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-real-time) applications. Most of fast implementations for spline interpolation and spline smoothing are realized through their careful 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, nonnegativity, 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 \( \Omega := [\eta_0, \eta_{M-1}] \subset \mathbb{R} \) s.t. \( \eta_0 < \eta_1 < \cdots < \eta_{M-1} \), and let \( d, \rho \in \mathbb{Z}_+ \) s.t. \( 0 \leq \rho < d \). Define \( S^\rho_d(\Delta_M) := \{ f \in C^\rho(\Omega) \mid \forall i \ f_i \in P_d \text{ over } [\eta_i, \eta_{i+1}] \} \) as the set of all spline functions of degree \( d \) and smoothness \( \rho \) on \( \Delta_M \), where \( C^\rho(\Omega) \) stands for the set of all \( \rho \)-times continuously differentiable functions over the interior of \( \Omega \) and \( P_d \) denotes the set of all polynomials whose degree is \( d \) at most.

Assume that we observe samples of a twice continuously differentiable function \( f_0 : \Omega \to \mathbb{R} \) with additive noise \( \epsilon_i \in \mathbb{R} \) on \( \Delta_M \), i.e., we observe \( \zeta_i = f_0(\eta_i) + \epsilon_i \) at \( \eta_i \ (i = 0, 1, \ldots, M - 1) \). In this situation, the problem of interest is to reconstruct the unknown function \( f_0 \in C^2(\Omega) \) by a \( C^2 \)-spline function \( s \in S^2_d(\Delta_M) \):

\[ \langle x, y \rangle = x^T y. \]
(Convexly Constrained Spline Smoothing Problem) Find $s^* \in S_2^2(\Delta M)$ minimizing

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

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

$$
\min_{x \in \mathbb{R}^N} x^T Q x =: \phi(x)
$$

s.t. $H x = 0$
$$
\mathcal{I} x - \zeta = \xi
$$
$$
\xi \in C,
$$

where $x \in \mathbb{R}^N$ is a coefficient vector of the spline function, the cost function $\phi$ 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. Here, all the matrices $Q \in \mathbb{R}^{N \times N}$, $H \in \mathbb{R}^{M \times N}$, $\mathcal{I} \in \mathbb{R}^{M \times N}$ are sparse. In addition, we can assume that $Q$ is symmetric and that $\mathcal{I}$ has specific properties.

$$
\mathcal{I} \mathcal{I}^T = I_M
$$
$$
P_{N(\mathcal{I})} = I_N - \mathcal{I}^T \mathcal{I}.
$$

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

$$
\min_{(\xi, x) \in \mathbb{R}^M \times \mathbb{R}^N} x^T Q x
$$

s.t. $H x = 0$
$$
\mathcal{I} x - \zeta = \xi
$$
$$
\xi \in C.
$$

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

$$
\min_{x \in \mathbb{R}^N} x^T Q x
$$

s.t. $H x = 0$
$$
\mathcal{I} x = \zeta + \xi,
$$

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

$$
\begin{pmatrix}
I_M + \mu \partial_\zeta C & F_\mu \\
G & J
\end{pmatrix}
\begin{pmatrix}
\xi^* \\
v_*
\end{pmatrix}
\succeq 0,
$$

where

$$
F_\mu := (-\mathcal{I} - O_{M \times M} - \mu I_M)
$$
$$
G :=
\begin{pmatrix}
O_{N \times M} \\
O_{M \times M} \\
I_M
\end{pmatrix}
$$
$$
J :=
\begin{pmatrix}
2Q & H^T & \mathcal{I}^T & O_{N \times M} \\
-\mathcal{I} & O_{M \times M} & O_{M \times M} & O_{M \times M} \\
O_{M \times N} & O_{M \times M} & O_{M} & I_M \\
O_{M \times N} & O_{M \times M} & -I_M & \partial_\xi C
\end{pmatrix}
$$
$$
v_* :=\begin{pmatrix} x_* \\
\lambda_{H \ast} \\
\lambda_{\zeta \ast}
\end{pmatrix} \in \mathbb{R}^N \times \mathbb{R}^{M \times N} \times \mathbb{R}^M =: \mathcal{H}
$$

with a user-defined parameter $\mu > 0$. That is, if $(\xi, v_*)$ satisfies (5), then its $(\xi, x_*)$ is a solution of the spline smoothing problem (3); conversely, if $(\xi, x_*)$ is a solution of (3), then there exists a pair $(\xi, v_*)$ satisfies (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 $\lambda_{H \ast}, \lambda_{\zeta \ast}$ are multipliers, i.e., $\lambda_{H \ast} \in \partial_\zeta(\mathcal{H} x_*)$ and $\lambda_{\zeta \ast} \in \partial_\zeta(\zeta)(y_*)$.

Theorem 1 (Gauss-Seidel-like Splitting) (a) For the characterization (5), we shall define a Gauss-Seidel-like splitting operator $
abla$; assume that the interpolation problem (4) has a solution for any $\xi \in C$. Then there exists an operator $J^\top_G$, which maps from $\xi \in C$ to $v \in \mathcal{H}$ s.t.

$$
J(v) = -G \xi.
$$

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

4 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 \( T : \mathbb{R}^M \times \mathcal{H} \to C \times \mathcal{H} \), \( (\xi, v) \mapsto (\xi_+, v_+) \) defined by
\[
\xi_+ = P_C(-F_{\mu}v)
\]
\[
v_+ = J_G^\dagger(\xi_+)
\]
satisfies that
\[
(I_M + \mu \partial_{\mathcal{H}} J_G) \begin{pmatrix} \xi_+ \\ v_+ \end{pmatrix} \geq \begin{pmatrix} O_M \quad -F_{\mu} \\ \partial_{\mathcal{H}} J_G \end{pmatrix} \begin{pmatrix} \xi \\ v \end{pmatrix}.
\]

(b) Define the iterative algorithm, with initial \( (\xi_0, v_0) \), 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 \[22\]. Then the sequence \( (\xi_k, v_k)_{k \geq 1} \) converges to a some \( (\xi_\infty, v_\infty) \in \text{Fix}(T) \), and a pair of \( x_\infty \in v_\infty \) and \( x_\infty \in \text{solution of problem (3)} \); (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
\[
\phi(x_k) - \phi(x_\ast) \leq \frac{\mu^{-1} \|\xi_1 - \xi_\ast\|^2}{k}, \quad \forall k \in \mathbb{Z}_+ \setminus \{0\}.
\]

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).

[Note: 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.]

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 \( \zeta \) 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,
Algorithm 2 achieves the fastest computation for the iterations by eliminating $\nu_k$ in the update. Therefore, for a 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


