STRUCTURE-PRESERVING FUNCTION APPROXIMATION VIA CONVEX OPTIMIZATION*

VIDHI ZALA†, ROBERT M. KIRBY‡, AND AKIL NARAYAN§

Abstract.
Approximations of functions with finite data often do not respect certain "structural" properties of the functions. For example, if a given function is non-negative, a polynomial approximation of the function is not necessarily also non-negative. We propose a formalism and algorithms for preserving certain types of such structure in function approximation. In particular, we consider structure corresponding to a convex constraint on the approximant (for which positivity is one example). The approximation problem then converts into a convex feasibility problem, but the feasible set is relatively complicated so that standard convex feasibility algorithms cannot be directly applied. We propose and discuss different algorithms for solving this problem. One of the features of our machinery is flexibility: relatively complicated constraints, such as simultaneously enforcing positivity, monotonicity, and convexity, are fairly straightforward to implement. We demonstrate the success of our algorithm on several problems in univariate function approximation.

Key words. structure-preserving approximation, high-order accuracy

AMS subject classifications. 41A29, 65D15, 65K05, 90C25, 42A16

1. Introduction.

The approximation of functions as a linear combination of basis functions is a foundational technique in numerical analysis and scientific computing. For example, such a linear combination or expansion is often used as an emulator for the original function, or as an ansatz for the solution to a differential equation. If, e.g., the original function is smooth, then such approximations are often accurate, but they may not adhere to other kinds of *structure* that the function possesses. The simplest example of such a structure is positivity: if f is a positive function, an accurate polynomial approximation of f need not also be positive. Other types of structure that arise in practice are monotonicity or maximum and minimum value constraints. If an approximation violates the implicit structure of a function, the resulting computation may produce unphysical predictions, and may cause solvability issues in numerical schemes for solving differential equations [26].

In this paper, we present a general framework for preserving structure in function approximation from a linear subspace. "Structure" in our context refers to fairly general types of linear inequality constraints, including positivity and monotonicity. However, we demonstrate that our setup can also handle more exotic types of constraints. The model by which we impose structure is straightforward: construct the approximation that best fits the available data, subject to the structural constraints. We observe that imposing our type of structure on the approximation corresponds to a convex constraint on the vector of expansion coefficients (i.e., the coordinates of the approximation in a basis of the linear space). Thus, our notion of structure-preserving approximation corresponds to a convex optimization problem. Unfortunately, the resulting convex set is "complicated", and we cannot utilize standard algorithms to solve this problem. We therefore develop two algorithms to solve this problem, each

*Submitted to the editors DATE.

†Scientific Computing and Imaging Institute and School of Computing, University of Utah, Salt Lake City, UT 84112 (vidhi.zala@utah.edu).

‡ Scientific Computing and Imaging Institute and School of Computing, University of Utah, Salt Lake City, UT 84112 (kirby@cs.utah.edu).

§ Scientific Computing and Imaging Institute and Department of Mathematics, University of Utah, Salt Lake City, UT 84112 (akil@sci.utah.edu).

of which is advantageous in different situations. We subsequently formulate a hybrid algorithm that achieves superior performance compared to the original two algorithms. In summary, the contributions of this paper are as follows:

- We formalize a new model for computing structure-preserving approximations of functions. This model can successfully compute function approximations that respect canonical structure such as positivity and/or monotonicity, but can also embed much richer, nontrivial structure, cf. Figure 11. A particular advantage of our approach is that the formalism is identical for all these structure; e.g., the procedure for preserving positivity versus monotonicity is fundamentally the same.
- We show that this model corresponds to a finite-dimensional convex optimization problem. We subsequently characterize the feasible set as an intersection of conic sets (Theorem 3.1), and show that the optimization problem, and hence our structure-preserving approximation model, has a unique solution. See Theorem 3.2.
- Our convex optimization problem can be cast as a problem of projecting onto a convex set (the feasible set). Unfortunately the feasible set is not, in general, a polytopic region in coefficient space. Hence, a finite number of linear inequality constraints cannot characterize the feasible set. We instead characterize the convex feasible set as one with an (uncountably) infinite number of supporting hyperplanes. We use this characterization to develop two types of algorithms for computing the solution to the optimization problem. We also combine these two algorithms into a hybrid approach that is more efficient than either algorithm alone. These three approaches are detailed in Section 4.
- We demonstrate with numerical results in one dimension with polynomial approximations that the resulting algorithm produces approximations satisfying desired constraints. We also show that, for our examples, rates of convergence of polynomial approximation are unchanged compared to the unconstrained case.

Our problem formulation (along with its mathematical properties) holds in the multivariate approximation case; the major drawback in such cases is that our algorithms require global optimization of multivariate functions, which is a difficult problem in general. In order to compute solutions to the constrained optimization problem, our algorithms iteratively "correct" an unconstrained initial guess. For one of our algorithms, these corrections are essentially Dirichlet kernels for the approximation space. We visualize some correction functions for enforcing positivity in polynomial approximation in Figure 1.

In Section 2, we introduce notation, describe the types of constraints we consider, and present the structure-preserving approximation model. Section 3 analyzes the feasible set of the model and shows that a unique solution exists. Section 4 presents our proposed algorithms for computing solutions. Finally, Section 5 contains numerical results and demonstrations.

1.1. Existing and alternative approaches.

There are several existing techniques for building special kinds of structure-preserving approximations. We will frequently use positivity as an explicit example below to make notions clear.

One simple technique in enforcing positivity in function approximation is to enforce positivity as a finite number of points in the domain. This technique makes the feasible set much easier to characterize and results in applicability of several off-the-

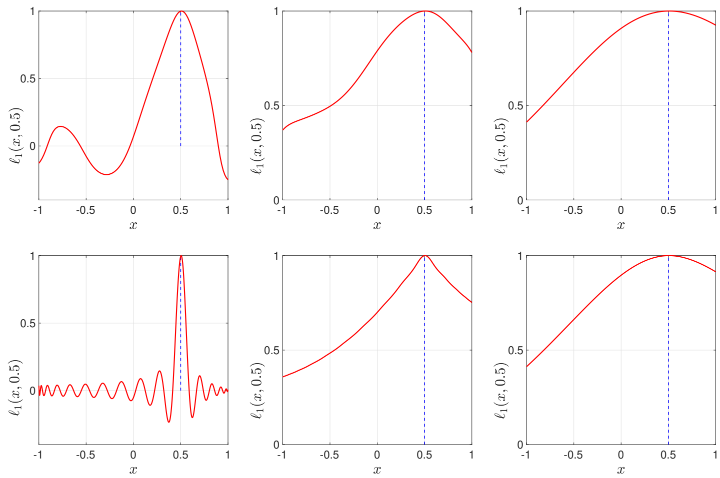

Fig. 1: Correction functions used to enforce positivity in a univariate polynomial approximation. Our algorithm adds scaled/combined versions of these functions to enforce constraints. Shown are corrections targeted to enforce positivity at $x = 0.5$. Correction functions are shown for polynomials of degree 5 (top) and 30 (bottom). The columns correspond to corrections in different ambient Hilbert spaces. Left: $L^2([-1, 1])$. Center: $H^1([-1, 1])$. Right: $H^2([-1, 1])$.

shelf algorithms [7]. However, these approaches do not guarantee positivity on the entire domain, which our structure-preserving model does enforce. Another class of techniques uses mapping methods. For example, if we approximate $\sqrt{f}$ and square the resulting approximation, then the squared approximation is guaranteed to be positive. Finally, there are more complicated successful approaches to construct positive approximations [10]. Although these approaches are attractive, such mapping functions are not easy to construct for more complicated constraints.

Another approach is to adapt the basis; for example, by expanding a function in Bernstein polynomials that are positive on some domain, we can ensure the positivity of the approximation on that domain if all the expansion coefficients are non-negative. Therefore, one forms an approximation subject to the positivity of the coefficients. However, this approach does not yield polynomial reproduction even in simple cases. Consider the following basis for quadratic polynomials in one dimension: $v_1(x) = 1 - x^2$, $v_2(x) = (1 - x)(x + 3)$ and $v_3(x) = (x + 1)(3 - x)$. Note that on $[-1, 1]$, these three functions are all non-negative. However, the (unique) representation of $f \equiv 1$ (that is also non-negative) in this basis is

$$f(x) = -\frac{1}{2}v_1(x) + \frac{1}{4}v_2(x) + \frac{1}{4}v_3(x),$$

which clearly does not have positive expansion coefficients. Alternative approaches use an adaptive construction scheme for certain kinds of constraints [5]; our framework allows much more general constraints and is not restricted by dimension, although in this paper we consider only univariate examples.

In general, each of the techniques above is different, and they must usually be nontrivially adapted when a new kind of structure is desired, or if a different approximation space is used. The model we employ in this work is general-purpose, handling rather general types of constraints and very general approximation spaces. Finally, we note the prior theoretical investigation of error estimates for best structure-preserving approximation [14, 4, 3, 21].

Our formulation constructs an optimization problem of the form

$$\min_{\widehat{\boldsymbol{v}} \in \mathbb{R}^N} \|\boldsymbol{A}\widehat{\boldsymbol{v}} - \boldsymbol{b}\|_2^2 , \quad \text{such that} \quad g(\widehat{v}, y) \leq 0 \; \forall y \in \Omega, \tag{1.1}$$

where $\boldsymbol{A}$ and $\boldsymbol{b}$ are a given matrix and vector (of appropriate sizes), and $g(\cdot, y)$ is a scalar-valued function depending on a parameter y that takes values in an infinite set Ω. Hence, our problem is a *semi-infinite programming* (SIP) problem [17] since the feasible set is described by an infinite number of constraints. As is well-known in SIP methods, even assessing feasibility of a candidate $\widehat{\boldsymbol{v}}$ would require certifying satisfaction of the constraints, i.e., certifying that the maximum of $g(\widehat{\boldsymbol{v}}, \cdot)$ over all Ω is non-positive. Globally solving this so-called lower-level problem is typically the main challenge in SIP algorithms, and is frequently circumvented by means of either discretization approaches (that replace Ω by a finite set) or by local reduction approaches (which partition Ω into subdomains and use specialized approaches on each subdomain). In both cases, there is a discrete approximation of Ω that is constructed (and perhaps refined). For generating positive approximations, this would correspond to requiring positivity at only a finite set of points on the domain.

Our formulation, upon discretization/division of Ω, can certainly leverage SIP algorithms. However, our aim in this paper is to discuss the solution of this problem *without* discretization of Ω, and hence we do not rely on existing SIP algorithms. In particular, we propose algorithms to solve the original SIP problem that presume the ability to compute global solutions to the SIP lower-level problem. Thus, our algorithms differ from many existing SIP algorithms [15, 23], but also inherit the general challenge that global solutions to lower-level SIP problems must be provided.

2. Setup.

Let $\Omega \subset \mathbb{R}^d$ be a spatial domain. Whereas our setup and theoretical results are valid for general Ω and $d \geq 1$, our numerical examples in this paper will primarily be restricted to $d = 1$ with $\Omega = [-1, 1]$. The restriction affects only algorithms and not the model or mathematical properties of our discussion. Consider a Hilbert space formed from scalar-valued functions over Ω:

$$H = H(\Omega) \coloneqq \left\{ f : \Omega \to \mathbb{R} \;\middle|\; \|f\| < \infty \right\}, \qquad \|f\|^2 \coloneqq \langle f, f \rangle ,$$

with $\langle \cdot, \cdot \rangle = \langle \cdot, \cdot \rangle_H$ the inner product on H. We are mostly concerned with "standard" function spaces such as $L^2(\Omega)$ or Sobolev spaces[1]. Let V be an N-dimensional subspace of H, with $\{v_n\}_{n=1}^N$ a collection of orthonormal basis functions,

$$V = \operatorname{span}\{v_1, \ldots, v_N\}, \qquad \langle v_j, v_k \rangle = \delta_{jk},$$

for $j, k = 1, \ldots, N$ and δ_{jk} the Kronecker delta function. For example, if H is $L^2(\Omega)$ with $\Omega = [-1, 1]$ and V is spanned by polynomials up to degree $N - 1$, then one choice for the v_j basis functions are orthonormal Legendre polynomials. We will consider this particular case as an example several times in this paper.

2.1. Riesz representors.

We consider the dual V^* of V, i.e., the space of all bounded linear functionals mapping V to $\mathbb{R}$. The Riesz representation theorem

[1] We formally define L^2 and some Sobolev spaces in Section 5.

guarantees that a functional $L \in V^*$ can be associated with a unique V-representor $\ell \in V$ satisfying

$$L(u) = \langle u, \ell \rangle , \qquad \forall \; u \in V.$$

Furthermore, this $L \leftrightarrow \ell$ identification is an isometry. We will use these facts in what follows. Given L that identifies ℓ, we consider the coordinates $\widehat{\ell}_j$ of ℓ in a V-orthonormal basis,

$$\ell(x) = \sum_{j=1}^{N} \widehat{\ell}_j v_j(x), \qquad \widehat{\ell}_j = \langle \ell, v_j \rangle = L(v_j).$$

Then we have the following relations:

$$\|L\|_{V^*} = \|\ell\|_V = \|\widehat{\boldsymbol{\ell}}\|_2, \qquad \widehat{\boldsymbol{\ell}} = \left(\widehat{\ell}_1,\, \widehat{\ell}_2,\, \ldots,\, \widehat{\ell_N}\right)^T,$$

where $\| \cdot \|_2$ is the Euclidean norm on vectors in $\mathbb{R}^N$.

2.2. Least squares problems. We are interested in a common least squares-type approximation problem. Suppose that $u \in H$ is an unknown function for which we have M pieces of data. We wish to construct an approximation $p \in V$ to u that best matches these data points. We now formulate this abstractly. Let $\phi_1, \ldots, \phi_M$ be M linear functionals on H that are bounded on V. We assume that the observations $\{u_j\}_{j=1}^M = \{\phi_j(u)\}_{j=1}^M \subset \mathbb{R}$ are available to us (and also bounded), and we seek to solve the optimization problem,

$$p = \operatorname*{argmin}_{v \in V} \sum_{j=1}^{M} \left(\phi_j(v) - u_j\right)^2 .$$

For example, if ϕ_j is a point-evaluation (the Dirac mass) at some location $x_j \in \Omega$ for all $j = 1, \ldots, M$, then the problem above is equivalent to

$$p = \operatorname*{argmin}_{v \in V} \sum_{j=1}^{M} \left(v(x_j) - u(x_j)\right)^2 .$$

This problem has a unique solution if the matrix $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ with entries

$$(A)_{m,n} = \phi_m(v_n), \qquad 1 \le n \le N, 1 \le m \le M,$$

has the rank equal to $\dim V = N$; otherwise, infinitely many solutions exist. This least squares problem is well understood and computational algorithms to solve it given data $\phi_j(u)$ are ubiquitous [8].

For an overdetermined system, where $M > N$, the method of ordinary least squares can be used to find an approximate solution. Details for this case are discussed in 2.4 after the discrete formulation of this problem.

2.3. Constraints. The previous section explains how a function p can be constructed from data. However, we are interested in a particular kind of *constrained* approximation. Our investigation can be motivated by the following examples of types of constraints:

- Positivity: $p(x) \ge 0$ for all $x \in \Omega$

- Monotonicity: $p'(x) \geq 0$ for all $x \in \Omega \subset \mathbb{R}$
- Boundedness: $0 \leq p(x) \leq 1$ for all $x \in \Omega$.

Thus, the central focus of this paper is solving a *linearly constrained* least squares problem, where constraints of the above type are imposed. We now give the abstract setup of our constraints, which specializes to the examples above.

Our abstraction defines K *families* of linear constraints; for fixed $k \in \{1, 2, \ldots, K\}$, family k is prescribed by the tuple (L_k, r_k, ω_k):

- ω_k: a subset of Ω.
- r_k: an element of V
- L_k: for $y \in \omega_k$ fixed, $L_k(\cdot, y)$ is a y-parameterized *unit norm* element of V^*.

Our kth family of constraints on v is

$$L_k(v, y) \leq L_k(r_k, y), \qquad y \in \omega_k. \tag{2.1}$$

The subset of V that satisfies constraint family k is

$$E_k \coloneqq \left\{ v \in V \;\middle|\; L_k(v, y) \leq L_k(r, y) \text{ for all } y \in \omega_k \right\}. \tag{2.2}$$

The elements in V that satisfy all K families of constraints simultaneously are

$$E \coloneqq \cap_{k=1}^{K} E_k. \tag{2.3}$$

We assume that E is nonempty, i.e., that the constraints are consistent. Constraints can be inconsistent, e.g., simultaneously enforcing $f(x) \leq 0$ and $f(x) \geq 1$. However, one can create more subtle inconsistencies in more complicated settings. Our procedure does not provide a means to detect inconsistent constraints (and in this case the algorithm will simply not converge). Thus, we rely on the user to ensure consistent constraints. (Note that a corresponding constrained problem has no solution if inconsistent constraints are prescribed.)

Particularly important later will be the formula for the $\{v_j\}_{j=1}^N$-coordinates of the Riesz representor L_k. As in Section 2.1, $L_k(\cdot, y)$ for fixed (k, y) can be identified with its Riesz representor $\ell_k(\cdot, y) \in V$ and its corresponding expansion coefficients $\widehat{\boldsymbol{\ell}}_k(y)$. The unit norm condition of L_k then implies

$$\|L_k(\cdot, y)\|_{V^*}^2 = \left\| \widehat{\boldsymbol{\ell}}_k(y) \right\|_2^2 = \sum_{j=1}^{N} \left(L_k(v_n, y)\right)^2 = 1. \tag{2.4}$$

We consider some examples.

Example 2.1 (Positivity). *Consider $\Omega = [-1, 1]$, and let V be any N-dimensional subspace of $L^2_\mu(\Omega) \cap L^\infty(\Omega)$. We seek to impose $p(x) \geq 0$ for all $x \in \Omega$. Thus, we have $K = 1$, and the linear operator L_1 should be point evaluation, appropriately normalized. Note that point evaluation is not a bounded functional in L^2, but it is on the finite-dimensional space V. Formally, this is*

$$L_1(v, y) \coloneqq -\lambda(y) v(y), \qquad v \in V,$$

where $\lambda(y)$ is chosen so that L_1 has unit norm for every $y \in \omega_k$; the negative sign is chosen so that we can reverse the inequality in (2.1). We set $\omega_k = \Omega$, and choose $r_k \equiv 0 \in V$. Then, the constraint (2.1) is equivalent to $v(y) \geq 0$ for every $y \in \Omega$. The constraint set E_1 defined in (2.2) is

$$E_1 = \left\{ u \in V \;\middle|\; -u(y) \leq 0 \text{ for all } y \in \omega_1 \right\}.$$

It will be useful here to also demonstrate how $\widehat{\boldsymbol{\ell}}_1(y)$ can be computed. For fixed y, we can identify $L_1(\cdot, y)$ via its Riesz representor $\ell_1(\cdot, y)$:

$$\ell_1(\cdot, y) \coloneqq -\lambda(y) \sum_{j=1}^{N} v_j(y) v_j(\cdot) \in V, \qquad \lambda(y) = \left[\sum_{j=1}^{N} v_j^2(y) \right]^{-1/2}, \tag{2.5}$$

so that $\{-\lambda(y) v_j(y)\}_{j=1}^N$ are the entries of $\widehat{\boldsymbol{\ell}}_1(y)$. The formula for λ results from the normalization condition (2.4). Thus, the coefficient vector $\widehat{\boldsymbol{\ell}}_1(y) \in \mathbb{R}^N$ has explicit entries in terms of y and the orthonormal basis $\{v_j\}_{j=1}^N$.

EXAMPLE 2.2 (Monotonicity). *With the same setup as the previous example, we take V as any N-dimensional subspace of $L^2_\mu(\Omega) \cap W^{1,\infty}(\Omega)$, where $W^{1,\infty}(\Omega)$ is the Sobolev space of functions that are in $L^\infty(\Omega)$ and whose derivatives are also in $L^\infty(\Omega)$. Again with $K = 1$, we define L_1 and its corresponding Riesz representor as*

$$L_1(v, y) \coloneqq -\tau(y) v'(y), \qquad v \in V,$$

$$\ell_1(\cdot, y) \coloneqq -\sum_{n=1}^{N} \tau(y) v_n'(y) v_n \in V, \qquad \tau(y) = \left[\sum_{j=1}^{N} \left(v_j'\right)^2(y) \right]^{-1/2},$$

where again τ is determined using the normalization condition (2.4). With $r_1 \equiv 0$, then (2.2) enforces $v'(y) \geq 0$ for all $y \in \Omega$.

EXAMPLE 2.3 (Boundedness). *With the same setup as Example 2.1, we take V as any N-dimensional subspace of $L^2_\mu \cap L^\infty$, and we further assume that V contains constant functions. Let $K = 2$, and define the operators L_1 and L_2 as*

$$L_1(v, y) \coloneqq -v(y), \qquad v \in V,$$
$$L_2(v, y) \coloneqq v(y), \qquad v \in V,$$

for each $y \in \omega_1 = \omega_2 = [-1, 1]$. Then, with constraint functions $r_1 \equiv 0$ and $r_2 \equiv 1 \in V$, we have that E_k, $k = 1, 2$ are the sets

$$E_1 = \left\{ u \in V \;\middle|\; -u(y) \leq 0 \;\forall\, y \in [-1, 1] \right\}, \quad E_2 = \left\{ u \in V \;\middle|\; u(y) \leq 1 \;\forall\, y \in [-1, 1] \right\},$$

so that their intersection E in (2.3) is the set of elements u in V such that $0 \leq u(x) \leq 1$ for each $x \in \Omega$.

EXAMPLE 2.4. *We can also form constraints on different subsets of Ω. With all the notation in the previous example, we change only:*

$$\omega_1 = [-1, 0), \qquad \omega_2 = (0, 1],$$

so that E contains functions u satisfying $u(x) \geq 0$ for $x \in [-1, 0)$ and $u(x) \leq 1$ for $x \in (0, 1]$.

The above examples illustrate the generality of our notation and the intuitive simplicity of the types of constraints that we consider. A constrained version of a least squares problem thus is formulated as

$$p = \operatorname*{argmin}_{v \in E} \sum_{j=1}^{M} \left(\phi_j(v) - u_j\right)^2. \tag{2.6}$$

2.4. Problem discretization. We now formulate the constrained problem (2.6) via coordinates in the basis $\{v_j\}_{j=1}^N$, which results in a discrete form amenable to numerical computation. Any $v \in V$ has the expansion

$$v(x) = \sum_{j=1}^{N} \widehat{v}_j v_j(x),$$

and the expansion coefficient vector $\widehat{\boldsymbol{v}} \coloneqq (\widehat{v}_1, \ldots, \widehat{v}_N)^T \in \mathbb{R}^N$ uniquely identifies the element $v \in V$. This identification defines subsets of $\mathbb{R}^N$ corresponding to the sets E_k:

$$C_k \coloneqq \left\{ \boldsymbol{c} \in \mathbb{R}^N \;\middle|\; \sum_{j=1}^{N} c_j v_j \in E_k \right\} \subset \mathbb{R}^N, \qquad C \coloneqq \bigcap_{k=1}^{K} C_k. \tag{2.7}$$

Then, the optimization problem (2.6) is equivalent to

$$\boldsymbol{c} = \operatorname*{argmin}_{\widehat{\boldsymbol{v}} \in C} \left\| \boldsymbol{A}\widehat{\boldsymbol{v}} - \boldsymbol{b} \right\|_2^2, \qquad b_j = \phi_j(u). \tag{2.8}$$

This problem is again a least squares problem and so is easily solved in principle, but unfortunately in practice the set C is a quite complicated subset of $\mathbb{R}^N$. Nevertheless, C is convex, which is a fact we exploit.

We now consider the case of an overdetermined system, where $M > N$ in (2.8). The method of ordinary least squares can be used to find an approximate solution to (2.8) in this case. This is achieved with the normal equations by decomposition of A as[1],

$$\widehat{\boldsymbol{v}} = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{b}$$

2.5. Geometry of sets. We recall some basic properties of cones and convex sets and functions that we utilize. In all the discussion below, the ambient space is $\mathbb{R}^N$. A set C is convex if, for every $x, y \in \partial C$,

$$\lambda x + (1 - \lambda) y \in C \quad \forall \;\; \lambda \in (0, 1),$$

A set C is a convex cone if, for every $x, y \in C$,

$$ax + by \in C \quad \forall \;\; a, b \geq 0.$$

The set C is an affine convex cone if it is the rigid translate of a convex cone, i.e., if $C = D + z$, where $z \in \mathbb{R}^N$ and D is a convex cone. In this case, we call z the vertex of the cone.

The convex sets we consider are generated by an uncountably infinite number of supporting hyperplanes. Given $y \in \mathbb{R}^N$ and $a \in \mathbb{R}$, a hyperplane H_0 is a set given by $H_0 = \left\{ x \;\middle|\; \langle x, y \rangle = a \right\}$. The hyperplane H_0 separates $\mathbb{R}^N$ into two halfspaces, one of which is

$$H(y, a) \coloneqq \left\{ x \in \mathbb{R}^N \;\middle|\; \langle x, y \rangle \leq a \right\}$$

Note that $H(y, a)$ is a closed set in $\mathbb{R}^N$. A hyperplane $H_0(y, a)$ with an associated halfspace $H(y, a)$ is a supporting hyperplane for a closed convex set C if $C \subset H(y, a)$ and if $H_0(y, a) \cap \partial C \neq \emptyset$.

3. Constrained optimization.

The main task in this paper is solving the optimization problem (2.8). This optimization problem appears simple since it features a quadratic objective, and the feasible set C is convex (which we show in the next section). The main difficulty here is that C is not a computationally simple convex set in $\mathbb{R}^N$, and hence computing, e.g., projections onto this set, is difficult. To begin, we establish that C is convex.

3.1. Constraint set properties.

This section is devoted to establishing that the sets C_k and C are convex cones in $\mathbb{R}^N$. These properties will be used in the construction of algorithms for solving (2.8).

Before proceeding, we note that each inequality function $r_k \in V$ for $k = 1, \ldots, K$, can be translated into its vector of expansion coefficients:

$$r_k(x) = \sum_{j=1}^{N} \widehat{r}_{k,j} v_j(x), \qquad \widehat{\boldsymbol{r}}_k = \left(\widehat{r}_{k,1}, \ldots, \widehat{r}_{k,N}\right)^T. \tag{3.1}$$

Now the definitions of C and C_k immediately yield convexity and conic properties of these sets.

THEOREM 3.1. *The set C is a closed convex set in $\mathbb{R}^N$, and each for $k = 1, \ldots, K$, C_k is a closed, affine convex cone in $\mathbb{R}^N$ with vertex located at $\widehat{\boldsymbol{r}}_k$.*

Proof. Convexity, closure, and conic structure are preserved under isometries. Due to the isometric relation between V and $\mathbb{R}^N$, we can thus prove properties in one space, which extends to the other space. We first show that C_k is closed directly in $\mathbb{R}^N$: Rewriting (2.7) using the definition of E_k, we have

$$C_k = \bigcap_{y \in \omega_k} \left\{ \boldsymbol{c} \in \mathbb{R}^N \mid L_k \left(\sum_{j=1}^{N} c_j v_j, y \right) \leq L_k(r_k, y) \right\} =: \bigcap_{y \in \omega_k} c_k(y).$$

By definition, $c_k(y)$ is actually a halfspace in $\mathbb{R}^N$,

$$c_k(y) = H\left(\widehat{\boldsymbol{\ell}}_k(y), L_k(r_k, y)\right)$$

and hence $c_k(y)$ is a closed set. Therefore, $C_k = \cap_y c_k(y)$ is also a closed set, and thus $C = \cap_k C_k$ is a closed set.
We will now show the convexity and conic properties in V: fix $k \in \{1, \ldots, K\}$ and $y \in \omega_k$. Let $v, w \in V$ be two elements in E_k. For any $\lambda \in [0, 1]$,

$$L_k(\lambda v + (1 - \lambda)w, y) = \lambda L_k(v, y) + (1 - \lambda) L_k(w, y) \leq L_k(r_k, y),$$

where the inequality is true since $v, w \in E_k$. Therefore E_k, and hence C_k, is convex. Thus we also have that C is convex since it's an intersection of convex sets.

We next show that E_k is a cone with the vertex at r_k, i.e., we must show that for any $\tau \geq 0$ and $v \in V$, we have $L_k(r_k + \tau(v - r_k)) \leq L_k(r_k)$. This is true since

$$L_k(r_k + \tau(v - r_k)) = L_k(r_k) + \tau \left[L_k(v) - L_k(r_k)\right] \leq L_k(r_k),$$

so indeed, E_k is a convex cone with the vertex at r_k, and hence C_k is a convex cone with the vertex at $\widehat{\boldsymbol{r}}_k$. ∎

Despite their conic convexity, the sets C_k are not polyhedral in general, and are hence "complicated" to computationally encode. Consider the setup of Example 2.1. If we change the definition of ω_1 to

$$\widetilde{\omega}_1 = \{x_1, \ldots, x_P\} \subset \Omega = [-1, 1].$$

for any arbitrary $P < \infty$, the new constraint set $\widetilde{C}_1 = C(L_1, r_1, \widetilde{\omega}_1)$ is strictly larger than the constraint set C_1 in Example 2.1. In particular, $p \in V$ satisfying $p(x_j) \geq 0$ for $j = 1, \ldots, P$ does not imply that $p(x) \geq 0$ for all $x \in [-1, 1]$ unless V has very special properties (for example, if V contains only certain piecewise constant functions). Note that the supporting hyperplanes of the constraint set $\widetilde{C}$ are $P < \infty$ halfspaces in $\mathbb{R}^N$ and hence $\widetilde{C}$ is polyhedral (if nonempty). However, if V contains polynomials, it is easy to construct a polynomial that is non-negative on $\widetilde{\omega}_1$ but *not* non-negative on Ω. Hence, the constraint set C_1 defined by (L_1, r_1, ω) in example 2.1 is strictly smaller than $\widetilde{C}_1$, here defined by $(L_1, r_1, \widetilde{\omega})$.

Nevertheless, such discretization approaches, i.e. approaches that use a finite set $\widetilde{\omega}_1$ as a surrogate for an infinite set Ω, are common and frequently effective algorithms for solving (2.8), as is commonly done in semi-infinite programming problems. However, in this manuscript we present algorithms that insist on global satisfaction of the constraints, and hence adopt alternative approaches. Thus, the main computational difficulty of our optimization problem is that the set C cannot be exactly represented as a polyhedron in general, and in particular that projections onto C are in general difficult to compute.

3.2. Solutions to (2.8). Our main goal in this section is to demonstrate the unique solution to our constrained optimization problem. The result is straightforward from the closed convexity of the constraint set and strict convexity of the objective function.

THEOREM 3.2. *Assume that the design matrix $\boldsymbol{A}$ has rank N and the feasible set C is nonempty. Then, the constrained optimization problem (2.8) has a unique solution.*

Proof. The first step is to observe that since $\boldsymbol{A}$ has full column rank, we can write the problem in transformed coordinates as a convex feasibility problem (specifically as a projection problem). Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ be the *reduced* singular value decomposition of $\boldsymbol{A}$. Since $\mathrm{rank}(\boldsymbol{A}) = N \leq M$, Σ is $N \times N$, diagonal, and invertible; $\boldsymbol{V}$ is $N \times N$ and orthogonal; and $\boldsymbol{U}$ is $M \times N$ with orthonormal columns.

With $\mathcal{P}_{\mathcal{W}}$ the $\mathbb{R}^N$-orthogonal projector onto a subspace $\mathcal{W}$, and $\mathcal{R}(\boldsymbol{A})$ the range of $\boldsymbol{A}$, then (2.8) can be written as

$$\begin{aligned}
\underset{\widehat{\boldsymbol{v}} \in C}{\mathrm{argmin}} \left\| \boldsymbol{A}\widehat{\boldsymbol{v}} - \boldsymbol{b} \right\|_2^2 &= \underset{\widehat{\boldsymbol{v}} \in C}{\mathrm{argmin}} \left\| \mathcal{P}_{\mathcal{R}(\boldsymbol{A})^\perp} \boldsymbol{b} \right\|_2^2 + \left\| \boldsymbol{A}\widehat{\boldsymbol{v}} - \mathcal{P}_{\mathcal{R}(\boldsymbol{A})} \boldsymbol{b} \right\|_2^2 \\
&= \underset{\widehat{\boldsymbol{v}} \in C}{\mathrm{argmin}} \left\| \boldsymbol{\Sigma}\boldsymbol{V}^* \widehat{\boldsymbol{v}} - \boldsymbol{U}^* \boldsymbol{b} \right\|_2^2 \\
&= \boldsymbol{V}\boldsymbol{\Sigma}^{-1} \underset{\boldsymbol{z} \in \boldsymbol{\Sigma}\boldsymbol{V}^* C}{\mathrm{argmin}} \left\| \boldsymbol{z} - \boldsymbol{U}^* \boldsymbol{b} \right\|_2^2, \qquad (3.2)
\end{aligned}$$

where $\boldsymbol{\Sigma}\boldsymbol{V}^* C \coloneqq \left\{ \boldsymbol{\Sigma}\boldsymbol{V}^* \boldsymbol{y} \in \mathbb{R}^N \;\middle|\; \boldsymbol{y} \in C \right\}$. Thus, (2.8) has a unique solution if and only if

$$\underset{\boldsymbol{z} \in \boldsymbol{\Sigma}\boldsymbol{V}^* C}{\mathrm{argmin}} \left\| \boldsymbol{z} - \boldsymbol{U}^* \boldsymbol{b} \right\|_2^2 \qquad (3.3)$$

has a unique solution. Theorem 3.1 establishes that C is closed and convex; thus, $\boldsymbol{\Sigma V}^* C$ is a linear transformation of a closed convex set, so it is also closed and convex. Therefore, (3.3) seeks the $\ell^2(\mathbb{R}^N)$-closest point to $\boldsymbol{U}^*\boldsymbol{b}$ from a nonempty, closed, convex set. The Hilbert Projection Theorem guarantees the existence and uniqueness of such a point. ∎

The study of existence and uniqueness of approximations under convex constraints is not new [22, 19]. Indeed, our result is a corollary of these earlier results, but we have presented a brief proof above in order to be self-contained.

4. Algorithms: Convex Feasibility.

We now concentrate on solving the problem defined by (2.8), equivalently (3.3). To simplify the presentation, we will assume first that $\boldsymbol{A} = \boldsymbol{I}$ so that both (3.3) and (2.8) reduce to

$$\operatorname*{argmin}_{\boldsymbol{c} \in C} \|\boldsymbol{c} - \boldsymbol{b}\|_2^2, \tag{4.1}$$

i.e., a standard problem of projecting $\boldsymbol{b}$ onto a convex set C. The main bottleneck to applying standard optimization tools is that the feasible set C is not easily defined in terms of a finite number of conditions on $\boldsymbol{c}$. The difficulty in our problem is not in minimizing the objective function, but instead the convex feasibility problem, i.e., to identify points in the convex feasible set.

Some of the most successful algorithms for solving the convex feasibility problem are alternating- or splitting-type algorithms. If $C_1, \ldots, C_r$ are convex sets with nonempty intersection C, these algorithms assume that projection onto any one of these sets is computationally feasible. A solution to (4.1) can be computed by alternating these individual projections. The original projection onto convex sets algorithm via iteration is due to Von Neumann [25], and much work has proceeded from this [9, 16, 2, 12, 18, 13]. When $r > 2$, the alternating algorithm becomes a cyclic one, and these cyclic projection algorithms have substantial theoretical underpinning, including convergence guarantees.

The difficulty in applying these algorithms to our situation is that they characterize the feasible region with a *finite* number of convex sets. Although our collection of sets $\{C_j\}_{j=1}^K$ is finite, we do not know how to project onto any of them individually. However, we have

$$C = \bigcap_{k=1}^K C_k = \bigcap_{k=1}^K \bigcap_{y \in \omega_k} H_k(y), \tag{4.2}$$
$$H_k(y) \coloneqq H\left(\boldsymbol{\ell}_{\boldsymbol{k}}(y), L_k(r_k, y)\right),$$

so that C is comprised of an (in general uncountably) *infinite* intersection of halfspaces, each of which is straightforward to project onto, see Figure 2 for a geometric visual. Our strategy here is to generalize certain types of cyclic/alternating algorithms to the case of an infinite number of convex sets (halfspaces). We broadly employ two strategies: greedy projection and averaged projection.

The major ingredient in our approaches is the ability to project onto any halfspace $H_k(y)$. Since the functionals $L_k(\cdot, y)$ are unit norm, a computation shows that the signed distance between some point $\boldsymbol{c} \in \mathbb{R}^N$ and $H_k(y)$ is

$$\operatorname{sdist}(\boldsymbol{c}, H_k(y)) = L_k(r_k, y) - \left\langle \widehat{\boldsymbol{\ell}}_k(y), \boldsymbol{c} \right\rangle, \tag{4.3}$$

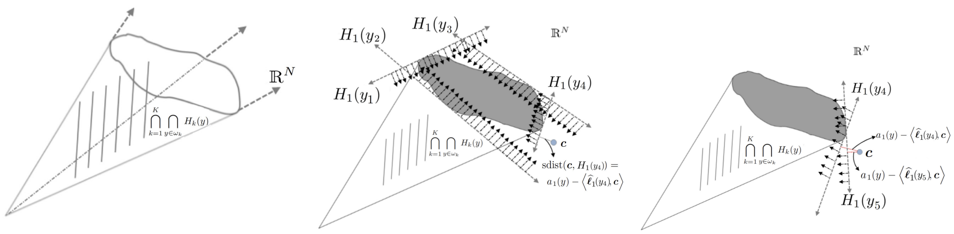

Fig. 2: Left: The hatched volume represents the closed convex cone C_1 Middle: Geometric depiction of intersecting hyperspaces $H_1(y)$ and their respective boundaries defined by hyperplanes parameterized by $y \in \Omega$. Also shown is the distance calculation corresponding to (4.3). Right: A scenario that demonstrates the greedy strategy to select the direction in which y moves in the next step of the algorithm: $H_1(y_4)$ is farther away from $\boldsymbol{c}$ than $H_1(y_5)$. The optimization (4.5) seeks the hyperplane that is farthest away from $\boldsymbol{c}$.

which is positive if $\boldsymbol{c} \in H_k(y)$ and negative otherwise. Thus, the nearest-distance projection of $\boldsymbol{c}$ onto $H_k(y)$ is

$$P_{H_k(y)}\boldsymbol{c} = \boldsymbol{c} + \boldsymbol{\ell}_k(y) \min\left\{0, \text{sdist}(\boldsymbol{c}, H_k(y))\right\}.$$

We consider an example to illustrate that these projections are easily computable.

Example 4.1. *Consider the positivity constraint setup of Example 2.1. The constraint functional $L_1(\cdot, y)$ is a (normalized, negative) point evaluation at y, and $\{v_n\}_{n=1}^N$ are the first N orthonormal Legendre polynomials on $[-1, 1]$. Then, the Riesz representor $\ell_1(y) \in V$ and its coordinates $\{\widehat{\ell}_{1,j}(y)\}_{j=1}^N$ are explicit in terms of the Legendre polynomials via (2.5). In the context of harmonic analysis, $\ell_1(y)$ is the y-centered, negative, normalized Dirichlet kernel for V. The function r_1 describing the constraint is $r_1 \equiv 0$, so that $\widehat{\boldsymbol{r}}_1 = \boldsymbol{0}$ and $L_1(r_1, y) = 0$. Now let $v \in V$ be any element with coordinates $\boldsymbol{c} \in \mathbb{R}^N$ in the orthonormal Legendre polynomials. Then,*

$$\text{sdist}(\boldsymbol{c}, H_1(y)) = -\left\langle \widehat{\boldsymbol{\ell}}_1(y), \boldsymbol{c} \right\rangle = \lambda(y) v(y). \tag{4.4}$$

Thus, the signed distance at $y \in \Omega$ is simply scaled evaluation of the original function v. The projection of $\boldsymbol{c}$ onto the halfspace defined by $H_k(y)$ is therefore

$$P_{H_k(y)}\boldsymbol{c} = \boldsymbol{c} + \widehat{\boldsymbol{\ell}}_1(y) \min\left\{0, v(y)\lambda(y)\right\}.$$

Note that since $\lambda(y) > 0$, this projection equals $\boldsymbol{c}$ if $v(y) \geq 0$, as expected.

4.1. Greedy projections. Since projections onto individual halfspaces defined by $H_k(y)$ are relatively simple to compute, we can devise one algorithm for computing the solution to (4.1) as a modification of cyclic projections. Although cyclic projection-type algorithms proceed by cycling through the enumerable constraint sets, our (uncountably) infinite collection of sets prevents such a simple cycling. Instead, we can project onto the *farthest* or most violated constraint, i.e., with

$$(y^*, k^*) \coloneqq \underset{y \in \omega_k, k \in [K]}{\text{argmin}} \; \text{sdist}(\boldsymbol{c}, H_k(y)), \tag{4.5}$$

We can update $\boldsymbol{c}$ via

$$\boldsymbol{c} \leftarrow \boldsymbol{c} + \boldsymbol{\ell}_{k^*}(y^*) \min\{0, \mathrm{sdist}(\boldsymbol{c}, H_{k^*}(y^*))\}. \tag{4.6}$$

The geometric picture associated to (4.5) is shown in the right panel of Figure 2. The update process (4.6) can be repeated, resulting in an iterative algorithm. We summarize this procedure in Algorithm 4.1. This algorithm proceeds by iteratively "correcting" the vector $\boldsymbol{c}$ in (4.6). The associated operation in the function space V is that an unconstrained function is additively augmented by the Riesz representor correction function $\ell_{k^*}(y^*) \in V$. These corrections are visualized in Figure 1 for polynomials. A more detailed understanding of these function is provided in Figures 4 and 5 where we show $\ell_k(y)(x)$ as a function of (x, y) for polynomials.

Algorithm 4.1 Iterative greedy projection algorithm to compute the solution to (4.1). The unspecified "extra termination criteria" can be standard metrics, such as number of iterations, improvement in objective function, etc.

1: Input: constraints $(L_k, r_k, \omega_k)_{k=1}^K$
2: Input: coordinates $\boldsymbol{c} \in \mathbb{R}^N$ of a function $v \in V$
3: **while** True **do**
4: Compute (y^*, k^*) via (4.5).
5: **if** $\mathrm{sdist}(\boldsymbol{c}, H_{k^*}(y^*)) \geq 0$ or extra termination criteria triggered **then**
6: Break
7: **end if**
8: Update $\boldsymbol{c}$ via (4.6).
9: **end while**
10: **return** $\boldsymbol{c}$

Note that the bulk of the computational effort in Algorithm 4.1 corresponds to line 4 where the Ω-global optimization problem (4.5) must be solved, which can be of considerable expense at each iteration. We explain in Appendix A how we accomplish this optimization for univariate polynomial spaces V.

It is straightforward to establish that under a special kind of termination in Algorithm 8, we obtain the solution to (4.1).

PROPOSITION 4.1. *If Algorithm 4.1, without any extra termination criteria, terminates after one only iteration of line 8, then the output* $\mathbf{c}$ *is the solution to* (4.1).

Proof. Assume without loss that the input to algorithm 4.1 **$\boldsymbol{c}$ is not in C**. By (4.2), we have

$$\mathrm{dist}\,(\boldsymbol{c}, C) \geq \mathrm{dist}\,(\boldsymbol{c}, H_k(y)),$$

for any (y, k). Let (y^*, k^*) be the solution to (4.5), and note that since $\boldsymbol{c} \notin C$,

$$\mathrm{dist}\,(\boldsymbol{c}, H_k(y)) = -\mathrm{sdist}(\boldsymbol{c}, H_{k^*}(y^*)) > 0.$$

The assumption that Algorithm 4.1 terminates after one iteration implies that

$$\boldsymbol{d} := \boldsymbol{c} + \widehat{\boldsymbol{\ell}}_{k^*}(y^*)\mathrm{sdist}(\boldsymbol{c}, H_{k^*}(y^*)) \in C.$$

Note $\boldsymbol{d}$ is returned by the algorithm. $\boldsymbol{c} \notin C$, $\boldsymbol{d} \in C$, $\left\| \widehat{\boldsymbol{\ell}}_{k^*}(y^*) \right\|_2 = 1$, and that

$$\mathrm{dist}\,(\boldsymbol{c}, C) \geq -\mathrm{sdist}(\boldsymbol{c}, H_{k^*}(y^*)),$$

all imply that the above inequality is actually an equality, and thus $\boldsymbol{d}$ solves (4.1). ◻

In standard cyclic projection algorithms, it is well known that directly projecting onto each set in each iteration produces a suboptimal trajectory for the iterates. The greedy algorithm described in this section suffers from this as well, which we show in the numerical results section. An improvement that somewhat ameliorates this deficiency is accomplished by averaging these projections.

4.2. Averaged projections. A simple strategy to mitigate the oscillatory iteration trajectory produced by iterative greedy projections is via averaging. Precisely, given a current iterate $\boldsymbol{c}$, we identify the subset of Ω where our constraints are violated:

$$\omega_k^- \coloneqq \left\{ y \in \omega_k \;\middle|\; \mathrm{sdist}(\boldsymbol{c}, H_k(y)) < 0 \right\}. \tag{4.7}$$

Under mild assumptions on V, e.g., that it contains only piecewise continuous functions, ω_k^- is either the trivial (empty) set, or of positive Lebesgue measure. (In other words, it cannot be a discrete or nontrivial measure-0 set.) Assume for simplicity that ω_k^- has a positive Lebesgue measure for each k. We then produce an update by a normalized average of corrections corresponding to values of y in ω_k^-:

$$\boldsymbol{c} \leftarrow \boldsymbol{c} + \sum_{k=1}^{K} \frac{1}{K|\omega_k^-|} \int_{\omega_k^-} \widehat{\boldsymbol{\ell}}_k(y) \mathrm{sdist}\left(\boldsymbol{c}, H_k(y)\right) \mathrm{d}y. \tag{4.8}$$

Above, $|\omega_k^-|$ is the measure of $\omega_k^- \subset \Omega$. We again illustrate with an example that these quantities are computable.

Example 4.2. *Consider the positivity constraint setup of Example 2.1. As we saw in Example 4.1, the signed distance for our single constraint is given by (4.4). Note that in this one-dimensional setup with finite-degree polynomials, the set ω_k^- is a finite union of subintervals of $[-1, 1]$, and hence the measure $|\omega_k^-|$ is just the sum of the lengths of these subintervals. Then, the correction term on right-hand side of the update scheme (4.8) is*

$$-\frac{1}{|\omega_k^-|} \int_{\omega_k^-} \widehat{\boldsymbol{\ell}}_1(y) \lambda(y) v(y) \mathrm{d}y = -\frac{1}{|\omega_k^-|} \sum_{j=1}^{N} \boldsymbol{e}_j \int_{\omega_k^-} \lambda^2(y) v(y) v_j(y) \mathrm{d}y,$$

where $\boldsymbol{e}_j$, $j \in [N]$ are the cardinal unit vectors in $\mathbb{R}^N$. Thus, the integrals that must be computed have smooth integrands and can be efficiently approximated by standard quadrature rules, assuming the endpoints of the subintervals defining ω_k^- can be identified.

A variation of Algorithm 4.1 that uses this averaging approach is nearly identical: the only change required is that the update of the coefficient vector $\boldsymbol{c}$ in line 8 should be replaced by the update in (4.8).

Figure 3 visually depicts both the greedy and averaged projections idea where V is a univariate space of polynomials and the constraint is positivity (i.e., Example 2.1). In particular, the value y^* that solves the greedy optimization problem (4.5) is shown, along with the averaging set ω_1^- identified in (4.7).

4.3. Hybrid algorithms. In experimentation, we have found that hybrid combinations of the greedy approach of Section 4.1 and the averaged approach of Section 4.2 work better than any algorithm alone. In particular, the greedy algorithm works well when $\boldsymbol{c}$ is "close" to the solution, but the averaged algorithm works better for an iterate that is "far" away. Thus, we utilize a standard switching procedure in optimization depending on the proximity to a basin of attraction.

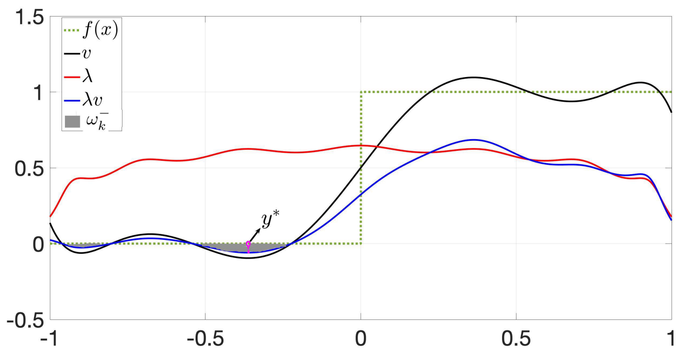

Fig. 3: v is the unconstrained $L^2([-1,1])$ projection of the step function $f(x)$ onto the space of degree-7 polynomials. For the positivity setup of Example 2.1, the greedy point y^* defined in (4.5) is shown, and the averaging set $\omega_1^- \subset [-1,1]$ defined in (4.7). Also plotted is the signed distance $\lambda(y)v(y)$ of v to $H_1(y)$.

Through experimentation, we have found that the following switching mechanism works well: We perform averaged projections until the norm of the correction (4.8) reaches a certain tolerance. After a condition is met, we switch to greedy projections. The switching condition is the following: if i is the iteration index, consider the ratio,

$$\alpha_i = \frac{\mathrm{sdist}(\boldsymbol{c}_i, H_{k_i^*}(y_i^*)}{\mathrm{sdist}(\boldsymbol{c}_{i-1}, H_{k_{i-1}^*}(y_{i-1}^*)}.$$

Our switching condition is triggered when $|\alpha_i - \alpha_{i-1}| \le \epsilon$, for a user-specified ϵ. At this point, we perform one more averaged update of the form (4.8), but multiply the right-hand side correction by $1/\alpha_i$. Subsequently, greedy projections as in (4.6) are performed. While this procedure is quite *ad hoc*, we have observed that it consistently performs better than other hybrid variants we have tried.

4.4. Algorithms for polynomial subspaces. As described in previous sections, the main computational expense in our convex optimization algorithm is the minimization of the signed distance function in (4.5) (for the greedy and hybrid algorithms) and identification and integration over the set ω_k^- in (4.7) (for the averaged and hybrid algorithms). Such problems for *general* function spaces are difficult to solve, and efficient algorithms will likely depend on what kinds of functions the subspace V contains.

When V contains univariate polynomials, all the tasks in the algorithm can be reduced to the problem of computing roots of polynomials, and hence are feasible in principle. We accomplish this computationally by computing the spectrum of a confederate matrix, although more sophisticated and practically effective methods are known. We describe this formulation and details of the approach in Appendix A.

4.5. Nonidentity matrices $\boldsymbol{A}$. The optimization problem we seek to solve is (2.8); the algorithms in this section have proceeded under the assumption that $\boldsymbol{A} = \boldsymbol{I}$. When this is not the case, we must first solve (3.3), so that the full solution is (3.2). Thus, we focus on the problem

$$\underset{\boldsymbol{z} \in \boldsymbol{\Sigma} \boldsymbol{V}^* C}{\operatorname{argmin}} \left\| \boldsymbol{z} - \boldsymbol{U}^* \boldsymbol{b} \right\|_2 . \tag{4.9}$$

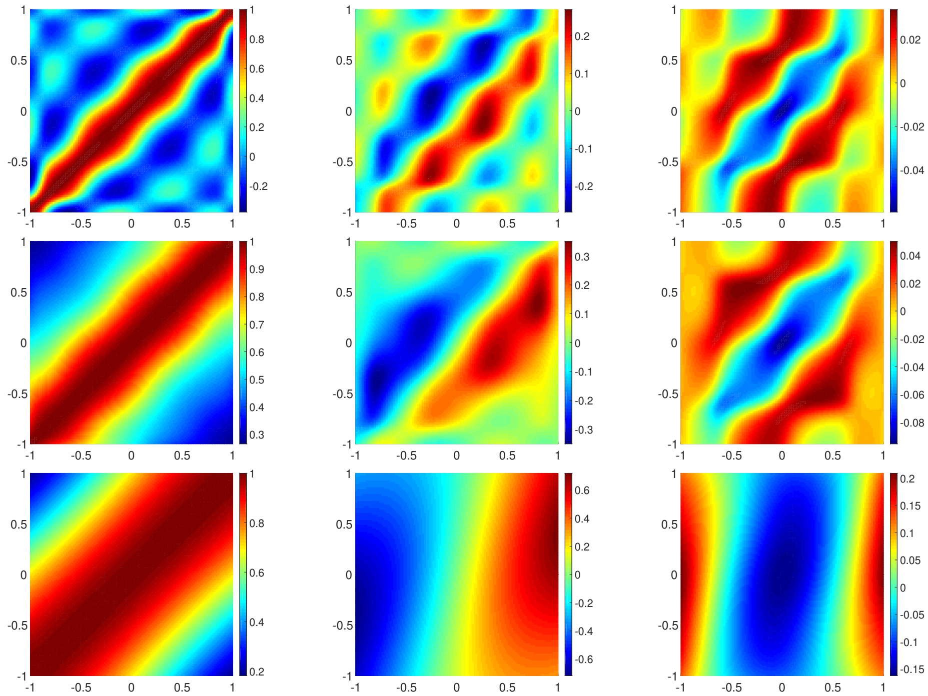

Fig. 4: Correction functions for degree-5 polynomial approximation. Plots of $\ell_k(y)(x)$ are shown as functions of (x, y) for various constraints enforcing positivity of the kth derivative (rows) and ambient Hilbert spaces (columns). Top: $k = 0$ positivity; middle: $k = 1$ monotonicity; bottom: $k = 2$ convexity. Left: $L^2([-1, 1])$; middle; $H^1([-1, 1])$; bottom: $H^2([-1, 1])$.

Note that the only difference between this optimization and the simplified version (4.1) is that the feasible set is $\boldsymbol{\Sigma V}^* C$ instead of C so that we need only address the presence of the linear map $\boldsymbol{\Sigma V}^*$. Since C is closed and convex, then $\boldsymbol{\Sigma V}^* C$ is also closed and convex, and in particular is defined as the intersection of closed, conic, convex sets $\widetilde{C}_k$:

$$\boldsymbol{\Sigma V}^* C =: \widetilde{C} = \bigcap_{k=1}^{K} \widetilde{C}_k \coloneqq \bigcap_{k=1}^{K} \boldsymbol{\Sigma V}^* \boldsymbol{C}_k.$$

Thus, all our previous algorithms apply, except that we need to only transform (L_k, r_k, ω_k) for C_k into the appropriate quantities for $\widetilde{C}_k$. These transformations are straightforward but technical, so we omit showing them explicitly.

5. Numerical results. In all that follows, f is a given function in a Hilbert space H. Given a finite-dimensional space $V \subset H$, the function v is the H-best projection onto V, which does not in general satisfy any structural constraints. (Note from discussion in Section 4.5 that extensions to, e.g., collocation-based approximations, are straightforward.) The function $\tilde{v}$ is the output of the constrained optimization procedure.

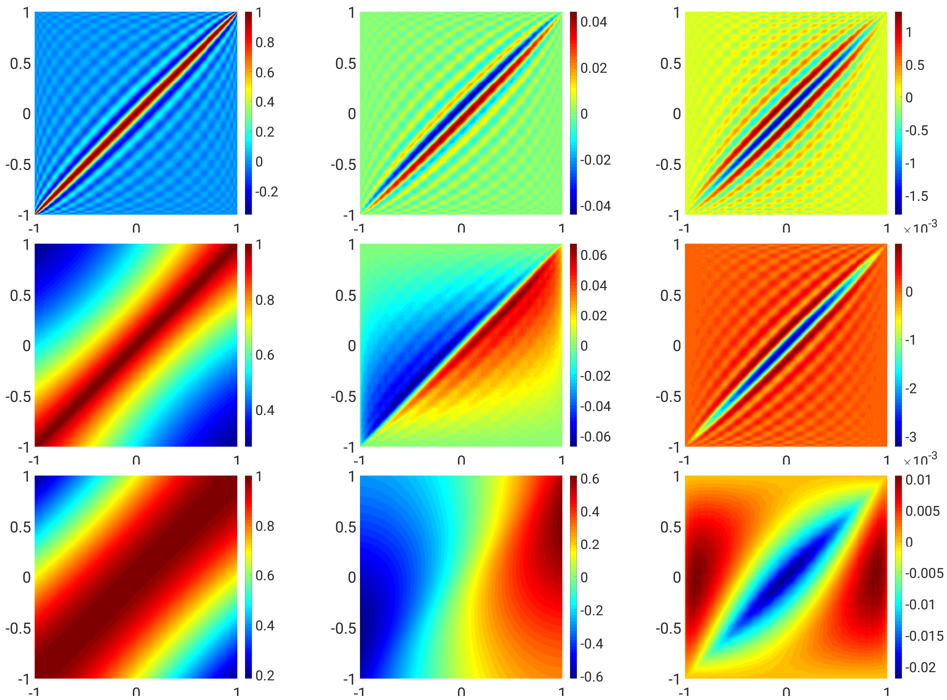

Fig. 5: Correction functions for degree-30 polynomial approximation. Plots of $\ell_k(y)(x)$ are shown as functions of (x, y) for various constraints enforcing positivity of the kth derivative (rows) and ambient Hilbert spaces (columns). Top: $k = 0$ positivity; middle: $k = 1$ monotonicity; bottom: $k = 2$ convexity. Left: $L^2([-1, 1])$; middle; $H^1([-1, 1])$; right: $H^2([-1, 1])$.

With the univariate Sobolev spaces,

$$H^q([-1,1]) \coloneqq \left\{ f : [-1,1] \to \mathbb{R} \;\middle|\; \|f\|_{H^2} < \infty \right\}, \quad \|f\|^2_{H^q} \coloneqq \sum_{j=0}^{q} \int_{-1}^{1} \left[f^{(j)}(x) \right]^2 \mathrm{d}x,$$

our examples will consider the ambient Hilbert space H as $H^0 (= L^2)$, H^1, or H^2. The subspace V in all our experiments is the space of polynomials up to degree $N - 1$:

$$V = \left\{ p : [-1,1] \to \mathbb{R} \;\middle|\; \deg p \le N \right\}.$$

Our test functions f_j are defined iteratively for $j \ge 1$ as,

$$f_{j+1}(x) = c_{j+1} \int_{-1}^{x} f_j(t) \mathrm{d}t, \qquad f_0(x) = \begin{cases} 0, & x \le 0, \\ 1, & x > 0 \end{cases},$$

where c_{j+1} are normalization constants chosen so that $f_{j+1}(1) = 1$. Thus, f_j has j weak L^2 derivatives. Finally, most of our results will consider intersections of the following four types of constraint sets in V:

- (Positivity) $F_0 \coloneqq \{ f \in H \mid f(x) \ge 0 \; \forall \, x \in [-1,1] \}$
- (Boundedness) $G_0 \coloneqq \{ f \in H \mid f(x) \le 1 \; \forall \, x \in [-1,1] \}$

- (Monotonicity) $F_1 \coloneqq \{f \in H \mid f'(x) \geq 0 \, \forall x \in [-1,1]\}$
- (Convexity) $F_2 \coloneqq \{f \in H \mid f''(x) \geq 0 \, \forall x \in [-1,1]\}$

Our final example considers a slightly more exotic set of constraints, which we discuss later.

In order to understand how much our algorithms "change" the input v when producing constrained approximation $\tilde{v}$, we measure the following quantity:

$$\eta \coloneqq \frac{\|v - \tilde{v}\|_H}{\|f - v\|_H}. \tag{5.1}$$

Since $f - v$ is H-orthogonal to V, then

$$\|f - \tilde{v}\|_H^2 = (1 + \eta^2)\|f - v\|_H^2.$$

Thus, $\sqrt{1+\eta^2}$ measures the error in the constrained approximation relative to the (best) unconstrained approximation. Values on the order of 1 imply that this optimization problem commits an additional error that is approximately the same as the error committed by the best (unconstrained) approximation.

Algorithm 4.1 is the greedy algorithm, but it is the template for the averaging and hybrid algorithms as well. For example, a hybrid algorithm needs to replace only line 8 in that algorithm by the update (4.8). However, we have left some details of the termination criterion in line 5 unexplained. For example, we do not actually enforce $\mathrm{sdist}(\boldsymbol{c}, H_{k^*}(y^*)) \leq 0$ as stated due to finite precision. Instead, we enforce

$$\mathrm{sdist}(\boldsymbol{c}, H_{k^*}(y^*)) \leq \delta, \qquad \delta > 0, \tag{5.2}$$

where we set $\delta = 10^{-10}$ and have implemented the procedure in double precision. In addition, the number of iterations I required before termination will also be reported.

5.1. Algorithm comparison. A short summary of all the experiments investigating the hybrid approaches and their comparison with the greedy and the averaging methods is given in the Table 1.

	$N=6$				$N=31$			
	I		η		I		η	
ϵ	10^{-3}	10^{-5}	10^{-3}	10^{-5}	10^{-3}	10^{-5}	10^{-3}	10^{-5}
Greedy	20	20	1.147	1.147	23	23	0.986	0.986
Averaging	36	36	1.148	1.148	383	383	0.985	0.985
Hybrid	4	16	1.1464	1.148	2	3	1.142	1.054

Table 1: Performance summary of three proposed algorithms on the test function $f = f_2$ for different values of ϵ, where ϵ is as described in Section 4.3. The constraint set is $E = F_0$.

5.2. Function approximation examples. We present two examples of function approximation to preserve structure in this section. The first example takes $H = H^0$ and the test function $f = f_0$, which is a step (discontinuous) function. We present results for different N (the dimension of V) and different constraints. Figure 6 illustrates the results of the greedy algorithm. We compare medium-degree polynomial approximation $N = 6$ with high-degree polynomial approximation $N = 31$. The three kinds of constraints are (a) positivity, (b) positivity and boundedness, and (c) positivity, boundedness, and monotonicity. We observe that both the positivity and monotonicity constraints accomplish what is desired: the approximation $\tilde{v}$ satisfies the desired constraints, but still features Gibbs'-type oscillations. However, enforcing monotonicity as well results in a nonoscillatory approximation. All computed values of $\eta < 1$ show that the constrained approximation commits an error that is comparable to that of the H-best approximation.

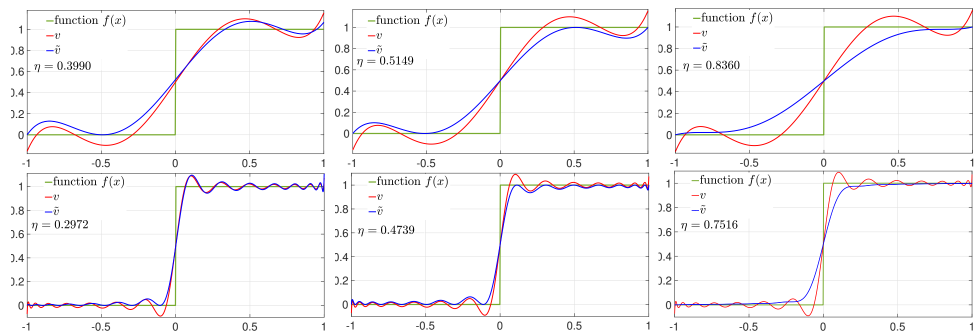

Fig. 6: Greedy algorithm results: Test function f_0 for different constraint sets E and polynomial spaces V. Top: $N = \dim V = 6$, bottom: $N = \dim V = 31$. Left: Constraint $E = F_0$. Center: Constraint $E = F_0 \cap G_0$. Right: Constraint $E = F_0 \cap G_0 \cap F_1$.

Our second experiment uses the test function $f = f_2$, which has a piecewise-constant second derivative. We use a fixed constraint: positivity, monotonicity, and convexity. Using again $N = 6$ and $N = 31$, we investigate the approximation for different ambient spaces $H = H^0$, H^1, and H^2. Results are displayed in Figure 7. We observe much larger values of η in this experiment, but note that the values of η decrease as the order of the Sobolev space increases. We also observe that the visual discrepancy between the constrained approximation and the underlying function is also considerably larger in this experiment. However, the approximation quality still appears good for the larger value of $N = 31$.

5.3. Constrained approximation as a nonlinear filter. The right-hand panels in Figure 6 show that the monotonicity constraint removes oscillations in the approximation. These empirical results suggest that the constrained optimization procedure is a type of spectral filter. There is a stronger theoretical motivation for this observation as well.

PROPOSITION 5.1. *Let $E \subset V$ be a nonempty, closed, convex set in H. Given some $v \in V$, let $\tilde{v}$ be the solution to* (2.6) *(i.e., also the solution to* (2.8)*). If $0 \in E$, then, $\|\tilde{v}\| \leq \|v\|$.*

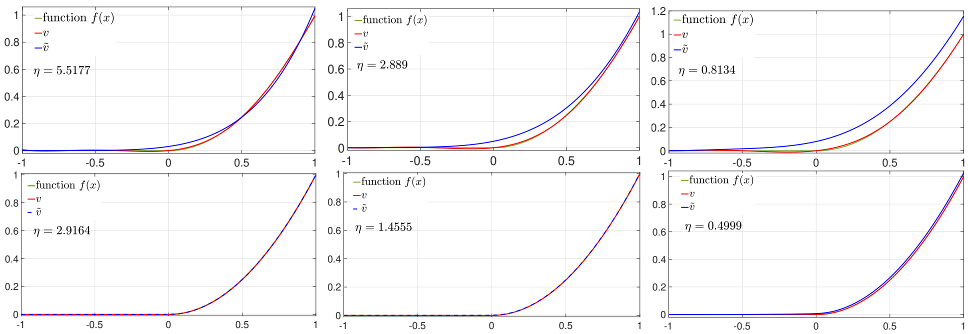

Fig. 7: Test function f_2 for different polynomial spaces V and ambient spaces H. The constraint is $E = F_0 \cap F_1 \cap F_2$. Top: $N = 6$, bottom: $N = 31$. Left: $H = H^0$. Center: $H = H^1$. Right: $H = H^2$.

Proof. Projections onto closed convex sets in Hilbert spaces are nonexpansive [11]. I.e., $\|\tilde{v} - P(0)\| \le \|v - 0\|$, where $P : V \to E$ is the projection operator from V to E. Since $0 \in E$, then $P(0) = 0$. ∎

In general, the assumption that E is closed and convex is automatically satisfied from our apparatus in Sections 2 and 3. The only nontrivial requirement is that $v = 0$ is a member of the constraint set E. All the examples in Figures 6 and 7 satisfy $0 \in E$, and thus we expect that the optimization problem decreases the norm of the function, just as a standard linear filter would. Note, however, that our "filter" (optimization) is a nonlinear map.

To illustrate this filter interpretation, we compare in Figures 8 and 9 the magnitude of the before-optimization and after-optimization expansion coefficients. These figures correspond to the experiments in Figures 6 and 7, respectively.

For the step function example shown in Figure 8, we see that when monotonicity is enforced, there is a steeper decay of the higher order coefficients in the constrained approximation. The stronger decay of coefficients is also observed when only positivity/boundedness is enforced, but the increase in decay is less pronounced. All these observations are qualitatively consistent with Figure 6. We emphasize that this constrained optimization procedure is nonlinear, so that our approximation cannot easily be written in coefficient space as a standard (linear) spectral filter.

5.4. Convergence rates. Optimal Hilbert space projections of smooth functions onto polynomial spaces converge at a rate commensurate with the function smoothness. We investigate in this section whether the corresponding *constrained* projections have similar convergence rates. In Figure 10 we show convergence of $H = L^2$-optimal (unconstrained) polynomial projections versus the output from our constrained optimization procedure. Our constrained approximations are less accurate, but the convergence *rates* are unchanged.

5.5. More complicated constraints. Finally, we show that our formalism allows for more complicated constraints than the ones we have previously shown. With $H = H^0$ and V a space of degree-$(N-1)$ polynomials as before, we consider two new kinds of constraints:

- $J_1 = \{f \in V \mid f(x) \ge |x| \ \ \forall\, x \in [-1, 1]\}$
- $J_2 = \{f \in V \mid -\operatorname{sign}(x) f(x) \ge |x| \ \ \forall\, x \in [-1, 1]\}$

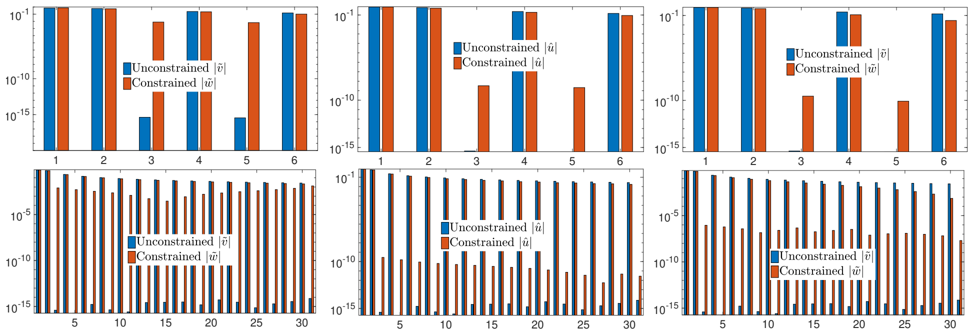

Fig. 8: Companion to Figure 6. Bar plot showing unconstrained projection coefficients magnitude $|\widetilde{v}_j|$ vs various constrained projection coefficients magnitude $|\widetilde{w}_j|$. Top: $N = 6$. Bottom: $N = 31$. Left: Constraint $E = F_0$. Center: Constraint $E = F_0 \cap G_0$. Right: Constraint $E = F_0 \cap G_0 \cap F_1$.

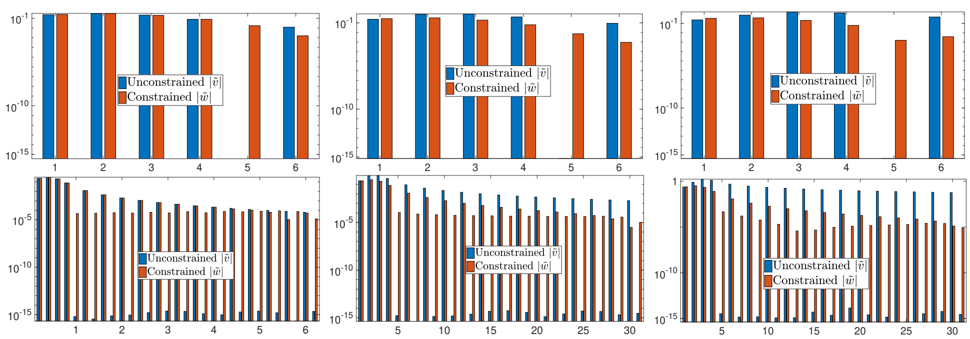

Fig. 9: Companion to Figure 7. Bar plot showing unconstrained projection coefficients magnitude $|\widetilde{v}_j|$ vs various constrained projection coefficients magnitude $|\widetilde{w}_j|$. Top: $N = 6$, bottom: $N = 31$. Left: $H = H^0$. Center: $H = H^1$. Right: $H = H^2$.

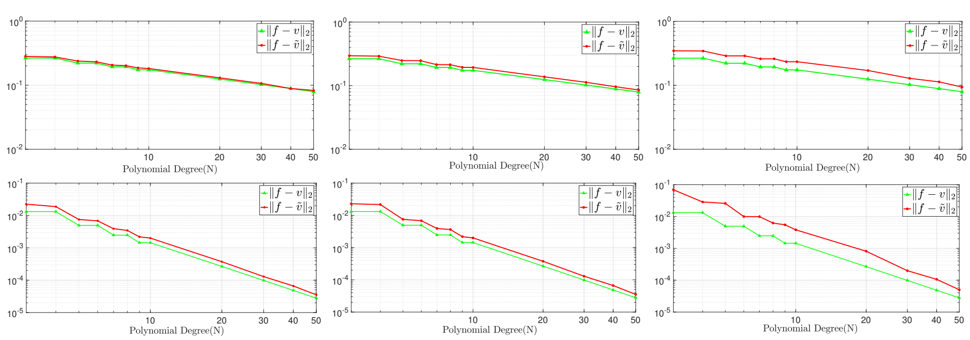

Fig. 10: $H = H^0$ convergence results for projection of test function $f = f_0$ (top row) and $f = f_2$ (bottom row). V is a space of polynomials of degree N. Left: Constraint $E = F_0$. Center: Constraint $E = F_0 \cap G_0$. Right: Constraint $E = F_0 \cap G_0 \cap F_1$.

Constraint set J_1 can be defined as the intersection of two conic constraints: for $x \in [-1, 0]$, we enforce $f(x) \geq -x$. For $x \in [0, 1]$ we enforce $f(x) \geq x$. Constraint set J_2 enforces $f(x) \geq -x$ for $x \in [-1, 0]$ as before, but now enforces $f(x) \leq x$ for $x \in [0, 1]$. Note that J_2 implicitly enforces $f(0) = 0$, but we do not explicitly require this in our algorithm. Since $x \in V$ when $N \geq 2$, we can handle these constraints with our setup.

We consider the test function $f(x) = |x|$; the optimization successfully terminates and results are shown in Figure 11.

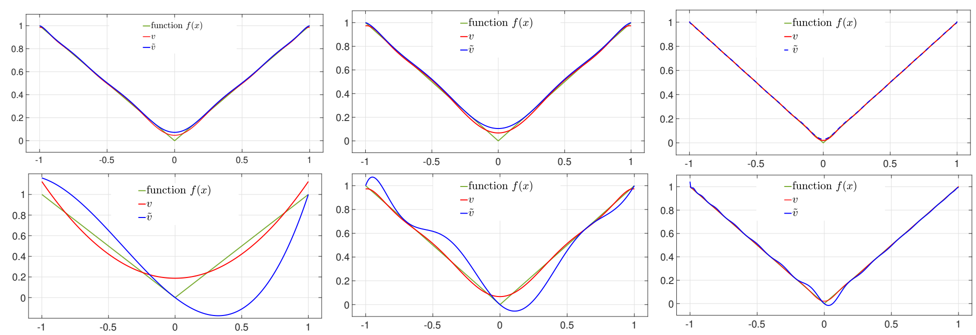

Fig. 11: Algorithm results from unusual constraints for $f(x) = |x|$. Top: Constraint set J_1. Bottom: constraint set J_2. Left: $N = 4$. Center: $N = 9$. Right: $N = 31$.

6. Conclusions. We have proposed a formalism for performing constrained function approximation. Restricting the class of possible constraints to those that are convex assures a unique solution to the constrained function approximation problem in Hilbert spaces. Typical constraints of interest such as positivity or monotonicity are specializations of our setup. We propose three iterative algorithms to compute solutions to the problem. Each algorithm requires minimization or level set detection on a weighted version of the current approximant, and thus can be expensive. In one dimension with polynomial approximation, our algorithms require only the ability to accurately compute roots of polynomials. We have demonstrated the flexibility, feasibility, and utility of our constrained approximation setup with many examples, including empirical investigation of convergence rates.

For higher dimensions, we require the ability to find the minimum of a non-polynomial multivariate function, and so our optimization problem becomes much more complex and expensive. Our difficulties in computing global minima correspond precisely to the known difficulty of globally solving the "lower-level" problem in semi-infinite programming methods, and our algorithms do not provide novel or constructive approaches to addressing this more general challenge in SIP algorithms. Therefore, identifying approaches to make our algorithm usable for multivariate approximation problems is the subject of ongoing research.

Acknowledgments. Vidhi Zala and Robert M. Kirby acknowledge support from the National Science Foundation under DMS-1521748 and the Army Research Office under ARO W911NF-15-1-0222 (Program Manager Dr. Mike Coyle. Akil Narayan was partially supported by NSF DMS-1848508.).

Appendix A. Algorithms for univariate polynomial subspaces.

We present procedures for solving the greedy and averaging optimization procedures in sections 4.1 and 4.2 under the assumption that V is a complete, univariate polynomial space. More formally, we make three specializing assumptions.

The first assumption is that H an L^2-type space. A typical setup in one dimension is that Ω is a interval in (and possibly equal to) $\mathbb{R}$, and a weighted L^2 space is defined by a probability density function ρ:

$$\langle u, v\rangle_{L^2_\rho} \coloneqq \int_\Omega u(x)v(x)\rho(x)\mathrm{d}x$$

The second specializing assumption in this section is that V is a complete polynomial space. For a finite $N \in \mathbb{N}$, the space V contains polynomials up to degree $N-1$. Then, $\{v_j\}_{j=1}^N$ can be chosen as the first N orthonormal polynomials under the weight ρ on Ω. It is classical knowledge that such a family of polynomials satisfies the three-term recurrence:

$$xv_n(x) = b_{n+1}v_{n+1}(x) + a_{n+1}v_n(x) + b_n v_{n-1}(x), \qquad n \geq 1,$$

with the starting conditions $v_0 \equiv 1$ and $v_{-1} \equiv 0$, where $a_n = a_n(\rho)$ and $b_n = b_n(\rho)$ are the recurrence coefficients [24].

The third specializing assumption is that we are in the setup of Example 2.1 where the constraints enforce positivity $v(x) \geq 0$ for every $x \in \Omega$. We will see that this assumption can be relaxed substantially; indeed we make this assumption here to only clarify some computations.

An important technique that we will need to exploit for this special setup is the ability to compute roots of polynomials from their expansion coefficients, i.e., if $v \in V$ has expansion coefficients $\{\widehat{v}_j\}_{j=1}^N$, then the $N-1$ (complex-valued) roots of v coincide with the spectrum of the $(N-1)\times(N-1)$ *confederate* matrix $\boldsymbol{T} = \boldsymbol{T}(v)$:

$$\text{(A.1)} \qquad \boldsymbol{T}(v) = \boldsymbol{J} - \frac{b_{N-1}}{\widehat{v}_N}\boldsymbol{e}_{N-1}\widetilde{\widehat{\boldsymbol{v}}}^T, \qquad \boldsymbol{J} = \begin{pmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & b_2 & a_3 & b_3 & \\ & & \ddots & \ddots & \\ & & & b_{N-2} & a_{N-1} \end{pmatrix}$$

where $\boldsymbol{e}_{N-1} \in \mathbb{R}^{N-1}$ is the cardinal unit vector in the $(N-1)$st direction and $\widetilde{\widehat{\boldsymbol{v}}}^T = (\widehat{v}_1, \ldots, \widehat{v}_{N-1})$. The matrix $\boldsymbol{J}$ is the Jacobi matrix and is independent of v. We use direct eigenvalue solvers to compute the spectrum of $\boldsymbol{T}(v) = v^{-1}(0)$. Note that there are backwards stable versions of the task of computing roots from the spectrum of related matrices [20]. An analogous approach that operates on expansion coefficients in a monomial basis uses the spectrum of the *companion* matrix. Note that our strategy is rather rudimentary compared to more sophisticated methods for computing roots of polynomials [6], e.g., one can compute polynomial roots on subintervals and perform refinement. However, this consideration is not the main innovation of our algorithm, and so we use the procedure above mainly for simplicity. We do perform a numerical stability check where we switch between companion and confederate matrices depending on which has smaller condition number. In all the examples we attempted for this manuscript, this check was sufficient to robustly and accurately compute roots of polynomials.

A.1. Greedy projections. With the setup of Example 2.1, the problem (4.5) requires us to compute

$$y^* = \operatorname*{argmin}_{y\in\Omega} \operatorname{sdist}\left(\widehat{\boldsymbol{v}}, H_1(y)\right) \overset{(4.4)}{=} \operatorname*{argmin}_{y\in\Omega} v(y)\lambda(y).$$

To minimize the last expression, we can compute the critical points, which are the roots of the derivative. Using (2.5), we have

$$\frac{\mathrm{d}}{\mathrm{d}y}[v(y)\lambda(y)] = \lambda^3(y)\left[v'(y)\sum_{j=1}^{N} v_j^2(y) - v(y)\sum_{j=1}^{N} v_j(y)v_j'(y)\right].$$

Note that λ^3 cannot vanish, so the critical points coincide with the roots of the bracketed expression above, which is a degree-$(3N-4)$ polynomial. Thus,

$$\frac{\frac{\mathrm{d}}{\mathrm{d}y}[v(y)\lambda(y)]}{\lambda^3(y)} = \sum_{j=1}^{3N-3} \widehat{g}_j v_j(y) =: g(y),$$

for some coefficients $\widehat{g}_j$. The computation $\{\widehat{v}_j\} \mapsto \{\widehat{g}_j\}$ can be accomplished using *only* the recurrence coefficients in $\mathcal{O}(N^2)$ time without resorting to, e.g., quadrature.

In summary, the global minimum in (4.5) can be computed by first computing the $\widehat{g}_j$ expansion coefficients defined above, and then by computing the spectrum of the $(3N-4)\times(3N-4)$ matrix $\boldsymbol{T}(g)$. To compute the global minimizer, we then need only evaluate the discrete minimum of $v(y)\lambda(y)$ over the eigenvalues located in Ω.

A.2. Averaged projections. The main task for the averaged projections procedure is to compute the integral in (4.8). In our specialized setup, this task reduces to computing

$$\frac{1}{|\omega_1^-|}\int_{\omega_1^-} \widehat{\boldsymbol{\ell}}_1(y)v(y)\lambda(y)\mathrm{d}y,$$

which is an N-component vector, where component j of this vector has the entry

$$\frac{1}{|\omega_1^-|}\int_{\omega_1^-} v_j(y)v(y)\lambda(y)\mathrm{d}y. \tag{A.2}$$

The first step is to identify the set ω_1^- defined in (4.7), which in this special case is equivalent to

$$\omega_1^- = \left\{y \in [-1,1] \;\middle|\; v(y) < 0\right\}.$$

Therefore, this set can be identified by examining the roots of v, which are the eigenvalues of $\boldsymbol{T}(v)$. Thus, we partition $[-1,1]$ into subintervals on which v is single-signed, after which determining the sign of v on an interval can be accomplished by evaluating v in this interval.

After ω_1^- is identified as a disjoint collection of subintervals of $[-1,1]$, we compute the components of the update (A.2) by employing an M-point Gaussian quadrature rule; since the integrand $v_j v\lambda$ is a smooth function on $[-1,1]$, this can be completed efficiently. We employ $M = N+1$ quadrature points for this same computation.

REFERENCES

[1] H. Anton and C. Rorres, *Elementary Linear Algebra, Binder Ready Version: Applications Version*, John Wiley & Sons, 2013.

[2] H. Bauschke and J. Borwein, *On Projection Algorithms for Solving Convex Feasibility Problems*, SIAM Review, 38 (1996), pp. 367–426, https://doi.org/10.1137/S0036144593251710, http://epubs.siam.org/doi/10.1137/S0036144593251710 (accessed 2018-02-15).

[3] R. Beatson, *Restricted Range Approximation by Splines and Variational Inequalities*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 372–380, https://doi.org/10.1137/0719023.

[4] R. K. Beatson, *The degree of monotone approximation.*, Pacific Journal of Mathematics, 74 (1978), pp. 5–14, https://projecteuclid.org/euclid.pjm/1102810431 (accessed 2018-10-30).

[5] M. Berzins, *Adaptive Polynomial Interpolation on Evenly Spaced Meshes*, SIAM Review, 49 (2007), pp. 604–627, https://doi.org/10.1137/050625667.

[6] J. P. Boyd, *Computing Zeros on a Real Interval through Chebyshev Expansion and Polynomial Rootfinding*, SIAM Journal on Numerical Analysis, 40 (2003), pp. 1666–1682.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization, With Corrections 2008*, Cambridge University Press, Cambridge, UK ; New York, 1 edition ed., Mar. 2004.

[8] S. Boyd and L. Vandenberghe, *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*, Cambridge University Press, Cambridge, UK ; New York, NY, 1 edition ed., Aug. 2018.

[9] L. Bregman, *The method of successive projection for finding a common point of convex sets*, Soviet Math Dokl., 6 (1965), pp. 688–692.

[10] M. Campos-Pinto, F. Charles, and B. Després, *Algorithms For Positive Polynomial Approximation*, SIAM Journal on Numerical Analysis, 57 (2019), pp. 148–172, https://doi.org/10.1137/17M1131891, https://epubs.siam.org/doi/abs/10.1137/17M1131891 (accessed 2019-11-06).

[11] W. Cheney and A. A. Goldstein, *Proximity Maps for Convex Sets*, Proceedings of the American Mathematical Society, 10 (1959), pp. 448–450, https://doi.org/10.2307/2032864, https://www.jstor.org/stable/2032864 (accessed 2019-11-04).

[12] F. Deutsch and H. Hundal, *The rate of convergence for the cyclic projections algorithm I: Angles between convex sets*, Journal of Approximation Theory, 142 (2006), pp. 36–55, https://doi.org/10.1016/j.jat.2006.02.005.

[13] F. R. Deutsch, *Best Approximation in Inner Product Spaces*, Springer Science & Business Media, Dec. 2012.

[14] R. A. DeVore, *Degree of Monotone Approximation*, International Series of Numerical Mathematics / Internationale Schriftenreihe zur Numerischen Mathematik / Série Internationale D'Analyse Numérique, Birkhäuser Basel, Basel, 1974, https://doi.org/10.1007/978-3-0348-5991-2_26, https://doi.org/10.1007/978-3-0348-5991-2_26 (accessed 2019-10-17).

[15] *Semi-Infinite Programming: Recent Advances*, Nonconvex Optimization and Its Applications, Springer US, 2001, https://doi.org/10.1007/978-1-4757-3403-4.

[16] L. G. Gubin, B. T. Polyak, and E. V. Raik, *The method of projections for finding the common point of convex sets*, USSR Computational Mathematics and Mathematical Physics, 7 (1967), pp. 1–24, https://doi.org/10.1016/0041-5553(67)90113-9.

[17] R. Hettich and K. O. Kortanek, *Semi-Infinite Programming: Theory, Methods, and Applications*, SIAM Review, 35 (1993), pp. 380–429, https://doi.org/10.1137/1035089, http://epubs.siam.org/doi/abs/10.1137/1035089 (accessed 2020-07-01). Publisher: Society for Industrial and Applied Mathematics.

[18] A. S. Lewis, D. R. Luke, and J. Malick, *Local Linear Convergence for Alternating and Averaged Nonconvex Projections*, Foundations of Computational Mathematics, 9 (2009), pp. 485–513, https://doi.org/10.1007/s10208-008-9036-y, https://doi.org/10.1007/s10208-008-9036-y (accessed 2019-09-28).

[19] J. Lewis, *Approximation with Convex Constraints*, SIAM Review, 15 (1973), pp. 193–217, https://doi.org/10.1137/1015006, https://epubs.siam.org/doi/abs/10.1137/1015006.

[20] Y. Nakatsukasa and V. Noferini, *On the stability of computing polynomial roots via confederate linearizations*, Mathematics of Computation, 85 (2016), pp. 2391–2425, https://doi.org/10.1090/mcom3049, https://www.ams.org/home/page/ (accessed 2018-08-28).

[21] R. Nochetto and L. Wahlbin, *Positivity preserving finite element approximation*, Mathematics of Computation, 71 (2002), pp. 1405–1419, https://doi.org/10.1090/S0025-5718-01-01369-2, https://www.ams.org/mcom/2002-71-240/S0025-5718-01-01369-2/ (accessed 2019-11-06).

[22] J. Rice, *Approximation with Convex Constraints*, Journal of the Society for Industrial and Applied Mathematics, 11 (1963), pp. 15–32, https://doi.org/10.1137/0111002, http://epubs.siam.org/doi/abs/10.1137/0111002.

[23] O. Stein, *How to solve a semi-infinite optimization problem*, European Journal of Operational Research, 223 (2012), pp. 312–320, https://doi.org/10.1016/j.ejor.2012.06.009.

[24] G. Szegö", *Orthogonal Polynomials*, American Mathematical Soc., 4th ed., 1975.

[25] J. Von Neumann, *Functional Operators (AM-22), Volume 2*, 1951, https://press.princeton.edu/titles/3136.html (accessed 2018-11-13).

[26] X. Zhang and C.-W. Shu, *Maximum-principle-satisfying and positivity-preserving high-order schemes for conservation laws: survey and new developments*, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, (2011), p. rspa20110153, https://doi.org/10.1098/rspa.2011.0153.