*Algorithms* **2010**, *3*, 311-328; doi:10.3390/a3030311

OPEN ACCESS

*algorithms*

**ISSN 1999-4893**

www.mdpi.com/journal/algorithms

*Article*

# Univariate Cubic $L_1$ Interpolating Splines: Spline Functional, Window Size and Analysis-based Algorithm

**Lu Yu[1,⋆], Qingwei Jin[1], John E. Lavery[1,2] and Shu-Cherng Fang[1]**

[1] Industrial and Systems Engineering Department, North Carolina State University, Raleigh, NC 27695-7906, USA; E-Mails: qjin2@ncsu.edu (Q.J.); john.lavery2@us.army.mil (J.L.); fang@ncsu.edu (S.-C.F.)

[2] Mathematical Sciences Division, Army Research Office, Army Research Laboratory, P.O. Box 12211, Research Triangle Park, NC 27709-2211, USA

⋆ Author to whom correspondence should be addressed; E-Mail: lyu@ncsu.edu; Tel.: +01-919-513-1909; Fax: +01-919-515-5281.

*Received: 11 July 2010 / Accepted: 10 August 2010 / Published: 20 August 2010*

---

**Abstract:** We compare univariate $L_1$ interpolating splines calculated on 5-point windows, on 7-point windows and on global data sets using four different spline functionals, namely, ones based on the second derivative, the first derivative, the function value and the antiderivative. Computational results indicate that second-derivative-based 5-point-window $L_1$ splines preserve shape as well as or better than the other types of $L_1$ splines. To calculate second-derivative-based 5-point-window $L_1$ splines, we introduce an analysis-based, parallelizable algorithm. This algorithm is orders of magnitude faster than the previously widely used primal affine algorithm.

**Keywords:** antiderivative; cubic $L_1$ spline; first derivative; 5-point window; function value; global; interpolation; locally calculated; second derivative; univariate

**Classification: MSC** 65D05, 65D07

---

## 1. Introduction

$L_1$ splines have been shown to provide superior shape preservation for interpolation and approximation of multiscale data, that is, data with sudden large changes in magnitude and/or spacing

used for modeling of natural and urban terrain, geophysical features, biological objects, robotic paths and many other irregular surfaces, processes and functions ([1–15]). The minimization principles for $L_1$ splines have typically (but not uniformly) been based on the $L_1$ norm of the second derivative (rather than on the $L_1$ norm of expressions involving other levels of derivatives). The resulting non-differentiable convex generalized geometric programs have been solved by active set [3], primal affine [5–7,10,11] and primal-dual [12,14] algorithms.

In the literature, there are a few indications of limitations of the primal affine and primal-dual algorithms for bivariate $L_1$ splines for large data sets [9,12]. There is also unpublished computational experience of the authors and others who have noticed issues of incomplete convergence or not completely correct convergence of the active set, primal affine and primal-dual algorithms for both univariate and bivariate $L_1$ splines. It is in this context that we seek in this paper a new algorithmic approach for calculating $L_1$ splines. Auquiert, Gibaru and Nyiri [16] have developed a subdifferential-based procedure for calculating second-derivative-based $L_1$ splines on 5-point windows. We propose here an algorithm for second-derivative-based 5-point-window $L_1$ splines based on the analysis in Section 2 of [17], which links, via analytical properties of the spline functional, local geometric properties of 5-point windows of the data set with geometric properties of the $L_1$ spline interpolant.

In considering second-derivative-based 5-point-window $L_1$ splines, three types of information are needed for full "situational awareness," namely, 1) information about whether use of 5-point windows is superior to use of windows of other sizes and to use of global data sets, 2) information about whether use of the second derivative in the spline functional is superior to use of the first derivative, function value or antiderivative and 3) information about whether the new, analysis-based algorithm mentioned above can achieve computational results superior to those of the primal affine algorithm, which was the previously most widely used algorithm for calculating $L_1$ splines. The third item here has not yet been considered in the literature and the information in the literature on the first and second items is sketchy at best. The computational and analytical results in [2,16,17] do suggest that 5-point windows have advantages vs. global calculations. However, it is not yet known whether windows of other sizes might also have advantages. Results presented in [10,11] indicate that first-derivative-based and function-value-based $L_1$ splines may have advantages over standard second-derivative-based $L_1$ splines. However, these two publications considered only the behavior of $L_1$ splines on the global scale and did not consider behavior on the fine, interval-to-interval scale.

The present paper addresses these needs. In Section 2, we give a brief description of the primal affine algorithm that has previously been widely used to calculate $L_1$ splines. In Section 3, we compare $L_1$ splines calculated by minimizing, on 5-point windows, on 7-point windows and on global data sets, four different spline functionals, namely, ones based on the second derivative, the first derivative, the function value and the antiderivative. These $L_1$ splines are calculated by the primal affine algorithm. The results of this section provide motivation for the development of a new, analysis-based algorithm for calculating 5-point-window, second-derivative-based $L_1$ splines. In Section 4, we present this new algorithm, which is based on the analysis in Section 2 of [17]. In Section 5, we show that, while the results of both the new algorithm and the primal affine algorithm look good on the macro level, there are differences on the micro level. Specifically, the results of the new algorithm are accurate on the micro level while those of

the primal affine algorithm are occasionally only approximate. Finally, in Section 6, we summarize the results presented in the previous sections and point out the potential for future algorithms for locally calculated univariate $L_1$ approximating splines and locally calculated bivariate $L_1$ interpolating and approximating splines.

All of the quantities in this paper are real quantities. The nodes $x_i$, $i = 0, 1, \ldots, I$, are a strictly monotonic but otherwise arbitrary partition of the finite interval $[x_0, x_I]$. Let $h_i = x_{i+1} - x_i$. At each node $x_i$, the function value $z_i$ is given, $i = 0, 1, \ldots, I$. The slope of the line segment connecting $(x_i, z_i)$ and $(x_{i+1}, z_{i+1})$ is

$$\triangle z_i := \frac{z_{i+1} - z_i}{h_i}, \quad i = 0, 1, \ldots, I-1\,. \tag{1}$$

The local and global cubic $L_1$ splines discussed in this paper are cubic polynomials in each interval $(x_i, x_{i+1})$, $i = 0, 1, \ldots, I-1$, and are $C^1$ continuous at the nodes. The first derivative of the spline at node $x_i$, $i = 0, 1, \ldots, I$, is denoted by $b_i$ (to be determined by minimization of the $L_1$ spline functional). We use $\delta_i$ to denote the slope of the chord between neighboring points:

$$\begin{aligned} \delta_i &= \frac{z_{i+1} - z_{i-1}}{x_{i+1} - x_{i-1}}, \quad i = 1, 2, \ldots, I-1, \\ \delta_0 &= \frac{z_1 - z_0}{h_0} \quad \text{and} \quad \delta_I = \frac{z_I - z_{I-1}}{h_{I-1}}\,. \end{aligned} \tag{2}$$

We use $\zeta$ to denote the linear spline:

$$\zeta(x) = \frac{(x_{i+1} - x)z_i + (x - x_i)z_{i+1}}{h_i}, \quad x \in [x_i, x_{i+1}], \quad i = 0, 1, \ldots, I-1\,. \tag{3}$$

For the interpolation problem under consideration in the present paper, the nodes and the function values at the nodes are given. Minimization of a spline functional means "determination of the first derivatives that yield the minimum." The first derivatives determined from the minimization along with the fixed nodal and functional values yield the piecewise cubic $L_1$ spline by the standard Hermite interpolation formula. For reference, we present here the spline functionals that define traditional, globally calculated $L_1$ splines ("global $L_1$ splines"). Second-derivative-based, first-derivative-based, function-value-based and antiderivative-based cubic $L_1$ splines are calculated by minimizing

$$\sum_{i=0}^{I-1} \int_{x_i}^{x_{i+1}} \left| \frac{\mathrm{d}^2 z}{\mathrm{d}x^2} \right| \mathrm{d}x\,, \tag{4}$$

$$\sum_{i=0}^{I-1} \int_{x_i}^{x_{i+1}} \frac{1}{h_i} \left| \frac{\mathrm{d}z}{\mathrm{d}x} - \frac{\mathrm{d}\zeta}{\mathrm{d}x} \right| \mathrm{d}x\,, \tag{5}$$

$$\sum_{i=0}^{I-1} \int_{x_i}^{x_{i+1}} \frac{1}{h_i^2} \left| z - \zeta \right| \mathrm{d}x \tag{6}$$

and

$$\sum_{i=0}^{I-1} \int_{x_i}^{x_{i+1}} \frac{1}{h_i^3} \left| \int_{(x_i + x_{i+1})/2}^{x} (z(\xi) - \zeta(\xi)) \mathrm{d}\xi \right| \mathrm{d}x\,, \tag{7}$$

respectively, over the finite-dimensional spline space of $C^1$ piecewise cubic polynomials $z$ that interpolate the data. The $L_1$ splines that minimize these functionals can be nonunique, a situation

that will be handled by adding "regularization terms" to the functionals when they are minimized by the primal affine algorithm or, in the new algorithm proposed in this paper, by applying a "choice procedure" as described below in Section 4. Second-derivative-based $L_1$ splines are the $L_1$ splines commonly encountered in the literature. First-derivative-based $L_1$ splines have been investigated in [10,11]. Function-value-based $L_1$ splines have been treated in [11]. Antiderivative-based $L_1$ splines are newly introduced in this present paper to provide additional insight into how the order of the derivative in the spline functional affects the geometric shape preservation properties of the $L_1$ spline.

In the present paper, computations of different algorithms are based on the following challenging data set consisting of 56 irregular data points that lie on flat, linear, quadratic, cubic and oscillatory functions and on protuberances, patched together with discontinuities of function values and first derivatives and with extreme irregular spacing—with lengths of neighboring intervals differing by up to a factor of 100:

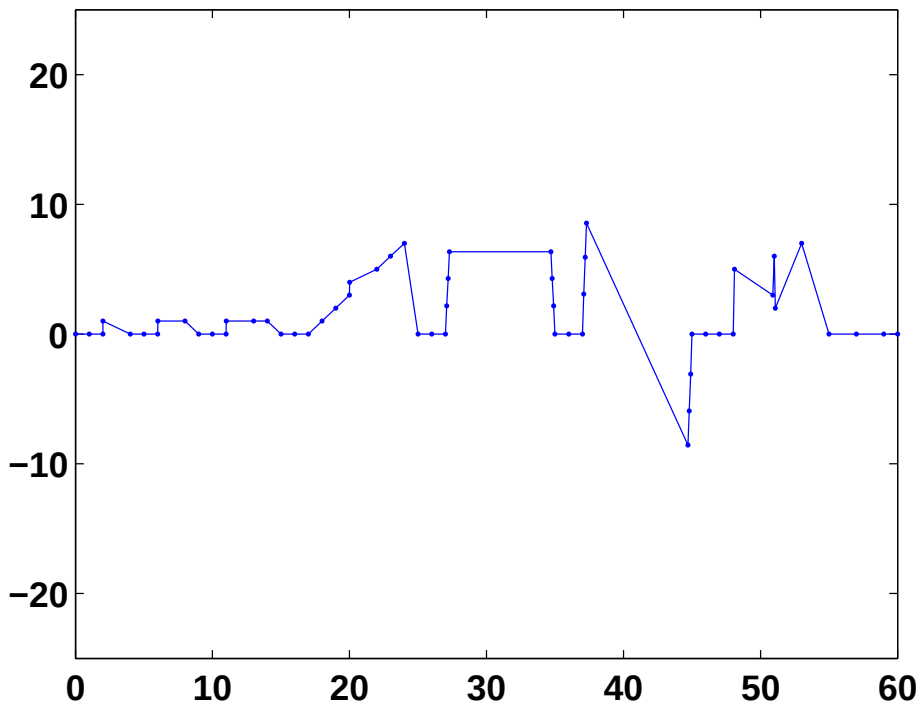
$$
\begin{array}{lllllllllll}
\boldsymbol{x} = & (0, & 1, & 2, & 2.01, & 4, & 5, & 6, & 6.01, & 8, & 9, & 10, \\
 & 11, & 11.01, & 13, & 14, & 15, & 16, & 17, & 18, & 19, & 20, & 20.01, \\
 & 22, & 23, & 24, & 25, & 26, & 27, & 27.1, & 27.2, & 27.3, & 34.7, & 34.8, \\
 & 34.9, & 35, & 36, & 37, & 37.1 & 37.2, & 37.3, & 44.7, & 44.8, & 44.9, & 45, \\
 & 46, & 47, & 48, & 48.1, & 50.9, & 51, & 51.08, & 53, & 55, & 57, & 59, \\
 & 60), & & & & & & & & & & \\
\boldsymbol{z} = & (0, & 0, & 0, & 1, & 0, & 0, & 0, & 1, & 1, & 0, & 0, \\
 & 0, & 1, & 1, & 1, & 0, & 0, & 0, & 1, & 2, & 3, & 4, \\
 & 5, & 6, & 7, & 0, & 0, & 0, & 2.1725, & 4.29, & 6.3525, & 6.3525, & 4.29, \\
 & 2.1725, & 0, & 0, & 0, & 3.081, & 5.928, & 8.547, & -8.547, & -5.928, & -3.081, & 0, \\
 & 0, & 0, & 0, & 5, & 3, & 6, & 2, & 7, & 0, & 0, & 0, \\
 & 0). & & & & & & & & & &
\end{array}
$$

This data set was used in [11]. The data $(x_i, z_i)$, $i = 27, 28, 29, 30, 31, 32, 33, 34$, in the region from $x = 27$ to $x = 35$ lie on the the quadratic function $44 - 2.75(x - 31)^2$. The data $(x_i, z_i)$, $i = 36, 37, 38, 39, 40, 41, 42, 43$, in the region from $x = 37$ to $x = 45$ lie on the cubic function $-16(x - 41) + (x - 41)^3$. Note the large gaps in the intervals $[x_{30}, x_{31}] = [27.3, 34.7]$ and $[x_{39}, x_{40}] = [37.3, 44.7]$. In the figures, the data will be represented by dots "·". The linear spline for these data is given in Figure 1.

## 2. Primal Affine and Other Previously Available Algorithms for $L_1$ Splines

Minimization of the global $L_1$ spline functionals (4), (5), (6) and (7) is a nonlinear programming problem. Direct minimization of (4) has been accomplished by an active set method [3]. Active set algorithms for minimizing (5), (6) and (7) have not been developed, but these functionals and functional (4) have been minimized by primal affine algorithms and primal-dual algorithms [12,14]. Primal affine and primal-dual algorithms are linear (not nonlinear) programming procedures. To create a linear program suitable for application of these algorithms, the integrals in the $L_1$ spline functionals need to be discretized. For the primal affine algorithm used in the present paper and in [5,6,8,10,11], the spline functionals were discretized by the midpoint rule with $K$ equal subintervals in each interval $(x_i, x_{i+1})$. In this paper, $K = 100$. For a detailed description of the primal affine algorithm, see [7].

**Figure 1**. Data set and linear spline.




For calculation of $L_1$ splines by the primal affine method using the global functionals (4), (5), (6) and (7), we add to these functionals the regularization terms

$$\varepsilon \sum_{0}^{I} |b_i - \delta_i|, \; i = 0, 1, \ldots, I \,. \tag{8}$$

When $\varepsilon$ is sufficiently small, the $L_1$ spline that minimizes the functional with the regularization term is unique. For the computational experiments of the present paper, $\varepsilon = 10^{-4}$.

In addition to calculating $L_1$ splines by minimizing the global functionals (4), (5), (6) and (7) by the primal affine method, we will calculate $L_1$ splines using these functionals on 5-point and 7-point windows. To calculate the derivative $b_i$ of a "windowed" $L_1$ spline at node $x_i$, we minimize functionals that are the same as (4), (5), (6) and (7) except that the integral is over a local set of intervals ("window") rather than over the global domain. For 5-point-window $L_1$ splines, the window $[x_{\hat{\imath}-2}, x_{\hat{\imath}+2}]$ is used for nodes $x_{\hat{\imath}}$, $\hat{\imath} = 2, 3, \ldots, I-2$, the window $[x_0, x_4]$ is used for nodes $x_{\hat{\imath}}$, $\hat{\imath} = 0, 1$ and the window $[x_{I-4}, x_I]$ is used for nodes $x_{\hat{\imath}}$, $\hat{\imath} = I-1, I$. For 7-point-window $L_1$ splines, the window $[x_{\hat{\imath}-3}, x_{\hat{\imath}+3}]$ is used for nodes $x_{\hat{\imath}}$, $\hat{\imath} = 3, 4, \ldots, I-3$, the window $[x_0, x_6]$ is used for nodes $x_{\hat{\imath}}$, $\hat{\imath} = 0, 1, 2$ and the window $[x_{I-6}, x_I]$ is used for nodes $x_{\hat{\imath}}$, $\hat{\imath} = I-2, I-1, I$. When minimizing a "windowed" $L_1$ spline functional for node $x_{\hat{\imath}}$ by the primal affine algorithm, we add to the spline functional the regularization term

$$\varepsilon |b_{\hat{\imath}} - \delta_{\hat{\imath}}| \,. \tag{9}$$

For the computational experiments of the present paper, $\varepsilon = 10^{-4}$.

## 3. Computational Results for Windowed and Global $L_1$ Splines

A window with an odd number of points is desirable, since it has a "middle point" at which one can use a locally calculated derivative as the derivative of the global interpolant. Conversely, a window with an even number of points is not desirable, because it lacks a middle point. Using the primal affine algorithm described in the previous section, we generated computational results for 5-point-window and 7-point-window $L_1$ splines as well as for global $L_1$ splines.

In the literature, there have been reports [10,11] about $L_1$ splines based on spline functionals involving first derivatives and function values, that is, derivatives of degree lower than the standard second degree. Results in [10,11] suggest that, for global $L_1$ splines, spline functionals based on the first derivative or function value (that is, on functional (5) or (6)) result in improved shape preservation. Those results emphasized overall global preservation of shape on the macro scale but did not treat local preservation of shape on the fine, interval-to-interval scale. Following up on these prior investigations, we revisit here the comparison of $L_1$ splines based on derivatives of degree 2, 1 and 0, add to the comparison $L_1$ splines based on derivatives of degree $-1$ (antiderivatives) and add the new dimension of considering the window size (5 points, 7 points or global). For this comparison, we use the primal affine algorithm described in Section 2.

Computational results are presented in Figures 2–13. In each figure, we highlight in dashed boxes the intervals $[25, 27]$, $[35, 37]$ and $[45, 55]$, where differences among the various splines are most prominent. The second-derivative-based 5-point-window spline of Figure 2 and the first-derivative-based and function-value-based global splines of Figures 7 and 10 preserve linearity in the intervals $[25, 27]$, $[35, 37]$ and $[45, 48]$ and avoid extraneous overshoot, extraneous undershoot and extraneous oscillation in the interval $[48.1, 55]$. None of the other nine splines of Figures 2–13 is able to preserve linearity and avoid extraneous overshoot, undershoot and oscillation in all of these intervals. In particular, the splines of Figures 3, 4, 8, 11, 12 and 13 do not preserve linearity well on the intervals $[25, 27]$, $[35, 37]$ and/or $[45, 48]$. The splines of Figures 3, 5, 8 and 11 have extraneous oscillation on the interval $[48.1, 50.9]$. The splines of Figures 6, 9 and 12 have undershoot in $[48.1, 50.9]$. The differences in the large intervals $[27.3, 34.7]$ and $[37.3, 44.7]$ is a separate issue that is discussed in the remark below.

**Remark** The data points immediately to the right and left of the large intervals $[27.3, 34.7]$ and $[37.3, 44.7]$ lie on a quadratic and a cubic function, respectively. The second- and first-derivative-based splines of Figures 2–7 approximate both the quadratic and the cubic function. In contrast, the function-value-based and antiderivative-based splines of Figures 8–13 avoid approximating the quadratic function in $[27.3, 34.7]$, even though they do approximate the cubic function in $[37.3, 44.7]$. In classical approximation theory, the ability to approximate or even reproduce quadratic, cubic and higher-degree functions is a goal. In geometric modeling of irregular data, however, reproducing any function of degree higher than 1 is generally considered disadvantageous because it leads to extraneous oscillation, overshoot or undershoot. It is not yet known how to construct $L_1$ splines that generically avoid approximating functions of degree higher than 1. In the present article, we acknowledge this issue but we do not take it into account when assessing the shape-preservation capabilities of the various types of $L_1$ splines.

**Figure 2**. Second-derivative-based 5-point-window $L_1$ spline

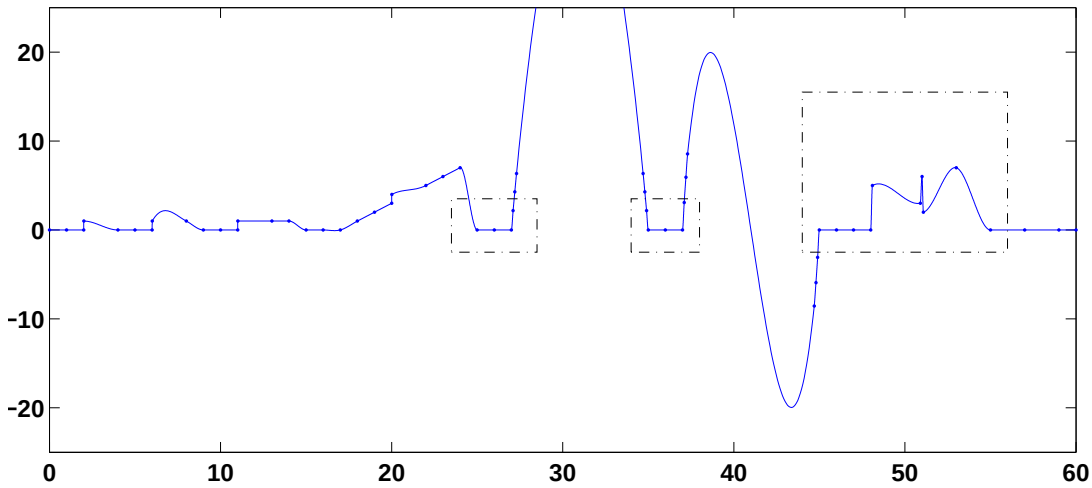



**Figure 3**. Second-derivative-based 7-point-window $L_1$ spline

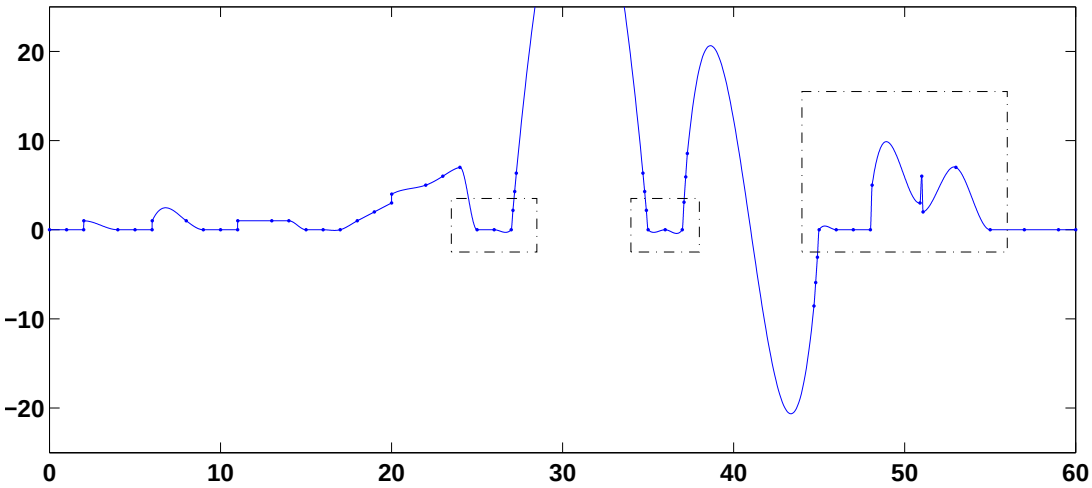



**Figure 4**. Second-derivative-based global $L_1$ spline

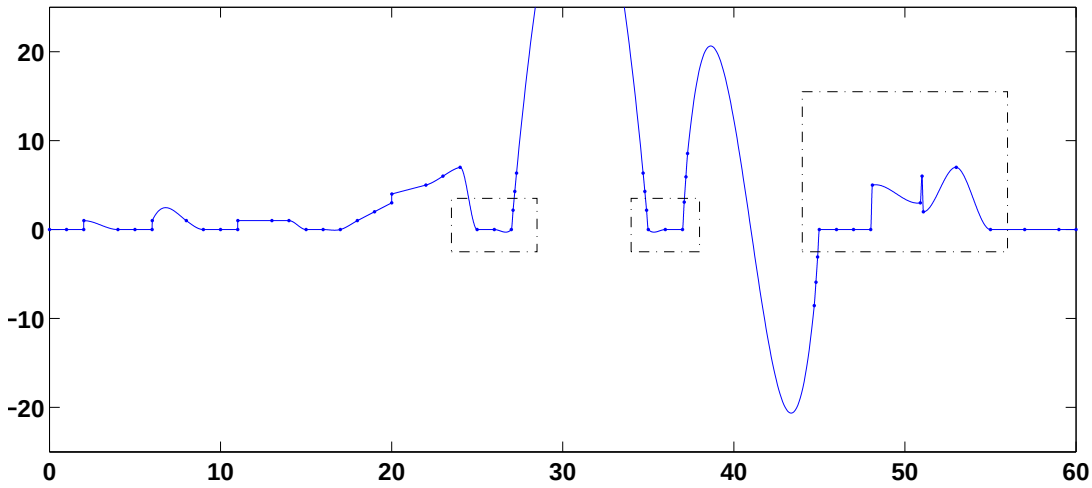


**Figure 5**. First-derivative-based 5-point-window $L_1$ spline

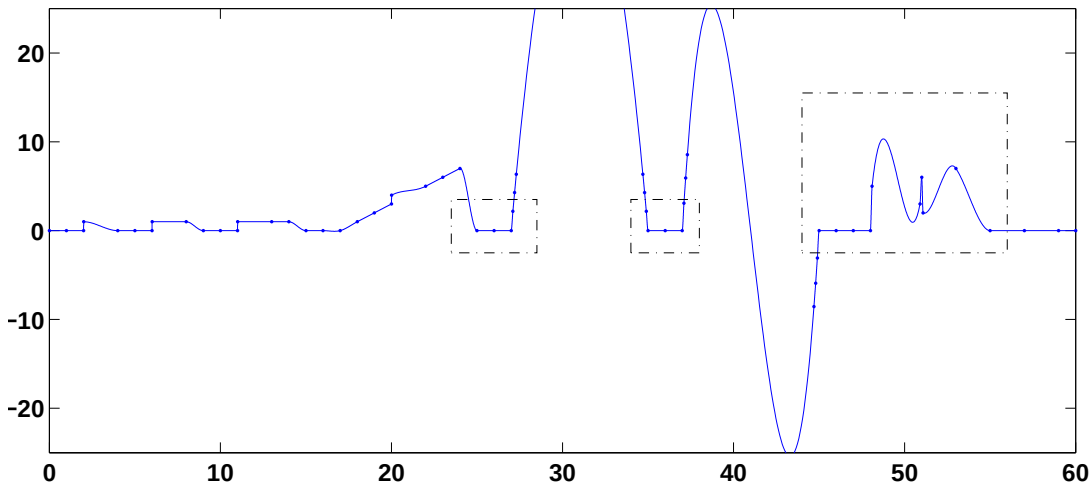



**Figure 6**. First-derivative-based 7-point-window $L_1$ spline

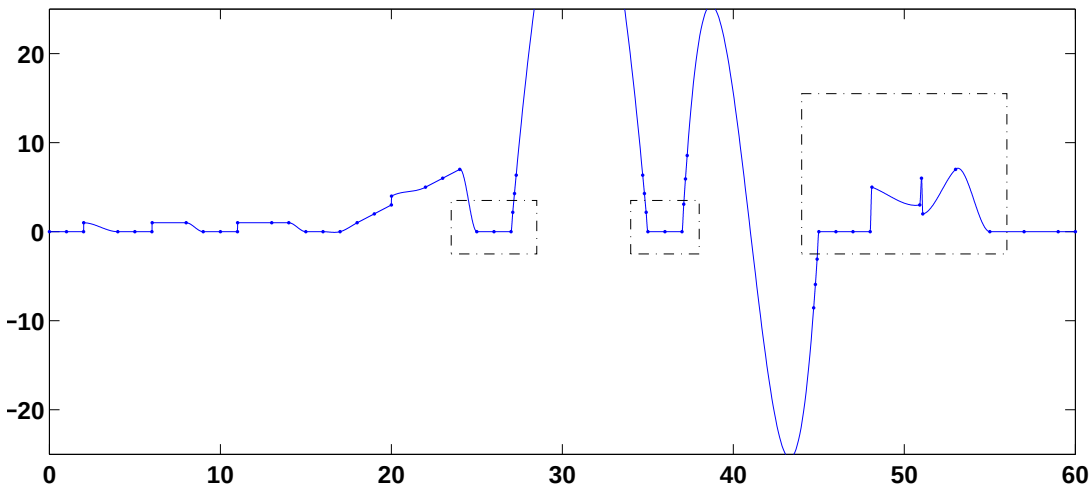



**Figure 7**. First-derivative-based global $L_1$ spline

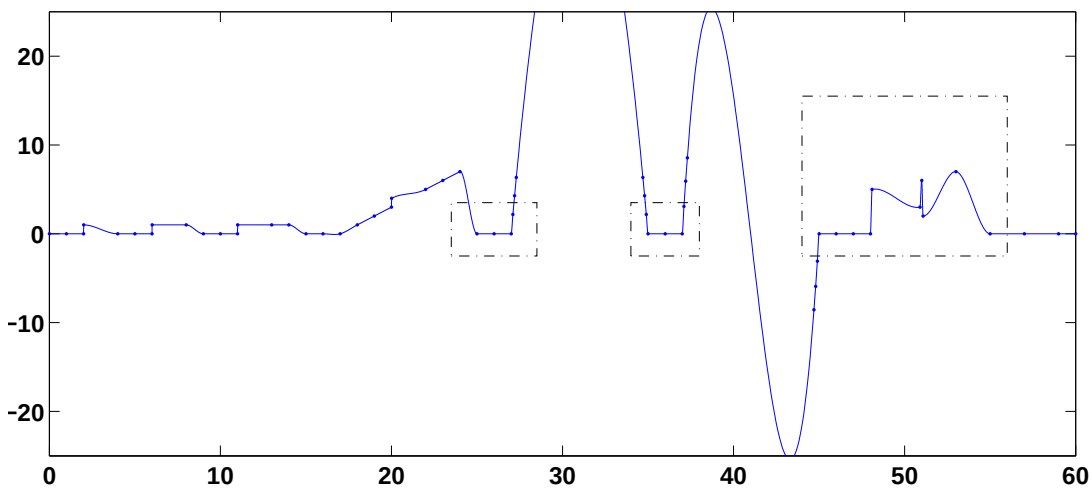


**Figure 8**. Function-value-based 5-point-window $L_1$ spline

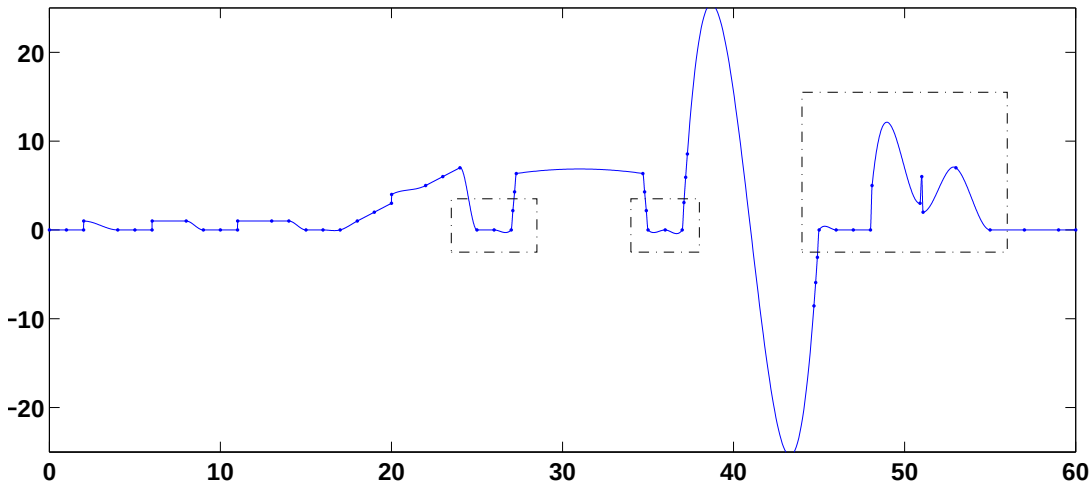



**Figure 9**. Function-value-based 7-point-window $L_1$ spline

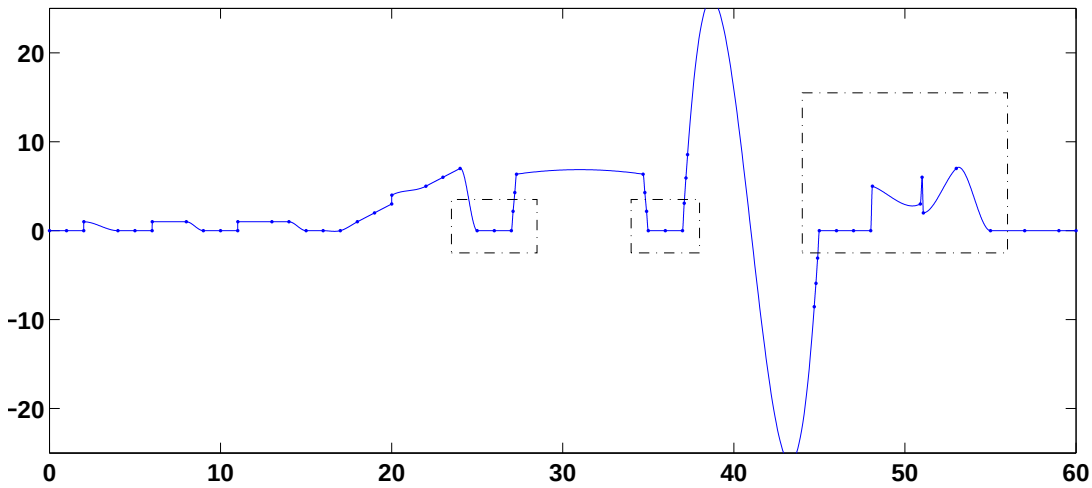



**Figure 10**. Function-value-based global $L_1$ spline

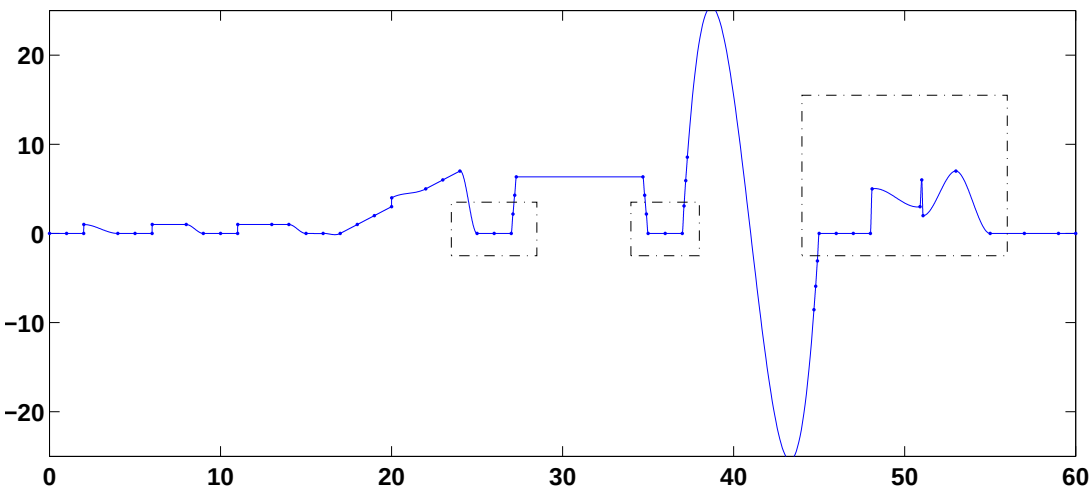


**Figure 11**. Antiderivative-based 5-point-window $L_1$ spline

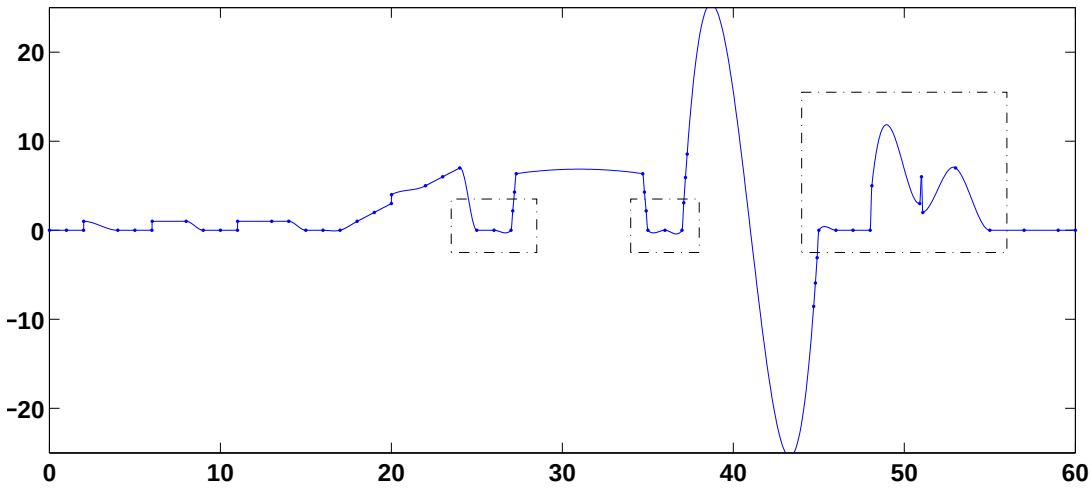



**Figure 12**. Antiderivative-based 7-point-window $L_1$ spline

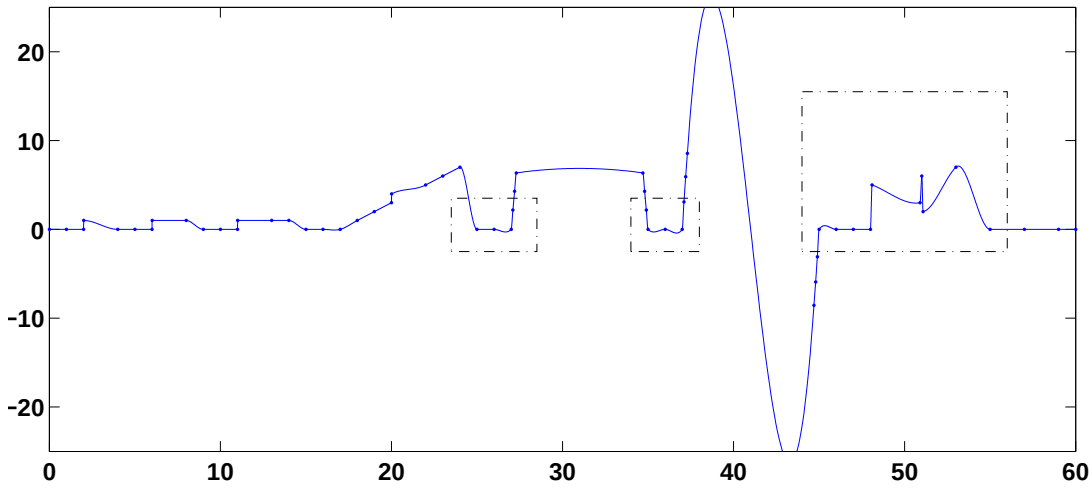
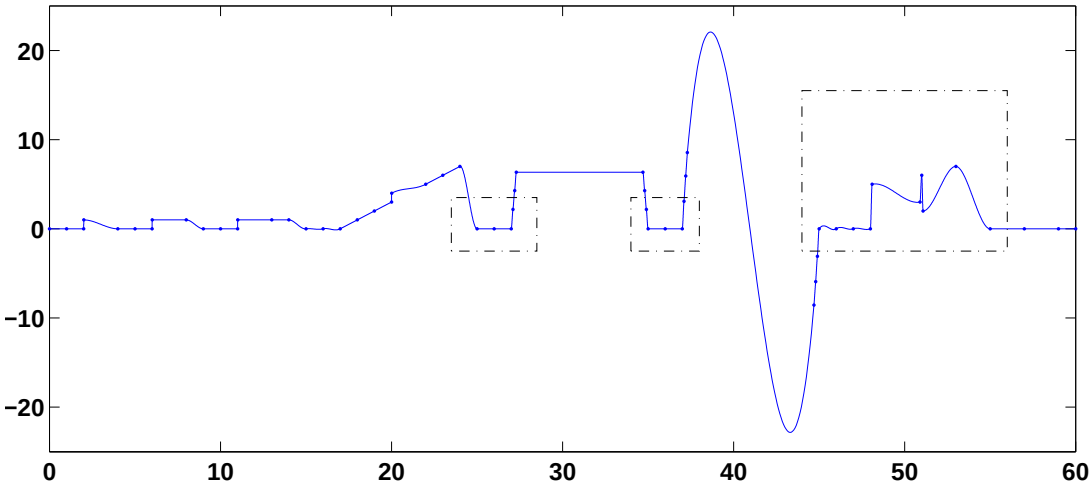



**Figure 13**. Antiderivative-based global $L_1$ spline



We do not claim here that the results presented in Figures 2–13 should yet be broadly interpreted as general descriptions of the shape-preservation capabilities of the various types of $L_1$ splines. However, these results do provide insight into the shape-preservation capabilities of these types of $L_1$ splines as well as a piece of evidence supporting the use of second-derivative-based 5-point-window $L_1$ splines. In these computational results, the performance of second-derivative-based $L_1$ splines, the hitherto most widely used variant of $L_1$ splines is improved by using 5-point windows. The other two "top performers" of Figures 2–13, namely, the first-derivative-based and function-value-based global $L_1$ splines of Figures 7 and 10 are computationally much more expensive than second-derivative-based 5-point-window $L_1$ splines. No theory has been published for function-value-based $L_1$ splines. Elementary theory has been published for first-derivative-based $L_1$ splines but only for the global case [15]. Based on these considerations, we choose the second-derivative-based 5-point-window $L_1$ spline as the $L_1$ spline to be used in the remainder of this paper.

## 4. Algorithm for Minimization of Second-derivative-based 5-point-window Spline Functional

We propose here an algorithm for generation of second-derivative-based 5-point-window $L_1$ splines based on the analytical results of Section 2 of [17]. Recall that, in the interpolation situation of interest in this paper, the task of calculating an $L_1$ spline is the task of calculating the first derivatives $b_i$ by minimization of a spline functional with given nodes $x_i$ and given function values $z_i$ at the nodes. In the $i$th 5-point window, $2 \le i \le I-2$, an objective function $E(b_{i-2}, b_{i-1}, b_i, b_{i+1}, b_{i+2})$ is minimized:

$$\begin{aligned} & \min_{\boldsymbol{b} \in R^5} E(b_{i-2}, b_{i-1}, b_i, b_{i+1}, b_{i+2}) \\ = \; & \min_{\boldsymbol{b}} \left\{ \sum_{j=i-2}^{i+1} \int_{-\frac{1}{2}}^{\frac{1}{2}} |(b_{j+1} - b_j) + 6t(b_j + b_{j+1} - 2\triangle z_j)| \, \mathrm{d}t \right\}. \end{aligned}$$

In [17], minimization of $E(b_{i-2}, b_{i-1}, b_i, b_{i+1}, b_{i+2})$ is treated as a bilevel minimization problem by representing $b_{i-2}$, $b_{i-1}$, $b_{i+1}$ and $b_{i+2}$ as functions of $b_i$. The optimal $b_i$, called $b_i^*$, is determined by solving

$$\min_{\boldsymbol{b}} E(b_{i-2}, b_{i-1}, b_i, b_{i+1}, b_{i+2}) = \min_{b_i} \{G_1(b_i) + G_2(b_i)\}, \tag{10}$$

where

$$G_1(b_i) = \frac{2(\sqrt{10}-1)}{3} |b_{i-1}(b_i) - \triangle z_{i-2}| + \int_{-\frac{1}{2}}^{\frac{1}{2}} |(b_i - b_{i-1}(b_i)) + 6t(b_{i-1}(b_i) + b_i - 2\triangle z_{i-1})| \, \mathrm{d}t \tag{11}$$

and

$$G_2(b_i) = \frac{2(\sqrt{10}-1)}{3} |b_{i+1}(b_i) - \triangle z_{i+1}| + \int_{-\frac{1}{2}}^{\frac{1}{2}} |(b_{i+1}(b_i) - b_i) + 6t(b_i + b_{i+1}(b_i) - 2\triangle z_i)| \, \mathrm{d}t \,. \tag{12}$$

$G_1(b_i)$ and $G_2(b_i)$ have similar structure, which leads to the representation

$$G_1(b_i) + G_2(b_i) = G(b_i - \triangle z_{i-1}; \triangle z_{i-2} - \triangle z_{i-1}) + G(b_i - \triangle z_i; \triangle z_{i+1} - \triangle z_i), \tag{13}$$

where $G(q; c)$ is a function that is defined and analyzed in [17].

The analysis of [17] is based on the signs of $\triangle z_{i-2} - \triangle z_{i-1}$, $\triangle z_{i-1} - \triangle z_i$ and $\triangle z_i - \triangle z_{i+1}$. There are 27 cases, as shown in Table 1. For the simple cases, the optimal $b_i^*$ are listed in the last column of the table. For cases 14, 15, 17, 18, 23, 24, 26 and 27, the $b_i^*$ will be discussed in Algorithm 1.

**Table 1**. 27 cases used in the 5-point window algorithm (see [17])

| Case | Sign of $\triangle z_{i-1} - \triangle z_{i-2}$ | $\triangle z_i - \triangle z_{i-1}$ | $\triangle z_{i+1} - \triangle z_i$ | Same as Case | $b_i^*$ |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | | $\triangle z_{i-1}$ |
| 2 | 0 | 0 | + | | $\triangle z_{i-1}$ |
| 3 | 0 | 0 | – | 2 | $\triangle z_{i-1}$ |
| 4 | 0 | + | 0 | | $\delta_i$ |
| 5 | 0 | + | + | | $\triangle z_{i-1}$ |
| 6 | 0 | + | – | | $\triangle z_{i-1}$ |
| 7 | 0 | – | 0 | 4 | $\delta_i$ |
| 8 | 0 | – | + | 6 | $\triangle z_{i-1}$ |
| 9 | 0 | – | – | 5 | $\triangle z_{i-1}$ |
| 10 | + | 0 | 0 | 2 | $\triangle z_i$ |
| 11 | + | 0 | + | | $\triangle z_{i-1}$ |
| 12 | + | 0 | – | | $\triangle z_{i-1}$ |
| 13 | + | + | 0 | 5 | $\triangle z_i$ |
| 14 | + | + | + | | [see text] |
| 15 | + | + | – | | [see text] |
| 16 | + | – | 0 | 6 | $\triangle z_i$ |
| 17 | + | – | + | | [see text] |
| 18 | + | – | – | 15 | [see text] |
| 19 | – | 0 | 0 | 2 | $\triangle z_i$ |
| 20 | – | 0 | + | 12 | $\triangle z_{i-1}$ |
| 21 | – | 0 | – | 11 | $\triangle z_{i-1}$ |
| 22 | – | + | 0 | 6 | $\triangle z_i$ |
| 23 | – | + | + | 15 | [see text] |
| 24 | – | + | – | 17 | [see text] |
| 25 | – | – | 0 | 5 | $\triangle z_i$ |
| 26 | – | – | + | 15 | [see text] |
| 27 | – | – | – | 14 | [see text] |

Whenever the optimal $b_i$ is non-unique at a node $x_i$, $2 \leq i \leq I-2$, we choose the solution $b_i^*$ to be the real number in the optimal set closest to $\delta_i$, that is, median$\{b_i^u, b_i^l, \delta_i\}$. In all 27 cases, the solution is either uniquely determined by a simple expression or lies inside the interval $(\triangle z_{i-1}, \triangle z_i)$ and can be found by a line search. The expression $G_1(b_i) + G_2(b_i)$ is always continuously differentiable on $(\triangle z_{i-1}, \triangle z_i)$. When line search is required, the solution of (10) is obtained by solving $\frac{\mathrm{d}G_1(b_i)}{\mathrm{d}b_i} + \frac{\mathrm{d}G_2(b_i)}{\mathrm{d}b_i} = 0$. (The explicit form of this expression is provided in [17].) For the results generated in this paper, we used the

secant method (previously used in an analogous way in [3]). Calculation of $b_i^*$ for $i = 0, 1, I-1$ and $I$ is carried out differently, as described in Step 3 of Algorithm 2.

The complete algorithm for calculating second-derivative-based 5-point-window $L_1$ splines consists of a subroutine (Algorithm 1) for the local window calculation embedded in an "outer loop" (Algorithm 2) as described in the remainder of this section. Recall that the quantities $\delta_i$ are defined in (2).

**Algorithm 1** (Subroutine($\triangle z_{i-2}, \triangle z_{i-1}, \triangle z_i, \triangle z_{i+1}; \delta_i$))**.**

*STEP 1* *Calculate $\triangle z_{i-2} - \triangle z_{i-1}$, $\triangle z_{i-1} - \triangle z_i$ and $\triangle z_i - \triangle z_{i+1}$ and determine the case to which these quantities correspond.*

*STEP 2* *Calculate $b_i^*$ using information from [17] contained in Table 1 or described below in this step. Here, $c_1 = \triangle z_{i-2} - \triangle z_{i-1}$ and $c_2 = \triangle z_{i+1} - \triangle z_i$.*

- *In Cases 1, 2, 3, 5, 6, 8, 9, 11, 12, 20 and 21, return $b_i^* = \triangle z_{i-1}$.*
- *In Cases 10, 13, 16, 19, 22 and 25, return $b_i^* = \triangle z_i$.*
- *In Cases 4 and 7, return $b_i^* = \delta_i$.*
- *Case 14:*
  - *Subcase 14-1: If $\triangle z_i - \triangle z_{i-1} \leq \frac{\sqrt{10}-2}{\sqrt{10}}(|c_1| + |c_2|)$, then*

$$b_i^* = \text{median}\{\max\{\triangle z_{i-1}, \triangle z_i + \frac{2-\sqrt{10}}{\sqrt{10}}c_2\}, \min\{\triangle z_{i-1} + \frac{2-\sqrt{10}}{\sqrt{10}}c_1, \triangle z_i\}, \delta_i\}.$$

  - *Subcase 14-2: If $\frac{\sqrt{10}-2}{\sqrt{10}}(|c_1| + |c_2|) < \triangle z_i - \triangle z_{i-1} < \frac{1}{2}(|c_1| + |c_2|)$, then $b_i^*$ lies in the interval*

$$\left[\max\{\triangle z_{i-1} + \frac{2-\sqrt{10}}{\sqrt{10}}c_1, \triangle z_i - \frac{1}{2}c_2\}, \min\{\triangle z_{i-1} - \frac{1}{2}c_1, \triangle z_i + \frac{2-\sqrt{10}}{\sqrt{10}}c_2\}\right].$$

    *Use a line search to find $b_i^*$ such that $\frac{\mathrm{d}G_1(b_i^*)}{\mathrm{d}b_i} + \frac{\mathrm{d}G_2(b_i^*)}{\mathrm{d}b_i} = 0$ in this interval.*
  - *Subcase 14-3: If $\frac{1}{2}(|c_1| + |c_2|) \leq \triangle z_i - \triangle z_{i-1} \leq 2(|c_1| + |c_2|)$, then*

$$b_i^* = \text{median}\{\max\{\triangle z_{i-1} - \frac{1}{2}c_1, \triangle z_i - 2c_2\}, \min\{\triangle z_{i-1} - 2c_1, \triangle z_i - \frac{1}{2}c_2\}, \delta_i\}.$$

  - *Subcase 14-4: If $2(|c_1| + |c_2|) < \triangle z_i - \triangle z_{i-1}$, then $b_i^*$ lies in the interval*

$$[\triangle z_{i-1} - 2c_1, \triangle z_i - 2c_2].$$

    *Use a line search to find $b_i^*$ such that $\frac{\mathrm{d}G_1(b_i^*)}{\mathrm{d}b_i} + \frac{\mathrm{d}G_2(b_i^*)}{\mathrm{d}b_i} = 0$ in this interval.*

  *Return $b_i^*$.*
- *Case 15:*
  - *Subcase 15-1: If $\triangle z_i - \triangle z_{i-1} \leq \frac{2-\sqrt{10}}{\sqrt{10}}c_1$, then $b_i^* = \triangle z_i$.*
  - *Subcase 15-2: If $\frac{2-\sqrt{10}}{\sqrt{10}}c_1 < \triangle z_i - \triangle z_{i-1} \leq -\frac{7+\sqrt{10}}{3}c_1$, then $b_i^* = \triangle z_i$.*

· *Subcase 15-3: If* $-\frac{7+\sqrt{10}}{3}c_1 < \triangle z_i - \triangle z_{i-1}$, *then* $b_i^*$ *lies in the interval*

$$[\triangle z_{i-1} - \frac{7+\sqrt{10}}{3}c_1, \triangle z_i].$$

*Use a line search to find* $b_i^*$ *such that* $\frac{\mathrm{d}G_1(b_i^*)}{\mathrm{d}b_i} + \frac{\mathrm{d}G_2(b_i^*)}{\mathrm{d}b_i} = 0$ *in this interval.*

*Return* $b_i^*$.

- *Case 17:*

· *Subcase 17-1: If* $\triangle z_{i-1} - \triangle z_i > \frac{\sqrt{10}+1}{3}(|c_1| + |c_2|)$, *then* $b_i^*$ *lies in the interval*

$$[\triangle z_i + \frac{\sqrt{10}+1}{3}c_2, \triangle z_{i-1} + \frac{\sqrt{10}+1}{3}c_1].$$

*Use a line search to find* $b_i^*$ *such that* $\frac{\mathrm{d}G_1(b_i^*)}{\mathrm{d}b_i} + \frac{\mathrm{d}G_2(b_i^*)}{\mathrm{d}b_i} = 0$ *in this interval.*

· *Subcase 17-2: If* $\triangle z_{i-1} - \triangle z_i \leq \frac{\sqrt{10}+1}{3}(|c_1| + |c_2|)$, *then*

$$b_i^* = \mathrm{median}\{\max\{\triangle z_i, \triangle z_{i-1} + \frac{\sqrt{10}+1}{3}c_1\}, \min\{\triangle z_{i-1}, \triangle z_i + \frac{\sqrt{10}+1}{3}c_2\}, \delta_i\}.$$

*Return* $b_i^*$.

- *In Case 18, let*

$$(\triangle z'_{i-2}, \triangle z'_{i-1}, \triangle z'_i, \triangle z'_{i+1}, \delta'_i) = (\triangle z_{i+1}, \triangle z_i, \triangle z_{i-1}, \triangle z_{i-2}, \delta_i).$$

*This transforms Case 18 into Case 15. Obtain the optimal solution* $b'_i$ *of the transformed problem. Return* $b_i^* = b'_i$.

- *In Case 23, let*

$$(\triangle z'_{i-2}, \triangle z'_{i-1}, \triangle z'_i, \triangle z'_{i+1}, \delta'_i) = -(\triangle z_{i+1}, \triangle z_i, \triangle z_{i-1}, \triangle z_{i-2}, \delta_i).$$

*This transforms Case 23 into Case 15. Obtain the optimal solution* $b'_i$ *of the transformed problem. Return* $b_i^* = -b'_i$.

- *In Case 24, let*

$$(\triangle z'_{i-2}, \triangle z'_{i-1}, \triangle z'_i, \triangle z'_{i+1}, \delta'_i) = -(\triangle z_{i-2}, \triangle z_{i-1}, \triangle z_i, \triangle z_{i+1}, \delta_i).$$

*This transforms Case 24 into Case 17. Obtain the optimal solution* $b'_i$ *of the transformed problem. Return* $b_i^* = -b'_i$.

- *In Case 26, let*

$$(\triangle z'_{i-2}, \triangle z'_{i-1}, \triangle z'_i, \triangle z'_{i+1}, \delta'_i) = -(\triangle z_{i-2}, \triangle z_{i-1}, \triangle z_i, \triangle z_{i+1}, \delta_i).$$

*This transforms Case 26 into Case 15. Obtain the optimal solution* $b'_i$ *of the transformed problem. Return* $b_i^* = -b'_i$.

- *In Case 27, let*

$$(\triangle z'_{i-2}, \triangle z'_{i-1}, \triangle z'_i, \triangle z'_{i+1}, \delta'_i) = -(\triangle z_{i-2}, \triangle z_{i-1}, \triangle z_i, \triangle z_{i+1}, \delta_i).$$

*This transforms Case 27 into Case 14. Obtain the optimal solution $b'_i$ of the transformed problem. Return $b^*_i = -b'_i$.*

*STEP 3 Stop.*

The "outer loop" (Algorithm 2) consists of repeated or parallel application of Algorithm 1.

**Algorithm 2** (Analysis-based Algorithm)**.**

*STEP 1 Given a set of points $(x_i, z_i)$, $i = 0, 1, \ldots, I$, calculate $\triangle z_i$, $i = 0, \ldots, I-1$, and $\delta_i$, $i = 2, 3, \ldots, I-2$, by formulas (1) and (2).*

*STEP 2 For $i = 2, 3, \ldots, I-2$, calculate $b^*_i$ using Subroutine($\triangle z_{i-2}, \triangle z_{i-1}, \triangle z_i, \triangle z_{i+1}$;$\delta_i$) (Algorithm 1).*

*STEP 3 Set*

$$b^*_1 = \triangle z_1 + \min\{\frac{\sqrt{10}-5}{7-2\sqrt{10}}(b^*_2 - \triangle z_1), \frac{3\sqrt{10}-9}{7-2\sqrt{10}}(b^*_2 - \triangle z_1), \triangle z_0 - \triangle z_1\},$$
$$b^*_0 = \triangle z_0 + \frac{2-\sqrt{10}}{\sqrt{10}}(b^*_1 - \triangle z_1),$$
$$b^*_{I-1} = \triangle z_{I-2} + \min\{\frac{\sqrt{10}-5}{7-2\sqrt{10}}(b^*_{I-2} - \triangle z_{I-2}), \frac{3\sqrt{10}-9}{7-2\sqrt{10}}(b^*_{I-2} - \triangle z_{I-2}), \triangle z_{I-1} - \triangle z_{I-2}\},$$
$$b^*_I = \triangle z_{I-1} + \frac{2-\sqrt{10}}{\sqrt{10}}(b^*_{I-1} - \triangle z_{I-1}).$$

*STEP 4 Stop.*

## 5. Analysis-based Algorithm *vs.* Primal Affine Algorithm for 5-point Windows

In this section, we compare computational results generated by the analysis-based algorithm introduced in Section 4 and the widely used primal affine algorithm described in Section 2. These results differ significantly (by more than $10^{-4}$) at a number of nodes as shown in Table 2. It was confirmed by hand calculation that the results for the analysis-based algorithm are accurate for the original nondiscretized $L_1$ spline functional (a confirmation of both the analysis-based algorithm and the code). The differences in the results produced by the primal affine algorithm and the analysis-based algorithm are due mainly to the discretization (midpoint rule with 100 subintervals, as described in Section 2).

**Remark** On 5-point windows, the primal affine algorithm converges well. However, for global $L_1$ splines on large data sets, the primal affine algorithm can converge slowly or not at all, especially in bivariate situations, as indicated in Section 6 of [9], a conclusion supported by additional unpublished computational experience of the authors. In such situations, the $L_1$ spline functional may result in a nearly degenerate nonlinear program and this near degeneracy is retained in the discretized version.

**Table 2**. $b_i^*$ generated by primal affine algorithm and analysis-based algorithm.

| $i$ | $x_i$ | $b_i^*$ from primal affine algorithm | $b_i^*$ from analysis-based algorithm |
|---|---|---|---|
| 7 | 6.01 | 3.4096 | 3.3874 |
| 29 | 27.2 | 20.9698 | 20.9729 |
| 30 | 27.3 | 19.5166 | 19.5250 |
| 31 | 34.7 | -19.5166 | -19.5250 |
| 32 | 34.8 | -20.9698 | -20.9729 |
| 38 | 37.2 | 27.5971 | 27.6099 |
| 39 | 37.3 | 18.4160 | 18.4667 |
| 40 | 44.7 | 18.4160 | 18.4667 |
| 41 | 44.8 | 27.5971 | 27.6099 |

The analysis-based algorithm introduced in this paper does not require discretization of the $L_1$ spline functional and does not require regularization terms to be added to this functional. The algorithm is simple, accurate and inherently parallelizable. Moreover, it is computationally much cheaper than the primal affine algorithm. Both of the algorithms were implemented in C++ 6.0. Computational results for the data set of Section 1 were generated on an IBM laptop running under the Windows XP operating system at 1.66 GHz CPU with 1.50 GB RAM and were presented in Section 3. The computing times of the primal affine algorithm were

- 177.7 milliseconds for a 5-point window (sequential calculations),
- 335.2 milliseconds for a 7-point window (sequential calculations),
- 94.9 milliseconds for global

for the second-derivative-based $L_1$ splines of Figures 2, 3 and 4, respectively. The computing time of the analysis-based algorithm was

- 0.0531 milliseconds for a 5-point window (sequential calculations)

for the second-derivative-based $L_1$ spline that corresponds to Figure 2. In these computational experiments, the sequential analysis-based algorithm is thus $177.7/0.0531 = 3347$ times faster than the sequential primal affine algorithm. The speed-up for the parallel versions of these algorithms would be roughly the same, since each parallel version would be faster than the corresponding sequential version by a factor roughly equal to the number of points in the data set (56 in this case). The sequential analysis-based algorithm is faster than the global primal affine algorithm by a factor of $94.9/0.0531 = 1787$. The speed-up of the parallel analysis-based algorithm vs. the global primal affine algorithm would, therefore, be roughly a factor of $1787.2 \times 56 \approx 10^5$, an impressive amount. For larger data sets, the speed-up is correspondingly larger.

**Remark** The computing time for the primal affine algorithm could be reduced by decreasing the number of subintervals $K$ for discretization (via the midpoint rule) of the spline functional from 100, as was used for the results generated for this paper, to, say, 10, or less. However, since the degree to which

a discretized spline functional approximates the continuum spline functional decreases as the number of subintervals is decreased, using only a few subintervals is generally not an advantageous choice when accuracy is an issue, as it is here.

## 6. Conclusion

The results presented here indicate that the new, analysis-based algorithm for calculating univariate second-derivative-based $L_1$ interpolating splines on 5-point windows is a highly computationally efficient algorithm for generating a type of $L_1$ spline that has good shape-preservation properties on the micro scale as well as (generally) the macro scale. The analysis-based algorithm produces computational results that are accurate, in contrast to the widely used primal affine algorithm, which produces computational results that are only approximate. The low computing time and the inherently parallelizable nature of the calculations in the analysis-based algorithm are strong advantages.

The extension of the results in the present paper to algorithms for locally calculated univariate $L_1$ approximating splines and for locally calculated bivariate $L_1$ interpolating and approximating splines is a topic for future research. For approximation, either smoothing splines [4,6,8,9,12] or spline fits [8] could be used. In the bivariate case, analytical results for windows consisting of contiguous triangles or rectangles will have to be developed and, on the basis of those analytical results, algorithms analogous to the univariate algorithm introduced in the present paper will need to be created. The success of the algorithm for univariate interpolation of the present paper indicates that further extension in the directions stated here may be fruitful.

## Acknowledgements

The authors wish to thank Olivier Gibaru and Eric Nyiri of the Ecole Nationale Supérieure d'Arts et Métiers de Lille and Philippe Auquiert of the Université de Valenciennes et du Hainaut-Cambrésis for discussions related to the topic of this paper. The reviewers of this paper provided insightful comments and questions that led to improvements in the paper. This work was generously supported by US Army Research Office Grant # W911NF-04-D-0003, the NCSU Edward P. Fitts Fellowship and US NSF Grant # DMI-0553310.

## References

1. Auquiert, P.; Gibaru, O.; Nyiri, E. $C^1$ and $C^2$-continuous polynomial parametric $L_p$ splines ($p \geq 1$). *Comput. Aided Geom. Design* **2007**, *24*, 373–394.
2. Auquiert, P.; Gibaru, O.; Nyiri, E. On the cubic $L_1$ spline interpolant to the Heaviside function. *Numer. Algorithms* **2007**, *46*, 321–332.
3. Cheng, H.; Fang, S.-C.; Lavery, J.E. An efficient algorithm for generating univariate cubic $L_1$ splines. *Comput. Optim. Appl.* **2004**, *29*, 219–253.
4. Cheng, H.; Fang, S.-C.; Lavery, J.E. A geometric programming framework for univariate cubic $L_1$ smoothing splines. *Ann. Oper. Res.* **2005**, *133*, 229–248.
5. Lavery, J.E. Univariate cubic $L_p$ splines and shape-preserving, multiscale interpolation by univariate cubic $L_1$ splines. *Comput. Aided Geom. Design* **2000**, *17*, 319–336.

6. Lavery, J.E. Shape-preserving, multiscale fitting of univariate data by cubic $L_1$ smoothing splines. *Comput. Aided Geom. Design* **2000**, *17*, 715–727.
7. Lavery, J.E. Shape-preserving, multiscale interpolation by bi- and multivariate cubic $L_1$ splines. *Comput. Aided Geom. Design* **2001**, *18*, 321–343.
8. Lavery, J.E. Shape-preserving approximation of multiscale univariate data by cubic $L_1$ spline fits. *Comput. Aided Geom. Design* **2004**, *21*, 43–64.
9. Lavery, J.E. The state of the art in shape preserving, multiscale modeling by $L_1$ splines. Proceedings of SIAM Conference on Geometric Design and Computing, Seattle, WA, USA, November 2003; Lucian, M.L., Neamtu, M., Eds.; Nashboro Press: Brentwood, TN, USA, 2004; pp. 365–376.
10. Lavery, J.E. Shape-preserving, first-derivative-based parametric and nonparametric cubic $L_1$ spline curves. *Comput. Aided Geom. Design* **2006**, *23*, 276–296.
11. Lavery, J.E. Shape-preserving univariate cubic and higher-degree $L_1$ splines with function-value-based and multistep minimization principles. *Comput. Aided Geom. Design* **2009**, *26*, 1–16.
12. Lin, Y.-M.; Zhang, W.; Wang, Y.; Fang, S.-C.; Lavery, J.E. Computationally efficient models of urban and natural terrain by non-iterative domain decomposition with $L_1$ smoothing splines. In Proceedings of the 25th Army Science Conference, Department of the Army, Washington, DC, USA, November 2006.
13. Wang, Y.; Fang, S.-C.; Lavery, J.E. A geometric programming approach for bivariate cubic $L_1$ splines. *Comput. Math. Appl.* **2005**, *49*, 481–514.
14. Wang, Y.; Fang, S.-C.; Lavery, J.E. A compressed primal-dual method for bivariate cubic $L_1$ splines. *Comput. Math. Appl.* **2007**, *201*, 69–87.
15. Zhao, Y.; Fang, S.-C.; Lavery, J.E. Geometric dual formulation for first-derivative-based univariate cubic $L_1$ splines. *J. Global Optim.* **2008**, *40*, 589–621.
16. Auquiert, P.; Gibaru, O.; Nyiri, E. Fast $L_1$–$C^k$ polynomial spline interpolation algorithm with shape-preserving properties. *Comput. Aided Geom. Design* **2010**, in press.
17. Jin, Q.; Lavery, J.E.; Fang, S.-C. Univariate cubic $L_1$ interpolating splines: Analytical results for linearity, convexity and oscillation on 5-point windows. *Algorithms*, **2010**, *3*, 276–293.
18. Bertsekas, D.P.; Nedić, A.; Ozdaglar, A.E. *Convex Analysis and Optimization*; Athena Scientific: Belmont, MA, USA, 2003.