ON ENFORCING NON-NEGATIVITY IN POLYNOMIAL APPROXIMATIONS IN HIGH DIMENSIONS

YUAN CHEN*, DONGBIN XIU* AND XIANGXIONG ZHANG†

Abstract. Polynomial approximations of functions are widely used in scientific computing. In certain applications, it is often desired to require the polynomial approximation to be non-negative (resp. non-positive), or bounded within a given range, due to constraints posed by the underlying physical problems. Efficient numerical methods are thus needed to enforce such conditions. In this paper, we discuss effective numerical algorithms for polynomial approximation under non-negativity constraints. We first formulate the constrained optimization problem, its primal and dual forms, and then discuss efficient first-order convex optimization methods, with a particular focus on high dimensional problems. Numerical examples are provided, for up to 200 dimensions, to demonstrate the effectiveness and scalability of the methods.

Key words. polynomial approximation, positivity preserving, Fast Iterative Shrinkage Thresholding Algorithm

MSC codes. 42C05, 41A10, 65K10, 90C25

1. Introduction.

We are interested in finding a polynomial approximation to an unknown multivariate function $f(\mathbf{x})$, via its samples $f(\mathbf{x}_i)$, $i = 1, \ldots, K$, $\mathbf{x}_i \in \mathbb{R}^d$, $d \geq 1$. In many applications, the function is expected to obey a set of given constraints, usually in the form of inequalities. For examples, density or mass is supposed to be positive, or non-negative. Consequently, it is highly desirable to enforce the polynomial approximations to satisfy the same constraints. Violation of the constraints results in non-physical numerical results and often catastrophic break down of the underlying numerical model.

For polynomial approximations with constraints, most of the existing efforts can be classified into two kinds of approaches: construction-based approach and optimization-based approach. The first approach resorts to characterization of the sought-after polynomials with given structures, such as positivity. Examples of such methods include the positivity-preserving interpolation [7, 19, 8, 11, 13], with an extension to bound-preserving [9]. These methods usually require precise and complicated algebraic derivations. They usually work with relatively simple constraints and are difficult to apply in high dimensions. The second approach is to enforce the constraints by solving a constrained optimization problem, where the constraints are usually replaced by their approximations in the polynomial space. Convex optimization tools are leveraged to obtain the solution. In [1], Bernstein basis that forms a nonnegative partition of unity was used, where the positivity constraint on the whole polynomial becomes the positivity on the expansion coefficients. The resulting approximation is constrained uniformly. However, this approach is not capable of reproducing polynomials [34]. In [34], the feasible set formed by the constraints is characterized by a set of hyperplanes. A greedy algorithm is then employed to iteratively enforce the constraints represented by the hyperplanes.

Although we do not consider solving partial differential equations (PDEs) in this paper, it is worth mentioning the approaches for positivity-preserving numerical meth-

*E-mail addresses: {chen.11050, xiu.16}@osu.edu. Department of Mathematics, The Ohio State University, Columbus, OH 43210, USA. Funding: This work was partially supported by AFOSR FA9550-22-1-0011.

†E-mail address: zhan1966@purdue.edu. Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, Indiana 47907. X.Z. was supported by NSF DMS-2208515.

ods for PDEs. It is often crucial for numerical schemes for solving PDEs to satisfy certain constraints for the sake of stability, such as positivity of density and pressure in fluid dynamics equations. Bound-preserving and positivity-preserving numerical schemes include [36, 37] for conservation laws, [15, 35] for compressible Navier–Stokes equations, [33] for magnetohydrodynamics, diffusion equations [32, 20], transport equations [26, 5] and [21, 2] for more general PDEs, to name a few.

The focus of this paper is on the development of a numerical framework for finding constrained polynomial approximation of functions, which should be scalable for high dimensions. For simplicity we focus on the non-negative constraint, which can be extended to more general inequality constraints. We formulate the non-negative polynomial approximation problem as a convex minimization with the constraints enforced over a set of finite number chosen points. In high dimensions, the cardinality of the polynomial space can be exceedingly large, resulting in a large scale minimizaation problem. To circumvent the challenge, we employ first-order convex optimization methods, which use only gradient of the cost functions and thus scales well with the problem size. Moreover, the corresponding dual problem is formulated in a space whose dimension is the number of the constraint points. This is a user's choice and controllable in practice.

There is a vast amount of literature on first-order optimization methods. In this paper, we investigate a few widely known methods, including Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [4, 24], accelerated Primal-Dual Hybrid Gradient (PDHG) method [10, 27], Douglas–Rachford splitting [22], which is equivalent to Alternating Direction Method of Multipliers (ADMM) [16] and split Bregman method [18] with special parameters. Moreover, the FISTA algorithm can be accelerated [3] by using an adaptive restart technique [24, 25].

In this paper, we demonstrate that the FISTA with restart method applied to a proper dual problem is efficient to enforce non-negativity at finite given locations for a polynomial approximation. For simpler problems, it is possible to design nearly optimal parameters to accelerate convergence for methods like the Douglas-Rachford splitting [23]. For simplicity, we do not consider tuning parameters. We design a comprehensive set of numerical examples, from one dimension to hundreds of dimensions, to compare these popular first-order methods. The numerical cost is estimated and verified in experiments to demonstrate the scalability and efficiency in practical use, which indicates that the restarted FISTA on the dual problem is a simple and effective method without tuning parameters.

The rest of the paper is organized as follows. In Section 2, we introduce the setup of the problem. In Section 3, we provide a detailed description of our method. A comprehensive numerical study is presented in Section 4.

2. Problem Setup.

2.1. Least squares polynomial approximation. Consider the problem of approximating an unknown function $f : D \mapsto \mathbb{R}$, $D \subset \mathbb{R}^d$, $d \geq 1$. Let $\mathbf{x} = (x_1, ..., x_d)$ be the variable and $f \in L^2(D)$. Consider the subspace of polynomials of degree up to $n \geq 1$

$$\Pi_n^d := \text{span}\left\{\mathbf{x}^{\mathbf{k}} = x_1^{k_1} \cdots x_d^{k_d}, |\mathbf{k}| \leq n\right\}, \tag{2.1}$$

where $\mathbf{k} = (k_1, \ldots, k_d)$ is multi-index with $|\mathbf{k}| = k_1 + \cdots + k_d$. The dimension of this subspace is

$$N = \dim \Pi_n^d = \begin{pmatrix} n+d \\ d \end{pmatrix} = \frac{(n+d)!}{n!d!}. \tag{2.2}$$

Let $\{\psi_j(\mathbf{x}), j = 1, ..., N\}$ be an orthonormal basis of Π_n^d, then any $\tilde{f} \in \Pi_n^d$ can be expressed as

$$\tilde{f}(\mathbf{x}) = \sum_{|\mathbf{k}|=0}^{n} c_{\mathbf{k}} \psi_{\mathbf{k}}(\mathbf{x}) = \sum_{k=1}^{N} c_k \psi_k(\mathbf{x}), \tag{2.3}$$

where we have used a linear ordering to map the multi-index $\mathbf{k}$ to a single index k. By using vector notation

$$\mathbf{\Psi}(\mathbf{x}) = [\psi_1(\mathbf{x}), \ldots, \psi_N(\mathbf{x})]^T, \tag{2.4}$$

the expression can be written as

$$\widetilde{f}(\mathbf{x}) = \langle \mathbf{c}, \mathbf{\Psi}(\mathbf{x}) \rangle, \tag{2.5}$$

where $\mathbf{c} = [c_1, ..., c_N]^T$ and $\langle \cdot, \cdot \rangle$ is the dot product.

We then consider approximating the function f using its samples. Let $\mathbf{x}_1, \ldots, \mathbf{x}_K$, be a sequence of sample locations in the domain D, $\mathbf{f} := [f(\mathbf{x}_1), ..., f(\mathbf{x}_K)]^T$ be the sample function values, and

$$\mathbf{\Psi}_a = \begin{bmatrix} \mathbf{\Psi}^T(\mathbf{x}_1) \\ \cdots \\ \mathbf{\Psi}^T(\mathbf{x}_K) \end{bmatrix} \in \mathbb{R}^{K \times N}, \tag{2.6}$$

be the Vandermonde-like matrix, the standard least squares approximation problem can be solved via

$$\min_{\mathbf{c}} \|\mathbf{\Psi}_a \mathbf{c} - \mathbf{f}\|_2^2, \tag{2.7}$$

under the assumption that the problem is over-determined with $K > N$.

2.2. Constrained Polynomial Approximation. Next we consider a polynomial approximation under linear inequality constraints on the coefficients $\mathbf{c}$:

$$\min_{\mathbf{c}} \|\mathbf{\Psi}_a \mathbf{c} - \mathbf{f}\|_2^2, \qquad \text{subject to} \qquad \mathbf{Bc} \geq \mathbf{b}, \tag{2.8}$$

where $\mathbf{B} \in \mathbb{R}^{C \times N}$, $\mathbf{b} \in \mathbb{R}^C$, C is the number of the constraints, and the inequality is enforced component-wise. For a set Ω, the indicator function is defined as

$$\iota_\Omega(x) = \begin{cases} 0, & x \in \Omega, \\ +\infty, & x \notin \Omega. \end{cases} \tag{2.9}$$

We can rewrite problem (2.8) in the following equivalent form with a constant $\alpha > 0$:

$$\min_{\mathbf{c} \in \mathbb{R}^N} \quad \frac{1}{2} \alpha \|\mathbf{\Psi}_a \mathbf{c} - \mathbf{f}\|_2^2 + \iota_\Lambda(\mathbf{Bc} - \mathbf{b}), \tag{2.10}$$

where

$$\Lambda = \{\mathbf{a} \in \mathbb{R}^C : \mathbf{a} \geq \mathbf{0}\}.$$

The indicator function term is non-differentiable but convex. The matrix $\mathbf{\Psi}_a^T \mathbf{\Psi}_a$ is positive semidefinite thus the l^2 approximation term is also convex.

Enforcing non-negativity at some points can be recasted as the constraint $\mathbf{Bc} \geq \mathbf{b}$. Let $\mathbf{y}_1$,...,$\mathbf{y}_M \in D$ be a sequence of points of interest, and let $\mathbf{\Psi}_p \in \mathbb{R}^{M \times N}$ be the Vandermonde-like matrix at these points,

$$\mathbf{\Psi}_p = \begin{bmatrix} — & \mathbf{\Psi}^T(\mathbf{y}_1) & — \\ & \cdots & \\ — & \mathbf{\Psi}^T(\mathbf{y}_M) & — \end{bmatrix}. \tag{2.11}$$

Let $\mathbf{B} = \mathbf{\Psi}_p$, and $\mathbf{b} = \mathbf{0}$. Then the minimizer of (2.10) would be the coefficients of an approximation polynomial which is non-negative at points $\mathbf{y}_1$,...,$\mathbf{y}_M$. In this case, the number of constraints $C = M$.

3. Efficient Iterative Methods. We now discuss splitting algorithms for solving convex optimization (2.10). In particular, we focus on the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA), which is an efficient simple algorithm if applied to a proper problem set up.

3.1. The primal and dual problems. Recall that the primal problem (2.10) is given by

$$\min_{\mathbf{c} \in \mathbb{R}^N} \quad \frac{1}{2}\alpha \left\| \mathbf{\Psi}_a \mathbf{c} - \mathbf{f} \right\|_2^2 + \iota_\Lambda(\mathbf{Bc} - \mathbf{b}), \tag{P}$$

Then (P) can be written as

$$\min_{\mathbf{c} \in \mathbb{R}^N} g(\mathbf{c}) + h(\mathbf{Bc} - \mathbf{b}),$$

where the two functions $g : \mathbb{R}^N \mapsto \mathbb{R}$ and $h : \mathbb{R}^C \mapsto \mathbb{R}$ are

$$g(\mathbf{c}) = \frac{1}{2}\alpha \left\| \mathbf{\Psi}_a \mathbf{c} - \mathbf{f} \right\|_2^2, \qquad h(\mathbf{x}) = \iota_\Lambda(\mathbf{x}). \tag{3.1}$$

For any convex function h, its convex conjugate is defined as

$$h^\star(\mathbf{y}) := \sup_{\mathbf{x} \in \mathbb{R}^C} \left\{ \langle \mathbf{y}, \mathbf{x} \rangle - h(\mathbf{x}) \right\}.$$

The primal problem (P) is equivalent to the following primal-dual form (P-D) and dual form (D) (e.g., see [28]):

$$\min_{\mathbf{c} \in \mathbb{R}^N} \max_{\mathbf{u} \in \mathbb{R}^C} \quad \left[\langle \mathbf{Bc} - \mathbf{b}, \mathbf{u} \rangle - h^\star(\mathbf{u}) + g(\mathbf{c}) \right], \tag{P-D}$$

$$- \min_{\mathbf{u} \in \mathbb{R}^C} \quad g^\star(-\mathbf{B}^T \mathbf{u}) + \mathbf{b}^T \mathbf{u} + h^\star(\mathbf{u}), \tag{D}$$

where $\mathbf{u} \in \mathbb{R}^C$ is the dual variable. For the two functions in (3.1), the conjugate functions $g^\star : \mathbb{R}^N \mapsto \mathbb{R}$, $h^\star : \mathbb{R}^C \mapsto \mathbb{R}$ are given by

$$g^\star(\mathbf{c}) = \frac{1}{2\alpha} \left(\mathbf{z} + \mathbf{c} \right)^T \mathbf{K}^\dagger \left(\mathbf{z} + \mathbf{c} \right) - \frac{1}{2}\alpha \mathbf{f}^T \mathbf{f}, \tag{3.2}$$

$$h^\star(\mathbf{x}) = \iota_{\Lambda^\star}(\mathbf{x}), \quad \text{with } \Lambda^\star = \{\mathbf{x} : \mathbf{x} \leq \mathbf{0}\}, \tag{3.3}$$

where $\mathbf{z} = \alpha \mathbf{\Psi}_a^T \mathbf{f}$, $\mathbf{K} = \mathbf{\Psi}_a^T \mathbf{\Psi}_a$, and $\mathbf{K}^\dagger$ is the Moore-Penrose pseudo inverse $\mathbf{K}$. Let

$$G^*(\mathbf{u}) = g^\star(-\mathbf{B}^T\mathbf{u}) + \mathbf{b}^T\mathbf{u},$$

then the dual problem (**D**) can be written as:

$$-\min_{\mathbf{u}\in\mathbb{R}^C} \quad G^*(\mathbf{u}) + h^\star(\mathbf{u}), \tag{3.4}$$

where

$$G^*(\mathbf{u}) = \frac{1}{2\alpha}\left(\mathbf{z} - \mathbf{B}^T\mathbf{u}\right)^T \mathbf{K}^\dagger \left(\mathbf{z} - \mathbf{B}^T\mathbf{u}\right) - \frac{1}{2}\alpha \mathbf{f}^T\mathbf{f} + \mathbf{b}^T\mathbf{u}. \tag{3.5}$$

3.2. Closed convex proper functions. In this subsection, we explain why the indicator function of a set must be defined using $+\infty$ in (2.9). For a set $\Omega \subset \mathbb{R}^n$, consider a function

$$I_\Omega(\mathbf{x}) = \begin{cases} 0, & \mathbf{x} \in \Omega, \\ M, & \mathbf{x} \notin \Omega. \end{cases}$$

for a very large number M. Then $I_\Omega(\mathbf{x})$ is a well defined function on the whole space $\mathbb{R}^n$, and it may seem that $I_\Omega(\mathbf{x})$ can serve the same purpose numerically as the indicator function (2.9). However, a convex function well defined on the whole space $\mathbb{R}^n$ must be a continuous function [29, Corollary 10.1.1], thus the function $I_\Omega(\mathbf{x})$ cannot be convex on $\mathbb{R}^n$. To this end, one must consider a closed convex proper function, which will be defined as follows.

Notice that the indicator function defined in (2.9) should be regarded as an *extended* function

$$f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{\pm\infty\}.$$

For an extended function $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{\pm\infty\}$, its *epigraph* is defined as

$$\mathrm{epi} f = \{(\mathbf{x}, a) \in \mathbb{R}^n \times \mathbb{R} : f(\mathbf{x}) \le a\}.$$

An extended function is called *closed* if its epigraph is a closed set in $\mathbb{R}^{n+1}$. The *domain* of an extended function is denoted as

$$\mathrm{dom} f = \{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) \in \mathbb{R}\}.$$

The indicator function of a set Ω defined in (2.9) is a closed extended function if and only if Ω is a closed set, see [3].

A *convex* extended function is defined as an extended function satisfying:

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \le \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}), \quad \forall \mathbf{x}, \mathbf{y} \in \mathrm{dom} f,\ \forall \lambda \in (0,1).$$

An extended function is called *proper* if it never maps negative infinity. For instance, the indicator function (2.9) is a proper function. Thus if Ω is a closed convex set, then the indicator function (2.9) is a closed convex proper function, to which many results about convex functions well defined on the whole space $\mathbb{R}^n$ can be extended.

It can be proven that a closed extended function is also a lower semi-continuous function [3, Theorem 2.6]. Thus for a closed convex set Ω, the indicator function (2.9) is also a lower semi-continuous convex proper function.

3.3. First order splitting methods. Both the primal problem (**P**) and the dual problem in the form of (3.4) can be written as a general composite minimization

$$\min_{\mathbf{x}} \quad g(\mathbf{x}) + h(\mathbf{x}), \tag{3.6}$$

where the functions $g(\mathbf{x})$ and $h(\mathbf{x})$ are convex closed proper functions. For a convex closed proper function $g(\mathbf{x})$ such as the indicator function (2.9), its subdifferential $\partial g(\mathbf{x}_0)$ at a point $\mathbf{x}_0$ is defined as a set of slopes of subtangent lines:

$$\partial g(\mathbf{x}_0) = \{\mathbf{v} \in \mathbb{R}^n : g(\mathbf{x}) \geq g(\mathbf{x}_0) + \langle \mathbf{v}, \mathbf{x} - \mathbf{x}_0 \rangle, \quad \forall \mathbf{x} \in \mathrm{dom}(g)\}.$$

An element in the subdifferential set ∂g is called a subgradient. If $g(\mathbf{x})$ is differentiable at $\mathbf{x}_0$, then the subgradients coincide with the gradient, i.e., $\partial g(\mathbf{x}_0) = \{\nabla g(\mathbf{x}_0)\}$. For example, for $f(x) = |x|$, the subdifferential set is $\partial f(x) = \begin{cases} \{1\}, & x > 0 \\ \{-1\}, & x < 0 \\ [-1, 1], & x = 0 \end{cases}$.

Let ∂g and ∂h be their subdifferentials, I the identity operator. The proximal operators of these two functions, i.e., the resolvents of subdifferentials, are

$$\mathtt{prox}_g^\gamma(\mathbf{x}) = (I + \gamma \partial g)^{-1}(\mathbf{x}) = \mathrm{argmin}_{\mathbf{z}}\, \gamma g(\mathbf{z}) + \frac{1}{2}\|\mathbf{z} - \mathbf{x}\|_2^2, \quad \gamma > 0, \tag{3.7}$$

$$\mathtt{prox}_h^\gamma(\mathbf{x}) = (I + \gamma \partial h)^{-1}(\mathbf{x}) = \mathrm{argmin}_{\mathbf{z}}\, \gamma h(\mathbf{z}) + \frac{1}{2}\|\mathbf{z} - \mathbf{x}\|_2^2, \quad \gamma > 0. \tag{3.8}$$

For a convex closed proper function $g(\mathbf{x})$, $\gamma g(\mathbf{z}) + \frac{1}{2}\|\mathbf{z} - \mathbf{x}\|_2^2$ is a closed strongly convex proper function and it has a unique minimizer [3, Theorem 5.25], thus the proximal operator $\mathtt{prox}_g^\gamma(\mathbf{x})$ is a well defined operator.

Assume the proximal operators have explicit formulae or can be efficiently approximated. If $g(x)$ is differentiable but $h(x)$ is not differentiable, then the simplest method for (3.6) is the subgradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \gamma_k(\nabla g(\mathbf{x}_k) + \mathbf{v}_k), \quad \mathbf{v}_k \in \partial h(\mathbf{x}_k),$$

where γ_k is a step size and any subgradient $\mathbf{v}_k$ can be chosen. However, the subgradient method may converge very slowly. Instead, a method using the proximal operator $\mathtt{prox}_h^\gamma(\mathbf{x})$ is usually much faster. The simplest splitting method to use the proximal operator is the forward-backward splitting:

$$\mathbf{x}_{k+1} = (I + \gamma \partial h)^{-1}(I - \gamma \nabla g)(\mathbf{x}_k) = \mathtt{prox}_h^\gamma(\mathbf{x}_k - \gamma \nabla g(\mathbf{x}_k)). \tag{3.9}$$

The splitting method (3.9) is also referred to as the *proximal gradient method*, or *projected gradient method* when $\mathtt{prox}_h^\gamma$ is a projection operator. The *fast proximal gradient method* [24] is also called the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [4]. For (3.6), FISTA is given by

$$\begin{cases} \mathbf{x}_{k+1} = \mathtt{prox}_h^\gamma(\mathbf{y}_k - \gamma \nabla g(\mathbf{y}_k)), \\ t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}, \\ \mathbf{y}_{k+1} = \mathbf{x}_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{x}_{k+1} - \mathbf{x}_k), \end{cases} \tag{3.10}$$

where $t_0 = 1$, $\mathbf{x}_0 = \mathbf{y}_0$, and $\gamma > 0$ is a step size. The convergence rate of (3.9) can be proven $\mathcal{O}(k^{-1})$, and the convergence rate of FISTA (3.10) is $\mathcal{O}(k^{-2})$, when the

step size is taken as $\gamma = \frac{1}{L}$, assuming that ∇g is Lipschitz continuous with Lipschitz constant L.

When both the proximal operators ~~are available~~ have explicit formulae or can be efficiently approximated, we can also consider the splitting methods with two proximal operators. Define reflection operators as

$$\mathtt{R}_g^\gamma = 2\mathtt{prox}_g^\gamma - I, \quad \mathtt{R}_h^\gamma = 2\mathtt{prox}_h^\gamma - I,$$

then the Douglas-Rachford splitting [22, 14] can be given as

$$\begin{cases} \mathbf{y}_{k+1} = \lambda \frac{\mathtt{R}_g^\gamma \mathtt{R}_h^\gamma + \mathtt{I}}{2} \mathbf{y}_k + (1-\lambda)\mathbf{y}_k, \\ \mathbf{x}_{k+1} = \mathtt{prox}_h^\gamma(\mathbf{y}_{k+1}), \end{cases} \tag{3.11}$$

where $\lambda \in (0, 2]$ is a relaxation parameter. When $\lambda = 2$, (3.11) converges only if at least one of the two functions g, h is strongly convex. The Douglas-Rachford splitting (3.11) is equivalent to the popular alternating direction method of multipliers (ADMM) method [16] and the split Bregman method [18] on the equivalent Fenchel dual problem of (3.6) if using special step sizes, see [12] and the references therein. When using ADMM and the split Bregman on the primal problem (3.6), it is equivalent to using Douglas-Rachford splitting on the Fenchel dual problem of (3.6). Usually, there is no significant difference in numerical performance between using the same splitting method on the primal problem and the dual problem.

Method	Iteration Schemes
Projected Gradient for (3.4)	$\begin{cases} \mathbf{u}_k = \mathbf{u}_k - \eta \nabla G^*(\mathbf{u}_k), \\ \mathbf{u}_k = \mathtt{prox}_{h^*}^\eta(\mathbf{u}_k). \end{cases}$
Douglas-Rachford for (3.4)	$\begin{cases} \mathbf{p}_{k+1} = \frac{\mathtt{R}_{G^*}^\gamma \mathtt{R}_{h^*}^\gamma + \mathtt{I}}{2} \mathbf{p}_k, \\ \mathbf{u}_{k+1} = \mathtt{prox}_{h^*}^\gamma(\mathbf{p}_{k+1}). \end{cases}$
Fast PDHG for (**P-D**)	$\begin{cases} \mathbf{u}_{k+1} = \mathtt{prox}_{h^*}^{\tau_k}\left(\mathbf{u}_k + \tau_k(\mathbf{B}\bar{\mathbf{c}}_k - \mathbf{b})\right), \\ \mathbf{c}_{k+1} = \mathtt{prox}_g^{\eta_k}\left(\mathbf{c}_k - \eta_k \mathbf{B}^T \mathbf{u}_{k+1}\right), \\ \theta_k = 1/\sqrt{1+2\mu\eta_k}, \eta_{k+1} = \theta_k \eta_k, \tau_{k+1} = \tau_k/\theta_k, \\ \bar{\mathbf{c}}_{k+1} = \mathbf{c}_{k+1} + \theta_k\left(\mathbf{c}_{k+1} - \mathbf{c}_k\right). \end{cases}$

TABLE 1

Examples of applying popular first-order splitting algorithms for solving (**P**)*, for which* $\mathbf{c}$ *is the primal variable,* $\mathbf{u}$ *is the dual variable,* $\mathbf{p}$ *is an auxiliary variable in Douglas-Rachford splitting and* $\mu > 0$ *is a parameter for accelerated PDHG method.*

Another popular splitting method for using two proximal operators is the accelerated Primal-Dual Hybrid Gradient (PDHG) method [10, 27], for solving the primal-dual form (**P-D**).

See [3, 31] for a comprehensive introduction of these first-order algorithms. In Table 1, we list several popular first-order algorithms that are used in our numerical tests to solve the optimization problem (**D**) (or in the form of (3.4)) and (**P-D**).

3.4. FISTA with restart on the dual. It is however inefficient to directly apply (3.10) to the problem (2.10). The main technical difficulty is the computation cost of the proximal operator for $\iota_\Lambda(\mathbf{Bc}-\mathbf{b})$ as a function of $\mathbf{c}$. The proximal operator of an indicator function is the projection operator to the domain it indicates. The domain $\{\mathbf{c} : \mathbf{Bc} - \mathbf{b} \geq 0\}$ is usually a high-dimensional hyper polygon defined by the linear inequalities. Although methods such as Dykstra's algorithm [6] can be used, the additional computational cost is undesired. We refer readers to [34] for a projection-based method for structure-preserving polynomial approximations. To circumvent the difficulty, we take advantage of the linear constraints and consider the equivalent dual problem (3.4). This allows us to compute the proximal operator of a simple function with a closed-form formula for each step.

Remark 3.1. Here we remark that the primal form (2.10) is in the space of $\mathbb{R}^N$, while the dual form is in $\mathbb{R}^C$. In practice, especially in high dimensions, we expect $N \gg C$. Thus the dual problem could significantly reduce the computational cost. For example, for a non-negative polynomial approximation, when d is large, the cardinality of polynomial space N can be exceedingly large even when polynomial order n is very small. However, in practice, usually only a small set of discrete samples is needed to enforce the positivity, e.g., positivity is needed only at certain locations of interest, thus $C \ll N$.

Since the function $G^*(\mathbf{u})$ in (3.4) is quadratic with respect to $\mathbf{u}$, the derivatives and proximal operators can be written explicitly as:

$$\nabla G^*(\mathbf{u}) = \frac{1}{\alpha}\left(\mathbf{B}\mathbf{K}^\dagger\mathbf{B}^T\mathbf{u} - \mathbf{B}\mathbf{K}^\dagger\mathbf{z}\right) + \mathbf{b}, \tag{3.12}$$

$$\mathtt{prox}^{\gamma}_{G^*}(\mathbf{u}) = \left(I + \frac{\gamma}{\alpha}\mathbf{B}\mathbf{K}^\dagger\mathbf{B}^T\right)^{-1}\left[\mathbf{u} + \gamma\left(\frac{1}{\alpha}\mathbf{B}\mathbf{K}^\dagger\mathbf{z} - \mathbf{b}\right)\right]. \tag{3.13}$$

Since $h^\star(\mathbf{u})$ in (3.4) is an indicator function on the negative half-space of $\mathbb{R}^C$. Its proximal operator is simply the cut-off operator:

$$\mathtt{prox}^{\tau}_{h^\star}(\mathbf{u}) = \left(\min\{u_i, 0\}\right)_{i=1}^{C}. \tag{3.14}$$

Now the FISTA method can be efficiently implemented on the dual problem (3.4). In practice, FISTA method can be further accelerated by various restarting strategies [25, 3], which is accomplished by reiterating the sequence t_k from the starting point t_0 after some iterations. The simplest restarting scheme is to repeat the standard FISTA algorithm (3.10) with a fixed frequency.

Here, we consider the following two criteria in [25]:

$$g(\mathbf{x}_k) > g(\mathbf{x}_{k-1}), \tag{A}$$

$$\left[\nabla g\left(\mathbf{y}_{k-1}\right)\right]^T\left(\mathbf{x}_k - \mathbf{x}_{k-1}\right) > 0. \tag{B}$$

The FISTA with adaptive restart, hereafter referred to as r-FISTA, usually achieves much faster convergence than the standard FISTA [3]. In our numerical tests, the r-FISTA applied to the dual problem is superior to all the methods listed in Table 1 using the same step size.

Then our main algorithm is to apply FISTA with adaptive restart (r-FISTA) on the dual problem (**D**) which results in an optimization in $\mathbb{R}^C$. After obtaining the minimizer $\mathbf{u}^\star$, we recover the primal minimizer $\mathbf{c}^\star$ by the primal-dual relation:

$$\mathbf{K}\mathbf{c}^\star = \frac{1}{\alpha}(\mathbf{z} - \mathbf{B}^T\mathbf{u}^\star). \tag{3.15}$$

Algorithm 3.1 Solve (D) by FISTA with Adaptive Restart

Input: Matrix $\mathbf{\Psi}_a$, $\mathbf{B}$; Vectors $\mathbf{f}$, $\mathbf{b}$; Constants α, η (step-size). Functions $G^*(\mathbf{u})$, $\nabla G^*(\mathbf{u})$; Stopping criteria S. Use condition (A) for restart.

Output: The basis coefficients $\mathbf{c}$.

Compute $\mathbf{K} = \mathbf{\Psi}_a^T \mathbf{\Psi}_a$, $\mathbf{K}^\dagger$, $\mathbf{z} = \alpha \mathbf{\Psi}_a^T \mathbf{f}$;
Initialize $\mathbf{u}_0 = \mathbf{u}_1 = \mathbf{p}_1$, $t_1 = 1$, $k = 1$;
while (S = **FALSE**) **do**
  $\mathbf{u}_{k+1} = \min(\mathbf{p}_k - \eta \nabla G^*(\mathbf{p}_k), \mathbf{0})$;
  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$;
  $\mathbf{p}_{k+1} = \mathbf{u}_k + \frac{t_k - 1}{t_{k+1}}(\mathbf{u}_{k+1} - \mathbf{u}_k)$;
  if $G^*(\mathbf{u}_{k+1}) > G^*(\mathbf{u}_k)$ **then**
    $t_{k+1} = 1$;
  end if
  $k = k + 1$;
end while
$\mathbf{c} = \mathbf{K}^\dagger(\mathbf{z} - \mathbf{B}^T \mathbf{u}_k)/\alpha$.

We summarize the above algorithm in Algorithm 3.1.

We briefly discuss the computational cost of the proposed method. As a preparation step, the matrix $\mathbf{B}\mathbf{K}^\dagger\mathbf{B}^T$ of size $C \times C$ and two vectors $\mathbf{B}(\mathbf{K}^\dagger)^T\mathbf{z}$, $\mathbf{b}$ of size $C \times 1$ need to be computed and stored. Within each iteration of Algorithm 3.1, it costs $\mathcal{O}(C^2)$ flops. As to the convergence rate, we mention some classical results of of FISTA for convex optimization in the next subsection. In particular, for the problem (3.4), the convergence rate of first-order methods is related to the ratio between the Lipschitz constant of G^* and the strong convexity parameter of the cost function. Since G^* is quadratic, such a ratio is naturally related to the condition number of the matrix $\mathbf{B}\mathbf{K}^\dagger\mathbf{B}^T$. It is also interesting to explore its performance dependency on the parameters, such as the number of constraints C, number of approximation points K, polynomial order n, and most importantly, the dimension d. However, it is difficult to conduct such analysis, as the properties the Vandermonde-like matrix $\mathbf{\Psi}_a$ depend critically on the geometric distribution of the samples $\mathbf{x}_1$,...,$\mathbf{x}_K$. Consequently, we rely on numerical testing in Section 4. Through numerical tests in both low and high dimensions, we discover that the number of iteration steps required for numerical convergence satisfies $k \sim \mathcal{O}(C)$. Under this condition, the proposed algorithm 3.1 has computational complexity $\mathcal{O}(C^3)$ and requires $\mathcal{O}(C^2)$ storage of real numbers.

The cost of computing matrices $\mathbf{\Psi}_a$, $\mathbf{K}$, $\mathbf{K}^\dagger$ grows dramatically as dimension d increases. This is due to the cardinality of the polynomial space $N \approx d^n/n!$ for large d. However, these computations need to be done only once and can be efficiently implemented in parallel, e.g., on modern Graphics Processing Units (GPUs). In our numerical tests in Section 4, the number of iterations needed for convergence does not increase when dimension d increases. When N dramatically grows, the matrix $\mathbf{B}$ of the size of $C \times N$ becomes larger, but the size of $\mathbf{B}\mathbf{K}^\dagger\mathbf{B}^T$ does not grow much when we use a mild number of constraints C.

3.5. The convergence rate of FISTA methods. The standard FISTA method has the following provable $\mathcal{O}(1/k^2)$ convergence rate when applied to problem (3.4). By [3, Theorem 10.34], we have

THEOREM 3.2. *Let* $\{\mathbf{u}_k\}_{k\geq 0}$ *be the sequence generated by FISTA* (3.10) *for solving*

problem (3.4). *Then for any* $k \geq 1$,

$$F\left(\mathbf{u}^k\right) - F_{\text{opt}} \leq \frac{2L\left\|\mathbf{u}^0 - \mathbf{u}^*\right\|^2}{(k+1)^2},$$

where L *is the maximum eigenvalue of the matrix* $\mathbf{B}\mathbf{K}^\dagger\mathbf{B}^T$, $F(\mathbf{u}) := G^*(\mathbf{u}) + h^\star(\mathbf{u})$ *is the objective function defined in* (3.4) *with optimal solution* $\mathbf{u}^*$ *and optimum* F_{opt}.

FISTA with a fixed restarting frequency is described in Algorithm 3.2.

Algorithm 3.2 Solve (D) by FISTA with a fixed restarting frequency N

Input: Matrix $\mathbf{\Psi}_a$, $\mathbf{B}$; Vectors $\mathbf{f}$, $\mathbf{b}$; Constants α, η (step-size). Functions $G^*(\mathbf{u})$, $\nabla G^*(\mathbf{u})$; a fixed restarting frequency N.
Output: The basis coefficients $\mathbf{c}$.

Compute $\mathbf{K} = \mathbf{\Psi}_a^T\mathbf{\Psi}_a$, $\mathbf{K}^\dagger$, $\mathbf{z} = \alpha\mathbf{\Psi}_a^T\mathbf{f}$;
Initialize $\mathbf{u}_0 = \mathbf{u}_1 = \mathbf{p}_1$, $t_1 = 1$, $k = 1$;
while ($S =$ **FALSE**) **do**
  $\mathbf{u}_{k+1} = \min(\mathbf{p}_k - \eta\nabla G^*(\mathbf{p}_k), \mathbf{0})$;
  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$;
  $\mathbf{p}_{k+1} = \mathbf{u}_k + \frac{t_k - 1}{t_{k+1}}(\mathbf{u}_{k+1} - \mathbf{u}_k)$;
  if $k \mod N = 0$ **then**
    $t_{k+1} = 1$;
    $\mathbf{p}_{k+1} = \mathbf{u}_{k+1}$;
  end if
  $k = k + 1$;
end while
$\mathbf{c} = \mathbf{K}^\dagger(\mathbf{z} - \mathbf{B}^T\mathbf{u}_k)/\alpha$.

With a stronger assumption on the matrix $\mathbf{B}\mathbf{K}^\dagger\mathbf{B}^T$, FISTA with a fixed restarting frequency can be proven to converge linearly when applied to (3.4). By [3, Theorem 10.41], we have

THEOREM 3.3. *Suppose the matrix* $\mathbf{B}\mathbf{K}^\dagger\mathbf{B}^T$ *is positive definite, and denote the minimum and maximum eigenvalue of* $\mathbf{B}\mathbf{K}^\dagger\mathbf{B}^T$ *by* μ *and* L, *condition number by* $\kappa := L/\mu$. *Let* $\{\mathbf{u}_k\}_{k\geq 0}$ *be the sequence generated by the restarted FISTA method employed with a fixed restarting frequency* $N = \lceil\sqrt{8\kappa} - 1\rceil$, *then for any* $k \geq 0$, *the following convergence result holds*

$$F\left(\mathbf{u}^k\right) - F_{\text{opt}} \leq \frac{L\left\|\mathbf{u}^0 - \mathbf{u}^*\right\|^2}{2}\left(\frac{1}{2}\right)^k,$$

where $F(\mathbf{u}) := G^*(\mathbf{u}) + h^\star(\mathbf{u})$ *is the objective function defined in* (3.4) *with optimal solution* $\mathbf{u}^*$ *and optimum* F_{opt}.

To implement this fixed-frequency restarting FISTA method, one has to choose the period that requires prior knowledge of condition number κ, which may not be available due to either lack of information or ill-conditioning of the system. The adaptive restart [24, 25] mitigates this issue by reiterating the sequence t_k when certain criterion is met.

Remark 3.4. When condition number κ is known, one can improve standard FISTA by choosing an optimal stepsize instead of restarting. It is achieved by replacing $\frac{t_k-1}{t_{k+1}}$ by constant $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ in (3.10). This method, referred to as v-FISTA, can also achieve a provable linear convergence rate $\mathcal{O}((1-\frac{1}{\sqrt{\kappa}})^k)$ given a strongly-convex objective function [3]. However, v-FISTA behaves worse than r-FISTA in our experiment due to the ill-conditioning system, which will be shown in numerical examples.

4. Numerical Examples. In this section, we present several numerical examples to demonstrate the effectiveness of our proposed method. In all of our numerical tests, we use Legendre polynomials for the basis. The constants in Algorithm 3.1 are set to be: $\alpha = 100$, $\eta = \alpha/|\sigma_{\texttt{max}}(\mathbf{B}\mathbf{K}^\dagger\mathbf{B}^T)|$, where $\sigma_{\texttt{max}}$ is the maximum eigenvalue. In our test, we use the following stopping criteria (i.e. S in Algorithm 3.1):

1. Convergence of primal variable: $\|\mathbf{c}_k - \mathbf{c}_{k-1}\|_2 \leq 1 \times 10^{-14}$. Note that this is at the machine accuracy level.
2. Satisfaction of constraints: $\mathbf{B}\mathbf{c}_k \geq \mathbf{b}$.

The dimensions of our examples include low dimensions $d = 1$, $d = 2$, intermediate dimension $d = 10$, and high dimensions $d = 100$, $d = 200$.

We consider the following functions as testing. In one dimension ($d = 1$), we consider the following 3 functions:

$$\text{(4.1)}\quad \textbf{Runge function}: f_1(x) = \frac{101}{100}\left(\frac{1}{1+100x^2} - \frac{1}{101}\right);$$

$$\text{(4.2)}\quad \textbf{Truncated sine function}: f_2(x) = \left[\sin(\frac{\pi(x+1)}{2}) - \sin(0.6\pi)\right]\mathbb{1}_{\{|x|<0.2\}};$$

$$\text{(4.3)}\quad \textbf{Simple indicator function}: f_3(x) = \mathbb{1}_{\{x>0\}},$$

where $\mathbb{1}_\Omega$ is the indicator function on set Ω,

$$\mathbb{1}_\Omega(x) = \begin{cases} 1, & x \in \Omega \\ 0, & x \notin \Omega \end{cases}.$$

In multi-dimension ($d > 1$), we test the following 3 functions ([17]) that have been widely used for multi-dimensional function approximation:

$$\text{(4.4)}\quad \textbf{Gaussian peak function}: f_4(\mathbf{x}) = \exp\left(-\sum_{i=1}^{d}\sigma_i^2\left(\frac{x_i+1}{2} - \omega_i\right)^2\right);$$

$$\text{(4.5)}\quad \textbf{Continuous peak function}: f_5(\mathbf{x}) = \exp\left(-\sum_{i=1}^{d}\sigma_i\left|\frac{x_i+1}{2} - \omega_i\right|\right);$$

$$\text{(4.6)}\quad \textbf{Corner peak function}: f_6(\mathbf{x}) = \left(1 + \sum_{i=1}^{d}\sigma_i\frac{(x_i+1)}{2}\right)^{-(d+1)}.$$

The parameters σ_i and ω_i, $i = 1, ..., d$ are designed to control the behavior of functions and specified in each example.

4.1. One-dimensional Examples. For one-dimensional cases, the domain of approximation is set to be $D = [-1, 1]$. The samples for approximation $\mathbf{x}_i$ are chosen to be Chebyshev nodes with the number of samples $K = 50$. The behavior of our proposed method is very similar in all the examples, so we only present a subset of the test results.

4.1.1. Runge function. We first consider the scaled Runge function $f_1(x)$ [30], which is positive. When the polynomial degree is large, its standard polynomial approximation produces oscillations around the edge, resulting in undesirable negative values. We take equidistant grid points, on which we enforce the positivity constraints, i.e. $\mathbf{y}_i = -1 + \frac{2(i-1)}{M-1}$, $i = 1, ..., M$. Following the notation in Section 2.2, the matrix $\mathbf{B} = \mathbf{\Psi}_p$ is generated by the samples $\mathbf{y}_i$, $i = 1, ..., M$ and number of constraints $C = M$. For finding a positive approximation, we take $\mathbf{b} = \mathbf{0} + \epsilon$, with $\epsilon = 10^{-5}$. The results for polynomial orders $n = 10, 20$ with $M = 201$ are in Figure 1. Compared with L^2 projection (red dot line), our approximation (blue dashed line) could almost preserve positivity on the domain. We test the convergence rate for approximation error $\|f - \tilde{f}\|_2$ with respect to polynomial order in Figure 2. It is observed that both unconstrained estimation and constrained estimation converge at an exponential rate. The constrained error is slightly larger than the unconstrained error when the polynomial order is small, while this difference is eliminated as the polynomial order gets larger. The positivity constraints limit the optimality of approximation. As the polynomial degree increases, L^2 estimation suffers less from the positivity issue and becomes close to the constrained estimation. This conclusion is also reflected in the percentage of negativity points out of the 201 constraint points, presented on the right of Figure 2.

We also conduct tests for f_1 with different settings of nonnegativity-enforcing points. We test 3 sets of points: equidistant, Chebyshev, and uniformly distributed random points with $M = 30$. The approximated polynomials are shown in Figure 3. It can be observed that the difference in the approximations is marginal. Though these approximated polynomials are guaranteed to be non-negative on the enforced points, it is still impossible to obtain non-negativity on all $x \in D$. This is also illustrated in Figure 3. In the figure, some violations can be seen in the area marked by dotted lines, which are zoomed in for clear presentation.

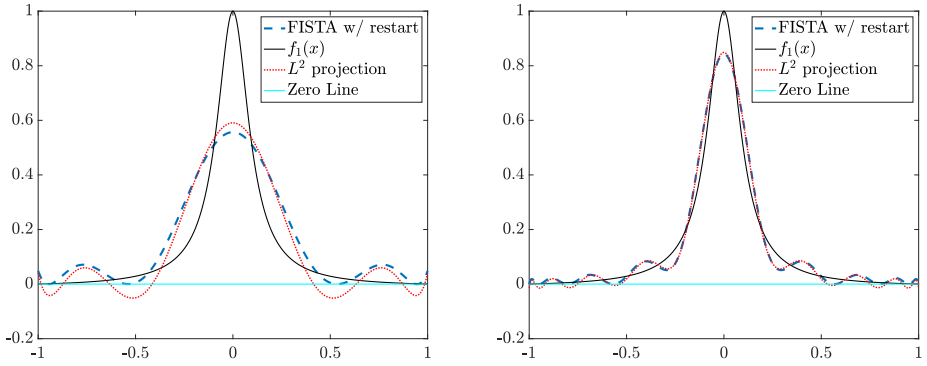

FIG. 1. *The approximated polynomials for* $f_1(x)$ (4.1) *with positivity constraints for* $n = 10$ *(Left),* $n = 20$ *(Right). The positive approximation is found via solving* (**P**) *by restarted FISTA on* (3.4).

4.1.2. Truncated sine function. The truncated sine function $f_2(x)$ (4.2) is a non-smooth, non-negative function in a bounded domain. We follow the same technical setting as $f_1(x)$ and seek a positive approximation. The results are shown in Figure 4 for polynomial orders $n = 5$ with $M = 101$ and $n = 20$ with $M = 201$.

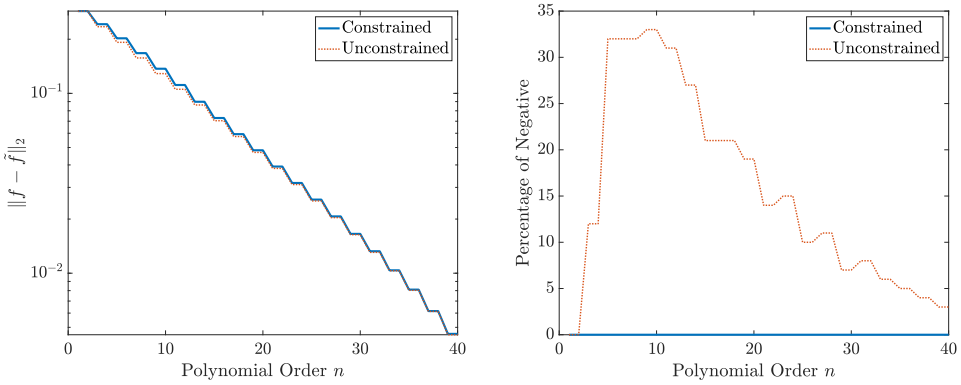

FIG. 2. *Comparasion for unconstrained approximation (L^2 projection) and non-negativity constrained approximation for $f_1(x)$ (4.1). The convergence with respect to polynomial order (left) and percentage of negative points in* 201 *non-negativity enforced sample points (right).*

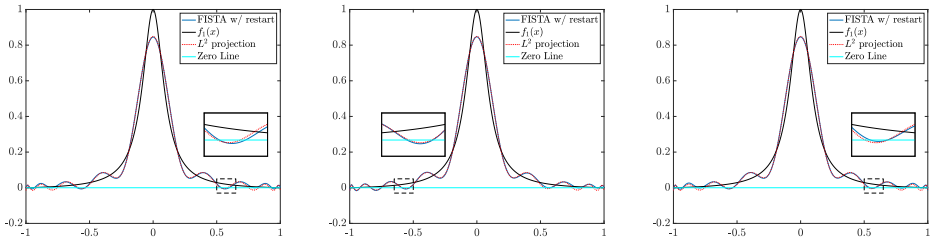

FIG. 3. *The approximated polynomials for $f_1(x)$ (4.1) with positivity constraints for $n = 20$ with $M = 31$ equispaced (left), Chebshev (middle) and random (right) constraint points. The areas enclosed by the dotted line are zoomed in for detailed presentation.*

We conduct further examination of the proposed method using this example. As mentioned in Section 3, FISTA with adaptive restart is not the only choice, so we show the comparison of convergence of several algorithms. The algorithms we compared include accelerated Primal-Dual Hybrid Gradient (PDHG) method [10, 27] for primal-dual problem (**P-D**); projected gradient method, FISTA without restart and Douglas–Rachford splitting [22] for dual problem (**D**). The schemes used in our experiment are shown in Table 1. The convergence of these methods for the case $n = 20$ is presented in Figure 5. Asymptotic linear convergence is observed for all the methods with different rates except accelerated PDHG. Our proposed method, FISTA with restart converges to the round-off error in around 600 iterations with the highest asymptotic rate. The effectiveness of adaptive restart could be observed through comparison to FISTA without restart. The v-FISTA method behaves even worse than FISTA without restart since the system $\mathbf{B}\mathbf{K}^{\dagger}\mathbf{B}^T$ is ill-conditioned as mentioned in Remark 3.4, with condition number $\kappa \sim \mathcal{O}(10^{17})$. We emphasize that in this test the parameter (step size η) is not tuned for Douglas-Rachford splitting, which could be much faster with tuned parameters.

We are also interested in the number of iterations needed for convergence, and the influence of the following parameters on the performance:

- Number of constraints C;

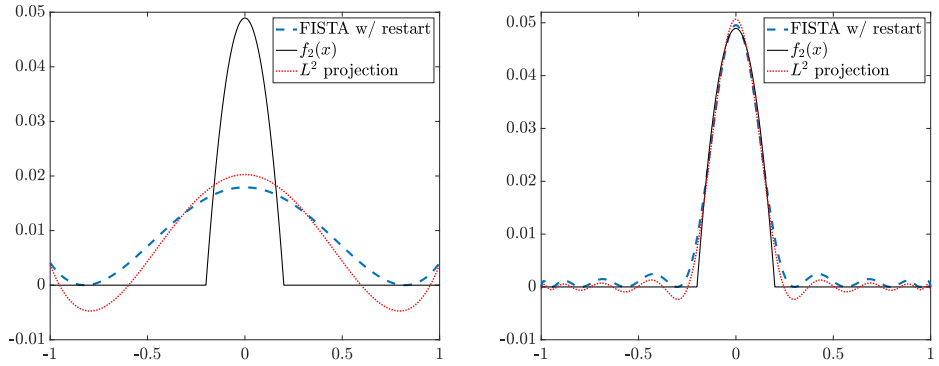

FIG. 4. *The approximated polynomials for $f_2(x)$ (4.2) with positivity constraints for $n = 5$ (Left), $n = 20$ (Right). The positive approximation is found via solving (**P**) by restarted FISTA on (3.4).*

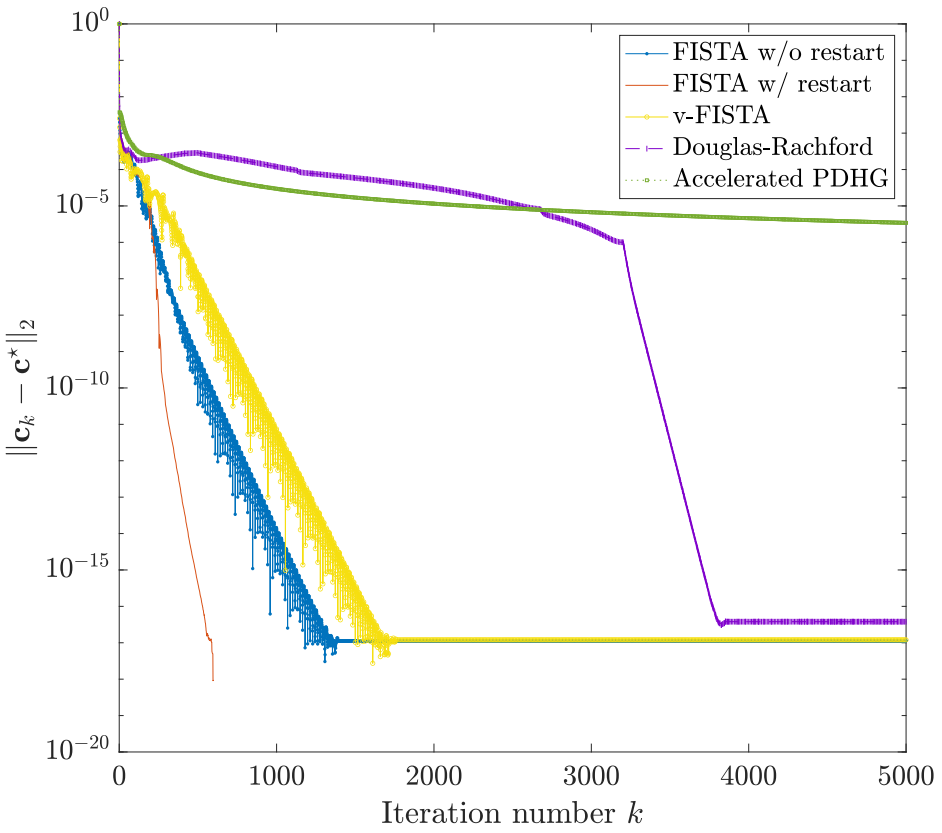

FIG. 5. *The comparison of convergence curve of several methods for approximating $f_2(x)$ (4.2) with positivity constraints when $n = 20$. The reference minimizer $\mathbf{c}^\star$ is obtained numerically via restarted FISTA with $5,000$ steps. For simplicity, Douglas-Rachford splitting uses the same step size as the FISTA method. We emphasize that Douglas-Rachford splitting could be much faster if tuning parameters.*

- Number of samples used for approximation K;
- Polynomial order n;
- Dimension d.

In this one-dimensional example, we design tests for the first three parameters. The tests for dimensions will be presented in the next subsection and Figure 17.

We record the number of iterations needed for restarted FISTA to converge for each parameter. The results are shown in Figure 6. For C, we keep $n = 5$, $K = 50$ and test different values of C from 10 to 1000. As C increases, numerically the number of iterations needed scales like $\mathcal{O}(C)$. For the other two parameters K and n, there is no significant growth of iteration numbers as these parameters are enlarged. These tests show that the parameter C affects the performance the most. This can be partially explained from a close look of (3.4), where C affects the dimension of the optimization problem while K and n only affect the condition.

Then another question is how would C affect the effectiveness of the positive approximation. In other words, the proposed method only strictly enforces the positivity at C chosen points. As C increases, we expect the minimizer to (**P**) will give a polynomial approximation producing fewer points where negative values emerge. We check the approximation polynomial values at 10^4 equidistant points on $[-1, 1]$, and compute the percentage of negative point values. For example, assume these test samples are denoted by $\mathbf{z}_i$, $i = 1, ..., L$, then we compute $|\{\mathbf{z}_i, \widetilde{f}(\mathbf{z}_i) \geq 0\}|/L$ for different C from 10 to 1000. The result is shown in the upper right of Figure 6. We observed a rapid descent of this percentage. The number of negative points concentrates at 0 for $C > 97$ which indicates the polynomial is positive mostly on the whole domain.

4.1.3. A step function. We consider a step function. The polynomial approximation of this function suffers from oscillations which result in overshoots and undershoots. So we consider a bound-preserving approximation using the proposed method. To achieve it, $M = 251$ equidistant grid points are used for bound-preserving constraints. We set $\mathbf{B} = \begin{bmatrix} \mathbf{\Psi}_p \\ -\mathbf{\Psi}_p \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} \mathbf{0} + \epsilon \\ -\mathbf{1} + \epsilon \end{bmatrix}$. This enforces the condition $0 < \widetilde{f}(\mathbf{y}_i) < 1$ for each $\mathbf{y}_i$, $i = 1, ..., M = 251$. The results for $n = 5$ and $n = 30$ are shown in Figure 7. It is observed, our approximation could almost preserve boundedness.

4.2. Two-dimensional Examples. In this section, we consider approximating two-dimensional functions. The domain is set to be $D = [-1, 1] \times [-1, 1]$. We use 31×31 uniform rectangular grids for the approximation points $\mathbf{x}_i$, thus $K = 961$.

4.2.1. Gaussian peak function. We set $\sigma_1 = \sigma_2 = 10$ and $\omega_1 = \omega_2 = 0.5$ in the formula (4.4). A smooth peak could be observed in the contour plot, see Figure 8. Standard L^2 Approximations around this peak will lead to negative oscillations. To construct positivity preserving polynomial approximation, we randomly sample $3,000$ points from uniform distribution and enforce the positive conditions on these points. The results with $n = 20$ are shown in Figure 8. In these contour plots, the color blue and red are used to indicate positive and negative values, respectively. A few contour lines are also shown in the graph using black color. It is clear to see that negative values occur widely around the peak of standard L^2 approximations. The approximation of our method shows a significant improvement.

4.2.2. Continuous peak function. With the same condition as $f_4(\mathbf{x})$, we tested the performance of $f_5(\mathbf{x})$. The parameters are similarly set to be $\sigma_1 = \sigma_2 = 10$ and $\omega_1 = \omega_2 = 0.5$. This function has a non-smooth peak around the origin. The

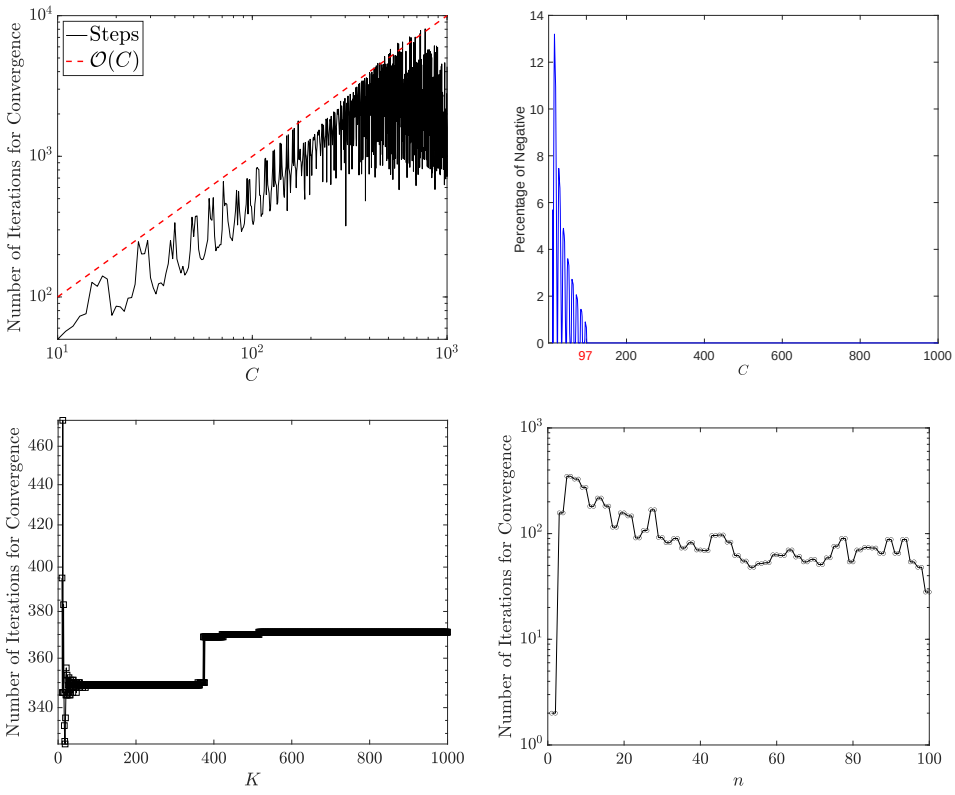

FIG. 6. *The required iteration number for restarted FISTA to converge, with respect to the number of constraints C (upper left), the number of approximation samples K (lower left), and polynomial order n (lower right) for finding a positive approximation to the function $f_2(x)$ (4.2). The percentage of negative sample points in 10^4 independent randomly tested samples with respect to C is shown in upper right.*

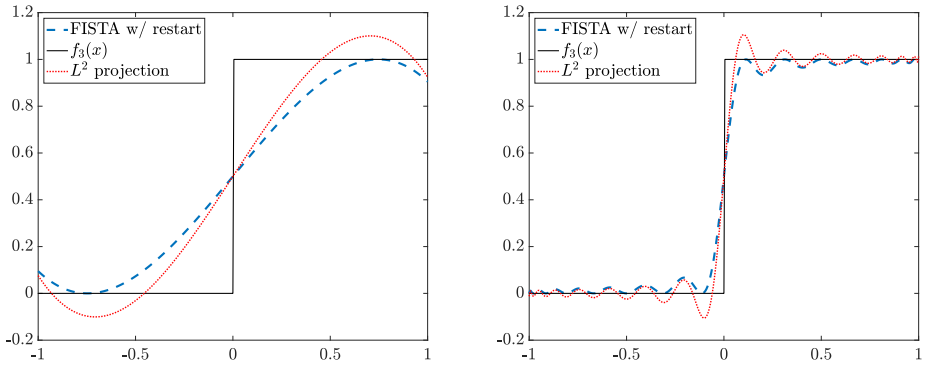

FIG. 7. *The approximated polynomial for $f_3(x)$ (4.3) with boundedness constraints for $n = 5$ (Left), $n = 30$ (Right). The bounded approximation is found via solving* (**P**) *by restarted FISTA on* (3.4)

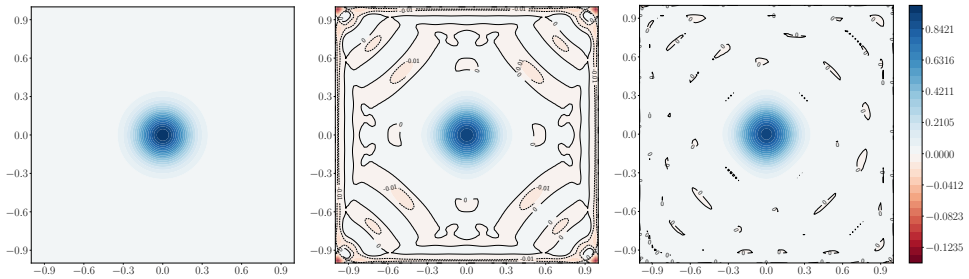

FIG. 8. *The contour plots of $f_4(\mathbf{x})$ (4.4) (left), L^2 projection when polynomial order $n = 20$ (Middle) and positive approximation via solving (**P**) by restarted FISTA on (3.4) (Right). In these graphs, positive values are marked in blue color, negative values are marked in red. A few contour lines are drawn using black solid (zero value) and dashed (a few negative values) lines.*

contour plots of the function $f_5(\mathbf{x})$, L^2 approximation, and the approximation using our methods are shown in Figure 9. We observed negative values for standard L^2 projection around the four edges of the domain, especially on the four corners. This is significantly improved with our method.

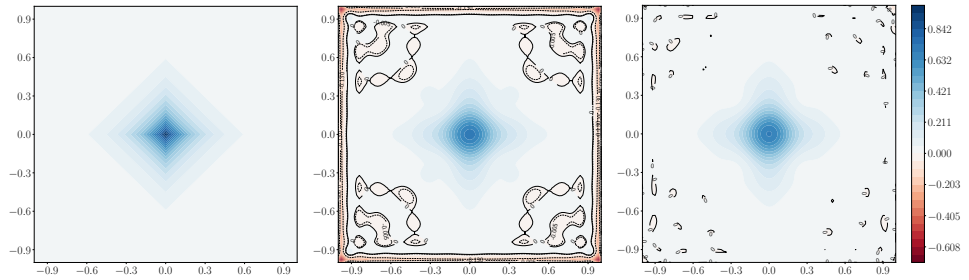

FIG. 9. *The contour plot of $f_5(\mathbf{x})$ (4.5) (left), L^2 projection when polynomial order $n = 20$ (Middle) and positive approximation via solving (**P**) by restarted FISTA on (3.4) (Right). In these graphs, positive values are marked in blue color, negative values are marked in red. A few contour lines are drawn using black solid (zero value) and dashed (a few negative values) lines.*

4.2.3. Corner peak function. We now consider the example $f_6(\mathbf{x})$ with parameters $\sigma_1 = \sigma_2 = 20$. This function is called "corner peak" because its value is concentrated in the lower-left corner of the domain, see the left of Figure 10. With standard L^2 approximation, it exhibits a large area of negative values along the diagonal of the domain D (Middle of Figure 10). This can be improved significantly by the proposed method, with positivity enforcement on a set of $3,000$ randomly sampled points from $\mathcal{U}(D)$. The result is on the right of Figure 10.

4.3. High dimensional Examples. We present results in dimensions $d \geq 3$, including the immediate dimension $d = 10$ and high dimensions $d = 100$, $d = 200$. We only focus on the Gaussian function $f_4(\mathbf{x})$ on domain $D = [-1, 1]^d$. We will examine the results with several metrics. One is the approximation error in L^2 norm. That is,

$$\|f - \widetilde{f}\|_{L^2(D)} \approx \sqrt{\frac{1}{L}\sum_{i=1}^{L}(f(\mathbf{z}_i) - \widetilde{f}(\mathbf{z}_i))^2}, \tag{4.7}$$

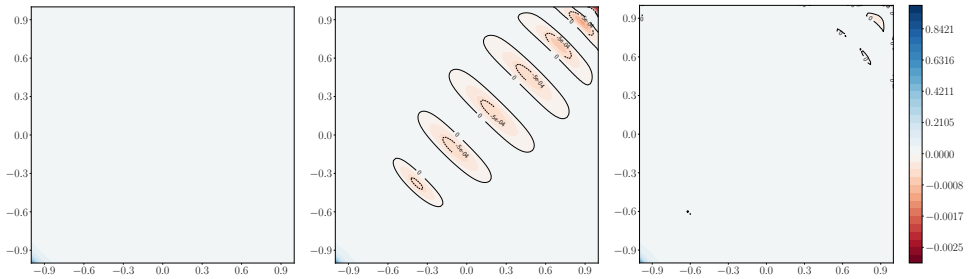

FIG. 10. *The contour plots of* $f_6(\mathbf{x})$ (4.6) *(left),* L^2 *projection when polynomial order* $n = 20$ *(Middle) and positive approximation via solving* (**P**) *by restarted FISTA on* (3.4) *(Right). In these graphs, positive values are marked in blue color, negative values are marked in red. A few contour lines are drawn using black solid (zero value) and dashed (a few negative values) lines.*

where $\mathbf{z}_i$, $i = 1, \ldots, L$, are sampled uniformly in the domain. These points are different from the ones used for function approximation. Another metric is the percentage of negative points, which has been defined and used in one-dimensional examples.

4.3.1. Dimension 10. We consider the case $d = 10$. The parameters of $f_4(\mathbf{x})$ are set as $\sigma_i = 10$, $i = 1, 2, \ldots, 10$. To approximate this function, we use $K = 2,000$ random generated samples points. The orders of the polynomial approximations are $n = 3, 4, 5, 6, 7, 8$, with the corresponding cardinality of the polynomial spaces $N = 286,\ 1,001,\ 3,003,\ 8,008,\ 19,448$ and $43,758$, respectively. The points to enforce positivity are randomly sampled with sizes from $C = 20$ to $12,000$. To evaluate the performance, we sample another $L = 5,000$ points to compute the approximated L^2 error and percentage of negative points. The results are shown in Figure 11. It can

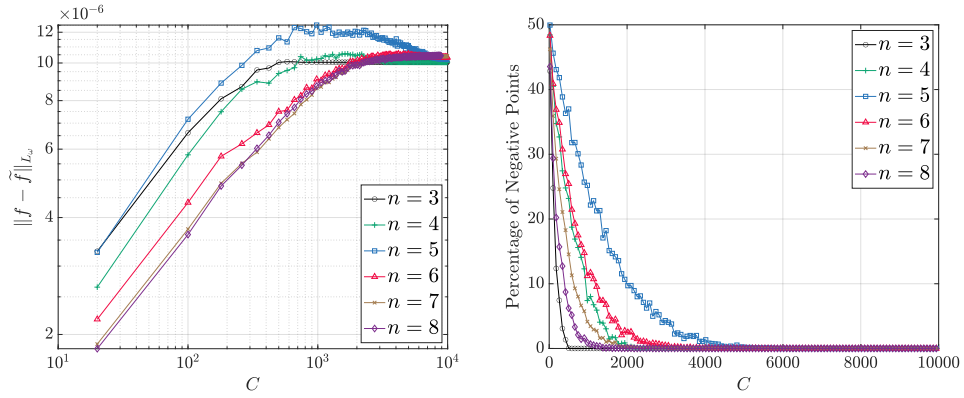

FIG. 11. *Comparasion of the estimated approximation error* (4.7) *(left) and percentage of negative sample points (right) with respect to the number of constraints* C *for* 10*-dimensional* $f_4(\mathbf{x})$ (4.4) *using restarted FISTA at polynomial orders* $n = 3, 4, 5, 6, 7, 8$. *These quantities are computed based on* $5,000$ *independent randomly tested samples.*

be seen that the percentage of the negative points converges to 0 as the number of positivity enforcing points C increases. The approximation errors do not vary too much with respect to C, staying within the range of $2 \times 10^{-6} \sim 1.2 \times 10^{-5}$. Its magnitude first grows as C increases, and then converges as C becomes larger. For large C, we do not observe a significant difference among the approximation errors at

the chosen polynomial orders.

In Figure 12, we show comparison of the proposed method with other splitting algorithms, for the case of $n = 3$ and $C = 1,000$. We observe superior convergence of the proposed method. We are also interested in the number of iterations needed

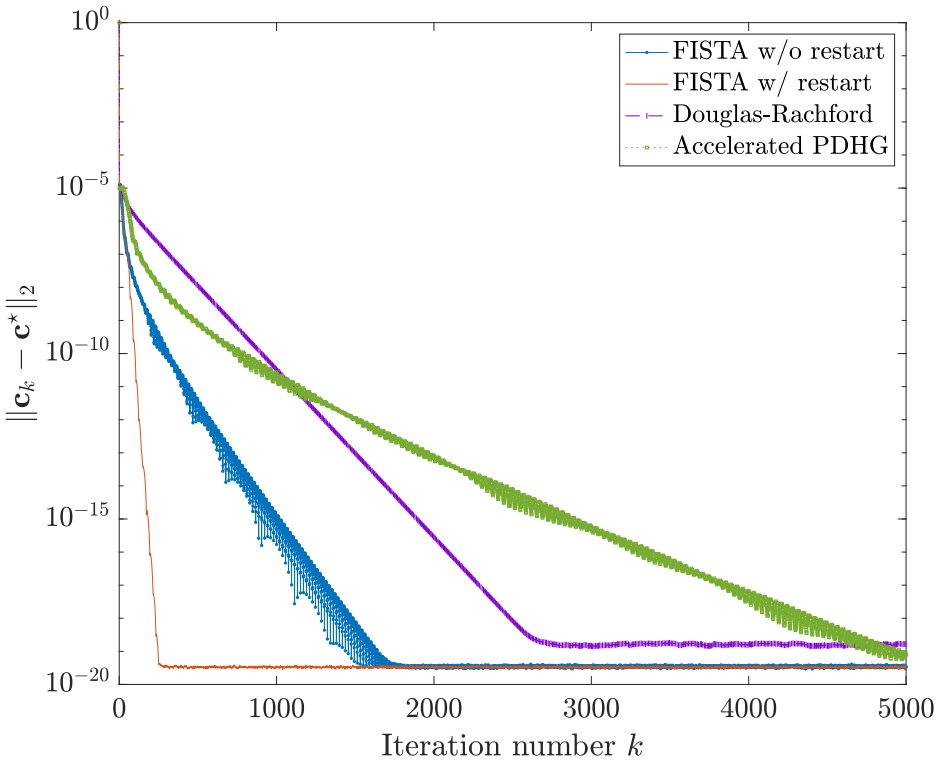

FIG. 12. *The comparison of convergence curve of several methods for approximating* 10 *dimensional* $f_4(\mathbf{x})$ (4.4) *with positivity constraints when* $n = 3$. *The reference minimizer* $\mathbf{c}^\star$ *is obtained numerically via restarted FISTA with* 5000 *steps. For simplicity, Douglas-Rachford splitting uses the same step size as the FISTA method. We emphasize that Douglas-Rachford splitting could be much faster if tuning parameters.*

for the proposed FISTA with restart, shown in Figure 13. An approximately linear growth w.r.t. C is observed, similar to the one-dimensional case. It is observed that polynomials with larger orders usually need fewer steps to converge because the condition of matrix $\mathbf{B}$ is better.

4.3.2. Dimensions of 100 and 200. In this section, we provide the approximation results for $f_6(\mathbf{x})$ at polynomial order $n = 2$ in dimensions $d = 100$ and $d = 200$, where the cardinality of the polynomial spaces is $N = 5,151$ and $N = 20,301$, respectively. We use $K = 3,000$ random sample points for the approximation, another $5,000$ independent random samples to evaluate the results. The results are in Figures 14, 15, and 16. It is observed that only a few hundreds iteration steps are needed for convergence, for the number of positivity constraint points $C \sim 10^3$ in these high-dimensional examples.

4.3.3. Effect of Dimensionality. Finally, we examine the impact of dimensionality on the convergence. We conduct tests for $f_6(\mathbf{x})$ with $2,000$ random approx-

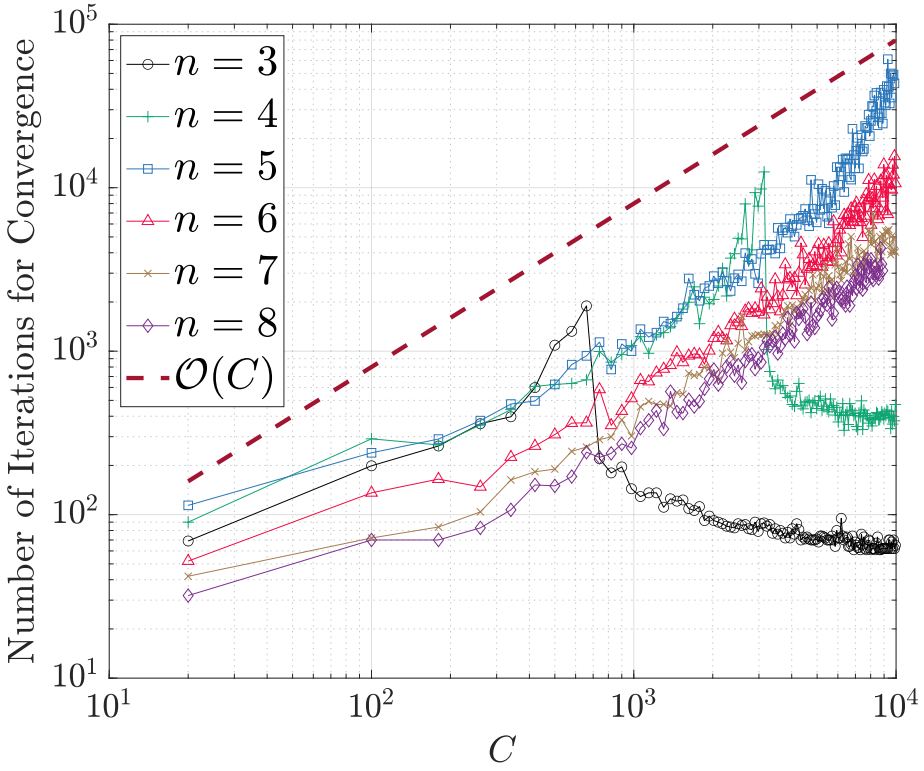

FIG. 13. *The number of iterations needed for restarted FISTA to converge for* 10*-dimensional* $f_4(\mathbf{x})$ (4.4) *with respect to number of constraints* C *given polynomial order* $n = 3, 4, 5, 6, 7, 8$.

imation points and 2,000 random constraint points, for dimensions from 2 to 280. The number of iterations needed for restarted FISTA to converge are shown in Figure 17. We observe that, at least for this test, the number of iterations needed for the proposed FISTA with restart of high dimensions ($d \geq 100$) is significantly smaller than that of lower dimensions. This is explained by the fact that the condition of matrix $\mathbf{B}\mathbf{K}^{\dagger}\mathbf{B}^T$ is better when d gets larger, with C remaining unchanged. Moreover, the iteration number is not sensitive to the dimensionality beyond $d = 100$.

5. Conclusion. In this paper, we have proposed a convex optimization based computational framework for approximating a function in high dimensions by polynomials with non-negative and bound-preserving constraints. In particular, the restarted FISTA method applied on a proper dual problem can be easily implemented, and scales well with the problem size. Numerical tests have verified the effectiveness of the method for problems in a few hundred dimensions. Future work consists of exploring how to design optimal parameters scuh as the step size for Douglas-Rachford splitting to achieve faster convergence and lower cost than the restarted FISTA method.

REFERENCES

[1] L. ALLEN AND R. C. KIRBY, *Bounds-constrained polynomial approximation using the Bernstein basis*, Numer. Math., 152 (2022), pp. 101–126, https://doi.org/10.1007/s00211-022-01311-1.

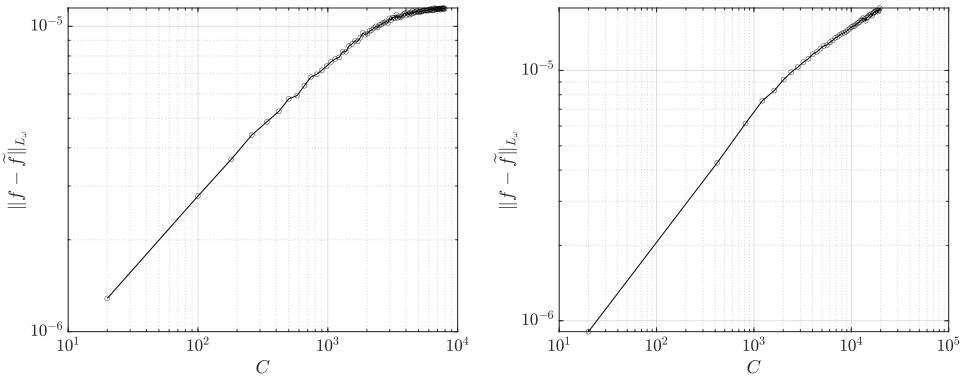

FIG. 14. *The estimated approximation error* (4.7) *for* 100*-dimensional (left) and* 200*-dimensional (right)* $f_4(\mathbf{x})$ (4.4) *with respect to the number of constraints* C *using restarted FISTA when polynomial orders* $n = 2$*. The errors are computed based on* 5,000 *independent randomly tested samples.*

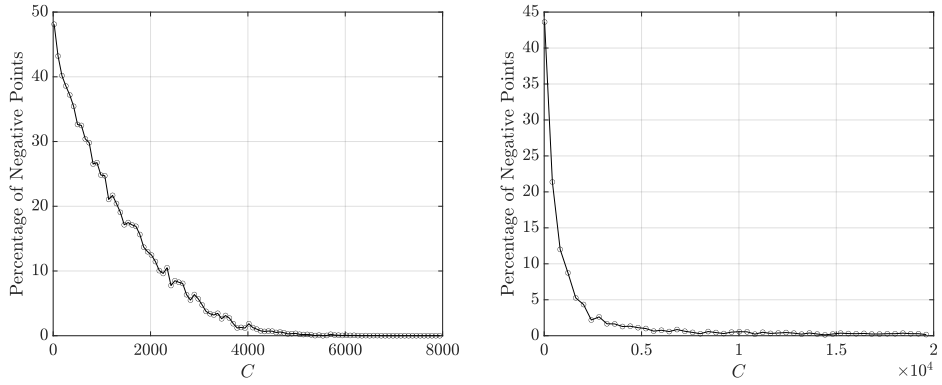

FIG. 15. *The percentage of negative sample test points of restarted FISTA method (based on* 5,000 *independent randomly tested samples) for* 100*-dimensional (left) and* 200*-dimensional (right)* $f_4(\mathbf{x})$ (4.4) *with respect to the number of constraints* C *when polynomial orders* $n = 2$*.*

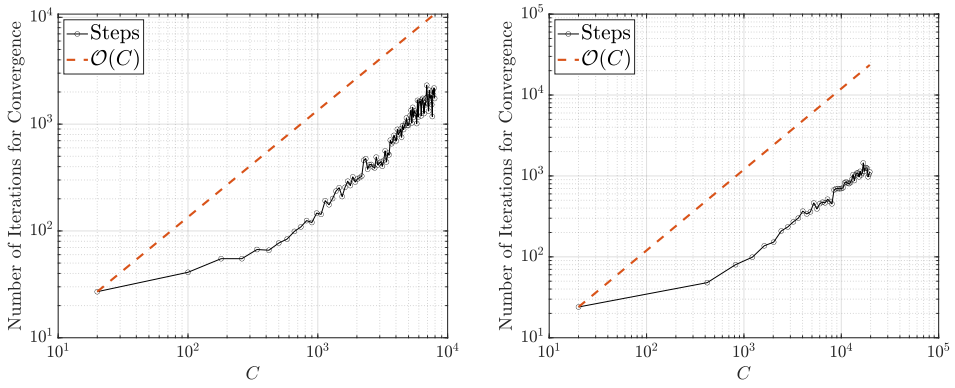

FIG. 16. *The number of iterations needed for restarted FISTA to converge for* 100*-dimensional (left) and* 200*-dimensional (right)* $f_4(\mathbf{x})$ (4.4) *with respect to the number of constraints* C *when polynomial orders* $n = 2$*.*

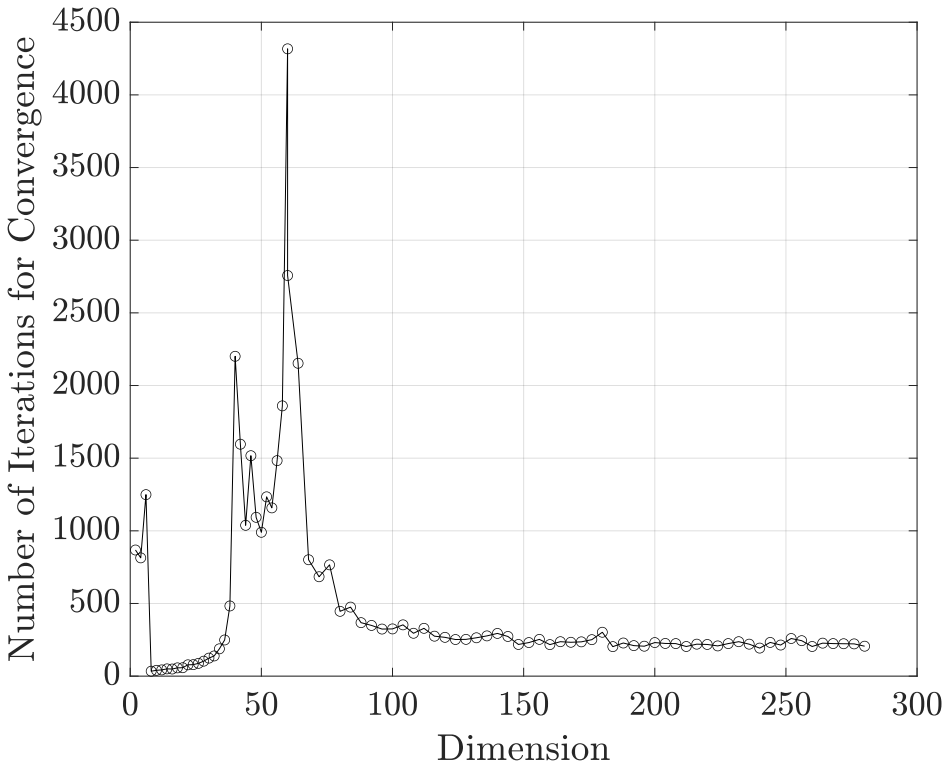

FIG. 17. *The number of iterations needed for restarted FISTA to converge for* $f_4(\mathbf{x})$ (4.4) *with respect to the number of dimensions using polynomial orders* $n = 2$*, number of approximation points* $N = 2,000$ *and number of constraints* $C = 2,000$.

[2] G. R. Barrenechea, E. H. Georgoulis, T. Pryer, and A. Veeser, *A nodally bound-preserving finite element method*, IMA Journal of Numerical Analysis, 44 (2024), pp. 2198–2219.
[3] A. Beck, *First-order methods in optimization*, SIAM, 2017.
[4] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202, https://doi.org/10.1137/080716542.
[5] P. Bochev, D. Ridzal, M. D'Elia, M. Perego, and K. Peterson, *Optimization-based, property-preserving finite element methods for scalar advection equations and their connection to algebraic flux correction*, Computer Methods in Applied Mechanics and Engineering, 367 (2020), p. 112982.
[6] J. P. Boyle and R. L. Dykstra, *A method for finding projections onto the intersection of convex sets in Hilbert spaces*, in Advances in Order Restricted Statistical Inference: Proceedings of the Symposium on Order Restricted Statistical Inference held in Iowa City, Iowa, September 11–13, 1985, Springer, 1986, pp. 28–47.
[7] S. Butt and K. Brodlie, *Preserving positivity using piecewise cubic interpolation*, Computers & Graphics, 17 (1993), pp. 55–64, https://doi.org/https://doi.org/10.1016/0097-8493(93)90051-A.
[8] M. Campos-Pinto, F. Charles, and B. Després, *Algorithms for positive polynomial approximation*, SIAM J. Numer. Anal., 57 (2019), pp. 148–172, https://doi.org/10.1137/17M1131891.
[9] M. Campos Pinto, F. Charles, B. Després, and M. Herda, *A projection algorithm on the set of polynomials with two bounds*, Numer. Algorithms, 85 (2020), pp. 1475–1498, https://doi.org/10.1007/s11075-019-00872-x.
[10] A. Chambolle and T. Pock, *A first-order primal-dual algorithm for convex problems with applications to imaging*, J. Math. Imaging Vision, 40 (2011), pp. 120–145, https://doi.org/

10.1007/s10851-010-0251-1.
[11] A. De Rossi and E. Perracchione, *Positive constrained approximation via RBF-based partition of unity method*, J. Comput. Appl. Math., 319 (2017), pp. 338–351, https://doi.org/10.1016/j.cam.2017.01.024.
[12] L. Demanet and X. Zhang, *Eventual linear convergence of the Douglas-Rachford iteration for basis pursuit*, Math. Comp., 85 (2016), pp. 209–238, https://doi.org/10.1090/mcom/2965.
[13] B. Després and M. Herda, *Computation of sum of squares polynomials from data points*, SIAM J. Numer. Anal., 58 (2020), pp. 1719–1743, https://doi.org/10.1137/19M1273955.
[14] J. Eckstein and D. P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Programming, 55 (1992), pp. 293–318, https://doi.org/10.1007/BF01581204.
[15] C. Fan, X. Zhang, and J. Qiu, *Positivity-preserving high order finite difference WENO schemes for compressible Navier-Stokes equations*, J. Comput. Phys., 467 (2022), pp. Paper No. 111446, 21, https://doi.org/10.1016/j.jcp.2022.111446.
[16] D. Gabay, *Chapter ix applications of the method of multipliers to variational inequalities*, in Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems, M. Fortin and R. Glowinski, eds., vol. 15 of Studies in Mathematics and Its Applications, Elsevier, 1983, pp. 299–331, https://doi.org/https://doi.org/10.1016/S0168-2024(08)70034-1.
[17] A. Genz, *Testing multidimensional integration routines*, in Proc. of international conference on Tools, methods and languages for scientific and engineering computation, 1984, pp. 81–94.
[18] T. Goldstein and S. Osher, *The split Bregman method for L1-regularized problems*, SIAM J. Img. Sci., 2 (2009), pp. 323–343, https://doi.org/10.1137/080725891, http://dx.doi.org/10.1137/080725891.
[19] M. Z. Hussain and M. Sarfraz, *Positivity-preserving interpolation of positive data by rational cubics*, J. Comput. Appl. Math., 218 (2008), pp. 446–458, https://doi.org/10.1016/j.cam.2007.05.023.
[20] L. Ju, X. Li, Z. Qiao, and J. Yang, *Maximum bound principle preserving integrating factor Runge-Kutta methods for semilinear parabolic equations*, J. Comput. Phys., 439 (2021), pp. Paper No. 110405, 18, https://doi.org/10.1016/j.jcp.2021.110405.
[21] R. C. Kirby and D. Shapero, *High-order bounds-satisfying approximation of partial differential equations via finite element variational inequalities*, Numerische Mathematik, (2024), pp. 1–21.
[22] P.-L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM J. Numer. Anal., 16 (1979), pp. 964–979, https://doi.org/10.1137/0716071.
[23] C. Liu, B. Riviere, J. Shen, and X. Zhang, *A simple and efficient convex optimization based bound-preserving high order accurate limiter for Cahn-Hilliard-Navier-Stokes system*, 2023, https://arxiv.org/abs/2307.09726.
[24] Y. Nesterov, *Gradient methods for minimizing composite functions*, Math. Program., 140 (2013), pp. 125–161, https://doi.org/10.1007/s10107-012-0629-5.
[25] B. O'Donoghue and E. Candès, *Adaptive restart for accelerated gradient schemes*, Found. Comput. Math., 15 (2015), pp. 715–732, https://doi.org/10.1007/s10208-013-9150-3.
[26] K. Peterson, P. Bochev, and D. Ridzal, *Optimization-based, property-preserving algorithm for passive tracer transport*, Computers & Mathematics with Applications, 159 (2024), pp. 267–286.
[27] T. Pock and A. Chambolle, *Diagonal preconditioning for first order primal-dual algorithms in convex optimization*, in 2011 International Conference on Computer Vision, 2011, pp. 1762–1769, https://doi.org/10.1109/ICCV.2011.6126441.
[28] R. T. Rockafellar, *Duality and stability in extremum problems involving convex functions*, Pacific J. Math., 21 (1967), pp. 167–187, http://projecteuclid.org/euclid.pjm/1102992608.
[29] R. T. Rockafellar, *Convex analysis*, (1970).
[30] C. Runge, *Über empirische funktionen und die interpolation zwischen äquidistanten ordinaten*, Zeitschrift für Mathematik und Physik, 46 (1901), p. 20.
[31] E. K. Ryu and W. Yin, *Large-scale convex optimization: algorithms & analyses via monotone operators*, Cambridge University Press, 2022.
[32] J. Shen and J. Xu, *Unconditionally bound preserving and energy dissipative schemes for a class of Keller-Segel equations*, SIAM J. Numer. Anal., 58 (2020), pp. 1674–1695, https://doi.org/10.1137/19M1246705.
[33] K. Wu, *Positivity-preserving analysis of numerical schemes for ideal magnetohydrodynamics*, SIAM J. Numer. Anal., 56 (2018), pp. 2124–2147, https://doi.org/10.1137/18M1168017.
[34] V. Zala, M. Kirby, and A. Narayan, *Structure-preserving function approximation via convex optimization*, SIAM J. Sci. Comput., 42 (2020), pp. A3006–A3029, https://doi.org/10.

1137/19M130128X.

[35] X. Zhang, *On positivity-preserving high order discontinuous Galerkin schemes for compressible Navier-Stokes equations*, J. Comput. Phys., 328 (2017), pp. 301–343, https://doi.org/10.1016/j.jcp.2016.10.002.

[36] X. Zhang and C.-W. Shu, *On maximum-principle-satisfying high order schemes for scalar conservation laws*, J. Comput. Phys., 229 (2010), pp. 3091–3120, https://doi.org/10.1016/j.jcp.2009.12.030.

[37] X. Zhang and C.-W. Shu, *On positivity-preserving high order discontinuous Galerkin schemes for compressible Euler equations on rectangular meshes*, J. Comput. Phys., 229 (2010), pp. 8918–8934, https://doi.org/10.1016/j.jcp.2010.08.016.