A FAST DUAL ITERATIVE ALGORITHM FOR CONVEXLY CONSTRAINED SPLINE SMOOTHING

Masao Yamagishi, Daichi Kitahara, Isao Yamada

Dept. of Communications and Computer Engineering, Tokyo Institute of Technology
E-mail: {myamagi, kitahara, isao}@sp.ce.titech.ac.jp

ABSTRACT

This paper proposes fast iterative algorithms for solving convexly constrained spline smoothing through a characterization of solutions in a primal-dual space. In view of achievements for fast implementations of spline interpolation, the update of the proposed algorithm is designed as the composition of solving a spline interpolation problem and computing the projection onto the constraint set. In addition, the update of the proposed algorithm is performed in an efficient dimensional space having the same size as given observations. These desired properties significantly reduce the computational cost in the update, which is demonstrated by a numerical example.

Index Terms— Bivariate spline smoothing, convexly constrained quadratic programming, iterative method, Nesterov's acceleration

1. INTRODUCTION

Spline smoothing is a regression technique of observed noisy data with a smooth piecewise-polynomial function [1], [2], [3], [4], which aims not only to denoise smoothly observed data, but also to explore the underlying system that generates the observed data and discover properties on the system. In fact, in feature extraction, rather than the resulting smoothed data itself, its first- and second-order derivatives provide us several features (of e.g. images and handwriting characters) [5], [6], [7]; in the so-called phase unwrapping problem in the SAR data processing [8], [9], [10], coefficients of each polynomial acts significant role to recovery unwrapped phase signal [11], [12].

Meanwhile, designing fast algorithms for spline smoothing is important in view of several (near-realtime) applications. Most of fast implementations for spline interpolation and spline smoothing are realized through their careful problem formalization, i.e., linearly constrained quadratic programming w.r.t. coefficients of all the polynomials of the spline, of which the solutions can be characterized in the large sparse (structured) linear equation [13]. Hence the solution can be obtained efficiently by suitable direct methods, e.g., LU, Cholesky, or QR decompositions [14], [15], [16]. On the other hand, to incorporate further prior information on noise of the observed data (e.g., boundedness, nonngativity, or probability distribution, etc.), convexly constrained quadratic programming provides a natural problem formulation, which can be solved by general iterative solvers [17], [18], [19], [20]. However, the auxiliary problems in their updates might not share the same structure with the spline interpolation/smoothing problems, and therefore the aforementioned successful fast implementations cannot be directly applied. In addition, these algorithms in general introduce auxiliary variables in their update, which require intensive computation and memory complexity.

In this paper, we resolve these two weaknesses by proposing fast iterative algorithms, for solving the convexly constrained spline smoothing problem, having two desired properties: (i) the update of the proposed algorithms is characterized by two auxiliary problems, the standard spline interpolation problem and computation of the metric projection onto the convex constraint; (ii) the update can be performed over the same dimensional space as the given observed data. Clearly, these properties resolve the above weaknesses, i.e., the first property implies that the former auxiliary problem can be solved efficiently by directly applying the successful fast implementations, and the second property provides efficient computational and memory complexity. In algorithm derivation, we extends idea of the Gauss-Seidel method in theory of linear system, to nonlinear system; That is, for a nonlinear system characterizing solution of the original spline smoothing problem, we split the nonlinear system into its upper and lower triangular systems and derive, with the two triangular systems, the update of iterative algorithm. In addition, we provide theoretical convergence analysis of the proposed algorithm. Furthermore, we propose an accelerated variant of the proposed algorithm by importing the Nesterov's technique [21], [22], [23]. Finally, a numerical example demonstrates that the proposed algorithms significantly reduce the computational cost.

This work was supported in part by JSPS Grants-in-Aid (26730128).

2. PRELIMINARIES

Let $\mathbb{R}$, $\mathbb{R}_+$ and $\mathbb{Z}_+$ be respectively the set of all real numbers, nonnegative real numbers and nonnegative integers. For any vector $x, y \in \mathbb{R}^N$, the inner product is defined by $\langle x, y\rangle = x^\top y$ and hence its induced norm is $\|x\| = \sqrt{\langle x, x\rangle}$ $(x \in \mathbb{R}^N)$.

2.1. Bivariate Spline Functions

Let $\Omega := [x_0, x_n] \times [y_0, y_m] \subset \mathbb{R}^2$. We shall partition it into triangles. Define a triangle $\mathcal{T}$ on $\mathbb{R}^2$, by specifying three vertices $v_k := (x_k, y_k) \in \mathbb{R}^2$ $(k = 1, 2, 3)$ which are not arranged linearly, i.e., $x_1y_2 - y_1x_2 + x_2y_3 - y_2x_3 + x_3y_1 - y_3x_1 \neq 0$, as

$$\mathcal{T} := \langle v_1, v_2, v_3\rangle := \left\{ rv_1 + sv_2 + tv_3 \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} r, s, t \in [0, 1] \\ r + s + t = 1 \end{array} \right\}.$$

Then create four kinds of triangles

$$\begin{cases} \mathcal{T}_{i,j,1} := \langle (x_{i-1}, y_{j-1}), (x_i, y_{j-1}), (\frac{x_{i-1}+x_i}{2}, \frac{y_{j-1}+y_j}{2})\rangle \\ \mathcal{T}_{i,j,2} := \langle (x_{i-1}, y_j), (x_{i-1}, y_{j-1}), (\frac{x_{i-1}+x_i}{2}, \frac{y_{j-1}+y_j}{2})\rangle \\ \mathcal{T}_{i,j,3} := \langle (x_i, y_j), (x_{i-1}, y_j), (\frac{x_{i-1}+x_i}{2}, \frac{y_{j-1}+y_j}{2})\rangle \\ \mathcal{T}_{i,j,4} := \langle (x_i, y_{j-1}), (x_i, y_j), (\frac{x_{i-1}+x_i}{2}, \frac{y_{j-1}+y_j}{2})\rangle \end{cases}$$

by diagonally cutting every rectangle $[x_{i-1}, x_i] \times [y_{j-1}, y_j]$ s.t. $x_i - x_{i-1} = h_x > 0$ for all $i = 1, 2, \ldots, n$ and $y_j - y_{j-1} = h_y > 0$ for all $j = 1, 2, \ldots, m$. Let $\boxtimes_{n,m} := \{\mathcal{T}_{i,j,1}, \mathcal{T}_{i,j,2}, \mathcal{T}_{i,j,3}, \mathcal{T}_{i,j,4}\}_{j=1,2,\ldots m}^{i=1,2,\ldots,n}$ be a set of all triangles $\mathcal{T}_{i,j,k}$ on Ω. For $\rho, d \in \mathbb{Z}_+$ s.t. $0 \leq \rho < d$, define

$$\mathcal{S}_d^\rho(\boxtimes_{n,m}) := \{f \in C^\rho(\Omega) \,|\, \forall \mathcal{T}_{i,j,k} \in \boxtimes_{n,m} \quad f|_{\mathcal{T}_{i,j,k}} \in \mathbb{P}_d\}$$

as the set of all bivariate splines of degree d and smoothness ρ on $\boxtimes_{n,m}$, where $C^\rho(\Omega)$ stands for the set of all ρ-times continuously differentiable functions over Ω, and $\mathbb{P}_d$ is the set of all bivariate polynomials whose degree is d at most, i.e., $\mathbb{P}_d := \{f : \mathbb{R}^2 \to \mathbb{R} : (x,y) \mapsto \sum_{p=0}^{d}\sum_{q=0}^{d-p} c_{p,q}x^p y^q \mid c_{p,q} \in \mathbb{R}\}$.

In this paper, we suppose that a bivariate spline $f \in \mathcal{S}_d^\rho(\boxtimes_{n,m})$ is expressed, over $\mathcal{T}_{i,j,k}$, as

$$f(x,y) = \sum_{p=0}^{d}\sum_{q=0}^{p} c^{i,j,k}_{p(p+1)/2+q+1}\frac{d!\, r^{d-p}s^{p-q}t^q}{(d-p)!(p-q)!q!}, \quad (1)$$

where $c^{i,j,k}_{p(p+1)/2+q+1} \in \mathbb{R}$ are coefficients of the bivariate spline, and $(r,s,t) \in [0,1]^3$ $(r+s+t=1)$ is called *barycentric coordinate* of (x,y) with respect to $\mathcal{T}_{i,j,k}$ [24], [25], e.g., the barycentric coordinate with respect to $\mathcal{T}_{i,j,1}$ is

$$(r,s,t) = (x_{i-1} - x + y_j - y, x - x_{i-1} + y_{j-1} - y, 2y - 2y_{j-1}).$$

2.2. Convexly Constrained Spline Smoothing

Assume that we observe samples of a twice continuously differentiable function $f_*\colon \Omega \to \mathbb{R}$ with additive noise $\epsilon_{i,j} \in \mathbb{R}$ on $\mathcal{G} := \{x_0, x_1, \ldots, x_n\} \times \{y_0, y_1, \ldots, y_m\}$, i.e., we observe $\zeta_{i,j} = f_*(x_i, y_j) + \epsilon_{i,j}$ at $(x_i, y_j) \in \mathcal{G}$. In this situation, the problem of interest is to reconstruct the unknown function $f_* \in C^2(\Omega)$ by a C^2-spline function $f \in \mathcal{S}_d^2(\boxtimes_{n,m})$:

Problem 1 (Convexly constrained two-dimensional spline smoothing) *Find* $f^* \in \mathcal{S}_d^2(\boxtimes_{n,m})$ *minimizing*

$$\iint_\Omega \left[\left|\frac{\partial^2 f}{\partial x^2}\right|^2 + 2\left|\frac{\partial^2 f}{\partial x \partial y}\right|^2 + \left|\frac{\partial^2 f}{\partial y^2}\right|^2\right] \mathrm{d}x\mathrm{d}y \quad (2)$$

subject to[1] $\mathrm{vec}(f(x_i,y_j) - \zeta_{i,j})_{j=0,1,\ldots,m}^{i=0,1,\ldots,n} \in C \subset \mathbb{R}^M$ *with* $M := (n+1)(m+1)$*, where* C *is a nonempty closed convex set.*

Note that existence of the minimizers is guaranteed if $d \geq 4$ [25]. Obviously, if $C = \{0\}$ then this problem is reduced to the spline interpolation problem. In this paper, we assume that C is simple, i.e., its projection[2] P_C can be computed efficiently.

The standard reformulation (see e.g. [26], [27]) of spline smoothing problems into problems of all the coefficients, of polynomials of the spline function, leads to a convexly constrained quadratic programming problem

$$\begin{aligned} \min_{c\in\mathbb{R}^N}\ & c^\top Q c =: \phi(c) \\ \text{s.t. } & Hc = 0 \\ & \mathcal{I}c - \zeta \in C, \end{aligned} \quad (3)$$

where $c \in \mathbb{R}^N$ is a coefficient vector of the spline function, the cost function ϕ is nothing but the integral of the squared second-order derivative in (2), the linear constraint represents continuity condition of the first- and the second-order derivatives of the spline, and the second constraint is identical to the one introduced in the original problem[3]. Here, all the matrices $Q \in \mathbb{R}^{N\times N}$, $H \in \mathbb{R}^{M_H \times N}$, $\mathcal{I} \in \mathbb{R}^{M\times N}$ are sparse, where we can assume that Q is symmetric positive semidefinite without loss of generality. In addition, $\mathcal{I}$ has specific properties[4]

$$\begin{aligned} \mathcal{I}\mathcal{I}^\top &= I_M \\ P_{\mathcal{N}(\mathcal{I})} &= I_N - \mathcal{I}^\top\mathcal{I}. \end{aligned} \quad (5)$$

We can clarify an obvious relationship of the spline smoothing problem and the spline interpolation problem by introducing a slack variable $\xi \in \mathbb{R}^M$ which represents difference of the observed sample $\zeta = \mathrm{vec}(\zeta_{i,j})_{j=0,1,\ldots,m}^{i=0,1,\ldots,n} \in \mathbb{R}^M$ and the spline function, i.e.,

$$\begin{aligned} \min_{(\xi,c)\in\mathbb{R}^M\times\mathbb{R}^N}\ & c^\top Q c \\ \text{s.t. } & Hc = 0 \\ & \mathcal{I}c - \zeta = \xi \\ & \xi \in C. \end{aligned} \quad (6)$$

Then if we fix the slack variable ξ, the problem (6) becomes a spline interpolation problem

$$\begin{aligned} \min_{c\in\mathbb{R}^N}\ & c^\top Q c \\ \text{s.t. } & Hc = 0 \\ & \mathcal{I}c = \zeta + \xi, \end{aligned} \quad (7)$$

of which solution can be characterized by the so-called KKT system [28]: if $(\tilde{c}, \tilde{\lambda}_H, \tilde{\lambda}_\zeta) \in \mathbb{R}^N \times \mathbb{R}^{M_H} \times \mathbb{R}^M$ satisfies[5]

$$\begin{pmatrix} 2Q & H^\top & \mathcal{I}^\top \\ -H & O_{M_H\times M_H} & O_{M_H\times M} \\ -\mathcal{I} & O_{M\times M_H} & O_M \end{pmatrix} \begin{pmatrix} \tilde{c} \\ \tilde{\lambda}_H \\ \tilde{\lambda}_\zeta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\zeta - \xi \end{pmatrix}, \quad (8)$$

$\tilde{c}$ is a solution of problem (7). Since (8) is a large sparse system of equations, $(\tilde{c}, \tilde{\lambda}_H, \tilde{\lambda}_\zeta)$ can be obtained efficiently by suitable direct methods, i.e., LU, Cholesky, or QR decompositions [16]. This fact provides us a guideline to design iterative algorithms.

3. PROPOSED METHOD

We shall propose an iterative algorithm to solve the spline smoothing problem (6). The relationship between (6) and (7) motivates us to consider whether it is possible to reformulate a spline smoothing problem into a sequence of spline interpolation problems. The answer is affirmative and it is embodied as an iterative algorithm to solve (6). Our derivation is two-fold: clarifying a characterization of the problem, which is a sort of KKT system; applying Gauss-Seidel-like splitting to introduce a candidate operator describing the update. Using iteratively the derived operator, we can introduce an iterative algorithm (see Algorithm 1) of which convergences are guaranteed (note: all proofs are omitted due to space limitation).

Lemma 1 (Characterization of Solutions) *The solution of the spline smoothing problem (6) can be characterized by*

$$\begin{pmatrix} I_M + \mu\partial\iota_C & F_\mu \\ G & J \end{pmatrix} \begin{pmatrix} \xi_* \\ v_* \end{pmatrix} \ni 0, \quad (9)$$

[1] vec denotes the vectorization of 2-dimensional array.

[2] For a given nonempty closed convex set $S \subset \mathbb{R}^M$, the projection $P_S\colon \mathbb{R}^M \to \mathbb{R}^M$ onto S is defined by

$$P_S(x) = \arg\min_{y\in S}\|y - x\|.$$

[3] Although we only focus on the case where the two equality constraints are feasible for any $\xi \in C$, such an assumption is natural (see e.g. [27]).

[4] We denote the identity matrix of size M as $I_M \in \mathbb{R}^{M\times M}$. For a given matrix A, we denote its null as $\mathcal{N}(A)$.

It is well-known that

$$\forall (x,y) \in \mathcal{G}\ \exists \mathcal{T} \in \boxtimes_{n,m} \text{ such that } (x,y) \text{ is a vertex of } \mathcal{T} \quad (4)$$

implies that each row vector of $\mathcal{I}$ has only one non-zero component and its value is one, i.e., (5) holds true. Obviously, the condition (4) holds true in our problem setting.

[5] We denote zero matrix of size $M \times N$ as $O_{M\times N}$. For squared zero matrix of size $M \times M$, we simply denote O_M.

where[6]

$$F_\mu := \begin{pmatrix} -\mathcal{I} & O_{M\times M_H} & -\mu I_M & I_M \end{pmatrix}$$

$$G := \begin{pmatrix} O_{N\times M} \\ O_{M_H\times M} \\ I_M \\ O_M \end{pmatrix}$$

$$J := \begin{pmatrix} 2Q & H^\top & \mathcal{I}^\top & O_{N\times M} \\ -H & O_{M_H\times M_H} & O_{M_H\times M} & O_{M_H\times M} \\ -\mathcal{I} & O_{M\times M_H} & O_M & I_M \\ O_{M\times N} & O_{M\times M_H} & -I_M & \partial\iota_{\{\zeta\}} \end{pmatrix}$$

$$v_* := \begin{pmatrix} c_* \\ \lambda_{H*} \\ \lambda_{\zeta*} \\ z_* \end{pmatrix} \in \mathbb{R}^N \times \mathbb{R}^{M_H} \times \mathbb{R}^M \times \mathbb{R}^M =: \mathcal{H}$$

with a user-defined parameter $\mu > 0$. That is, if (ξ_*, v_*) satisfies (9), then its (ξ_*, c_*) is a solution of the spline smoothing problem (6); conversely, if (ξ_*, c_*) is a solution of (6), then there exists a pair (ξ_*, v_*) satisfies that (9) and the first upper block of v_* is identical to c_*. Note that z_* is an auxiliary variable s.t. $z_* = \mathcal{I}c_* - \xi_*$, and $\lambda_{H*}, \lambda_{\zeta*}$ are multipliers, i.e., $\lambda_{H*} \in \partial\iota_{\{0\}}(Hc_*)$ and $\lambda_{\zeta*} \in \partial\iota_{\{\zeta\}}(z_*)$.

Theorem 1 (Gauss-Seidel-like Splitting) *(a) For the characterization (9), we shall define a Gauss-Seidel-like splitting operator[7]: assume that the interpolation problem (7) has a solution for any $\xi \in C$. Then there exists an operator $J_G^\dagger$, which maps from $\xi \in C$ to $v \in \mathcal{H}$ s.t.*

$$J(v) = -G\xi. \tag{10}$$

Moreover, the operator $T\colon \mathbb{R}^M \times \mathcal{H} \to C \times \mathcal{H} : (\xi, v) \mapsto (\xi_+, v_+)$ defined by

$$\xi_+ = P_C(-F_\mu v) \tag{11}$$

$$v_+ = J_G^\dagger(\xi_+) \tag{12}$$

satisfies that

$$\begin{pmatrix} I_M + \mu\partial\iota_C & O_{M\times M_\mathcal{H}} \\ G & J \end{pmatrix} \begin{pmatrix} \xi_+ \\ v_+ \end{pmatrix} \ni \begin{pmatrix} O_M & -F_\mu \\ O_{M_\mathcal{H}\times M} & O_{M_\mathcal{H}} \end{pmatrix} \begin{pmatrix} \xi \\ v \end{pmatrix}, \tag{13}$$

where we denote the dimension of $\mathcal{H}$ as $M_\mathcal{H} := N + M_H + 2M$.
(b) Define the iterative algorithm, with initial (ξ_0, v_0), by

$$\begin{pmatrix} \xi_{k+1} \\ v_{k+1} \end{pmatrix} = T \begin{pmatrix} \xi_k \\ v_k \end{pmatrix}.$$

Then the following two convergences are guaranteed:
(i) assume that $J_G^\dagger$ is continuous and $-F_\mu \circ J_G^\dagger$ is averaged nonexpansive[8] [29]. Then the sequence $(\xi_k, v_k)_{k\geq 1}$ converges to a some $(\xi_\infty, v_\infty) \in \mathrm{Fix}(T)$, and (ξ_∞, c_∞) with c_∞ in v_∞ is a solution of problem (6);
(ii) suppose that

$$\sqrt{2\mu\sigma_{\max}(Q)}\|c_{k+1} - c_k\| \leq \|\xi_{k+1} - \xi_k\|$$

holds true for a some $\mu > 0$, where $\sigma_{\max}(Q)$ is the largest eigenvalue of Q. Then we have

$$\phi(c_k) - \phi(c_*) \leq \frac{\mu^{-1}\|\xi_1 - \xi_*\|^2}{k}, \quad \forall k \in \mathbb{Z}_+ \setminus \{0\}. \tag{14}$$

[6]For a given nonempty closed convex set C, the indicator function is defined by

$$\iota_C(x) := \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{otherwise.} \end{cases}$$

[7]Inspired by the Gauss-Seidel method in linear algebra, we split the characterization into its upper and lower triangular parts, and utilize them to define an operator to satisfy the relation (13). For this reason, we refer to our proposed operator as Gauss-Seidel-like splitting operator.

[8]The operator $T\colon \mathbb{R}^N \to \mathbb{R}^N$ is called nonexpansive if

$$\|Tx - Ty\| \leq \|x - y\|, \ \forall x, y \in \mathbb{R}^N.$$

In addition, the nonexpansive operator T is called averaged if there exists some $\alpha \in (0, 1)$ and some nonexpansive mapping U such that

$$T = (1 - \alpha)I + \alpha U,$$

where I is the identity operator.

Algorithm 1: A Gauss-Seidel-like splitting algorithm

Init:	$\xi_0 \in \mathbb{R}^M$, stepsize $\mu > 0$
Step 1:	Update $v_{k+1} = J_G^\dagger(\xi_k)$ $\xi_{k+1} = P_C(-F_\mu v_{k+1})$ for $k = 0, 1, \ldots, K$

Algorithm 2: An efficient implementation of Alg. 1

Init 1:	$\xi_0 \in \mathbb{R}^M$, stepsize $\mu > 0$
Init 2:	Compute the LU decomposition L, U of $\widetilde{\mathcal{M}} := \begin{pmatrix} 2P_{\mathcal{N}(\mathcal{I})}QP_{\mathcal{N}(\mathcal{I})} & P_{\mathcal{N}(\mathcal{I})}H^\top \\ -HP_{\mathcal{N}(\mathcal{I})} & O_{M_H} \end{pmatrix}$ and construct $\mathcal{M} := \mathcal{I}\left[\begin{pmatrix} 2QP_{\mathcal{N}(\mathcal{I})} & H^\top \end{pmatrix} U^\dagger L^{-1} \begin{pmatrix} -2P_{\mathcal{N}(\mathcal{I})}Q \\ H \end{pmatrix} + 2Q\right]\mathcal{I}^\top.$
Step 1:	Update $\xi_{k+1} = P_C\left(\xi_k - \mu\mathcal{M}(\xi_k + d)\right)$ for $k = 0, 1, \ldots, K-1$.
Step 2:	Obtain a solution by $v_K = J_G^\dagger(\xi_K)$

Remark 1: (First desired property) The specially designed operator $J_G^\dagger$ in the update (12) is nothing but a map from the slack variable ξ to a solution of the spline interpolation problem (7): it is easy to show that (10) is identical to the KKT system (8). Hence its implementation can be realized with several successful techniques [14], [15], [16]. □

Fortunately, we can reduce computational complexity of Algorithm 1 by explicitly eliminating $(v_k)_{k\geq 0}$ in the update. Since the update (11), (12) has a compact expression

$$\xi_{k+1} = P_C(-F_\mu J_G^\dagger(\xi_k)),$$

we do not require to store $(v_k)_{k\geq 0}$ in the update. Moreover, the following expression of $-F_\mu \circ J_G^\dagger$ as an affine operator reduces computational complexity on each iteration (see Algorithm 2).

Proposition 1 *The operator $-F_\mu \circ J_G^\dagger$ is expressed by*

$$-F_\mu J_G^\dagger(\xi_+) = \xi_+ - \mu\mathcal{M}(\xi_+ + \zeta)$$

for any $\xi_+ \in C$, where $\mathcal{M} \in \mathbb{R}^{M\times M}$ is defined by

$$\mathcal{M} := \mathcal{I}\left[\begin{pmatrix} 2QP_{\mathcal{N}(\mathcal{I})} & H^\top \end{pmatrix} U^\dagger L^{-1} \begin{pmatrix} -2P_{\mathcal{N}(\mathcal{I})}Q \\ H \end{pmatrix} + 2Q\right]\mathcal{I}^\top \tag{15}$$

using the LU decomposition L, U of

$$\widetilde{\mathcal{M}} := \begin{pmatrix} 2P_{\mathcal{N}(\mathcal{I})}QP_{\mathcal{N}(\mathcal{I})} & P_{\mathcal{N}(\mathcal{I})}H^\top \\ -HP_{\mathcal{N}(\mathcal{I})} & O_{M_H} \end{pmatrix}. \tag{16}$$

Remark 2: (Second desired property) Algorithm 2 using Proposition 1 realizes the second desired property: since $\mathcal{M}$ is of size $M \times M$, the entire update is performed on $\mathbb{R}^M$, which is significantly efficient compared with that of the characterization (9) (cf. $M_\mathcal{H} > M$). Moreover, $\mathcal{M}$ can be computed and stored before starting iterations, such properties are appropriate to accelerate the

iterations. Note that, to the best of our knowledge, other general solvers including proximal methods [17], [18], [19], [20] do not achieve this property. In fact, direct application of these methods requires a variable whose size is strictly larger than M. □

Finally, by extending the idea of the so-called Nesterov's acceleration [21], [22], [23], we propose an acceleration of the proposed algorithm, which has improved convergence rate $\mathcal{O}(1/k^2)$ (c.f. (14)).

Theorem 2 *Let $\xi_{1/2} = \xi_0 \in \mathbb{R}^M$, and $\alpha_1 = 1$. For $k = 1, 2, \ldots$*

$$\begin{aligned}\xi_k &= P_C(-F_\mu J_G^\dagger(\xi_{k-1/2}))\\ \alpha_{k+1} &= \frac{1+\sqrt{1+4\alpha_k^2}}{2}\\ \xi_{k+1/2} &= \xi_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(\xi_k - \xi_{k-1}).\end{aligned}$$

Under the assumption

$$\sqrt{2\mu\sigma_{\max}(Q)}\|c_k - c_{k-1/2}\| \le \|\xi_k - \xi_{k-1/2}\|,$$

we have

$$\phi(c_k) - \phi(c_*) \le \frac{2\mu^{-1}\|\xi_0 - \xi_*\|^2}{(k+1)^2}, \quad \forall k \in \mathbb{Z}_+ \setminus \{0\}.$$

Remark that Theorem 2 is not trivial: the standard Nesterov's acceleration is only applicable to the minimization of the sum of a smooth convex function and a nonsmooth convex function. However, since the spline smoothing problem (6) has three constraints, the acceleration technique cannot be applied directly to problem (6). For that reason, we introduce a special treatment for the linear constraints in convergence analysis, which is omitted in this paper.

4. NUMERICAL EXAMPLE

We examine efficacy of the propose method in the sense of computational complexity. Consider the 2D spline smoothing problem for InSAR application in [11].[9] C is set as a box constraint $\{\xi \in \mathbb{R}^M \mid \|\xi\|_\infty \le 0.1\}$ as an example. The data ζ is generated from zero mean Gaussian distribution with unit variance, where the size of $\zeta \in \mathbb{R}^M$ varies from $M = 5^2$ to 30^2.

Three algorithms are evaluated: Algorithm 1 with a poor implementation of $J_G^\dagger$; Algorithm 1 with a sophisticated LU decomposition; Algorithm 2 with a sophisticated LU decomposition. We refer to these three algorithms as Algorithm 1 (poor), Algorithm 1 (UMFPACK), and Algorithm 2 (UMFPACK) in this experiment, respectively. In the first method, i.e. Algorithm 1 (poor), the operator $J_G^\dagger$ to solve the KKT system (8) is implemented by using the inverse matrix directly constructed, which results in the intensive computation cost in the preprocessing (see pre. in Table 1). The second method, i.e. Algorithm 1 (UMFPACK), avoids such intensive computation by a sophisticated LU decomposition (UMFPACK [15] implemented in MATLAB) and hence the CPUtime is reduced significantly in the preprocessing. The third method eliminates the variable $(v_k)_{k\ge 0}$ in its update to reduce the computational cost on the iterations. We adopt UMFPACK as an implementation of the LU decomposition of $\widetilde{\mathcal{M}}$ in (16) (or Init 2 of Algorithm 2) to avoid intensive computation for the preprocessing. Unlike the other two methods, the third method requires to generate v_K in Step 2. Its computational cost is also evaluated (see post. in Table 1). We would like to emphasize that the comparison between the first and second method shows that the advantage, indicated in Remark 1, to use a fast implementation of spline interpolation, and the comparison between the second and third method demonstrates that the improvement, indicated in Remark 2, by performing the iterations in an efficient dimensional space.

[9] In [11], it is considered a 2-dimensional spline smoothing same as Problem 1 of degree $d = 4$ in Sect. 2.2.

Table 1. Comparison of CPUtime, which is evaluated separately for the following three processes: "pre." implies the preprocess beforehand iterations (i.e., the computation of the matrix inversion involving (8) required in the implementation of $J_G^\dagger$ in Algorithm 1 (poor), an LU decomposition involving (8) in Algorithm 1 (UMFPACK), or Init 2 in Algorithm 2 (UMFPACK)); "iter." shows the CPUtime for one iteration; "post." implies Step 2 in Algorithm 2 (UMFPACK).

size		Algorithm 1 (poor)	Algorithm 1 (UMFPACK)	Algorithm 2 (UMFPACK)
5^2	pre. [sec]	3.6×10^{-1}	$\mathbf{2.0 \times 10^{-2}}$	$\mathbf{2.0 \times 10^{-2}}$
	iter. [sec/iter]	4.3×10^{-3}	1.1×10^{-3}	$\mathbf{9.0 \times 10^{-5}}$
	post. [sec]	*	*	1.2×10^{-3}
	total [sec]	5.5×10^{-0}	4.6×10^{-1}	$\mathbf{2.2 \times 10^{-1}}$
10^2	pre. [sec]	12×10^{-0}	$\mathbf{9.9 \times 10^{-2}}$	3.2×10^{-1}
	iter. [sec/iter]	5.9×10^{-2}	2.9×10^{-3}	$\mathbf{1.1 \times 10^{-4}}$
	post. [sec]	*	*	1.8×10^{-3}
	total [sec]	89×10^{-0}	2.0×10^{-0}	$\mathbf{1.4 \times 10^{-0}}$
15^2	pre. [sec]	95×10^{-0}	$\mathbf{4.1 \times 10^{-1}}$	1.7×10^{-0}
	iter. [sec/iter]	2.3×10^{-1}	6.5×10^{-3}	$\mathbf{3.3 \times 10^{-4}}$
	post. [sec]	*	*	6.4×10^{-3}
	total [sec]	425×10^{-0}	$\mathbf{3.5 \times 10^{-0}}$	4.0×10^{-0}
20^2	pre. [sec]	too long	$\mathbf{1.1 \times 10^{-0}}$	6.1×10^{-0}
	iter. [sec/iter]	not eval.	1.3×10^{-2}	$\mathbf{1.1 \times 10^{-3}}$
	post. [sec]	*	*	1.2×10^{-2}
	total [sec]	too long	$\mathbf{10 \times 10^{-0}}$	$\mathbf{10 \times 10^{-0}}$
25^2	pre. [sec]	too long	$\mathbf{2.5 \times 10^{-0}}$	16×10^{-0}
	iter. [sec/iter]	not eval.	1.8×10^{-2}	$\mathbf{2.7 \times 10^{-3}}$
	post. [sec]	*	*	2.1×10^{-2}
	total [sec]	too long	$\mathbf{17 \times 10^{-0}}$	24×10^{-0}
30^2	pre. [sec]	too long	$\mathbf{39 \times 10^{-0}}$	234×10^{-0}
	iter. [sec/iter]	not eval.	1.4×10^{-1}	$\mathbf{5.4 \times 10^{-3}}$
	post. [sec]	*	*	1.4×10^{-1}
	total [sec]	too long	$\mathbf{71 \times 10^{-0}}$	246×10^{-0}

Table 1 shows a CPUtime comparison for the three algorithms. In the sense of the total CPUtime, Algorithm 1 (UMFPACK) is best for relatively large problems. Meanwhile, Algorithm 2 (UMFPACK) achieves the fastest computation for the iterations, and its postprocessing is computed efficiently. Therefore, for (near-)realtime InSAR application, Algorithm 2 (UMFPACK) is the best choice because the preprocessing can be performed before starting measurement from the sky or from the cosmic space, i.e., the CPUtime for the iterations and postprocessing is dominant.

5. CONCLUDING REMARKS

In this paper, we have proposed a novel iterative algorithm to solve convexly constrained spline smoothing problems. The update of the proposed algorithm is designed as the composition of computing the projection onto convex constraint and solving a spline interpolation problem, so that we can utilize fast implementation techniques for spline interpolation problems. In addition, the update can be performed on a reasonable dimensional space, i.e., the update variable belongs to the same dimensional space as the size of observation data, which reflects the computational efficacy of the proposed algorithm. In addition, we have extended the spirit behind the Nesterov's acceleration technique to the proposed method.

Our future work includes further acceleration of the proposed algorithm by extending an over-relaxation for the Nesterov's technique [23].

6. REFERENCES

[1] C. De Boor, *A Practical Guide to Splines*, Applied Mathematical Sciences. Springer Verlag, 2001.

[2] G. Wahba, "Improper priors, spline smoothing and the problem of guarding against model errors in regression," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 364–372, 1978.

[3] B. W. Silverman, "Some aspects of the spline smoothing approach to non-parametric regression curve fitting," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–52, 1985.

[4] M. Unser, "Splines: A perfect fit for signal and image processing," *IEEE Signal Processing Magazine*, vol. 16, no. 6, pp. 22–38, 1999.

[5] I. Sekita, K. Toraichi, R. Mori, K. Yamamoto, and H. Yamada, "Feature extraction of handwritten japanese characters by spline functions for relaxation matching," *Pattern Recognition*, vol. 21, no. 1, pp. 9–17, 1988.

[6] Ø. Due Trier, A. K. Jain, and T. Taxt, "Feature extraction methods for character recognition—a survey," *Pattern Recognition*, vol. 29, no. 4, pp. 641–662, 1996.

[7] G. Medioni and Y. Yasumoto, "Corner detection and curve representation using cubic *B*-splines," *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 3, pp. 267–278, 1987.

[8] R. M. Goldstein, H. A. Zebker, and C. L. Werner, "Satellite radar interferometry: Two-dimensional phase unwrapping," *Radio Science*, vol. 23, no. 4, pp. 713–720, 1988.

[9] C. W. Chen and H. A. Zebker, "Network approaches to two-dimensional phase unwrapping: intractability and two new algorithms," *JOSA A*, vol. 17, no. 3, pp. 401–414, 2000.

[10] A. Reigber, R. Scheiber, M. Jager, P. Prats-Iraola, I. Hajnsek, T. Jagdhuber, K. P. Papathanassiou, M. Nannini, E. Aguilera, S. Baumgartner, et al., "Very-high-resolution airborne synthetic aperture radar imaging: Signal processing and applications," *Proceedings of the IEEE*, vol. 101, no. 3, pp. 759–783, 2013.

[11] D. Kitahara and I. Yamada, "Algebraic phase unwrapping over collection of triangles based on two-dimensional spline smoothing," in *IEEE ICASSP*, 2014.

[12] D. Kitahara and I. Yamada, "Algebraic phase unwrapping based on two-dimensional spline smoothing over triangles," *IEEE Transactions on Signal Processing*, 16pp., 2015, available online.

[13] M. Unser, A. Aldroubi, and M. Eden, "B-spline signal processing. II. efficiency design and applications," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 834–848, 1993.

[14] G. H. Golub and C. F. Van Loan, *Matrix Computations*, vol. 3, JHU Press, 2012.

[15] T. A. Davis, "Algorithm 832: UMFPACK v4. 3—an unsymmetric-pattern multifrontal method," *ACM Transactions on Mathematical Software (TOMS)*, vol. 30, no. 2, pp. 196–199, 2004.

[16] T. A. Davis, *Direct methods for sparse linear systems*, vol. 2, SIAM, 2006.

[17] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Modeling & Simulation*, vol. 4, no. 4, pp. 1168–1200, 2005.

[18] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pp. 185–212. Springer, 2011.

[19] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 123–231, 2013.

[20] H. Raguet, J. Fadili, and G. Peyré, "A generalized forward-backward splitting," *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1199–1226, 2013.

[21] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$," in *Doklady AN SSSR*, 1983, vol. 269, pp. 543–547.

[22] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[23] M. Yamagishi and I. Yamada, "Over-relaxation of the fast iterative shrinkage-thresholding algorithm with variable stepsize," *Inverse Problems*, vol. 27, no. 10, pp. 105008, 2011.

[24] G. Farin, "Triangular Bernstein-Bèzier patches," *Computer Aided Geometric Design*, vol. 3, no. 2, pp. 83–127, 1986.

[25] C. K. Chui, *Multivariate Splines*, vol. 54 of *CBMS-NSF Regional Conference Series in Applied Mathematics*, Philadelphia, PA: SIAM, 1988.

[26] E. Quak and L.L. Schumaker, "Calculation of the energy of a piecewise polynomial surface," in *Algorithms for approximation II*, pp. 134–143. Springer, 1990.

[27] M.-J. Lai, "Multivariate splines and their applications," in *Computational Complexity: Theory, Techniques, and Applications*, pp. 1939–1980. NewYork, NY: Springer, 2012.

[28] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge university press, 2009.

[29] I. Yamada, M. Yukawa, and M. Yamagishi, "Minimizing the Moreau envelope of nonsmooth convex functions over the fixed point set of certain quasi-nonexpansive mappings," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pp. 345–390. Springer, 2011.