not restricted to finding polynomial approximations.

4.2.2 Rational approximation to $|x|$

A rational approximation to $|x|$ is presented by Newman in [41].

Theorem 6 (Theorem A of [41])**.** *For every $n \geq 4$, if*

$$z = e^{-1/\sqrt{n}}, \quad p(x) = \prod_{k=0}^{n-1}(x + z^k), \quad r(x) = x\frac{p(x) - p(-x)}{p(x) + p(-x)}, \tag{22}$$

then

$$\forall x \in [-1, 1], \quad ||x| - r(x)| \leq 3e^{-\sqrt{n}}. \tag{23}$$

Verifying the Newman bound We use the method of Section 2 to verify this theorem for fixed n. If $p(x) + p(-x) \geq 0$,

$$\begin{aligned} 3e^{-\sqrt{n}}(p(x) + p(-x)) - |x|(p(x) + p(-x)) + x(p(x) - p(-x)) \geq 0, \\ \text{and } 3e^{-\sqrt{n}}(p(x) + p(-x)) + |x|(p(x) + p(-x)) - x(p(x) - p(-x)) \geq 0 \end{aligned} \tag{24}$$

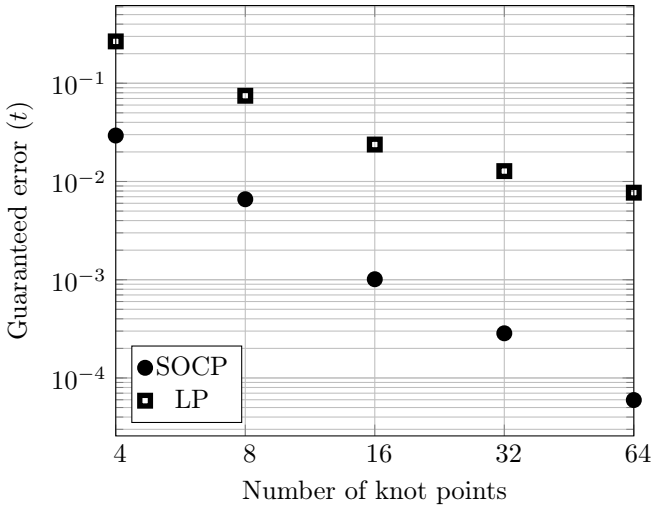


Figure 3: The error guaranteed in the estimation of a Gaussian function approximation of $\text{erf}(x)$ using the SOCP method and the LP method with various numbers of knot points.

on $[-1, 1]$, then Theorem 6 is satisfied. These positivity conditions can be verified with Lemma 1. We compute the smallest number of knot points that guarantee the bounds $p(x) + p(-x) \geq 0$ and (24) with the LP and SOCP methods of Section 2. Figure 4 shows how the LP method requires orders of magnitude more knot points than the SOCP.

A few details are needed to fully specify the LP and SOCP. First, we always ensure that $x = 0$ is a knot point so the nondifferentiability of $|x|$ at the origin is not an issue. Second, we need derivative bounds. For every $x \in [-1, 1]$ we have the derivative bound

$$|p^{(k)}(x)| \leq n^k 2^{n-k} \tag{25}$$

so that for every $x \in [-1, 1]$, $|(p(x) + p(-x))^{(k)}| \leq 2 \cdot n^k 2^{n-k}$. Let $g(x) = p(x) + p(-x)$ and bound the derivatives of the functions in (24) by

$$\begin{aligned} |(3e^{-\sqrt{n}} g(x) - |x|g(x) + xg(x))^{(k)}| &\leq |3e^{-\sqrt{n}} g^{(k)}(x)| + |(xg(x))^{(k)}| + (|x|g(x))^{(k)}| \\ &\leq 3e^{-\sqrt{n}} |g^{(k)}(x)| + 2|xg^{(k)}(x)| + 2k|g^{(k-1)}(x)| \\ &\leq 6e^{-\sqrt{n}}(n^k 2^{n-k}) + 4(n^k 2^{n-k}) + 4k(n^{k-1} 2^{n-(k-1)}). \end{aligned} \tag{26}$$

This is enough to certify the nonnegativity constraints with the methods of Sections 2.1 and 2.2.

Optimal rational approximation While Newman's construction provides a good rational approximation to $|x|$ with guaranteed asymptotic error, if n is fixed, there may exist better rational approximations with numerator and denominator both of degree n. We can use the SOCP to find such approximations that may be of use in practical settings.

In particular, we seek the degree n polynomials p and q, such that $\sup_{x \in [-1,1]} ||x| - \frac{p(x)}{q(x)}|$ is as small as possible. We let the coefficients of p and q be the decision variables, and we perform a binary search over possible errors so that the program is feasible. We use a knot spacing of .001. For $n = 6$, an approximation written in the monomial basis (rounded/truncated) is

$$\frac{.0047x^2 + .60x^4 + 2.2x^6}{.000075 + .086x^2 + 1.8x^4 + .00025x^5 + .96x^6}. \tag{27}$$

We compare this approximation to the $n = 6$ approximation of Newman. Notice that if $n = 6$, then Newman's numerator is actually of degree 7. The unnormalized errors are show in Figure 5.

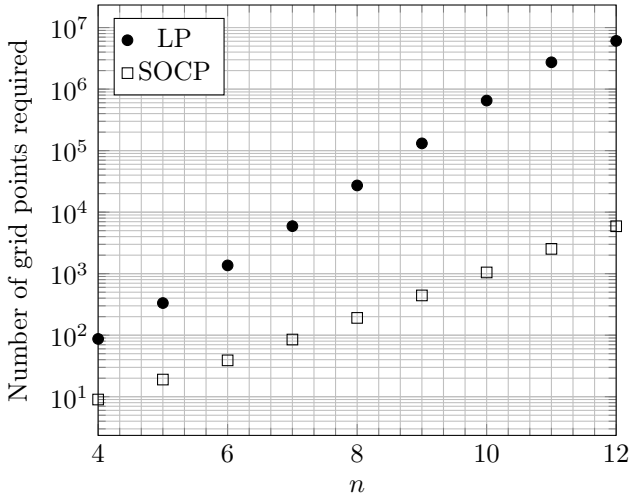


Figure 4: The number of grid points required to certify the Newman bound of degree n with each method

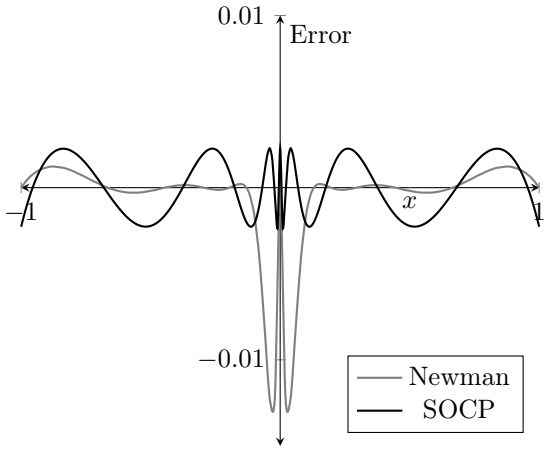


Figure 5: Errors in degree 6 rational approximations of $|x|$. One approximation is given by Newman's rational approximation (22) (which actually has a degree 7 numerator). Another rational approximation is computed by the SOCP method and reported in (27). Near the origin, the error in Newman's approximation is much greater.

4.3 Filter design

Objective Compare SOCP to existing algorithms for an SIP problem in signal processing.

Application background In this section we consider the problem of designing a filter bank for signal processing applications. The filter with maximum coding gain but with the desirable constraint of perfect reconstruction is shown in [31] and [39] to be described by the following SIP. Let a be the vector whose nth entry is a_n, and let r_n be constants that are derived from the statistics of the input signals.

Application SIP The optimization problem is

$$\begin{array}{ll} \max\limits_{a\in\mathbb{R}^N} & \dfrac{r_0}{2} + \displaystyle\sum_{n=0}^{N-1} a_n r_{2n+1} \\ \text{s.t.} & -2\displaystyle\sum_{n=0}^{N-1} a_n \cos((2n+1)2\pi f) \le 1, \qquad \forall f \in [0, 0.5]. \end{array} \tag{Filter-SIP}$$

Application SOCP We apply the approximation from Section 3.2. Let

$$g(a, f) = -1 - 2\sum_{n=0}^{N-1} a_n \cos((2n+1)2\pi f).$$

The approximation of (Filter-SIP) is given by

$$\begin{array}{lll} \max\limits_{a\in\mathbb{R}^N} & \dfrac{r_0}{2} + \displaystyle\sum_{n=0}^{N-1} a_n r_{2n+1} & \\ \text{s.t.} & p = \text{CubicInterpolant}(g) & \\ & \mu_4 \ge |g^{(4)}(x)| & \forall x \in [0, 0.5] \\ & \delta \ge C_0 h^4 \mu_4 & \\ & p(x) \le -\delta & \forall x \in [0, 0.5]. \end{array} \tag{Filter-SOCP}$$

To find a bound on the fourth derivative, each term in g can be differentiated directly four times. The sum of the absolute values of the terms is an SOCP-representable bound on the value of $g^{(4)}$.

Numerical experiments As is always the case with the SOCP approximation, we are guaranteed that our solution is feasible for the original problem. In this case, that means our filter has perfect reconstruction. Therefore a solution to (Filter-SOCP) provides a lower bound on (Filter-SIP). To see how well the lower bound competes with other semi-infinite optimization algorithms for this problem, we test our formulation on some examples presented by [31] and [39].

- The AR(1) process is a simple image model. Let $\rho = 0.95$, and then set $r_n = \rho^n$.
- The AR(2) process models image textures. Let $\rho = 0.975$ and $\theta = \frac{\pi}{3}$. Set $r_0 = 1$ and $r_1 = \frac{2\rho\cos(\theta)}{1+\rho^2}$. Finally, let $r_n = 2\rho\cos(\theta)r_{n-1} - \rho^2 r_{n-2}$.
- The box-spec process is a lowpass process with a box spectrum. Let $f_s = 0.225$. Then set $r_n = \frac{\sin(2\pi n f_s)}{2\pi n f_s}$.

We compare the coding gain from (Filter-SOCP) to the results from the discretization and cutting plane methods of [31] and the logarithmic barrier method of [1] when $N = 4$. Kortanek and Moulin have two discretization algorithms. We emphasize that this is not the same as the linear programming method of Section 3.1, and we refer the reader to the reference for a complete description. Whichever gave a better coding gain is presented along with the other results in Table 2. All results other than the SOCP coding

gains were taken from their respective publications. The SOCP finds a filter with a coding gain that matches the maximum coding gain found by any competing method. (None of the methods guarantee they find the optimal filter.) The coding gains of the filters found by the SOCP with 61 knot points are at least the coding gains with the discretization method of [31] using 91 discretization points.

Problem	SOCP (61 points)	Log Barrier [1]	Cutting Plane [31]	Discretization [31] (91 points)
AR(1)	5.862	5.860	5.862	5.862
AR(2)	6.070	6.069	6.070	6.042
box-spec	4.885	4.884	4.885	4.878

Table 2: Coding gain in dB for filters computed by various algorithms.

In Figure 6 we present the coding gains for each problem found by the discretization algorithms of [31] alongside our SOCP with varying numbers of discretization points. There are two main observations: (1) the SOCP is similar to or better than the better of the two discretization algorithms for each problem, and (2) the coding gains found by the discretization algorithms do not monotonically improve with more knot points as is the case with the SOCP formulation.

5 Software

In solving problems like those in Section 4, we repeatedly make use of the second order cone constraints described in Lemma 5 that constrain a cubic interpolant to be nonnegative. We factor out these constraints in the CubicSOS.jl package. The two primary methods of the package are:

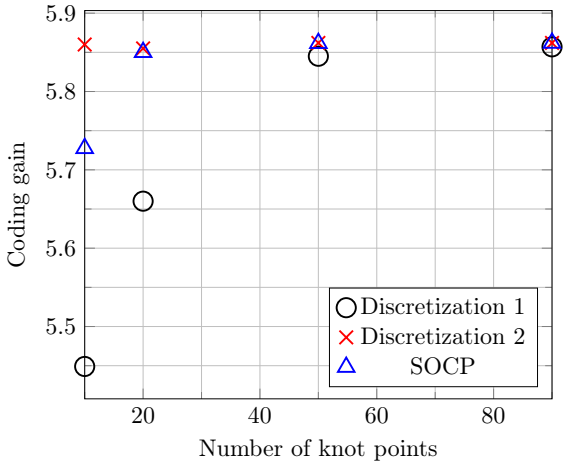
- The constructor `CubicInterpolant(x_vals, y_vals, deriv_vals)` instantiates an interpolant object with given values and derivatives, which can be real numbers or JuMP variable types.
- The method `constrain_interpolant_nonnegative!(model, p)` adds the second order cone constraints that $p \geq 0$ on its domain to `model`.

Furthermore, primitive operations like addition and subtraction of interpolants, multiplication by a constant, and addition or subtraction of a scalar from a interpolants, are defined in the mathematically equivalent ways. Users can evaluate interpolants anywhere on their domains. More detailed documentation is available in the software readme.

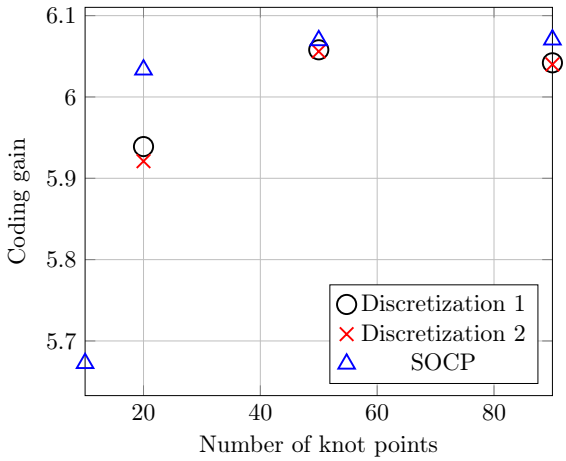
Here is a demonstration of the code used to approximate (1) with $N = 5$ and 20 knot points.

Listing 1: Approximating(1) with `CubicSOS`

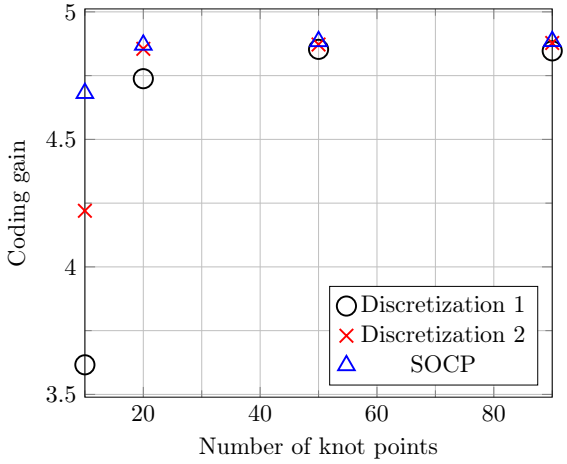
```
# Simple demonstration of package
using CubicSOS
using JuMP, MosekTools
N = 5; mu4p = 10 # bound on 4th derivative of p
mu4f = N * (N-1) * (N-2) # bound on 4th derivative of x^N = f
knots = range(-1, 1, length = 20)
xN = knots .^ N # true values of x^N
xNp = N * knots .^ (N-1) # true values of x^N's derivative

model = Model(); set_optimizer(model, () -> Mosek.Optimizer())
@variable(model, a[1:N]) # coefficients of approximation
approx = [sum([a[i+1] * knots[j] ^ i for i = 0:N-1])
        for j = 1:lastindex(knots)] # value of approximation
approxderiv = [sum([a[i+1] * i * knots[j] ^ (i-1) for i = 1:N-1])
        for j = 1:lastindex(knots)] # value of approximation's derivative
p = CubicInterpolant(knots, approx, approxderiv)
f = CubicInterpolant(knots, xN, xNp)
```



(a) AR(1)



(b) AR(2). The results for 10 knot points are not reported for the discretization algorithms in [31].



(c) box-spec

Figure 6: Coding gain of SOCP and discretization algorithms in [31] with varying number of knot points. The data for the discretization algorithms are taken from Table 1 in [31]. The coding gains found by the discretization method do not monotonically improve when increasing number of discretization points.

```
@variable(model, error)
@variable(model, delta)
@constraint(model, delta >= maximum(diff(knots))^4 * (mu4p + mu4f) / 384)
# enforce p - f <= error
constrain_interpolant_nonnegative!(model, p - f + error - delta)
# enforce f - p <= error
constrain_interpolant_nonnegative!(model, f - p + error - delta)
@objective(model, Min, error); optimize!(model)
@show(value.(a))
```

The output (rounded) of the code is `[1.37e-11, -0.313, -1.11e-10, 1.25, 9.98e-11]`, which nearly matches the correct polynomial coefficients of $[0, -\frac{5}{16}, 0, \frac{5}{4}, 0]$ corresponding to the Chebyshev polynomial $16x^5 - 20x^3 + 5x$. In this case, the 4th derivative is nearly 0, which is certainly less than the a priori bound `mu4p` of 10. When increasing the number of knot points, the numerical difference from the correct polynomial monotonically decreases.

6 Multidimensional interpolants

Extending this procedure to multiple dimensions is possible. We describe one way to extend to problems in the case in which $f(a)$ is bivariate.

Consider a constraint of the form $f(a; x, y) \geq 0$. In that case, one can use a bicubic interpolant. Such an approximation is computable by the method described in [16]. In addition, if the mesh discretization is the same in both coordinates, the error in function values still scales as $\mathcal{O}(h^4)$. In this case, the constants depend on both bounds for $f^{(4,0)}(a; x, y)$ and $f^{(0,4)}(a; x, y)$ as well as $f^{(2,2)}(a; x, y)$. The exact error can be found in Theorem 3 of [9].

The difficulty of higher dimensions is not interpolation but in the certification that such multidimensional cubic interpolants are nonnegative. One obstruction appears in Lemma 5 because sum of squares and nonnegativity are not equivalent in the bivariate case. Even if sufficiency were enough, the bigger issue is that we would have larger semidefinite constraints, which do not reduce to a second order cone problem.

In general, an extension to higher dimensions will have a sequence of $(d+1) \times (d+1)$ semidefinite constraints for a constraint in d variables. While solving such an optimization problem would not be as fast as solving the second order cones arising in the univariate case, it would still be much faster than solving a problem with a nonnegativity constraint on a degree n polynomial in dimension d. If we were approximating some complicated function, we may want n to increase. Even with a simple generalization of the cubic interpolant method, we can increase approximation accuracy without increasing the sizes of the constraint matrices.

7 Conclusion

In the context of semi-infinite programming methods that optimize over a feasible subset of solutions, the new method strikes a balance between sum of squares optimization and linear interpolation. On the one hand, sum of squares methods provide a global nonnegativity certificate on the full interval but require the solution of an expensive semidefinite program whose complexity depends on the degree of the function approximation over the full interval. On the other hand, linear interpolation does not necessarily make use of all the function data that is available. Using a cubic interpolant allows us to use the power of sum of squares at a local level.

The new second order cone method not only has theoretical guarantees of success but also performs well in practice. We have demonstrated its superior scaling compared to the linear programming technique both in theory and in practice. We expect that in moderate dimension, the ideas from Section 6 may provide a useful generalization of this technique, which is the subject of future work.

References

[1] Lars Abbe. Two logarithmic barrier methods for convex semi-infinite problems. In *Semi-infinite Programming*, pages 169–195. Springer, 2001. 15, 16

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

[3] MOSEK ApS. MOSEK optimization suite, 2019. 10

[4] Federico Bergenti, Stefania Monica, et al. Satisfaction of polynomial constraints over finite domains using function values. In *ICTCS/CILC*, pages 262–275, 2017. 3

[5] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017. 10

[6] Pierre Bézier. Definition numerique des courbes et surface. *Automatisme*, 11(4):625–632, 1966. 4

[7] Richard L Burden. *Numerical analysis*. Brooks/Cole Cengage Learning, 2011. 3, 5

[8] GT Cargo and Oved Shisha. The bernstein form of a polynomial. *J. Res. Nat. Bur. Standards B*, 70:79–81, 1966. 8

[9] RE Carlson and CA Hall. Error bounds for bicubic spline interpolation. *Journal of Approximation Theory*, 7(1):41–47, 1973. 18

[10] Neal L Carothers. A short course on approximation theory. *Bowling Green State University, Bowling Green, OH*, 1998. 1

[11] Abraham Charnes, William W Cooper, and K Kortanek. Duality, haar programs, and finite sequence spaces. *Proceedings of the National Academy of Sciences*, 48(5):783–786, 1962. 1

[12] Elliot Ward Cheney and Allen A Goldstein. Newton's method for convex programming and tchebycheff approximation. *Numerische Mathematik*, 1(1):253–268, 1959. 2

[13] PG Ciarlet, MH Schultz, and RS Varga. Numerical methods of high-order accuracy for nonlinear boundary value Problems: I. One Dimensional Problem. *Numerische Mathematik*, 9:394–430, 1967. 5

[14] Philip J Davis. *Interpolation and approximation*. Courier Corporation, 1975. 3

[15] Carl De Boor. *A practical guide to splines*, volume 27. 4

[16] Carl De Boor. Bicubic spline interpolation. *Journal of mathematics and physics*, 41(1-4):212–218, 1962. 18

[17] Carl De Boor. On calculating with b-splines. *Journal of Approximation theory*, 6(1):50–62, 1972. 4

[18] Etienne de Klerk, EG Elabwabi, and Dick Den Hertog. Optimization of univariate functions on bounded intervals by interpolation and semidefinite programming. *CentER Discussion Paper*, 2006. 3

[19] P. Delsarte, J. M. Goethals, and J. J. Seidel. Spherical codes and designs. *Geometriae Dedicata*, 6(3):363–388, 1977. 9

[20] Gerald Farin. *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier, 2014. 4

[21] Rida T Farouki. The bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design*, 29(6):379–419, 2012. 4, 8

[22] Miguel A Goberna and MA Lopez-Cerda. Linear semi-infinite optimization. *(No Title)*, 1998. 1

[23] Mitchell Tong Harris and Pablo A Parrilo. Improved nonnegativity testing in the bernstein basis via geometric means. *arXiv preprint arXiv:2309.10675*, 2023. 22

[24] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019. 4

[25] Rainer Hettich. An implementation of a discretization method for semi-infinite programming. *Mathematical Programming*, 34:354–361, 1986. 2

[26] Rainer Hettich and Kenneth O Kortanek. Semi-infinite programming: theory, methods, and applications. *SIAM review*, 35(3):380–429, 1993. 2, 3

[27] Francis Begnaud Hildebrand. *Introduction to numerical analysis*. Courier Corporation, 1987. 3

[28] Hsieh Hou and H Andrews. Cubic splines for image interpolation and digital filtering. *IEEE Transactions on acoustics, speech, and signal processing*, 26(6):508–517, 1978. 4

[29] Hui Hu. A one-phase algorithm for semi-infinite linear programming. *Mathematical programming*, 46:85–103, 1990. 2

[30] Mikael Johansson and Anders Rantzer. Computation of piecewise quadratic lyapunov functions for hybrid systems. In *1997 European Control Conference (ECC)*, pages 2005–2010. IEEE, 1997. 4

[31] KO Kortanek and Pierre Moulin. Semi-infinite programming in orthogonal wavelet filter design. In *Semi-Infinite Programming*, pages 323–360. Springer, 1998. 15, 16, 17

[32] Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001. 3

[33] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear algebra and its applications*, 284(1-3):193–228, 1998. 6

[34] Marco López and Georg Still. References in Semi-infinite Optimization. http://virt.dacs.utwente.nl/~stillgj/sip/lit-sip.pdf. Accessed: 2023-10-03. 1

[35] Marco López and Georg Still. Semi-infinite programming. *European journal of operational research*, 180(2):491–518, 2007. 1

[36] Miles Lubin, Oscar Dowson, Joaquim Dias Garcia, Joey Huchette, Benoît Legat, and Juan Pablo Vielma. JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, 2023. 10

[37] Gregory Miller and David Trebotich. An embedded boundary method for the navier–stokes equations on a time-dependent domain. *Communications in Applied Mathematics and Computational Science*, 7(1):1–31, 2011. 3

[38] Michael E Mortenson. *Geometric modeling*. John Wiley & Sons, Inc., 1997. 4

[39] Pierre Moulin, Mihai Anitescu, Kenneth O Kortanek, and Florian A Potra. The role of linear semi-infinite programming in signal-adapted qmf bank design. *IEEE Transactions on Signal Processing*, 45(9):2160–2174, 1997. 15

[40] Yurii Nesterov. Squared functional systems and optimization problems. In *High performance optimization*, pages 405–440. Springer, 2000. 3

[41] Donald J Newman. Rational approximation to $|x|$. *Michigan Mathematical Journal*, 11(1):11–14, 1964. 2, 12

[42] Dávid Papp. *Optimization models for shape-constrained function estimation problems involving nonnegative polynomials and their restrictions*. Rutgers The State University of New Jersey, School of Graduate Studies, 2011. 4

[43] Dávid Papp and Farid Alizadeh. Shape-constrained estimation using nonnegative splines. *Journal of Computational and graphical Statistics*, 23(1):211–231, 2014. 4

[44] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000. 3

[45] Florian Pfender. Improved delsarte bounds for spherical codes in small dimensions. *Journal of Combinatorial Theory, Series A*, 114(6):1133–1147, 2007. 10

[46] Stephen B Pope. Pdf methods for turbulent reactive flows. *Progress in energy and combustion science*, 11(2):119–192, 1985. 4

[47] Anthony Ralston and Philip Rabinowitz. *A first course in numerical analysis*. Courier Corporation, 2001. 3

[48] Rembert Reemtsen. Discretization methods for the solution of semi-infinite programming problems. *Journal of Optimization Theory and Applications*, 71:85–103, 1991. 2

[49] Rembert Reemtsen and Jan-J Rückmann. *Semi-infinite programming*, volume 25. Springer Science & Business Media, 1998. 1

[50] T. J. Rivlin. Bounds on a polynomial. *Journal of Research of the National Bureau of Standards–B. Mathematical Sciences*, 74B(1), 1970. 3, 4

[51] IJ Schoenberg. Cardinal interpolation and spline functions. *Journal of Approximation theory*, 2(2):167–206, 1969. 4

[52] Ahmad Shahsavaran and Akbar Shahsavaran. Application of lagrange interpolation for nonlinear integro differential equations. *Applied Mathematical Sciences*, 18:887–892, 2012. 3

[53] Naum Z Shor. Class of global minimum bounds of polynomial functions. *Cybernetics*, 23(6):731–734, 1987. 3

[54] Gabor Szegő. *Orthogonal polynomials*, volume 23, page 4. American Mathematical Soc., 1939. 3, 9

[55] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996. 3

[56] Yu Xia and Farid Alizadeh. Second-order cone programming for p-spline simulation metamodeling. *arXiv preprint arXiv:1506.05536*, 2015. 4

A Necessity and sufficiency of linear programming approximation

Theorem 1 (Soundness of (General-LP))**.** *Let a be feasible for* (General-LP)*. Then a is feasible for* (General-SIP)*, and so $c^T a$ is an upper bound for* (General-SIP)*.*

Proof of Theorem 1. If a satisfies the constraints for (General-LP), then by Lemmas 1 and 2, $f(a;x) \geq 0$ for all $x \in [-1,1]$, so a is feasible for (General-SIP). □

Theorem 2 (Completeness of (General-LP))**.** *Suppose that there exists a feasible a^* for* (General-SIP) *and $\epsilon \geq 0$ such that for every $x \in [-1,1]$, $f(a^*;x) \geq \epsilon$. Suppose μ_2 satisfies $|f^{(2)}(a^*;x)| \leq \mu_2$ for every $x \in [-1,1]$. If the knot points are chosen such that h satisfies*

$$\epsilon \geq \frac{1}{8}\mu_2 h^2, \tag{13}$$

then a^* is feasible for (General-LP)*.*

Proof of Theorem 2. We argue that the problem is feasible by using μ_2 as the given bound on the second derivative, $a = a^*$ and $\delta = \epsilon$. The first two constraints of (General-LP) are then satisfied by assumption on ϵ.

For the last constraint, we note that

$$\min_{y \in \{y_1, \ldots, y_K\}} f(a^*; y) \geq \min_{x \in [-1,1]} f(a^*; x) \geq \epsilon. \tag{28}$$

The first inequality follows because the knot points are a subset of the intervals. The second inequality follows from the assumption on a^*. Therefore the second constraint is satisfied for $\delta = \epsilon$. Hence, (General-LP) is feasible. □

B Necessity and sufficiency of second order cone approximation

In this section, we provide the background needed to prove Theorems 3 and 4. We first prove a few technical lemmas about the representability of nonnegative splines and the functions they approximate.

Lemma 5. *A cubic polynomial p is nonnegative on an interval $[x_0, x_1]$ with length h if and only if there exist $a, b \in \mathbb{R}$ such that*

$$(2p(x_0),\ 3p(x_1) - hp'(x_1) - a;\ b) \in \mathcal{Q} \qquad \text{and} \qquad (3p(x_0) + hp'(x_0) - b,\ 2p(x_1);\ a) \in \mathcal{Q}. \tag{11}$$

Proof. The coefficients of the cubic polynomial p in the Bernstein basis on the interval $[x_0, x_1]$ of length h are

$$p(x_0), \quad p(x_0) + \frac{h}{3}p'(x_0), \quad p(x_1) - \frac{h}{3}p'(x_1), \quad p(x_1). \tag{29}$$

Lemma 2 of [23] says that a polynomial in the Bernstein basis is nonnegative on the interval $[0, 1]$ if and only if its coefficients p_0, p_1, p_2, p_3 satisfy

$$\left(p_0,\ p_2 - \frac{c_2}{3};\ \frac{c_1}{\sqrt{6}}\right) \in \mathcal{Q} \qquad \text{and} \qquad \left(p_1 - \frac{c_1}{3},\ p_3;\ \frac{c_2}{\sqrt{6}}\right) \in \mathcal{Q}. \tag{30}$$

In fact, that proof extends to the case of nonnegativity over any interval: as long as the Bernstein basis is adjusted accordingly, the relationship amongst the coefficients remains the same. Plugging in the coefficients of our cubic polynomial we get exactly the condition we need to show. □

Theorem 3 (Soundness of (General-SOCP))**.** *Let a be feasible for* (General-SOCP)*. Then a is feasible for* (General-SIP)*, and so $c^T a$ is an upper bound for* (General-SIP)*.*

Proof of Theorem 3. If a feasible point to the constraints of (General-SOCP) is discovered, the three constraints prove that $f(a; x) \geq 0$ for all $x \in [-1, 1]$ by Lemmas 1 and 4. □

Theorem 4 (Completeness of (General-SOCP))**.** *Suppose that there exists a feasible a^* for* (General-SIP) *and $\epsilon \geq 0$ such that for every $x \in [-1, 1]$, $f(a^*; x) \geq 2\epsilon$. Suppose μ_4 satisfies $|f^{(4)}(a^*; x)| \leq \mu_4$ for every $x \in [-1, 1]$. If the knot points are chosen such that h satisfies*

$$\epsilon \geq \frac{\mu_4 h^4}{384}, \tag{14}$$

then a^* is also feasible for (General-SOCP)*.*

Proof of Theorem 4. Let $a = a^*$, $\delta = \epsilon$, $p = \text{CubicInterpolant}(f(a^*))$, and μ_4 be the given bound on the fourth derivative. Then the first three constraints are satisfied by assumption.

For the last constraint, we have by Lemma 4 for all $x \in [-1, 1]$,

$$f(a^*; x) - p(x) \leq \frac{h^4 \mu_4}{384} \tag{31}$$

and rearranging

$$\begin{aligned} p(x) &\geq f(a^*; x) - \frac{h^4 \mu_4}{384} \\ &\geq 2\epsilon - \epsilon \\ &= \epsilon, \end{aligned} \tag{32}$$

which follow from the theorem hypothesis on $f(a^*)$ and $\frac{h^4 \mu_4}{384}$. Hence $p \geq \epsilon = \delta$ is feasible as well. Therefore, since $p - \delta \geq 0$, the second order cone constraints are feasible by Lemma 5. □