A fast Gauss-Seidel-like splitting 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 a fast iterative algorithm for solving convexly constrained spline smoothing. The update of the proposed algorithm has two desired properties: achievements for fast spline interpolation can be directly incorporated; it can be performed on an efficient dimensional space, of which dimension is the same as the number of given observations. These properties are established by extending the so-called Gauss-Seidel method in linear algebra to nonlinear system of equations. The efficacy of the proposed algorithm is demonstrated in a numerical example in InSAR signal processing.

1 Introduction

Spline smoothing is a regression technique of observed noisy data with a smooth piecewise-polynomial function [1]–[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]–[7]; in the so-called phase unwrapping problem in the SAR data processing [8]–[10], coefficients of each polynomial acts significant role to recovery unwrapped phase signal [11].

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 [12]. Hence the solution can be obtained efficiently by suitable direct methods, e.g., LU, Cholesky, or QR decompositions [13]–[15]. 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 [16]–[18]. 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. Finally, a numerical example demonstrates that the proposed algorithms significantly reduce the computational cost.

2 Preliminaries

Let $\Delta_M := \{\eta_i\}_{i=0}^{M-1}$ be a grid on an area[1] $\Omega := [\eta_0, \eta_{M-1}] \subset \mathbb{R}$ s.t. $\eta_0 < \eta_1 < \cdots < \eta_{M-1}$, and let $d, \rho \in \mathbb{Z}_+$ s.t. $0 \le \rho < d$. Define

$$\mathcal{S}_d^\rho(\Delta_M) := \{f \in C^\rho(\Omega) \mid \forall i \ f = f_i \in \mathbb{P}_d \text{ over } [\eta_i, \eta_{i+1}]\}$$

as the set of all spline functions of degree d and smoothness ρ on Δ_M, where $C^\rho(\Omega)$ stands for the set of all ρ-times continuously differentiable functions over the interior of Ω and $\mathbb{P}_d$ denotes the set of all polynomials whose degree is d at most.

Assume that we observe samples of a twice continuously differentiable function $f_*\colon \Omega \to \mathbb{R}$ with additive noise $\epsilon_i \in \mathbb{R}$ on Δ_M, i.e., we observe $\zeta_i = f_*(\eta_i) + \epsilon_i$ at η_i $(i = 0, 1, \ldots, M-1)$. In this situation, the problem of interest is to reconstruct the unknown function $f_* \in C^2(\Omega)$ by a C^2-spline function $s \in \mathcal{S}_d^2(\Delta_M)$:

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

(Convexly Constrained Spline Smoothing Problem) Find $s_* \in \mathcal{S}_d^2(\Delta_M)$ minimizing

$$\int_{\eta_0}^{\eta_M} |s''(\eta)|^2 d\eta \tag{1}$$

subject to $(s(\eta_i) - \zeta_i)_{i=0}^{M-1} \in C$, where $C \subset \mathbb{R}^M$ is a nonempty closed convex set. Note that obviously, if $C = \{0\}$ then this problem is reduced to the spline interpolation problem.

The standard reformulation (see e.g. [19], [20]) of spline smoothing problems into problems of all the coefficients, of polynomials of the spline function, leads to a convexly constrained quadratic programming problem

$$\begin{aligned} &\min_{x\in\mathbb{R}^N} \ x^\top Qx =: \phi(x) \\ &\quad \text{s.t. } Hx = 0 \\ &\qquad \mathcal{I}x - \zeta \in C, \end{aligned} \tag{2}$$

where $x \in \mathbb{R}^N$ is a coefficient vector of the spline function, the cost function ϕ is nothing but the integral of the squared second-order derivative in (1), 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[2]. 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. In addition, we can assume that Q is symmetric and that $\mathcal{I}$ has specific properties[3]

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

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 $\zeta = (\zeta_i)_{i=0}^{M-1} \in \mathbb{R}^M$ and the spline function, i.e.,

$$\begin{aligned} &\min_{(\xi,x)\in\mathbb{R}^M\times\mathbb{R}^N} \ x^\top Qx \\ &\quad \text{s.t. } Hx = 0 \\ &\qquad \mathcal{I}x - \zeta = \xi \\ &\qquad \xi \in C. \end{aligned} \tag{3}$$

Then if we fix the slack variable ξ, the problem (3) becomes a spline interpolation problem

$$\begin{aligned} &\min_{x\in\mathbb{R}^N} \ x^\top Qx \\ &\quad \text{s.t. } Hx = 0 \\ &\qquad \mathcal{I}x = \zeta + \xi, \end{aligned} \tag{4}$$

of which solution can be characterized by the so-called KKT system [21]. Then since the system becomes a large sparse system of equations, the solution can be obtained efficiently by suitable direct methods, i.e., LU, Cholesky, or QR decompositions [15]. 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 (3). Our derivation is two-fold: clarifying a characterization of the problem; applying Gauss-Seidel-like splitting to introduce a candidate operator describing the update. Applying iteratively the derived operator, we can introduce an iterative algorithm of which convergences are guaranteed.

Lemma 1 (Characterization of Solutions) *The solution of the spline smoothing problem (3) can be characterized by*[4]

$$\begin{pmatrix} I_M + \mu\partial\iota_C & F_\mu \\ G & J \end{pmatrix} \begin{pmatrix} \xi_* \\ v_* \end{pmatrix} \ni 0, \tag{5}$$

where

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

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

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

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

with a user-defined parameter $\mu > 0$. That is, if (ξ_*, v_*) satisfies (5), then its (ξ_*, x_*) is a solution of the spline smoothing problem (3); conversely, if (ξ_*, x_*) is a solution of (3), then there exists a pair (ξ_*, v_*) satisfies that (5) and the first upper block of v_* is identical to x_*. Note that y_* is an auxiliary variable s.t. $y_* = \mathcal{I}x_* - \xi_*$, and $\lambda_{H*}, \lambda_{\zeta*}$ are multipliers, i.e., $\lambda_{H*} \in \partial\iota_{\{0\}}(Hx_*)$ and $\lambda_{\zeta*} \in \partial\iota_{\{\zeta\}}(y_*)$.

Theorem 1 (Gauss-Seidel-like Splitting) *(a) For the characterization (5), we shall define a Gauss-Seidel-like splitting operator*[5]*: assume that the interpolation problem (4) 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.*

$$J(v) = -G\xi. \tag{6}$$

[2] 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. [20]).

[3] 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 $\mathcal{N}(A)$. For a given nonempty closed convex set $S \subset \mathbb{R}^N$, the projection $P_S\colon \mathbb{R}^N \to \mathbb{R}^N$ onto S is defined by

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

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

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

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.

[5] Inspired by the Gauss-Seidel method, we split the characterization into its upper and lower triangular parts, and utilize them to define an operator to satisfy the relation (9).

Algorithm 1: A Gauss-Seidel-like splitting algorithm

Init:	$\xi_0 \in \mathbb{R}^M$, stepsize $\mu > 0$
Step 1:	Update $\xi_{k+1} = P_C(-F_\mu J_G^\dagger(\xi_k))$ for $k = 0, 1, \ldots, K$
Step 2:	Obtain a solution by $v_K = J_G^\dagger(\xi_K)$

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

$$\xi_+ = P_C(-F_\mu v) \tag{7}$$

$$v_+ = J_G^\dagger(\xi_+) \tag{8}$$

satisfies that[6]

$$\begin{pmatrix} I_M + \mu\partial\iota_C & O_{M\times M_\mathcal{H}} \\ G & J \end{pmatrix}\begin{pmatrix} \xi_+ \\ v_+ \end{pmatrix} \ni \begin{pmatrix} O_M & -F_\mu \\ O_{M_\mathcal{H}\times M} & O_{M_\mathcal{H}} \end{pmatrix}\begin{pmatrix} \xi \\ v \end{pmatrix}. \tag{9}$$

(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 [22]*[7]*. Then the sequence* $(\xi_k, v_k)_{k\geq 1}$ *converges to a some* $(\xi_\infty, v_\infty) \in \mathrm{Fix}(T)$*, and a pair of* x_∞ *in* v_∞ *and* ξ_∞ *is a solution of problem (3);*
(ii) suppose that

$$\sqrt{\mu L}\|x_{k+1} - x_k\| \leq \|\xi_{k+1} - \xi_k\|$$

holds true for a some $\mu > 0$*. Then we have*

$$\phi(x_k) - \phi(x_*) \leq \frac{\mu^{-1}\|\xi_1 - \xi_*\|^2}{k}, \quad \forall k \in \mathbb{Z}_+ \setminus \{0\}. \tag{10}$$

Remark 1: (First desired property) The specially designed operator $J_G^\dagger$ in the update (8) is nothing but a map from the slack variable to a solution of the spline interpolation problem (4): it is easy to show that (6) is identical to the KKT system of (4). Hence its implementation can be realized with several successful techniques. □

Fortunately, we can reduce computational complexity of the algorithm by explicitly eliminating $(v_k)_{k\geq 0}$ in the update. Since the update (7), (8) 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 further computational complexity on each iteration (see Algorithm 2).

[6]We denote the dimension of $\mathcal{H}$ as $M_\mathcal{H} := N + M_H + 2M$.

[7]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 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 $v_K = J_G^\dagger(\xi_K)$

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*

$$\mathcal{M} := \mathcal{I}\left[\begin{pmatrix} 2QP_{\mathcal{N}(\mathcal{I})} & H^\top \end{pmatrix} U^\dagger L^{-1}\begin{pmatrix} -2P_{\mathcal{N}(\mathcal{I})}Q \\ H \end{pmatrix} + 2Q\right]\mathcal{I}^\top \tag{11}$$

using the LU decomposition L, U *of*

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

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 (5) (cf. $M_\mathcal{H} > M$). Moreover, $\mathcal{M}$ can be computed and stored before starting iterations, such properties are appropriate to accelerate the iterations. □

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].[8] C is set as a box constraint $\{\xi \in \mathbb{R}^M \mid \|\xi\|_\infty \leq 0.1\}$ 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.

Table 1 shows CPUtime comparison for three algorithms: Algorithm 1 with a poor implementation of $J_G^\dagger$; Algorithm 1 with a sophisticated LU decomposition; Algorithm 2. In the Algorithm 1 with a poor implementation of $J_G^\dagger$, inverse matrices are directly constructed, which results in the intensive computation cost in the preprocessing. The second method avoids such intensive computation by a sophisticated LU decomposition (UMFPACK [14] implemented in MATLAB) and hence the CPUtime is reduced significantly in the preprocessing[9]. In the sense of the total CPUtime, Algorithm 1 is best for relatively large problem. Meanwhile,

[8]In [11], the 2D spline smoothing problem is considered. The 2D space is partitioned in triangular regions, in a way of the so-called crisscross partition, and on each region the Bernstein-Bézier polynomial of degree 4 is utilized.

[9]Note that in our implementation Algorithm 1 with UMFPACK computes all the auxiliary variables in each iteration, hence the post processing is not required.

Table 1: Comparison of CPUtime, which is evaluated separately for the following three processes: "pre." implies the preprocess (i.e., the computation of matrix inversion and $\mathcal{M}$ in (11) beforehand iterations); "iter." shows the CPUtime for one iteration; "post." implies Step 2 in Alg. 2 (i.e., generating v_K from ξ_K).

size		Algorithm 1 (poor)	Algorithm 1 (UMFPACK)	Algorithm 2
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}

Algorithm 2 achieves the fastest computation for the iterations by eliminating v_k in the update. Therefore, for (near-)realtime InSAR application, Algorithm 2 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 finding spline interpolation, 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.

Our future work includes further acceleration of the proposed algorithm by extending an over-relaxation for the Nesterov's technique [23]–[25].

Acknowledgment

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

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] 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.

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

[14] 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.

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

[16] 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.

[17] 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.

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

[19] 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.

[20] M.-J. Lai, "Multivariate splines and their applications," in *Encyclopedia of Complexity and Systems Science*, pp. 5800–5841. Springer, 2009.

[21] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2009.

[22] 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.

[23] 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.

[24] 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.

[25] 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.