ELSEVIER

Journal of Computational and Applied Mathematics 143 (2002) 145–188

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

An energy-minimization framework for monotonic cubic spline interpolation

George Wolberg *, Itzik Alfy

Department of Computer Science, City College of New York, New York, NY 10031, USA

Received 11 October 2000; received in revised form 3 May 2001

Abstract

This paper describes the use of cubic splines for interpolating monotonic data sets. Interpolating cubic splines are popular for fitting data because they use low-order polynomials and have C^2 continuity, a property that permits them to satisfy a desirable smoothness constraint. Unfortunately, that same constraint often violates another desirable property: monotonicity. It is possible for a set of monotonically increasing (or decreasing) data points to yield a curve that is not monotonic, i.e., the spline may oscillate. In such cases, it is necessary to sacrifice some smoothness in order to preserve monotonicity.

The goal of this work is to determine the *smoothest* possible curve that passes through its control points while simultaneously satisfying the monotonicity constraint. We first describe a set of conditions that form the basis of the monotonic cubic spline interpolation algorithm presented in this paper. The conditions are simplified and consolidated to yield a fast method for determining monotonicity. This result is applied within an energy minimization framework to yield linear and nonlinear optimization-based methods. We consider various energy measures for the optimization objective functions. Comparisons among the different techniques are given, and superior monotonic C^2 cubic spline interpolation results are presented. Extensions to shape preserving splines and data smoothing are described.

1. Introduction

Cubic splines are widely used to fit a smooth continuous function through discrete data. They play an important role in such fields as computer graphics and image processing, where smooth interpolation is essential in modeling, animation, and image scaling. In computer graphics, for instance, interpolating cubic splines are often used to define the smooth motion of objects and cameras passing through user-specified positions in a keyframe animation system. In image processing, splines prove useful in implementing high-quality image magnification.

Cubic splines interpolate (pass through) the data with piecewise cubic polynomials. The use of low-order polynomials is especially attractive for curve fitting because they reduce the computational

* Corresponding author.

E-mail address: wolberg@cs.ccny.cuny.edu (G. Wolberg).

0377-0427/02/$ - see front matter
PII: S0377-0427(01)00506-4

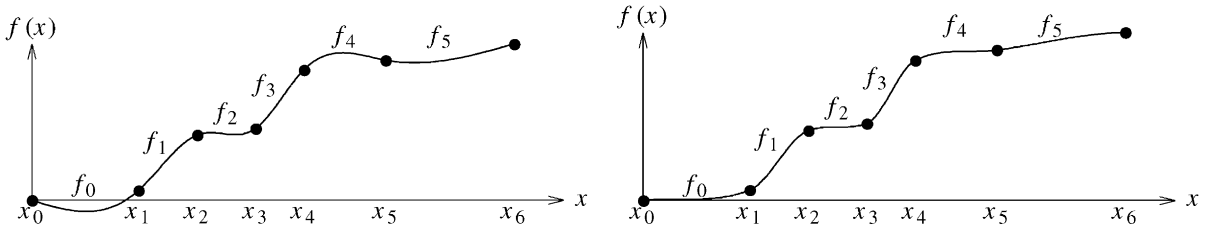

Fig. 1. Interpolating cubic splines.

requirements and numerical instabilities that arise with higher degree curves. These instabilities cause undesirable oscillations when several points are joined in a common curve. Cubic polynomials are most commonly used because no lower-degree polynomial allows a curve to pass through two specified endpoints with specified derivatives at each endpoint. The most compelling reason for their use, though, is their C^2 continuity, which guarantees continuous first and second derivatives across all polynomial segments.

C^2 continuity imposes an intuitive smoothness constraint on the curve. Unfortunately, that same constraint sometimes violates another desirable property: monotonicity. Simply stated, monotonic input data should give rise to an interpolating curve that is smooth *and* monotonic. For instance, consider the interpolating cubic spline passing through the seven marked points in Fig. 1(a). Although the seven data points are monotonically increasing in $f(x_i)$ for $0 \leqslant i \leqslant 6$, the cubic spline is not monotonic: it contains overshoots and undershoots, i.e., wiggles.

The goal of this work is to derive the *smoothest* possible cubic spline that simultaneously interpolates the data and satisfies the monotonicity constraint. Fig. 1(b) shows an example of such a C^2 monotone spline. In cases where the input is not monotonic, the data can be partitioned into consecutive intervals of monotonically increasing and decreasing data. For now, though, we shall limit our attention to one strictly monotonic interval spanning all the points. We begin with a review of the literature in Section 2 and a review of cubic spline interpolation in Section 3. The monotonicity constraint is discussed in Section 4. This paper advances an energy minimizing framework to produce monotone curves. Optimization-based solutions central to this framework are introduced in Section 5. There is a large family of monotone curves that interpolate the data. Section 6 derives bounds on the error between any two such curves. Section 7 demonstrates the monotone curves applied to various data sets. Extensions of the proposed techniques to handle arbitrary data sets with changing monotonicity are presented in Section 8. In addition, extensions to shape-preserving splines, knot insertion, and data smoothing are presented in Section 8 as well. The various methods are compared in Section 9. Finally, a discussion and summary of the work is presented in Sections 10 and 11, respectively. Appendix A derives the monotonicity constraints. Appendix B lists MATLAB code to demonstrate the monotonic cubic spline interpolation algorithm.

2. Previous work

There is a large body of work in the field of monotonic cubic spline interpolation. The earliest work in this area can be traced back to that of Chebyshev [3,2]. His work was motivated by the

need to design a stable governor for a steam engine. Currently, work in this area is motivated by diverse applications in many industrial problems, including CAD/CAM, VLSI, and signal processing. Recent work in this area dates back to Schweikert's work on splines in tension, where exponential splines were used as approximants [27]. Various other exponential and cubic spline interpolants were considered in [29,20–22,9]. Tension parameters were used to control shape. All of these methods were global, interpolatory, and C^2. Automatic algorithms to determine free parameters to control shape and monotonicity were complicated. In [19], an algorithm was presented to generate shape preserving curves of arbitrary smoothness based on the properties of Bernstein polynomials. However, C^2 smoothness required the use of piecewise polynomials whose degree exceeded three. There is also the possibility of using piecewise rational interpolants [10,16], although these are usually only C^1 or are intended for strictly monotone or strictly convex data.

In 1980, Fritsch and Carlson proposed a two-pass algorithm for computing a monotone cubic interpolant [15]. The first pass computes an interpolant using any method of choice. The authors used the standard three-point difference formula, i.e., the Catmull–Rom spline [12]. The second pass visits each interval in sequence and updates the derivative values to satisfy the monotonicity constraint. The algorithm has been shown to yield third-order approximation to a C^3 monotone function [11].

In 1984, Fritsch and Butland proposed a modified technique to simplify the Fritsch–Carlson algorithm [14]. In this method, the first derivatives at the knots are calculated using Brodlie's nonlinear averaging function to give the most visually pleasing results. A rather complete analysis of the essential properties of several nonlinear averaging functions is given in [17]. The Fritsch-Butland technique is available in Netlib (PCHIM.FOR) and can be downloaded from `www.math.iastate.edu/cmlib/pchipd.html`. The Fritsch–Carlson and the Fritsch–Butland algorithms are both local and yield C^1 continuous curves, even if a global C^2 solution exists. Furthermore, there is no flexibility in defining an application's specific properties for the desired spline, e.g., the objective function or constraints for a given optimization problem. Finally, Fritsch–Butland interpolants tend to exhibit high tension, i.e., they are a little "flat" [17].

In [8,7,4,5], several algorithms were proposed to compute shape preserving splines that are monotone and convex. The algorithms are based on [15] and iteratively compute a set of first derivatives that simultaneously satisfy the monotonicity and convexity constraints.

Schumaker [26] proposed a shape preserving interpolation algorithm to produce a C^1 quadratic spline with additional knots where necessary. The algorithm is interactive, and the user has flexibility in adjusting the shape of the interpolating spline under some relationship rules.

Several researchers have investigated other approaches involving the use of additional knots between data points [9,22,6,2,23]. If two extra break points are allowed between each data subinterval, then there are enough degrees of freedom to construct a globally C^2 cubic spline interpolant which is local and which has slopes and curvatures at the data points as free parameters [23]. Additional breakpoints, however, require more storage and increased search time during evaluation [15].

This paper presents several key enhancements beyond the work described in [31]. It presents a more efficient solution due to the use of the Hermite representation of cubic splines. As a result, fewer unknowns and constraints need to be solved and applied, respectively. We also extend the results to handle data of changing monotonicity, shape preserving splines, knot insertion, and data smoothing.

3. Cubic splines: a review

A cubic spline $f(x)$ interpolating on the partition $x_0 < x_1 < \cdots < x_{n-1}$ is a function for which $f(x_k) = y_k$. It is a piecewise polynomial function that consists of $n-1$ cubic polynomials f_k defined on the ranges $[x_k, x_{k+1}]$. Furthermore, each f_k is joined at x_k, for $k = 1, \ldots, n-2$, such that $y'_k = f'(x_k)$ and $y''_k = f''(x_k)$ are continuous. An example of a cubic spline passing through $n = 7$ data points is illustrated in Fig. 1.

The kth polynomial curve, f_k, is defined over the fixed interval $[x_k, x_{k+1}]$ and has the cubic form

$$f_k(x) = a_k(x - x_k)^3 + b_k(x - x_k)^2 + c_k(x - x_k) + d_k, \tag{1}$$

where

$$a_k = \frac{1}{\Delta x_k^2}\left(-2\frac{\Delta y_k}{\Delta x_k} + y'_k + y'_{k+1}\right), \tag{2a}$$

$$b_k = \frac{1}{\Delta x_k}\left(3\frac{\Delta y_k}{\Delta x_k} - 2y'_k - y'_{k+1}\right), \tag{2b}$$

$$c_k = y'_k, \tag{2c}$$

$$d_k = y_k. \tag{2d}$$

In the expressions for a_k and b_k, $\Delta x_k = x_{k+1} - x_k$ and $\Delta y_k = y_{k+1} - y_k$, for $k = 0, \ldots, n-2$.

The expressions for the cubic polynomial coefficients in Eq. (2) are given in terms of position data and derivatives. In cases where only position data is supplied, the derivative values may be evaluated by solving a tridiagonal system of equations that relate the unknown derivatives to the known position data. Derivations can be found in [25,30].

The role of position data and derivatives in cubic splines can be made explicit by rewriting Eq. (1) as

$$f_k(x) = H_0\left(\frac{x - x_k}{\Delta x_k}\right) y_k + H_1\left(\frac{x - x_k}{\Delta x_k}\right) y_{k+1} + \Delta x H_2\left(\frac{x - x_k}{\Delta x_k}\right) y'_k + \Delta x H_3\left(\frac{x - x_k}{\Delta x_k}\right) y'_{k+1}, \tag{3}$$

where H_0, H_1, H_2, and H_3 are the cubic Hermite basis functions [12], defined over $0 \leqslant u \leqslant 1$:

$$H_0(u) = 2u^3 - 3u^2 + 1, \tag{4}$$

$$H_1(u) = -2u^3 + 3u^2, \tag{5}$$

$$H_2(u) = u^3 - 2u^2 + u, \tag{6}$$

$$H_3(u) = u^3 - u^2. \tag{7}$$

The Hermite basis functions are derived directly from Eq. (2) by rearranging terms to find the weights associated with y_k, y_{k+1}, y'_k, and y'_{k+1}. In this manner, the Hermite expression for cubic

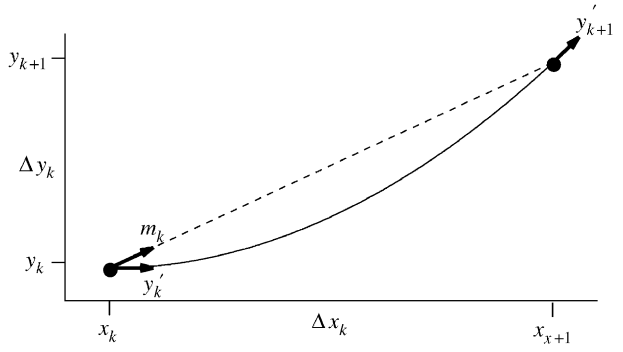

Fig. 2. A single cubic polynomial segment.

curves explicitly represents the function as a linear combination of position and derivative values. In contrast, Eq. (1) represented the cubic curve as a linear combination of powers of x with the position and derivate values embedded in the coefficients.

Fig. 2 depicts a cubic polynomial segment that is fully specified with four constraints: position vectors (x_k, y_k) and (x_{k+1}, y_{k+1}), and derivatives y'_k and y'_{k+1}. The segment passes through the two endpoints, and the derivatives at both ends are depicted with bold tangent vectors. A dashed line with derivative (slope) $m_k = \Delta y_k/\Delta x_k$ at both endpoints is shown as well.

To make the derivatives invariant to scale change, we shall find it useful to relate the derivatives in terms of slope m_k:

$$y'_k = \alpha_k m_k, \tag{8a}$$

$$y'_{k+1} = \beta_k m_k \tag{8b}$$

for $\alpha_k \geqslant 0$ and $\beta_k \geqslant 0$. The cubic curve in Fig. 2 was generated using $\alpha_k = 0$ and $\beta_k = 2$.

Although the user-supplied data points are fixed, the derivatives can be changed to yield a large family of interpolating cubic splines. We are interested in determining the range of derivative values for which the spline remains monotonic. To motivate the need for determining this range of derivative values, Fig. 3 shows a set of five cubic curves, each with derivative $y'_k = 0$ and increasing values for y'_{k+1}. In particular, $\alpha_k = 0$ and $1 \leqslant \beta_k \leqslant 5$ for integer values of β_k. Tangent vectors are shown for the $\beta_k = 1$ and $\beta_k = 5$ cases. Note that the monotonic constraint is violated when $\beta_k > 3$, i.e., a local minima is present in the span.

4. Monotonicity

In this section, we consider a single cubic polynomial $f_k(x)$ in the interval $[x_k, x_{k+1}]$ and derive necessary and sufficient conditions for which $f_k(x)$ is monotonic in the interval. These conditions form the basis of the monotonic cubic spline interpolation algorithm presented in this paper. The conditions derived below closely follow that of [15] and are reviewed here to make the presentation self-contained. These conditions are further simplified here to yield a fast method for determining monotonicity.

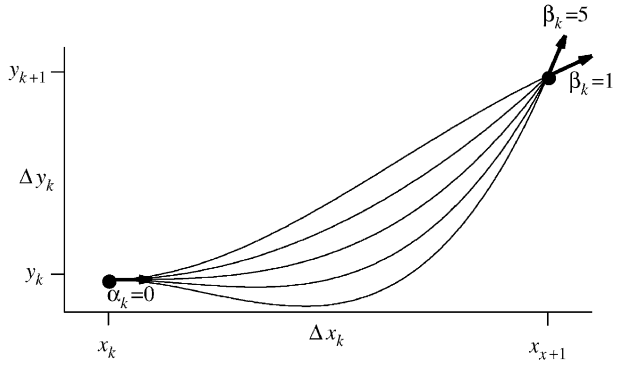

Fig. 3. A family of interpolating cubic polynomials.

A curve is monotonic in an interval $[x_k, x_{k+1}]$ if and only if there is no sign change in the derivative value along any part of the curve in the interval. Therefore, a necessary condition for monotonicity is that

$$\operatorname{sgn}(y'_k) = \operatorname{sgn}(y'_{k+1}) = \operatorname{sgn}(m_k). \tag{9}$$

Furthermore, if $m_k = 0$, then $f_k(x)$ is monotone (constant) in the interval if and only if $y'_k = y'_{k+1} = 0$.

In the remainder of the presentation, we will assume that $m_k \neq 0$ and that Eq. (9) is satisfied. As a result, $f_k(x)$ is strictly monotonic in the interval $[x_k, x_{k+1}]$ if $f'_k(x) \neq 0$ for $x_k \leqslant x \leqslant x_{k+1}$. This implies that there are no local extrema (minima/maxima) in that span.

Since the problem of determining monotonicity is translation-invariant, the x-coordinates can be shifted so that $x_k = 0$. Furthermore, without loss of generality, both Δx_k and Δy_k can be divided by Δx_k to yield $\Delta x_k = 1$ and $\Delta y_k = m_k$. Substituting these expressions into Eq. (1) yields the following cubic curve between (x_k, y_k) and (x_{k+1}, y_{k+1}):

$$f_k(x) = a_k x^3 + b_k x^2 + c_k x + d_k, \tag{10}$$

where

$$a_k = -2m_k + \alpha_k m_k + \beta_k m_k, \tag{11a}$$

$$b_k = 3m_k - 2\alpha_k m_k - \beta_k m_k, \tag{11b}$$

$$c_k = \alpha_k m_k, \tag{11c}$$

$$d_k = y_k. \tag{11d}$$

Note that the interval of interest here is [0,1] since $x_k = 0$ and Δx has been normalized to 1. The expressions for the first and second derivatives are:

$$f'_k(x) = 3a_k x^2 + 2b_k x + c_k, \tag{12}$$

$$f''_k(x) = 6a_k x + 2b_k. \tag{13}$$

We now consider several cases to determine necessary and sufficient conditions for monotonicity.

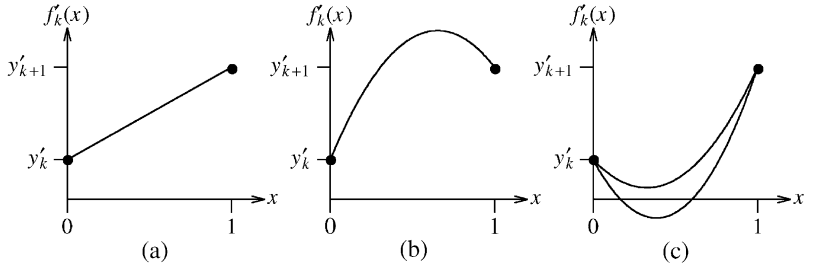

Fig. 4. Three $f'_k(x)$ cases: (a) linear; (b) concave down; and (c) concave up.

4.1. Case 1: $a_k = 0 \rightarrow \alpha_k + \beta_k - 2 = 0$

If $a_k = 0$, then $f_k(x)$ is quadratic (or linear) and $f'_k(x)$ is linear (or constant). Since $f'_k(x)$ constitutes the line between y'_k and y'_{k+1}, no sign change in $f'_k(x)$ is possible and Eq. (9) is a sufficient condition for monotonicity. This case is depicted in Fig. 4(a).

4.2. Case 2: $a_k \neq 0 \rightarrow \alpha_k + \beta_k - 2 \neq 0$

If $a_k \neq 0$, then $f'_k(x)$ is quadratic. If endpoints y'_k and y'_{k+1} of the quadratic are positive, the quadratic function $f'_k(x)$ is guaranteed to remain positive along the entire interval if the curve is concave down. This condition is met if $a_k < 0$. The opposite is true if y'_k and y'_{k+1} are negative. Therefore, if $\alpha_k + \beta_k - 2 < 0$ and Eq. (9) is satisfied, $f_k(x)$ is monotone. This case is depicted in Fig. 4(b). Note that we can accommodate the monotonic increasing and decreasing cases in a single condition by dividing a_k by m_k, where $m_k \neq 0$.

In the event that $a_k > 0$, the function $f'_k(x)$ is concave up and $f_k(x)$ may or may not be monotone. Fig. 4(c) illustrates two concave upward functions. The strictly positive $f'_k(x)$ function corresponds to a monotone $f_k(x)$. The other function corresponds to a non-monotonic $f_k(x)$.

We may distinguish between the two cases in Fig. 4(c) by finding the local minima of $f'_k(x)$. This is derived by solving for x^* in $f''_k(x^*) = 0$:

$$6a_k x^* + 2b_k = 0, \tag{14a}$$

$$3(\alpha_k + \beta_k - 2)x^* = 2\alpha_k + \beta_k - 3, \tag{14b}$$

$$x^* = \frac{2\alpha_k + \beta_k - 3}{3(\alpha_k + \beta_k - 2)}. \tag{14c}$$

The concave upward $f'_k(x)$ function is associated with a monotone $f_k(x)$ function if and only if it satisfies any of the following conditions:

(1) $x^* < 0$,
(2) $x^* > 1$,
(3) $0 < x^* < 1$ and $\operatorname{sgn}(f'_k(x^*)) = \operatorname{sgn}(m_k)$.

Conditions (1) and (2) imply that any sign change in $f'_k(x)$ takes place outside the normalized interval of interest, i.e, the function is monotone in the [0,1] interval. The two conditions can be

written as $2\alpha_k + \beta_k - 3 \leqslant 0$ and $\alpha_k + 2\beta_k - 3 \leqslant 0$, respectively. Condition (3) corresponds to the monotone case depicted in the strictly positive function in Fig. 4(c). We may write this condition as follows:

$$f'_k(x^*) = \left(\frac{3a_k(x^*)^2 + 2b_k x^* + c_k}{m_k} \right) m_k, \tag{15}$$

$$= \left(\alpha_k - \frac{(2\alpha_k + \beta_k - 3)^2}{3(\alpha_k + \beta_k - 2)} \right) m_k. \tag{16}$$

In order for $f'_k(x^*)$ to retain the same sign as m_k,

$$\alpha_k - \frac{(2\alpha_k + \beta_k - 3)^2}{3(\alpha_k + \beta_k - 2)} \geqslant 0. \tag{17}$$

Expanding Eq. (17) yields

$$\alpha_k^2 + \alpha_k(\beta_k - 6) + (\beta_k - 3)^2 \leqslant 0. \tag{18}$$

The same expression can be derived by computing $f'_k(x) = 0$ in [0,1]:

$$f'(x) = 3a_k x^2 + 2b_k x + c_k = 0$$

$$= Ax^2 + Bx + C = 0. \tag{19}$$

The solution for x in the quadratic expression of Eq. (19) is

$$x = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}. \tag{20}$$

No solution exists if $2A = 0$ or $B^2 < 4AC$:

$$2A = 0 \rightarrow 3(\alpha_k + \beta_k) - 6 = 0$$

$$\rightarrow \alpha_k + \beta_k = 2, \tag{21}$$

$$B^2 < 4AC \rightarrow 16\alpha_k^2 + 4\beta_k^2 - 48\alpha_k - 24\beta_k + 16\alpha_k b_k + 36 < 12\alpha_k^2 - 24\alpha_k + 12\alpha_k\beta_k$$

$$\rightarrow \alpha_k^2 + \alpha_k(\beta_k - 6) + (\beta_k - 3)^2 < 0. \tag{22}$$

Notice that the expression in Eq. (21) is identical to the $a_k = 0$ case given in Section 4.1, and the expression in Eq. (22) is identical to that of Eq. (18). The fact that no solution exists for the above expressions implies that no local extrema is present in the $[0, 1]$ interval.

4.3. Monotonicity conditions

The monotonicity constraints derived above can be summarized by the following two lemmas:

(1) If $\alpha_k + \beta_k - 2 \leqslant 0$, then $f_k(x)$ is monotone if and only if Eq. (9) is satisfied.
(2) If $\alpha_k + 2\beta_k - 2 > 0$, then $f_k(x)$ is monotone if and only if Eq. (9) *and* one of the following conditions is satisfied:

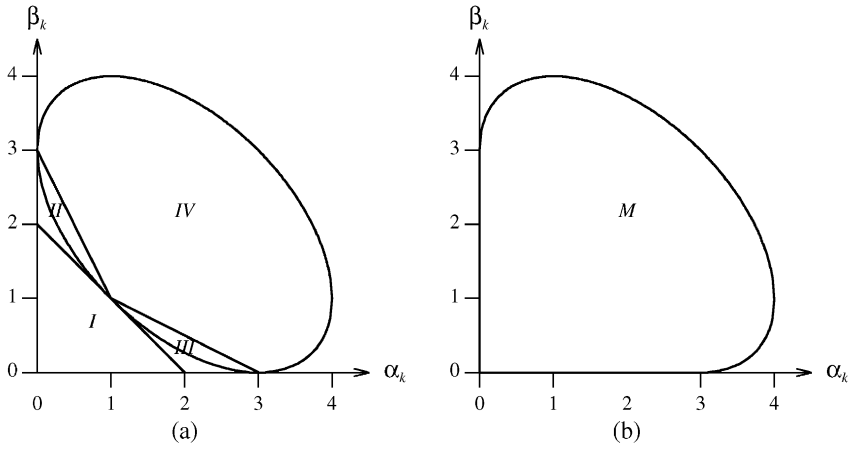

Fig. 5. (α, β) pairs for a monotonic curve.

(a) $2\alpha_k + \beta_k - 3 \leqslant 0$,
(b) $\alpha_k + 2\beta_k - 3 \leqslant 0$,
(c) $\alpha_k^2 + \alpha_k(\beta_k - 6) + (\beta_k - 3)^2 < 0$.

Conditions (1), (2a), (2b), and (2c) are depicted graphically as regions I, II, III, and IV in Fig. 5(a). The union of these regions, shown in Fig. 5(b), is bounded by lines $\alpha_k = 0$, $\beta_k = 0$, and the ellipse. A simple expression for this region is derived below:

$$\begin{aligned}
&\alpha_k^2 + \alpha_k(\beta_k - 6) + (\beta_k - 3)^2 < 0,\\
&(\alpha_k + \beta_k)^2 - 6(\alpha_k + \beta_k) + 9 - \alpha_k\beta_k < 0,\\
&[(\alpha_k + \beta_k) - 3]^2 < \alpha_k\beta_k,\\
&\alpha_k + \beta_k - 3 < \sqrt{\alpha_k\beta_k},\\
&\alpha_k + \beta_k < 3 + \sqrt{\alpha_k\beta_k}.
\end{aligned} \tag{23}$$

Eq. (23) defines the full ellipse of region IV. The monotonicity region M in Fig. 5(b) is expressed in terms of this result as follows:

$$M = \begin{cases} 0 < \alpha_k + \beta_k < 3 + s & \text{if } 0 \leqslant \beta_k \leqslant 3,\\ 3 - s < \alpha_k + \beta_k < 3 + s & \text{if } 3 < \beta_k \leqslant 4, \end{cases} \tag{24}$$

where $s = \sqrt{\alpha_k\beta_k}$. All points in region M denote valid (α_k, β_k) pairs that preserve monotonicity.

4.4. Greedy algorithm

A reasonable approach for generating a smooth monotonic curve is to apply the standard cubic spline interpolation algorithm [30] to the data and visit each interval to verify if it will produce a

monotonic segment. The interpolation algorithm will compute the y'_k derivatives at each knot. We evaluate α_k and β_k for each interval by dividing y'_k at each knot by the slope of the interval. If Eq. (24) is satisfied, the interval is monotone and we can leave the computed y'_k derivative alone. Otherwise, y'_k must be altered to arrive at an (α_k, β_k) pair that lies in the monotonicity region M illustrated in Fig. 5(b). To do this, we refer to the equation of an ellipse given in Eq. (22) to find (α_k, β_k) pairs that yield no solution to $f'_k(x) = 0$. For a given β_k, we solve for α_k in Eq. (22). Setting that quadratic expression to 0 yields the following limits for α_k:

$$\begin{aligned}\alpha_k &= \frac{-(\beta_k - 6) \pm \sqrt{(\beta_k - 6)^2 - 4(\beta_k - 3)^2}}{2} \\ &= \frac{-(\beta_k - 6) \pm \sqrt{\beta_k^2 - 12\beta_k + 36 - 4\beta_k^2 + 24\beta_k - 36}}{2} \\ &= \frac{-(\beta_k - 6) \pm \sqrt{3\beta_k(4 - \beta_k)}}{2}. \end{aligned} \tag{25}$$

A solution exists for α_k as long as $0 \leqslant \beta_k \leqslant 4$:

$$\alpha_{\min} \leqslant \alpha_k \leqslant \alpha_{\max} \tag{26}$$

where

$$\alpha_{\min} = \begin{cases} 0 & \text{if } 0 \leqslant \beta_k \leqslant 3, \\ \dfrac{-(\beta_k - 6) - \sqrt{3\beta_k(4 - \beta_k)}}{2} & \text{if } 3 < \beta_k < 4, \\ 1 & \beta_k > 4. \end{cases}$$

and

$$\alpha_{\max} = \begin{cases} \dfrac{-(\beta_k - 6) + \sqrt{3\beta_k(4 - \beta_k)}}{2} & \text{if } 0 < \beta_k < 4, \\ 1 & \beta_k > 4. \end{cases}$$

Therefore, we clamp α_k to the range $[\alpha_{\min}, \alpha_{\max}]$ when $\beta_k < 4$. If $\beta_k > 4$, then we clamp β_k as well as α_k to $(\alpha_k, \beta_k) = (1, 4)$.

Updating an (α_k, β_k) pair for interval k will alter neighboring intervals. Consequently, the updating process is reserved for the interval k whose (α_k, β_k) values lie furthest outside monotonicity region M. Only that interval is "fixed" and the standard cubic spline interpolation algorithm is re-applied to the remaining intervals of the curve. This algorithm is greedy in the sense that it sequentially attempts to remedy the problem by patching up the most offending interval, one at a time. It is possible that clamping the (α_k, β_k) values in one interval can fix problems that had existed in other intervals. Conversely, it can also introduce problems in neighboring intervals, where none may have existed before. With each pass, though, the standard cubic spline interpolation algorithm is applied to ever-smaller data sets since the derivatives fixed in previous iterations remain fixed throughout the remainder of the processing. The algorithm iterates until all intervals are found to be monotonic.

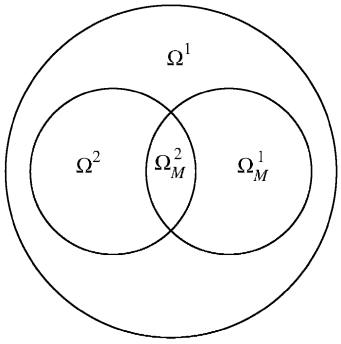

Fig. 6. Relationship between the cubic spline domains.

It is important to note that by fixing the derivative values in one interval of a C^2 curve, we have introduced a discontinuity in the second derivative. Implicit in this statement is the fact that the boundary conditions remained fixed. If the boundary conditions were allowed to be free, then it is possible that the modified curve could have remained C^2. However, optimization to solve for the boundary conditions would be required in this instance. In the absence of this optimization, the greedy algorithm produces C^1 curves. The greedy algorithm thereby iteratively breaks an initial C^2 curve at the offending interval, introduces boundary conditions there, and reapplies a C^2 fit on the remaining subcurves. The final C^1 curve is therefore a composite of C^2 subcurves.

4.5. Discussion

The presentation in this section has focused on monotonicity conditions for a single cubic polynomial $f_k(x)$ in the interval $[x_k, x_{k+1}]$. The suboptimal greedy algorithm used those conditions to iteratively generate a monotone C^1 curve. The most desirable solution will require that all intervals be smoothly tied together satisfying C^2 continuity. This is a global problem that will require optimization to determine the unknown derivative values at the data points to yield the smoothest monotone cubic spline.

Let Ω^i and Ω_M^i denote the domains of C^i cubic splines and C^i monotone cubic splines, respectively. The relationship between these domains may be given as

$$\begin{aligned} &\Omega_M^i \subset \Omega^i, && i = 0, 1, 2, \\ &\Omega^i \subset \Omega^{i-1}, && i = 1, 2, \\ &\Omega_M^i \subset \Omega_M^{i-1}, && i = 1, 2, \\ &\Omega^i \not\subset \Omega_M^{i-1}, && i = 1, 2. \end{aligned} \tag{27}$$

The last relation implies that there are C^2 solutions that do not yield monotone C^1 curves. These relationships are graphically depicted in Fig. 6 for $i = 2$.

For some data sets, it is possible that no monotone C^2 solution exists, i.e., $\Omega_M^2 = \emptyset$. Note that it is always possible to achieve a monotone C^1 solution, i.e., $\Omega_M^1 \neq \emptyset$, because we can always

force $y'_k = 0$ at all data points. Therefore, we will only consider using the best monotone C^1 solution if no C^2 solution exists.

5. Optimization-based solutions

In this section, we consider several solutions to the monotonic interpolation problem based on optimization techniques. We will investigate solutions derived by linear and quadratic programming techniques subject to various constraints on first and second derivative continuity.

The objective criterion for the optimization techniques will be based on energy measures of the curves. We begin with a review of the classic cubic spline energy measure in Section 5.1. Since the original cubic spline formulation is not guaranteed to be monotonic, we will impose the monotonicity constraint in Section 5.2. This will furnish a solution that may be solved using quadratic programming. That result will be further simplified in Section 5.3 to yield a solution that may be solved using linear programming. Since monotonic cubic splines are not guaranteed to be C^2 continuous, a new energy measure is introduced in Section 5.5 that addresses the extent of second derivative discontinuity in the spline. That result is further simplified in Section 5.6 to yield a solution that may be solved using linear programming.

5.1. Spline energy

Cubic splines originally arose as a mathematical model for a draftman's spline. Cubic splines mimic the position of a flexible thin beam that is forced to pass through the given data points [18,9]. The strain energy of the beam is given as the integral of the curvature

$$E = \int_{x_0}^{x_{n-1}} \frac{f''(x)^2}{(1 + [f'(x)]^2)^{5/2}}\, \mathrm{d}x. \tag{28}$$

The *elastica* is the ideal interpolating spline, i.e., the function $f(x)$ that minimizes E. Although any interpolating function that minimizes E is known as the elastica, we shall be interested in the C^2 cubic spline elastica (CSE) that minimizes E. Due to the inherent difficulty in solving for $f(x)$ under this formulation, a simpler linearized energy measure is commonly used [28,9,14,13]:

$$E_{\mathrm{L}} = \int_{x_0}^{x_{n-1}} f''(x)^2\, \mathrm{d}x. \tag{29}$$

This expression is valid only if one makes the simplifying assumption that $f'(x)^2 \ll 1$ everywhere. Despite the fact that this assumption is often violated in practice, it is widely used since it facilitates a computationally tractable solution for minimizing Eq. (28). The E_{L} energy measure given in Eq. (29) is often coupled with the free-end (FE) boundary condition $f''(x_0) = f''(x_{n-1}) = 0$ to produce the *natural spline*. It has been shown that the FE boundary condition minimizes Eq. (29) among all C^2 cubic polynomials [9,28].

5.2. Linearized energy (LE_QP)

The expression for E_{L} may be written in terms of the first derivatives of Eq. (3) as follows:

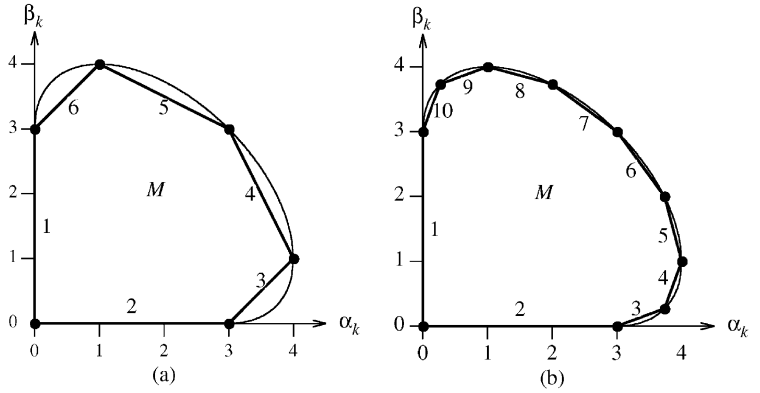

Fig. 7. Linear approximation of region M: (a) $n=6$; (b) $n=10$.

$$E_{\mathrm{L}} = \sum_{k=0}^{n-2} \int_{x=x_k}^{x_{k+1}} f_k''(x)^2 \, \mathrm{d}x$$

$$= \sum_{k=0}^{n-2} \int_{x=x_k}^{x_{k+1}} \frac{1}{\Delta x_k^2} \left[H_0'' \left(\frac{x-x_k}{\Delta x_k} \right) y_k + H_1'' \left(\frac{x-x_k}{\Delta x_k} \right) y_{k+1} + \Delta x H_2'' \left(\frac{x-x_k}{\Delta x_k} \right) y_k' \right.$$

$$\left. + \Delta x H_3'' \left(\frac{x-x_k}{\Delta x_k} \right) y_{k+1}' \right]^2 \mathrm{d}x, \tag{30}$$

where

$$H_0''(u) = 12u - 6, \tag{31}$$

$$H_1''(u) = -12u + 6, \tag{32}$$

$$H_2''(u) = 6u - 4, \tag{33}$$

$$H_3''(u) = 6u - 2. \tag{34}$$

The solution to the problem of minimizing E_{L} over all possible first derivatives is not guaranteed to preserve monotonicity. We may address this problem by adding linear monotonicity constraints. Fig. 5 shows the valid range of values for α_k and β_k to yield a monotonic curve segment. We may obtain linear constraints by approximating the closed region in Fig. 5 with an n-sided polygon. In the example below, we use $n=6$ and $n=10$ to demonstrate our approach.

Fig. 7 illustrates the two polygons we used to approximate region M in our work. The 6- and 10-sided polygons shown in Fig. 7 cover 90.53% and 97.53% of region M, respectively. The 6-sided polygon depicted in Fig. 7(a) consists of the intersection of the following six half-planes:

$$\alpha_k \geqslant 0, \tag{35a}$$

$$\beta_k \geqslant 0, \tag{35b}$$

$$\beta_k - \alpha_k + 3 \geqslant 0, \tag{35c}$$

$$\beta_k + 2\alpha_k - 9 \leqslant 0, \tag{35d}$$

$$2\beta_k + \alpha_k - 9 \leqslant 0, \tag{35e}$$

$$\beta_k - \alpha_k - 3 \leqslant 0, \tag{35f}$$

where $\alpha_k = y'_k/m_k$, and $\beta_k = y'_{k+1}/m_k$. Expressing these results in terms of the unknown first derivatives we have the following six monotonicity constraints:

$$\operatorname{sign}(m_k) \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & -1 \\ -1 & 1 \\ 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} y'_k \\ y'_{k+1} \end{bmatrix} \leqslant |m_k| \begin{bmatrix} 0 \\ 0 \\ 3 \\ 3 \\ 9 \\ 9 \end{bmatrix}. \tag{36}$$

Note that Eq. (36) can be applied to any increasing or decreasing data interval. In order to retain the same inequality direction for either case, we factored out the sign of m_k in Eq. (36).

A general method for computing the boundary of M is given in Appendix A. We use that method to derive the set of inequalities that constitute the monotonicity constraints. The constraints for the $n = 10$ case are given below:

$$\operatorname{sign}(m_k) \begin{bmatrix} -1.0000 & 0 \\ 0.0000 & -1 \\ 0.3660 & -1 \\ 2.7317 & -1 \\ 3.7313 & 1 \\ 1.3661 & 1 \\ 0.7320 & 1 \\ 0.2680 & 1 \\ -0.3661 & 1 \\ -2.7324 & 1 \end{bmatrix} \begin{bmatrix} y'_k \\ y'_{k+1} \end{bmatrix} \leqslant |m_k| \begin{bmatrix} 0.0000 \\ 0.0000 \\ 1.0980 \\ 9.9269 \\ 15.9254 \\ 7.0984 \\ 5.1960 \\ 4.2680 \\ 3.6339 \\ 3.0000 \end{bmatrix}. \tag{37}$$

Since E_L is quadratic in y'_k, and Eq. (36) and Eq. (37) are linear in y'_k, we can use quadratic programming to minimize E_L subject to the following constraints:

1. Second derivative continuity: $f''(x_k^+) = f''(x_k^-)$ (Eq. (38)).
2. Monotonicity constraints: Eqs. (36) or (37).

The second derivative continuity constraint can be expressed as follows:

$$-y'_{k-1}\frac{1}{\Delta x_{k-1}} - y'_k\left[\frac{2}{\Delta x_k} + \frac{2}{\Delta x_{k-1}}\right] - y'_{k+1}\frac{1}{\Delta x_k} - y_{k-1}\frac{3}{\Delta x_{k-1}^2} + y_k\left[\frac{3}{\Delta x_{k-1}^2} - \frac{3}{\Delta x_k^2}\right] + y_{k+1}\frac{3}{\Delta x_k^2} = 0. \tag{38}$$

Interpolation and first derivative continuity constraints are not required since they are implicit in the cubic Hermite form of Eq. (3).

Note that it is possible that a feasible C^2 solution does not always exist. In that case, we must solve the minimization problem without the second derivative continuity constraint. If a C^2 solution exists and the natural spline is monotone, then the solution consists of the first derivatives of the natural spline.

5.3. Modified linearized energy (LE_LP)

Quadratic programming can be solved by using linear programming [24,32]. However, for a fixed number of variables in an objective function F, quadratic expressions for F require a larger system of equations than a linear expression for F. As a result, we simplify the linearized energy measure to be linear in the first derivatives so that a computationally simpler linear programming procedure can be applied.

We define our objective function in terms of E_L, the linearized energy of the curve. In order to minimize Eq. (29) using linear programming, we approximate it with the following expression:

$$\tilde{E}_L = \sum_{k=0}^{n-2} \int_{x_k}^{x_{k+1}} |f_k''(x)| \, dx. \tag{39}$$

The nonlinear absolute value operation, however, makes it difficult to readily solve for the unknown first derivatives of the cubic piecewise polynomial. Instead, we propose a different approach: add a constant K to $f_k''(x)$ such that $f_k''(x) + K \geqslant 0$ everywhere. This yields $\bar{E}_L$, the objective function for our linear programming solution:

$$\bar{E}_L = \sum_{k=0}^{n-2} \int_{x_k}^{x_{k+1}} (f_k''(x) + K) \, dx. \tag{40}$$

Since the second derivative of a cubic is linear in x, its extrema are at the interval borders. There always exists a K such that $f_k''(x) + K \geqslant 0$. The positivity is obtained by having this condition hold at the interval borders $x = x_k$ and $x = x_{k+1}$. We therefore add the following two equations as optimization constraints where K is one of the optimization unknowns:

$$f_k''(x_k) + K = \frac{1}{\Delta x_k^2}(-3y_k + 3y_{k+1} - 2\Delta x_k y_k' - \Delta x_k y_{k+1}') + K \geqslant 0, \tag{41}$$

$$f_k''(x_{k+1}) + K = \frac{1}{\Delta x_k^2}(3y_k - 3y_{k+1} + \Delta x_k y_k' + 2\Delta x_k y_{k+1}') + K \geqslant 0 \tag{42}$$

for $0 \leqslant k \leqslant n-2$. Eq. (40) can therefore be rewritten as

$$\bar{E}_L = \sum_{k=0}^{n-2} \int_{x=x_k}^{x_{k+1}} (f_k''(x) + K) \, dx = \sum_{k=0}^{n-2} y_{k+1}' - y_k' + K\Delta x_k = y_{n-1}' - y_0' + K(x_{n-1} - x_0). \tag{43}$$

Note that this expression is a linear equation with respect to the first derivatives and K. We use linear programming to minimize $\bar{E}_L$ subject to the following constraints:

1. Second derivative continuity: $f''(x_k^+) = f''(x_k^-)$ (Eq. (38)).
2. Second derivative on left side: $f_k''(x_k) + K \geqslant 0$ (Eq. (41)).
3. Second derivative on right side: $f_k''(x_{k+1}) + K \geqslant 0$ (Eq. (42)).
4. Monotonicity constraints: Eqs. (36) or (37).

Note that it is possible that a feasible C^2 solution does not always exist. In that case, we must solve the minimization problem without the second derivative continuity constraint. Conversely, it is possible that many C^2 solutions exists, and we will find one such solution from that set.

5.4. Linearized energy properties

When no feasible C^2 solution exists, the second derivative discontinuity may be visually prominent at the spline joints. Even if the C^2 solution exists, it may not necessarily minimize E_L. In fact, a C^1 solution may have a lower E_L. This is due to the fact that the domain of the C^1 solutions is a superset of the C^2 solution domain (see Fig. 6). Note that the solution for C^0 monotone constraints is the linear interpolation with $E_L = 0$. This implies that the E_L measure is not valid for C^0 or C^1 solutions. It is not even always valid for C^2 solutions since it is based on the (possibly false) assumption that $f'(x)^2 \ll 1$ everywhere. The following example demonstrates this argument. Consider the data set

$$\mathbf{X} = [0 \quad 1 \quad 2 \quad 3],$$

$$\mathbf{Y} = [0 \quad 400 \quad 400 \quad 800].$$

Fig. 8(a) shows the data fitted with a spline satisfying the free-end condition, i.e., the natural spline. Notice that although the data is monotone, the curve is not monotonic. Figs. 8(b) and (c) show the curves that minimize E_L with C^1 and C^2 constraints, respectively. The value of the second derivative difference in Fig. 8(b) at (1,400) is high and visually prominent.

Table 1 summarizes the energy measures for Fig. 8. Note that although the C^2 LE_QP curve in Fig. 8(c) has a higher E_L, it is unquestionably smoother. This fact is properly reflected in the accurate E energy measure. All of the solutions used the 6-sided polygonal approximation to region M shown in Fig. 7(a).

5.5. Second derivative discontinuity energy (SDDE_QP)

Due to the limitations of the approaches based on minimizing E_L, we seek to solve for the closest C^2 spline among all C^1 curves, thereby producing a more natural looking curve. This process requires us to introduce an energy measure based on the second derivative discontinuities:

$$E_D = \sum_{k=1}^{n-2} (f''(x_k^-) - f''(x_k^+))^2 = \sum_{k=1}^{n-2} \left(\frac{1}{\Delta x_k^2}[-3y_k + 3y_{k+1} - 2\Delta x_k y_k' - \Delta x_k y_{k+1}'] \right.$$

$$\left. - \frac{1}{\Delta x_{k-1}^2}[3y_{k-1} - 3y_k + \Delta x_{k-1} y_{k-1}' + 2\Delta x_{k-1} y_k'] \right)^2. \tag{44}$$

A similar objective function was suggested by Nielson in his work on v-splines [20,13].

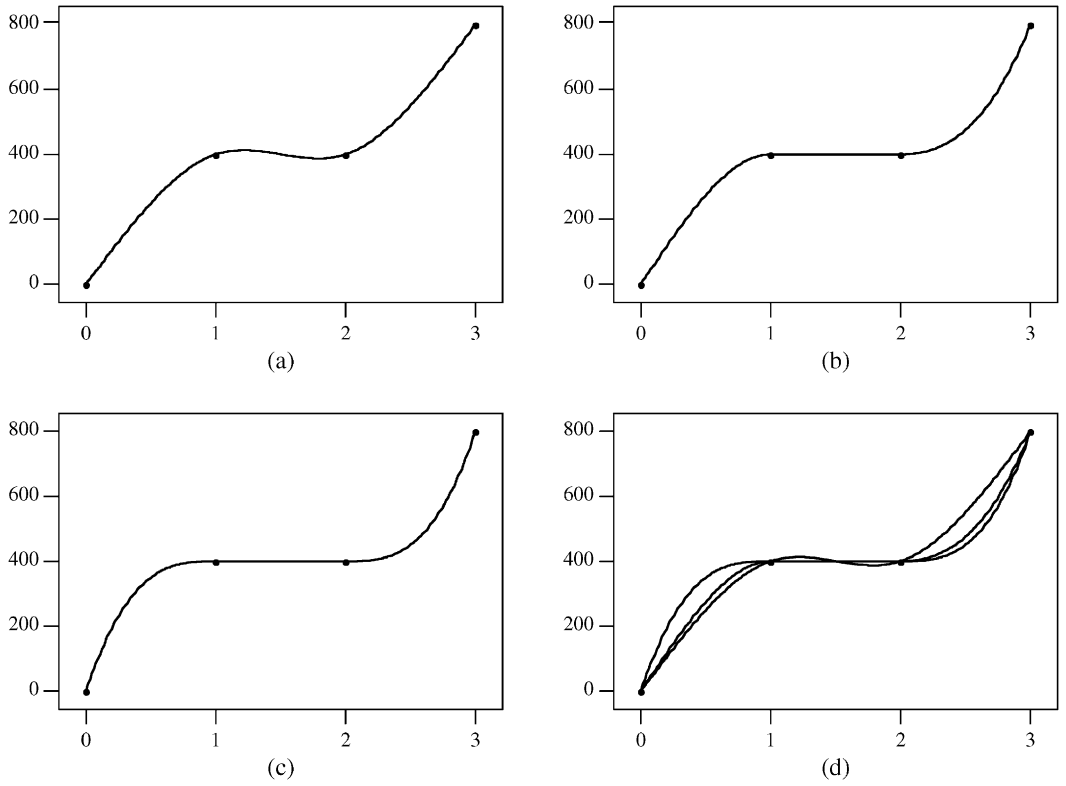

Fig. 8. (a) FE; (b) C^1 LE_QP; (c) C^2 LE_QP; (d) overlay.

The energy measure E_D can be minimized by using quadratic programming subject to the monotonicity constraints given in Eqs. (36) or (37). In the minimization formulations for the E_L and $\bar{E}_L$ energy methods, a second derivative continuity constraint was included among the set of constraints. This was a byproduct of the fact that E_L and $\bar{E}_L$ are poor energy measures, whereby a C^1 solution may be deemed to have lower energy than a C^2 solution. Note that no such constraint is necessary in minimizing E_D since a monotone C^2 spline, if it exists, will be the solution to the problem.

Table 1
Energy measures for Fig. 8

Method	E	E_L
FE	1231.66	640 000
LE_QP (C^1)	1599.34	960 000
LE_QP (C^2)	58.70	3 840 000

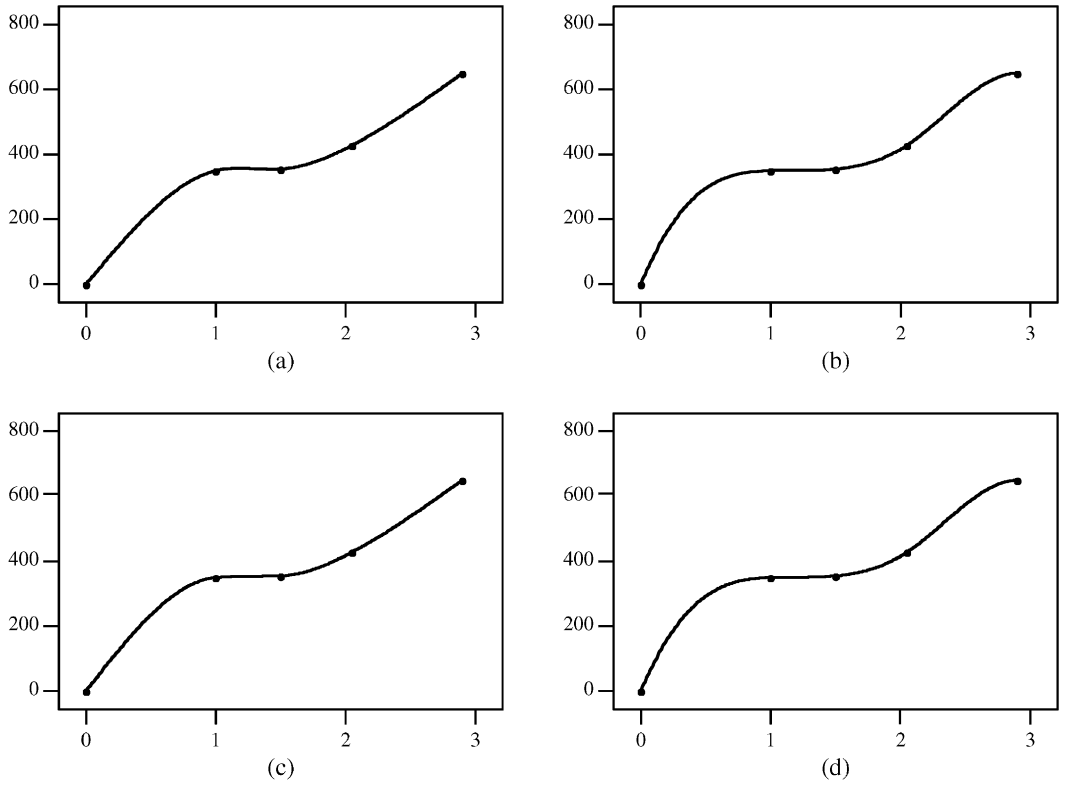

Fig. 9. (a) FE; (b) SDDE_QP (c) C^1 LE_QP (d) CSE.

The following example demonstrates the advantages of the SDDE_QP approach. Consider the data set:

$$\mathbf{X} = [0.000 \quad 1.00 \quad 1.05 \quad 2.05 \quad 2.90]$$
$$\mathbf{Y} = [0.00 \quad 350.00 \quad 354.65 \quad 428.00 \quad 650.00].$$

Fig. 9(a) shows the spline satisfying the free-end condition. Notice that although the data is monotone, the curve is not monotonic. Figs 9(b) and 9(c) show the C^1 curve that solves the quadratic programming problems required in minimizing E_D and E_L, respectively, using the 6-sided polygonal approximation to region M shown in Fig. 7(a). Fig. 9(d) shows the cubic spline elastica (CSE) obtained by minimizing Eq. (28). Table 2 summarizes the energy measures for Fig. 9. Note that although the CSE is a C^2 monotone curve, the LE_QP and SDDE_QP curves could not yield that solution since its (α_k, β_k) set exists in the area lying outside the polygon and inside M. It is evident that the SDDE_QP curve and its energy measures are much closer to those of CSE than LE_QP.

5.6. Modified discontinuity energy (SDDE_LP)

We simplify the discontinuity energy measure E_D to be linear with the first derivatives so that a linear programming procedure can be applied. The simplification is done by adding an unknown

Table 2
Energy measures for Fig. 9

Method	E	E_L	E_D
FE	855.84	343408.02	0
SDDE_QP	N/A	N/A	0.70
LE_QP (C^1)	N/A	N/A	666782.99
CSE	27.15	1504779.88	0

constant K to each second derivative difference, such that the term $f''(x_k^-) - f''(x_k^+) + K$ is positive. The positivity of each term is obtained by adding it as an optimization constraint where K is one of the optimization unknowns. We thus have

$$\hat{E}_D = \sum_{k=1}^{n-2} f''(x_k^-) - f''(x_k^+) + K, \tag{45}$$

where $f''(x_k^-) - f''(x_k^+)$ was defined in Eq. (44). The energy measure $\bar{E}_D$ can be minimized by using linear programming subject to the following constraints:

1. Positivity constraint: $f''(x_k^-) - f''(x_k^+) + K \geqslant 0$.
2. Monotonicity constraints: Eqs. (36) or (37).

An alternate approach can be used to linearize E_D using the absolute values of the discontinuities:

$$\bar{E}_D = \sum_{k=1}^{n-2} |f''(x_k^-) - f''(x_k^+)|. \tag{46}$$

For each discontinuity point we define a slack variable s_k whose value is forced to be the absolute value of the discontinuity, using the following inequality constraints:

$$f''(x_k^-) - f''(x_k^+) \leqslant s_k, \tag{47}$$

$$-[f''(x_k^-) - f''(x_k^+)] \leqslant s_k. \tag{48}$$

$\bar{E}_D$ can be rewritten as

$$\bar{E}_D = \sum_{k=1}^{n-2} s_k. \tag{49}$$

The energy measure $\bar{E}_D$ can be minimized by using linear programming subject to the following constraints:

1. Absolute value constraints: Eqs. (47) and (48).
2. Monotonicity constraints: Eqs. (36) or (37).

Note that no second derivative continuity constraint is necessary since a monotone C^2 spline is the solution to the problem.

5.7. Minmax discontinuity energy (SDDE_MM)

The SDDE approach minimizes the sum of the second derivative discontinuity energy across the knots. A reasonable alternative is to minimize $\tilde{E}_D$, the maximum second derivative discontinuity:

$$\tilde{E}_D = \mathrm{MAX}\{|f''(x_k^-) - f''(x_k^+)|\}, \tag{50}$$

where $f''(x_k^-) - f''(x_k^+)$ was defined in Eq. (44). The energy measure $\tilde{E}_D$ can be minimized by using linear programming subject to the monotonicity constraints given in Eqs. (36) or (37). For each discontinuity point we define a slack variable s_k whose value is forced to be the absolute value of the discontinuity, using Eqs. (47) and (48). Next, we define a new slack variable S whose value is forced to be the maximum value of all absolute discontinuities using the following inequality constraints:

$$s_k \leqslant S. \tag{51}$$

$\tilde{E}_D$ can be rewritten as $\tilde{E}_D = S$ and it can be minimized by using linear programming subject to the following constraints:

1. Absolute value constraints: Eqs. (47) and (48).
2. Maximum constraint: Eq. (51).
3. Monotonicity constraints: Eqs. (36) or (37).

Note that no second derivative continuity constraint is explicitly required because if a monotone C^2 spline exists it will naturally be chosen since $\tilde{E}_D = 0$ for C^2 splines.

6. Bounds on approximation error

The methods described in the previous sections interpolate monotone data with a cubic function $f(x)$. That same function approximates values of a monotone function $g(x)$ anywhere in $[x_1, x_{n-1}]$. In the following section, we derive bounds on the approximation error $e_k(x) = |f_k(x) - g_k(x)|$ for cubic polynomials. We also show the relationship between the approximation error and the size of the polygon used to approximate the monotone region M. We shall consider the six-sided polygon and the 3×3 square. The latter approximation is used in the popular Fritsch–Butland algorithm.

Lemma 1. *If $f(x)$ and $g(x)$ are two monotone polynomials of degree three or less that satisfy $f(x_k) = g(x_k) = y_k$ and $f(x_{k+1}) = g(x_{k+1}) = y_{k+1}$ then $e_k(x) \leqslant 0.866|\Delta y_k|$.*

Proof. Evaluating $e_k(x)$ using the Hermite basis functions (Eq. (3)) yields

$$e_k(x) = \left| \Delta x_k H_2\left(\frac{x - x_k}{\Delta x_k}\right)(f'(x_k) - g'(x_k)) + \Delta x_k H_3\left(\frac{x - x_k}{\Delta x_k}\right)(f'(x_{k+1}) - g'(x_{k+1})) \right|$$

$$= \left| \Delta y_k \left(H_2 \left(\frac{x - x_k}{\Delta x_k} \right) (\alpha_k^f - \alpha_k^g) + H_3 \left(\frac{x - x_k}{\Delta x_k} \right) (\beta_k^f - \beta_k^g) \right) \right|, \tag{52}$$

where

$$f'(x_k) = \alpha_k^f \Delta m_k,$$
$$f'(x_{k+1}) = \beta_k^f \Delta m_k,$$
$$g'(x_k) = \alpha_k^g \Delta m_k,$$
$$g'(x_{k+1}) = \beta_k^g \Delta m_k.$$

To determine the upper bound of $e(x)$ we solve the following problem:

$$\text{MAX } \tilde{e}_k(x) = (u^3 - 2u^2 + u)(\alpha_k^f - \alpha_k^g) + (u^3 - u^2)(\beta_k^f - \beta_k^g), \tag{53}$$

subject to:

$$0 \leqslant u \leqslant 1,$$
$$(\alpha_k^f, \beta_k^f) \in M,$$
$$(\alpha_k^g, \beta_k^g) \in M,$$

where the unknowns are $(\alpha_k^f, \beta_k^f), (\alpha_k^g, \beta_k^g)$ and u. The solution to this nonlinear problem is

$$u = 0.5,$$
$$(\alpha_k^f, \beta_k^f) = (3.7320, 0.2679),$$
$$(\alpha_k^g, \beta_k^g) = (0.2679, 3.7320),$$
$$\widetilde{e_k(x)} = 0.866.$$

Combining this result with Eq. (52) gives $e_k(x) \leqslant 0.866|\Delta y_k|$. □

Lemma 2. *If $f(x)$ and $g(x)$ are two monotone polynomials of degree three or less that satisfy $f(x_k) = g(x_k) = y_k$ and $f(x_{k+1}) = g(x_{k+1}) = y_{k+1}$ and $(\alpha_k^f, \beta_k^f) \in \tilde{M} \subset M$ then*

$$\text{If } (\alpha_k^g, \beta_k^g) \in M - \tilde{M} \quad \text{MAXMIN } e_k(x) = \begin{cases} 0 & \tilde{M} = M, \\ 0.058|\Delta y_k| & \tilde{M} = \textit{six-sided polygon}, \\ 0.1481|\Delta y_k| & \tilde{M} = \textit{square}. \end{cases} \tag{54}$$

Proof. If $(\alpha_k^g, \beta_k^g) \in \tilde{M}$ then the approximation algorithm can end with first derivatives such that $(\alpha_k^g, \beta_k^g) = (\alpha_k^f, \beta_k^f)$ i.e. $e_k(x) = 0$ for $x \in [x_k, x_{k+1}]$, and therefore MAXMIN $e_k(x) = 0$. If $(\alpha_k^g, \beta_k^g) \notin \tilde{M}$ then the approximation algorithm can not yield the correct first derivatives. The worst case in terms of maximum of minimum error can be obtained by solving the following problem:

$$\text{MAX } \widetilde{e_k(x)} = (u^3 - 2u^2 + u)(\alpha_k^f - \alpha_k^g) + (u^3 - u^2)(\beta_k^f - \beta_k^g), \tag{55}$$

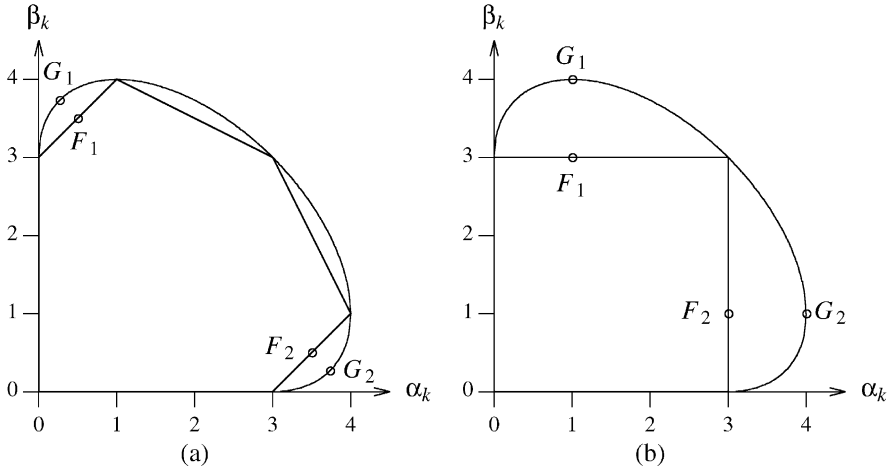

Fig. 10. MAXMIN solution for (a) six-sided polygon, and (b) square.

subject to:

$$0 \leqslant u \leqslant 1,$$

$$(\alpha_k^f, \beta_k^f) \in \tilde{M},$$

$$(\alpha_k^g, \beta_k^g) \in M - \tilde{M}.$$

There are two solutions to Eq. (55) for the case where $\tilde{M}$ is the 3×3 square used in [14].

$$[G_1, F_1, u_1] = [(\alpha_k^g, \beta_k^g), (\alpha_k^f, \beta_k^f), u_1] = [(4,1), (3,1), 0.66], \tag{56a}$$

$$[G_2, F_2, u_2] = [(\alpha_k^g, \beta_k^g), (\alpha_k^f, \beta_k^f), u_2] = [(1,4), (1,3), 0.33]. \tag{56b}$$

The error associated with either of these point pairs is $e_k(x) = 0.1481|\Delta y_k|$.

There are two solutions to Eq. (56) for the case where $\tilde{M}$ is a six-sided polygon:

$$[G_1, F_1, u_1] = [(\alpha_k^g, \beta_k^g), (\alpha_k^f, \beta_k^f), u_1] = [(3.7320, 0.2679), (3.5, 0.5), 0.5], \tag{57a}$$

$$[G_2, F_2, u_2] = [(\alpha_k^g, \beta_k^g), (\alpha_k^f, \beta_k^f), u_2] = [(0.2679, 3.7320), (0.5, 3.5), 0.5]. \tag{57b}$$

The error associated with either of these point pairs is $e_k(x) = 0.058|\Delta y_k|$. Fig. 10 shows the location of these two solutions for both cases. □

The above results show the relationship between the approximation error and the allowed region for (α_k^f, β_k^f). It is clear that by increasing the coverage of $\tilde{M}$ we can improve the approximation of $g(x)$.

Example. Consider the cubic function $g(x) = 6.5x^3 - 1.9x^2 + 0.2x$ sampled uniformly in the $[0,1]$ interval with $\Delta x = 0.1$. Fig. 11 shows the approximation error, $e(x)$, for the monotone cubic spline elastica (MCSE) and FB splines, respectively. The FB algorithm utilizes the 3×3 square in M.

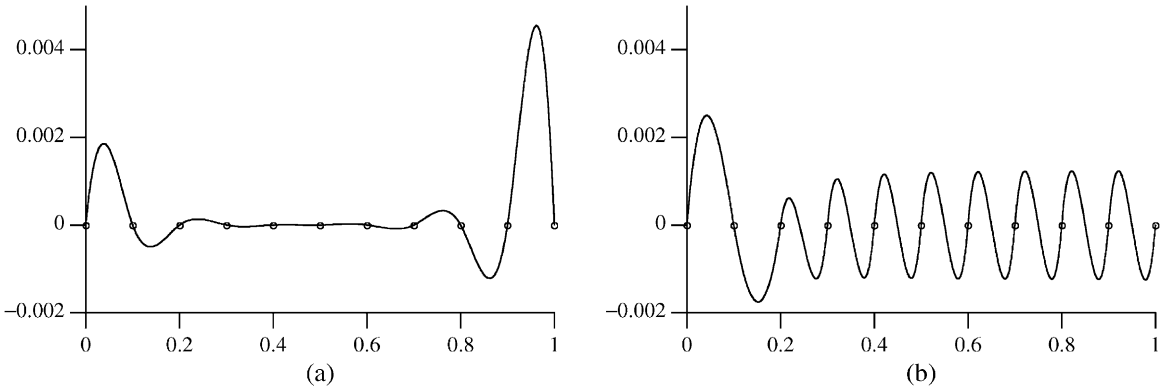

Fig. 11. Approximation error. (a) MCSE; (b) FB.

The MCSE splines are obtained by minimizing Eq. (28) subject to the six monotonicity constraints. The MCSE interpolates exactly, i.e., $e(x) < 0.0002$ in the [0.2,0.7] interval, while the approximation error of the FB curve is bounded by 0.0013.

7. Results

In this section, we compare the results of the different techniques described in this paper. They include:

1. Cubic spline elastica (CSE)—The cubic spline coefficients are obtained by minimizing Eq. (28) subject to the constraint that $f(x)$ is a C^2 cubic function. Since the energy given in Eq. (28) is a nonlinear objective function, global minimization is difficult. We used various initial estimates for the minimization procedure, including $\{y'_k\} = 0$ and $\{y'_k\} = m_k$. We selected the solution $\{y'_k\}$ associated with the minimum energy value computed.
2. Monotone cubic spline elastica (MCSE)—The first derivatives are obtained by minimizing E subject to the constraint that $f(x)$ is monotone and a C^2 cubic function. Since the energy E is a nonlinear objective function, global minimization is difficult. We used various initial estimates for the minimization procedure, including $\{y'_k\} = 0$, $\{y'_k\} = m_k$, as well as $\{y'_k\}$ computed by the SDDE_QP method. We selected the solution $\{y'_k\}$ associated with the minimum energy value computed. Note that MCSE solution may not exist if there is no monotone C^2 solution.
3. Free-end (FE) boundary condition—The first derivatives $\{y'_k\}$ are obtained by minimizing Eq. (29), assuming $f(x)$ is a C^2 cubic function that satisfies the free-end condition: $f''(x_0) = f''(x_{n-1}) = 0$.
4. Fritsch and Butland (FB): The algorithm is described in [14] and implemented in PCHIM. FOR, NETLIB's PCHIPD package for piecewise cubic hermite interpolation by Fritsch. The first derivatives are calculated using Brodlie's formula with $\alpha = (\Delta x_{k-1} + 2\Delta x_k)/3(\Delta x_{k-1} + \Delta x_k)$:
$$f'(x_k) = \frac{m_{k-1} m_k}{\alpha m_k + (1-\alpha) m_{k-1}}. \tag{58}$$
5. Second derivative discontinuity energy (SDDE_QP)—Minimize E_D.

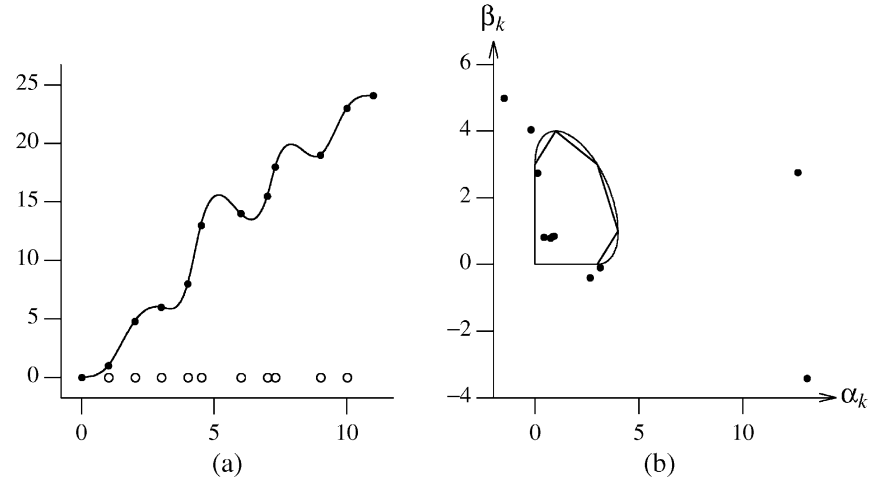

Fig. 12. Free end (FE): (a) interpolating spline; (b) $\{\alpha, \beta\}$ points.

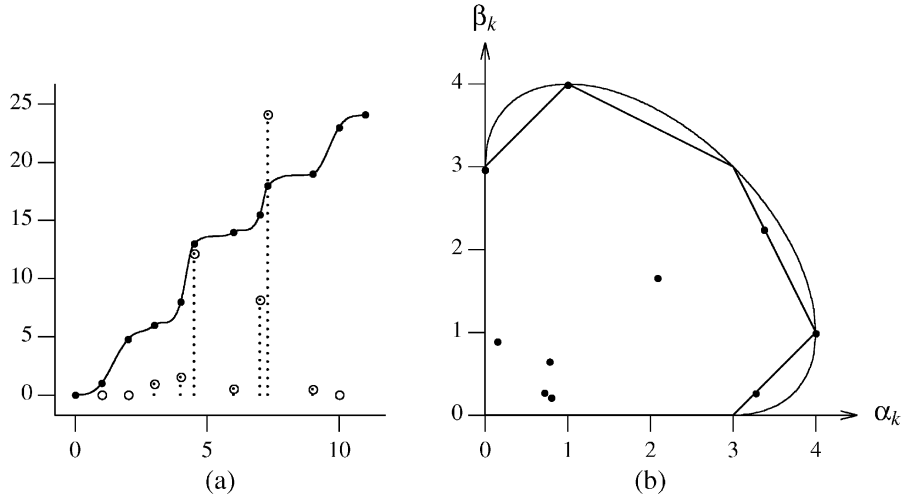

Fig. 13. Second derivative discontinuity energy (SDDE_LP): (a) interpolating spline; (b) $\{\alpha, \beta\}$ points.

6. Modified discontinuity energy (SDDE_LP)—Minimize $\bar{E}_{\mathrm{D}}$.
7. Minmax discontinuity energy (SDDE_MM)—Minimize $\tilde{E}_{\mathrm{D}}$.

We used the optimization toolbox of Matlab 5.3 to solve the linear and quadratic programming problems above. Note that Matlab indicates when no feasible solution exists. We will demonstrate the techniques on the following two data sets. The first set is the following:

$\mathbf{X} = [0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 4.5 \quad 6 \quad 7 \quad 7.3 \quad 9 \quad 10 \quad 11],$

$\mathbf{Y} = [0 \quad 1 \quad 4.8 \quad 6 \quad 8 \quad 13 \quad 14 \quad 15.5 \quad 18 \quad 19 \quad 23 \quad 24.1].$

Figs. 12–14 depict the curves produced by the various methods. Each curve is presented in two coordinate systems: $(x, f(x))$ and (α, β). The purpose of this representation is to highlight

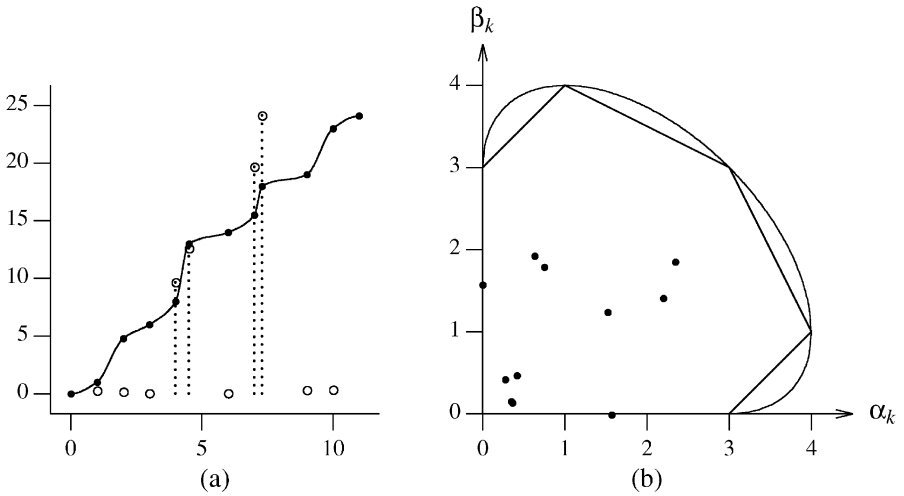

Fig. 14. Fritsch–Butland (FB): (a) interpolating spline; (b) $\{\alpha, \beta\}$ points.

Table 3
Energy measures

Method	E	E_{D}	$\tilde{E}_{\mathrm{D}}$
CSE	53.47	0.00	0.00
FE	54.27	0.00	0.00
SDDE_QP	N/A	16445.26	8306.84
SDDE_LP	N/A	16472.55	8306.84
FB	N/A	44460.52	15995.29

the monotone characteristics of the curves. The spline figures contain vertical lines ended with circles to represent the second derivative difference at the knots, where the difference is measured by $D_k = (f''(x_k^-) - f''(x_k^+))^2$. Note that E_{D} is defined as the sum of all the D_k's. The scale for the second derivative differences is normalized to fit the curve scale, while its maximum value is given in Table 3. Note also that for C^2 curves, e.g., $E_{\mathrm{D}} = 0$, the differences are zero.

The curves produced by the CSE and FE methods above are C^2 and not monotone. Note that there is no monotone C^2 solution and therefore the MCSE curve does not exist. The SDDE_QP, SDDE_LP, and Fritsch–Butland curves are C^1 and monotone. Since the SDDE_QP and SDDE_LP curves are virtually identical, we showed only the latter curve. The distribution of the $\{\alpha_k, \beta_k\}$ points for the Fritsch–Butland curve are concentrated near the origin, i.e., biased towards low values. This has a noticeable effect on the smoothness of the curve. In particular, the resulting curve exhibits more tension and is biased towards linear interpolation where $(\alpha, \beta) = (1, 1)$. It is also salient that the second derivative discontinuities are prominent at the spline joints. Table 3 summarizes these results. Note that the N/A entries in the table apply to those curves which are only C^1. Any attempt to minimize the sum of their piecewise squared second derivatives will yield a linear interpolant with $E = 0$, which is not smooth. E is therefore not a meaningful energy measure for C^1 curves.

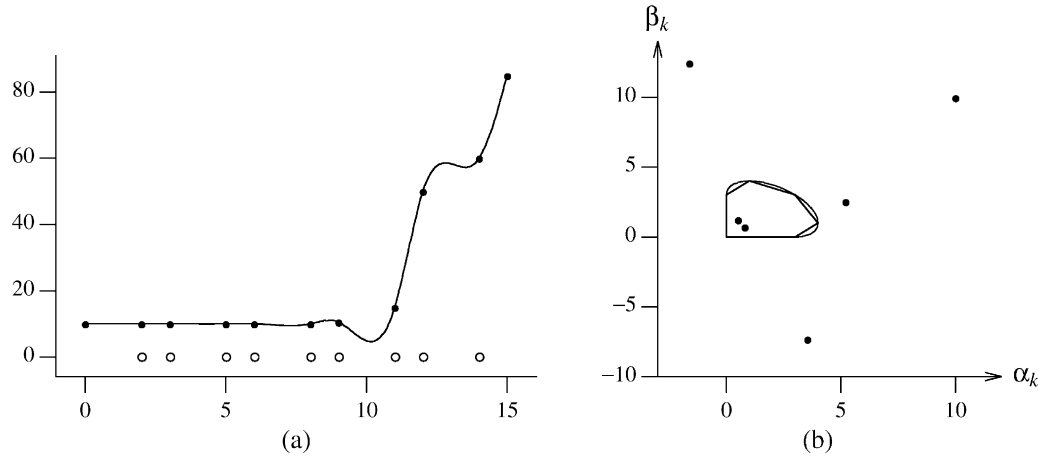

Fig. 15. Free end (FE): (a) interpolating spline; (b) $\{\alpha, \beta\}$ points.

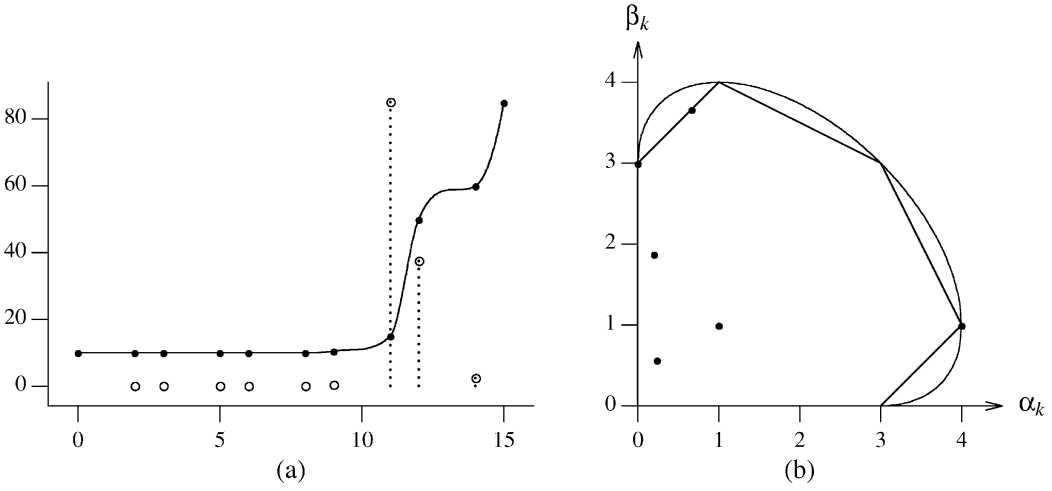

Fig. 16. SDDE_LP method: (a) interpolating spline; (b) $\{\alpha, \beta\}$ points.

The second example is the third set used by Akima [1].

$\mathbf{X} = [0 \quad 2 \quad 3 \quad 5 \quad 6 \quad 8 \quad 9 \quad 11 \quad 12 \quad 14 \quad 15],$
$\mathbf{Y} = [10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10.5 \quad 15 \quad 50 \quad 60 \quad 85].$

Figs. 15–17 depict the curves produced by methods described in this section. In this case, there is no feasible monotonic C^2 solution. Therefore, all the methods produced C^1 curves. Again, the energy minimization solutions are visually more pleasing. This can be explained by the $\{\alpha, \beta\}$ pairs that are spread more widely across the M region. Table 4 summarizes these results. Note that energy measure E is not applicable for C^1 solutions produced by the SDDE_QP, SDDE_LP, and FB methods.

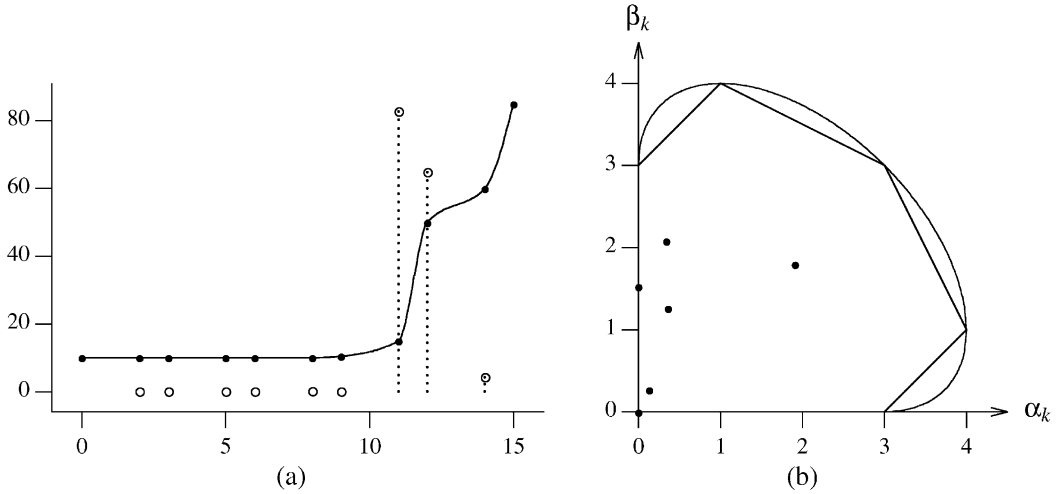

Fig. 17. Fritsch–Butland (FB): (a) interpolating spline; (b) $\{\alpha, \beta\}$ points.

Table 4
Energy measures for Akima data interpolants

Method	E	E_D	$\tilde{E}_D$
CSE	73.68	0.00	0.00
FE	81.02	0.00	0.00
SDDE_QP	N/A	22841.56	15813.06
SDDE_LP	N/A	22841.56	15813.06
FB	N/A	52249.08	28486.43

8. Extensions

In this section we extend the techniques described earlier to solve the following problems:

1. Handling data of changing monotonicity.
2. Construction of monotone and convex (or concave) interpolating cubic splines.
3. Construction of C^2 interpolating cubic splines with up to two extra knots inserted between each pair of data points.
4. Smoothing noisy data with C^2 cubic splines.

8.1. Handling data of changing monotonicity

All of the methods presented thus far have been demonstrated on monotonic increasing or decreasing data sets. The methods can be easily extended to handle data of changing monotonicity. Arbitrary data sets, for instance, can be partitioned into a sequence of monotonic increasing and decreasing sets. Enforcing monotonicity in each of these sets helps reduce undesirable ripples. This may be useful for applications such as signal resampling, e.g., image magnification.

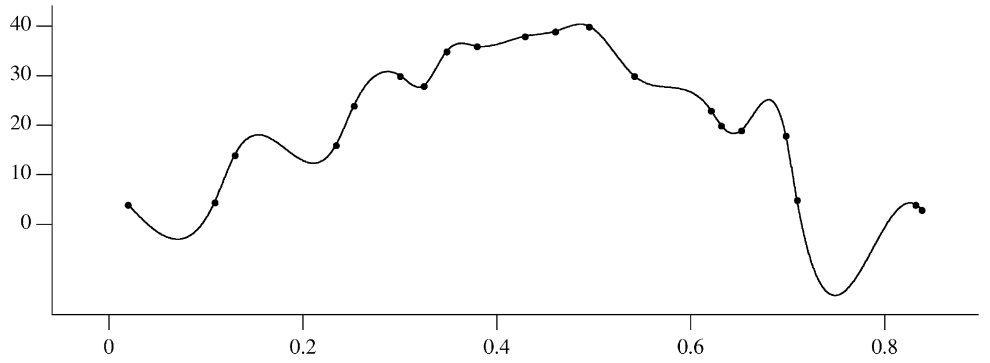

Fig. 18. Data fitting with natural spline (free end condition).

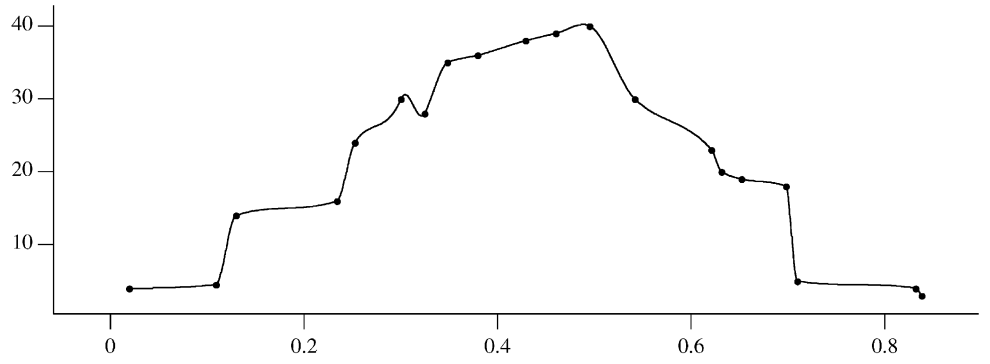

Fig. 19. Data fitting with Fritsch–Butland (FB) algorithm.

We visit all intervals and determine which monotonicity constraints to apply, based on whether the data is increasing, decreasing, or constant. For each increasing and decreasing interval, we have six monotonicity constraints. For each constant (horizontal) interval, we have two constraints: $y'_k = y'_{k+1} = 0$. When two intervals meet to form a local extrema, no monotonicity constraints are applied to either interval. Details are provided in the supplied MATLAB code in Appendix B.

Figs. 18–20 show the free-end, Fritsch–Butland (FB), and SDDE methods applied to data having three local extrema. The data set is given below. Note that the local nature of the FB algorithm forces a tense fit through the first extrema. The SDDE_LP method relaxes the monotonicity constraint to allow a smooth fit in the vicinity of the extrema. Similarly, the extrema on the right side of the figure demonstrates the smoother SDDE_LP fit. Fig. 21 shows the $\{\alpha, \beta\}$ points for the FB and SDDE_LP methods. Notice that the FB algorithm restricts the $\{\alpha, \beta\}$ positions to the square delimited by (0,0) and (3,3). The SDDE_LP method, on the other hand, permits the $\{\alpha, \beta\}$ positions to fall freely in

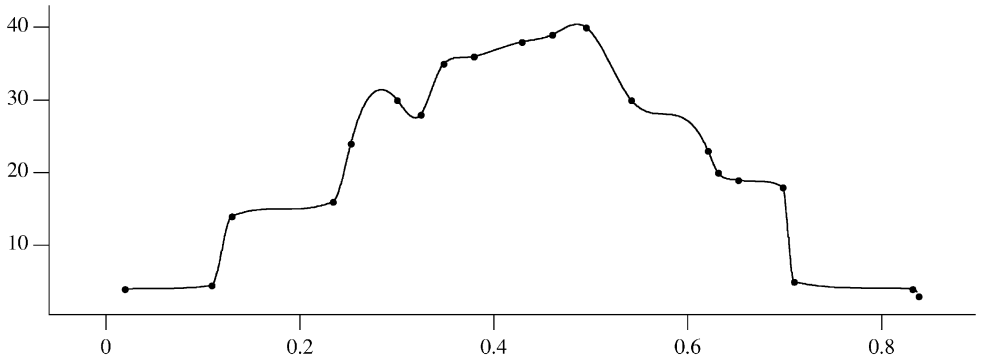

Fig. 20. Data fitting with SDDE_LP method.

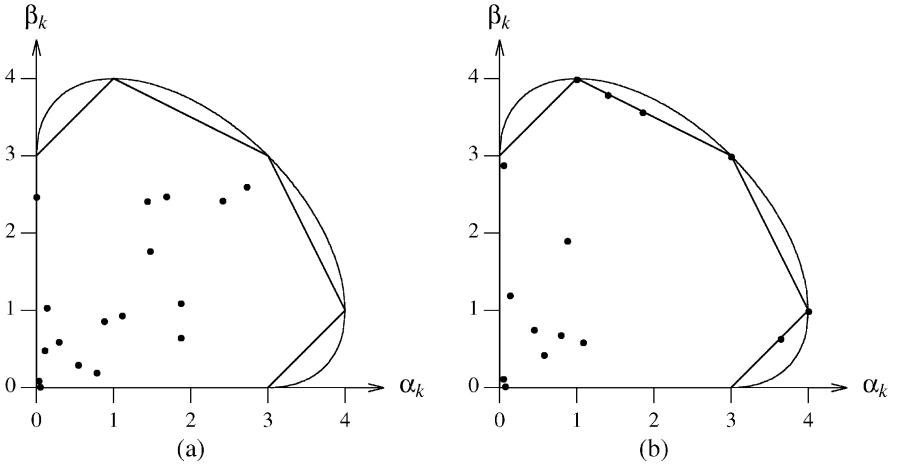

Fig. 21. $\{\alpha, \beta\}$ points for (a) FB; (b) SDDE_LP.

the whole monotonicity region.

$$\mathbf{X} = [0.0196 \quad 0.1090 \quad 0.1297 \quad 0.2340 \quad 0.2526 \quad 0.3003 \quad 0.3246 \quad 0.3484 \quad 0.3795 \quad 0.4289$$
$$0.4603 \quad 0.4952 \quad 0.5417 \quad 0.6210 \quad 0.6313 \quad 0.6522 \quad 0.6979 \quad 0.7095 \quad 0.8318 \quad 0.8381],$$
$$\mathbf{Y} = [\ 4 \quad 4.5 \quad 14 \quad 16 \quad 24 \quad 30 \quad 28 \quad 35 \quad 36 \quad 38$$
$$39 \quad 40 \quad 30 \quad 23 \quad 20 \quad 19 \quad 18 \quad 5 \quad 4 \quad 3].$$

8.2. Shape preserving cubic splines

A spline is said to be shape preserving if it produces convex splines for convex data. Several algorithms were proposed to compute shape preserving splines that are monotone and convex [8,7,4,5]. The algorithms are based on [15] and iteratively compute a set of first derivatives that simultaneously satisfy the monotonicity and convexity constraints to produce a C^1 spline. We seek to introduce shape preserving constraints in our optimization framework to produce a C^2 solution, if it exists.

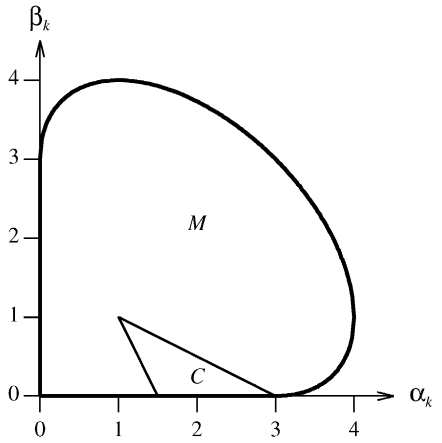

Fig. 22. Convexity region C is embedded in monotonicity region M.

Function $f(x)$ is said to be increasing convex in $[x_k, x_{k+1}]$ if

$$f'(x) \geqslant 0 \quad x \in [x_k, x_{k+1}] \tag{59}$$

and

$$f''(x) \geqslant 0 \quad x \in [x_k, x_{k+1}]. \tag{60}$$

The increasing condition of Eq. (59) is a monotonicity requirement discussed in Section 4.

Since the second derivative of $f(x)$ is linear in x, its extrema are at the interval borders. For the second derivative to be positive in the interval $[x_k, x_{k+1}]$ it is sufficient that $f''_k(x_k) \geqslant 0$ and $f''_k(x_{k+1}) \geqslant 0$. This yields the following convexity constraints:

$$2y'_k + y'_{k+1} \leqslant 3m_k, \tag{61a}$$

$$-y'_k - 2y'_{k+1} \leqslant -3m_k. \tag{61b}$$

Fig. 22 shows the convexity region C embedded in monotonicity region M. The intersection of both regions yields the shape preserving constraints:

$$2y'_k + y'_{k+1} \leqslant 3m_k, \tag{62a}$$

$$-y'_k - 2y'_{k+1} \leqslant -3m_k, \tag{62b}$$

$$y'_k \geqslant 0, \tag{62c}$$

$$y'_{k+1} \geqslant 0. \tag{62d}$$

These shape preserving constraints can be used in place of the monotonicity constraints (Eqs. (36) or (37)) for minimizing E_L, $\bar{E}_L$, E_D, and $\bar{E}_D$.

8.3. Knot insertion

In each subinterval $[x_k, x_{k+1}]$, we shall introduce two additional knots at locations $(x_k + \tau)$ and $(x_{k+1} - \tau)$, where $\tau \leqslant \Delta x_k/3$. It has been shown that this knot insertion process will guarantee the existence of a monotone C^2 cubic spline interpolant [23].

Let $D = \{x_k, y_k\}_{k=0}^{n-1}$ be the original set of data points and let $K = \{\tilde{x}_i, \tilde{y}_i\}_{i=0}^{2(n-1)}$ denote the inserted knots. The full set of points through which the interpolant must pass is $P = D \cup K$. That is, $P = \{X_m, Y_m\}_{m=0}^{3(n-1)}$ consists of the original data points and the inserted knots such that each original data point with index i lies at position $3i$ in the set P, i.e., $\{X_{3i}, Y_{3i}\} = \{x_i, y_i\}$.

In order to solve for the interpolant, we will introduce a new set of constraints that will be used to minimize E_L, $\bar{E}_L$, E_D, and $\bar{E}_D$. We will need to solve for the unknown $\{Y_m\}$ values in K and first derivative values $\{Y'_m\}$ in P. In order to simplify the implementation, we will consider both $\{Y_m\}$ and $\{Y'_m\}$ to be unknown. The original data values $\{y_k\}$ is applied to $\{Y_m\}$ by means of the interpolation constraint $Y_{3i} = y_i$. The following are the constraints for the optimization problem:

1. Interpolation: $Y_{3i} = y_i$,
2. 2nd derivative continuity: $f''(X_m^+) = f''(X_m^-)$ (Eq. (38)),
3. Monotonicity constraints: Eqs. (36) or (37).

Note that m_k in Eqs. (36) and (37) refers to $(Y_{m+1} - Y_m)/\Delta X_m$ in this case.

The manner in which we formulated the constraints is independent of the number of knots in K. Therefore, one may consider the use of 0, 1, or 2 additional knots between any two data points. Furthermore, the number of additional knots can be made to vary among intervals. This suggests that a progressive method may be applied in which we solve the optimization problem with no additional knots. If no feasible C^2 solution exists, then may add one knot between each pair of data points and solve the new problem. If a feasible C^2 solution does not exist, then we may add two knots between the data points, as suggested by Pruess [23]. This time we are guaranteed to have a feasible C^2 solution.

8.4. Smoothing

In case the data $\{y_k\}$ is not accurate due to noise or measurement error, we suggest the following method to compute a monotone C^2 approximating curve. This method can work even if the data is not monotone.

Let $\{f_k\}$ be the values of the approximating curve at the knots $\{x_k\}$. We use the mean squared error as our objective function

$$E_S = \sum_{k=0}^{n-1} |(y_k - f_k)|. \tag{63}$$

The following are the constraints for the optimization problem:

1. Second derivative continuity: $f''(x_k^+) = f''(x_k^-)$ (Eq. (38)),
2. Monotonicity constraints: Eqs. (36) or (37).

Since we are performing approximation rather than interpolation, we use f_k rather than y_k in Eq. (38). Also, note that m_k in Eqs. (36) and (37) refers to $(f_{k+1} - f_k)/\Delta x_k$ in this case. Finally, the E_S

Table 5
Comparison of proposed methods

Method	Energy	Optimization	Benefits	Drawbacks
MCSE	E	Nonlinear	Optimal C^2	Slow; does not always exist
SDDE_QP	E_D	QP	C^2 or closest C^1	Suboptimal C^2
SDDE_LP	$\bar{E}_D$	LP	C^2 or closest C^1	Suboptimal C^2
SDDE_MM	$\tilde{E}_D$	LP	C^2 or closest C^1	Suboptimal C^2
LE_QP	E_L	QP	Suboptimal C^2	Poor C^1
LE_LP	$\bar{E}_L$	LP	Suboptimal C^2	Poor C^1
FB	None	None	Fast	C^1, even if C^2 exists

energy measure can be minimized by using quadratic programming subject to the constraints given above.

9. Comparison

In this section, we compare the various techniques. We consider the MCSE, SDDE_QP, SDDE_LP, SDDE_MM, LE_QP, LE_LP, and FB methods. Note that this order corresponds to the quality of the splines produced, beginning with the MCSE's optimal C^2 spline. These methods are outlined in Table 5, where LP and QP are used to refer to linear and quadratic programming, respectively.

- The MCSE method yields the optimal C^2 solution, if one exists. Its primary drawback is that it requires a costly nonlinear constrained optimization technique to minimize E. Furthermore, if there is no C^2 solution, then an alternate method must be considered.
- The SDDE_QP method requires quadratic programming to minimize E_D. It produces a suboptimal C^2 solution, if it exists. Otherwise, it yields a C^1 solution that is closest to C^2.
- The SDDE_LP method requires linear programming to minimize $\bar{E}_D$. It produces a suboptimal C^2 solution, if it exists. Otherwise, it yields a C^1 solution that is closest to C^2. The SDDE_LP technique has proven to be our method of choice. It produces curves that are virtually indentical to SDDE_QP curves at much lower cost.
- The LE_QP and LE_LP methods require quadratic and linear programming, respectively, to minimize energy measures E_L and $\bar{E}_L$, respectively. They produce a suboptimal C^2 solution, if it exists. Otherwise, they yield a poor C^1 solution due to prominent second derivative discontinuities.
- The popular Fritsch–Butland algorithm uses a local method to compute a monotone C^1 interpolant. It is important to note that the method does not attempt to find a C^2 solution, even if one exists.
- All of the energy minimization methods compared here can benefit from knot insertion to guarantee the existence of a C^2 solution. Knot insertion can employ any energy measure and any optimization method to derive a C^2 solution. The major drawback to the use of knot insertion is that the number of constraints grows to solve for the additional knots.
- The E_D for SDDE_MM may be greater than that of SDDE_QP.

10. Discussion

In this section, we review several key points about monotonic cubic spline interpolation and linear programming.

1. This paper presents several key enhancements beyond the work described in [31]. In that paper, cubic splines were represented using four coefficients, requiring the solution of three unknowns and the use of nine constraints per interval. This work makes use of the Hermite representation of splines. As a result, the interpolation and C^1 continuity constraints are implicit and do not need to be considered when minimizing any objective function. Furthermore, this approach requires the solution of only one unknown and the use of seven constraints per interval. Note that we assume the use of six monotonicity constraints above.
2. The assumption used to define E_L is correct for the subset of C^2 curves that comply with the condition $f'(x)^2 \ll 1$. This condition is often violated in practice. This makes the use of linearized energy E_L meaningless as an objective function for energy minimization methods. For instance, in comparing the FE and the SDDE_QP curves in Fig. 9, higher E for the free-end (FE) curve does not translate to higher E_L (see Table 2).
3. By examining the physical definition of the strain energy, it is implicit that E and E_L are applicable for C^2 curves only. This means that the energy measures are only meaningful when comparing C^2 monotone splines. For instance, we cannot use such energy measures to compare C^0 curves, such as those produced by linear interpolation. It is apparent that such curves produce low values for E and E_L, although they are not smooth at the spline joints. That is, they satisfy $f''(x) = 0$ nearly everywhere. Only at the spline joints is this condition possibly violated. Therefore, C^2 energy measures E and E_L are not appropriate objective functions for monotone splines, since the monotonicity constraint may sometimes force the spline to be C^1 continuous.
4. The Fritsch–Butland algorithm clamps the α and β values to the $[0, 3]$ range, thereby utilizing only 67.9% of monotone region M. This has a tendency of biasing the solution away from smoother alternatives. The optimization-based solutions presented in this paper more fully utilize region M, yielding the smoother SDDE_LP curve shown in the figures above. For instance, the six-sided and fifteen-sided polygonal approximation of M cover 90.53% and 97.53% of region M, respectively. Furthermore, the SDDE_LP algorithm can produce C^2 solutions, whereas the local Fritsch–Butland algorithm is limited to C^1 solutions.
5. The approaches presented here are general in the sense that they can be easily modified to solve different applications where additional constraints are imposed or linear combination of objective functions are used. Section 8 demonstrated how to apply shape preserving, knot insertion, and data smoothing constraints in our framework.
6. The space of linear programming (LP) problems with n unknowns is in $\Re^n$. Each constraint represents a hyperplane. Equality constraints force the feasible region onto hyperplanes, while inequalities divide the feasible region into allowed and disallowed halfplanes. When all the constraints are imposed, either we are left with some feasible region or else there is no feasible solution. The feasible region for LP problems is a convex polygon and the optimum value occurs at a vertex of the feasible region [24]. The two-phase simplex method is commonly used to obtain the optimal solution. Phase I determines whether the LP problem has a feasible solution. If a feasible solution exists, phase I provides a basic solution that complies with the constraints

but is not necessarily optimal. Phase II, in turn, finds the basic solution that is optimal. Wolfe [32] proposed a method for converting a quadratic programming problem into an LP problem requiring only phase I computation. This allows us to make use of the simplex method to solve quadratic programming problems.

7. The observations made in Section 9 suggest the following sequence for determining the optimal spline. First, we apply the SDDE_LP method to derive a solution. If a C^2 solution exists, we may either apply the MCSE method to derive the optimal C^2 solution or be satisfied with the suboptimal C^2 SDDE_LP spline. If a C^2 solution does not exist, then SDDE_LP leaves us with a C^1 spline that is closest to C^2.

11. Summary

The goal of this work has been to determine the *smoothest* possible curve that passes through its control points while simultaneously satisfying the montonicity constraint. We presented a simple monotonicity test that may be applied to each pair of control points in the spline. That result is used as the basis for all the methods described in this paper, including linear and nonlinear optimization-based methods. We cast the monotonic cubic spline interpolation problem within an energy minimizing framework. Various energy measures were considered for the optimization objective functions.

We showed how to apply quadratic programming to minimize the objective functions used in this paper. Modifications were introduced to simplify the problem and facilitate the use of linear programming. The interpolation methods considered in this paper include cubic spline elastica (CSE), free end (FE), linearized energy (LE_QP), modified linearized energy (LE_LP), second derivative discontinuity energy (SDDE_QP), modified discontinuity energy (SDDE_LP), and the Fritsch–Butland algorithm. We found that energy minimization methods yielded superior results to the popular Fritsch–Butland algorithm [14]. We suggested that the SDDE_LP energy measure be used as an optimization objective. It minimized the second derivative discontinuity and provided visually pleasing results.

Since there is a large family of monotone curves that interpolate the data, we derived bounds on the error between any two such curves. We also showed that the traditional linearized energy measure E_L is based on invalid assumptions and is of limited value in determining C^1 monotonic solutions. Finally, we presented extensions to handle arbitrary data sets with changing monotonicity, shape-preserving splines, knot insertion, and data smoothing. MATLAB code is furnished to demonstrate the monotonic cubic spline interpolation algorithm.

Appendix A. Monotonicity constraints

The boundary of monotone region M is approximated using an n-sided polygon. The vertices of the polygon are given by coordinates (α_i, β_i), where $0 \leqslant i < n$. Let s_i be the slopes of the polygon sides:

$$s_i = \frac{\beta_i - \beta_{i-1}}{\alpha_i - \alpha_{i-1}}. \tag{A.1}$$

Table 6
Approximating monotonicity region M with a 6-sided polygon

α_{i-1}	β_{i-1}	α_i	β_i	s_i	y'_k	y'_{k+1}	m_k	Inequality direction $m_k > 0$	Inequality direction $m_k < 0$
0	3	0	0	∞	0	1	0	$\geqslant 0$	$\leqslant 0$
0	0	3	0	0	1	0	0	$\geqslant 0$	$\leqslant 0$
3	0	4	1	1	−1	1	3	$\geqslant 0$	$\leqslant 0$
4	1	3	3	−2	2	1	−9	$\leqslant 0$	$\geqslant 0$
3	3	1	4	−0.5	0.5	1	−4.5	$\leqslant 0$	$\geqslant 0$
1	4	0	3	1	−1	1	−3	$\leqslant 0$	$\geqslant 0$

The equation of the ith line is

$$\beta - s_i\alpha + s_i\alpha_{i-1} - \beta_{i-1} = 0. \tag{A.2}$$

Note that α is a free variable, while α_i is a fixed coordinate value. The same applies to β and β_i.

Since $(0,0)$ is a point in M, we may represent the interior half-plane (containing M) as follows:

$$\textbf{IF } s_i\alpha_{i-1} - \beta_{i-1} > 0 \textbf{ THEN } \beta - s_i\alpha + s_i\alpha_{i-1} - \beta_{i-1} \geqslant 0,$$
$$\textbf{ELSE } \beta - s_i\alpha + s_i\alpha_{i-1} - \beta_{i-1} \leqslant 0.$$

We may write the line equation in terms of first derivatives y'_k by letting $\alpha = \alpha_k = y'_k/m_k$ and $\beta = \beta_k = y'_{k+1}/m_k$. This yields the following inequality when $m_k > 0$:

$$\textbf{IF } s_i\alpha_{i-1} - \beta_{i-1} > 0 \textbf{ THEN } y'_{k+1} - s_i y'_k + m_k(s_i\alpha_{i-1} - \beta_{i-1}) \geqslant 0,$$
$$\textbf{ELSE } y'_{k+1} - s_i y'_k + m_k(s_i\alpha_{i-1} - \beta_{i-1}) \leqslant 0.$$

When $m_k < 0$, we have:

$$\textbf{IF } s_i\alpha_{i-1} - \beta_{i-1} > 0 \textbf{ THEN } y'_{k+1} - s_i y'_k + m_k(s_i\alpha_{i-1} - \beta_{i-1}) \leqslant 0,$$
$$\textbf{ELSE } y'_{k+1} - s_i y'_k + m_k(s_i\alpha_{i-1} - \beta_{i-1}) \geqslant 0.$$

Note that linear programming requires all inequalities to be less than the free variable (on the right-hand side). This may require multiplying both sides by -1 to switch the inequality direction. Tables 6 and 7 show the polygon vertices and their respective coefficients for y'_k, y'_{k+1}, and m_k. These results are used to form the monotonicity conditions in Eqs. (36) and (37).

Appendix B. MATLAB code

The following MATLAB code computes an interpolating monotonic cubic spline. First, we apply the SDDE_LP method to derive a solution. If a C^2 solution exists, we apply the MCSE method to derive the optimal monotone C^2 solution. If a C^2 solution does not exist, then SDDE leaves us with a C^1 spline that is closest to C^2.

Table 7
Approximating monotonicity region M with a 10-sided polygon

α_{i-1}	β_{i-1}	α_i	β_i	s_i	y'_k	y'_{k+1}	m_k	Inequality direction $m_k > 0$	Inequality direction $m_k < 0$
0	3	0	0	∞	0	1	0	$\geqslant 0$	$\leqslant 0$
0	0	3	0	0	1	0	0	$\geqslant 0$	$\leqslant 0$
3	0	3.7320	0.2679	0.3660	−0.3660	1	1.0980	$\geqslant 0$	$\leqslant 0$
3.7320	0.2679	4	1	2.7317	−2.7317	1	9.9269	$\geqslant 0$	$\leqslant 0$
4	1	3.7320	2	−3.7313	3.7313	1	−15.9254	$\leqslant 0$	$\geqslant 0$
3.7320	2	3	3	−1.3661	1.3661	1	−7.0984	$\leqslant 0$	$\geqslant 0$
3	3	2	3.7320	−0.7320	0.7320	1	−5.1960	$\leqslant 0$	$\geqslant 0$
2	3.7320	1	4	−0.2680	0.2680	1	−4.2680	$\leqslant 0$	$\geqslant 0$
1	4	0.2679	3.7320	0.3661	−0.3661	1	−3.6339	$\leqslant 0$	$\geqslant 0$
0.2679	3.7320	0	3	2.7324	−2.7324	1	−3	$\leqslant 0$	$\geqslant 0$

B.1. File monotone.m

```
% ------------------------------------------------------------------------
%
% montone.m - Compute interpolating monotonic cubic spline.
% Based on work described in our paper:
% An Energy-Minimization Framework for
% Monotonic Cubic Spline Interpolation
%
% Written by: Itzik Alfy and George Wolberg, 2000
% Copyright (C) 2000 by Itzik Alfy and George Wolberg
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% FUNCTION:
%     D = monotone(X,Y)
% INPUT:
%     X <- monotonic input data vector (size: n x 1)
%     Y <- monotonic input data vector (size: n x 1)
% OUTPUT:
%     D <- 1st derivatives of interpolating monotonic cubic spline (n x 1)
% DESCRIPTION:
%     Compute an interpolating monotonic cubic spline passing through
%     n input data points. The spline is fully specified in terms of
%     the nx1 output vector of first derivatives.
%
%     monotone() first attempts to fit a C2 spline minimizing E_D, the
%     SDDELP energy measure. If a C2 solution exists, then we seek the
```

```
%     optimal C2 spline by minimizing the strain energy E to compute the
%     MCSE. If a C2 solution does not exist, we are left with the best
%     C1 spline.
%
%     monotone() calls MATLAB function linprog() to perform linear
%     programming with the SDDELP energy objective function.
%     There are a total of 2n-2 unknowns that we wish to solve for:
%     n first derivatives and n-2 slack variables.
%     The slack variables are defined to be equal to the absolute value
%     of the second derivative discontinuity at each data point.
%     linprog() will minimize E_D, the sum of the n-2 slack variables.
%     E_D = FK
%         = [0 0 ... 0 1 1 .. 1][f'(1) f'(2) ... f'(n) s(2) s(3) ... s(n-1)]^T
%            n zeros   n-2 ones    first derivatives    slack variables
%
%     The full syntax for linprog() is:
%     linprog(coefficients of objective function (F),
%          linear inequality constraints: coefficient matrix (A Aabs),
%          linear inequality constraints: free variable vector (B Babs),
%          linear equality constraints: coefficient matrix, (Aeq)
%          linear equality constraints: free variable vector(Beq)
%     )
%
%     If linprog() indicates that a C2 solution exists, then we call
%     MATLAB function fmincon() to perform constrained minimization to the
%     objective function in energy(), in file energy.m.
%     The energy function will be called with input X and Y.
%     The MCSE energy measure is used.
%     The full syntax for fmincon() is:
%
%     fmincon(objective function ('energy'),
%          initial guess (previous derivative values and 0 for slack vars),
%          linear inequality constraints: coefficient matrix (A),
%          linear inequality constraints: free variable vector (B),
%          linear equality constraints: coefficient matrix, (Aeq)
%          linear equality constraints: free variable vector(Beq)
%          lower bound of first derivatives (null),
%          upper bound of first derivatives (null),
%          nonlinear constraints function (null),
%          options (default=null),
%          input parameters passed to objective function (X),
%          input parameters passed to objective function (Y),
%     )
% ------------------------------------------------------------------------
```

```
function D = monotone(X,Y)

[A,B,Aeq,Beq] = mono_constr(X,Y);
[Aabs,Babs]   = abs_constr(X,Y);

n = length(Y);
F = [zeros(1,n) ones(1,n-2)];

[D,E_D] = linprog(F,[A' Aabs']',[B' Babs']',Aeq,Beq);

% if E_D is small, then a C2 solution exists and we compute the MCSE (optimal)
if E_D <= 1e-10*max(D)
        [C2Aeq, C2Beq] = C2_constr(X,Y);
        Aeq = [Aeq' C2Aeq']';  % append C2 constraints to Aeq
        Beq = [Beq' C2Beq']';  % append C2 constraints to Beq
        D   = fmincon('energy',[D zeros(1,n-2)],A,B,Aeq,Beq,[],[],[],[],X,Y);
end
% ------------------------------------------------------------------------
% FUNCTION:
%    [A,B,Aeq,Beq] = mono_constr(X,Y);
% INPUT:
%    X <- monotonic input data vector (size: n x 1)
%    Y <- monotonic input data vector (size: n x 1)
% OUTPUT:
%    A <- monotonicity constraint coefficient matrix (size: 6k1 x 2n-2)
%    B <- free-variable vector (size: 6k1 x 1)
%    Aeq <-linear equality constraint coefficient matrix (size: 2k2 x 2n-2)
%    Beq <-linear equality constraint free variable vector (size: 2k2 x 2n-2)
%    (See below for details about k1 and k2)
% DESCRIPTION:
%    AD <= B are the inequality monotonicity constraints, where D is
%    the 2n-2 vector of unknowns that we wish to solve for:
%    n first derivatives and n-2 slack variables.
%    The function visits all intervals and determines which monotonicity
%    constraints to apply, based on whether the data is increasing,
%    decreasing, or constant.
%    CASE #1: increasing and decreasing intervals
%        A and B are updated with 6 monotonicity constraints.
%    CASE #2: constant (horizontal) interval
%        Aeq and Beq are updated to force y' to 0 at both ends.
%    CASE #3: interval is part of local extrema
%        No monotonicity constraints are imposed.
%
```

```
%     A,B: 6k1 x 2n-2 arrays, where k1 = #stricly inc/dec monotone intervals
%     Aeq,Beq: 2k2 x 2n-2 arrays, where k2 = #constant (horizontal) intervals
% ------------------------------------------------------------------------
function [A,B,Aeq,Beq] = mono_constr(X,Y)

n = length(Y);
S = sign(diff(Y));        % 1=increasing; -1 =decreasing; 0=constant
for i=1:n-2
    if S(i) == -S(i+1) & abs(S(i)) == 1;    % local extrema
            S(i   )=2;                      % flag condition at both
            S(i+1)=2;                       % intervals of local extrema
    end;
end;
A=[]; B=[]; Aeq=[]; Beq=[];
for i=1:n-1
    switch S(i)
    case 0
        Atmp=zeros(2,2*n-2);
        Atmp(1,i )=1;               % coefficient of y'(i)
        Atmp(2,i+1)=1;              % coefficient of y'(i+1)
        Aeq=[Aeq' Atmp']';          % append unity coeffs to Aeq
        Beq=[Beq' 0 0]';            % append zeros to Beq for y'=0
    case {1,-1}
        As=zeros(6,2*n-2);
        Bs=zeros(6,1);
        m = (Y(i+1)-Y(i)) / (X(i+1)-X(i));
        As(1,i) = -1; As(1,i+1) =  0; Bs(1) = 0;
        As(2,i) =  0; As(2,i+1) = -1; Bs(2) = 0;
        As(3,i) =  1; As(3,i+1) = -1; Bs(3) = 3*m;
        As(4,i) = -1; As(4,i+1) =  1; Bs(4) = 3*m;
        As(5,i) =  2; As(5,i+1) =  1; Bs(5) = 9*m;
        As(6,i) =  1; As(6,i+1) =  2; Bs(6) = 9*m;
        A=[A' S(i)*As']';
        B=[B' S(i)*Bs']';
    end;
end;
%-------------------------------------------------------------
%    FUNCTION:
%        [A,B] = C2_constr(X,Y)
%    INPUT:
```

```
%         X <- monotonic input data vector (size: n x 1)
%         Y <- monotonic input data vector (size: n x 1)
%     OUTPUT:
%         A <- C2 constraint coefficient matrix (size: (n-2) x n)
%         B <- free-variable vector (size: (n-2) x 1)
%     DESCRIPTION:
%         AD = B are the equality C2 constraints, where D is
%         the vector of unknown first derivatives (size: n x 1).
%         There are n-2 rows in A and B because there are n-2 interior
%         knots where we can enforce C2 continuity.
%-------------------------------------------------------------------
function [A,B] = C2_constr(X,Y)

n = length(Y);
A = zeros(n-2,2*n-2);
B = zeros(n-2,1);
for i=2:n-1
        dx1 = X(i)     - X(i-1);
        dx2 = X(i+1)   - X(i);
        A(i-1,i-1)  = -1/dx1;
        A(i-1,i)    = -2*(1/dx2+1/dx1);
        A(i-1,i+1)  = -1/dx2;
        B(i-1)      = 3*( Y(i-1)/dx1^2 - Y(i)*(1/dx1^2-1/dx2^2) -
                      Y(i+1)/dx2^2);
end;
%-------------------------------------------------------------------------------
%     FUNCTION:
%         [A,B] = abs_constr(X,Y)
%     INPUT:
%         X <- monotonic input data vector (size: n x 1)
%         Y <- monotonic input data vector (size: n x 1)
%     OUTPUT:
%         A <- absolute 2nd deriv discontinuity coefficient matrix
          (size: 2*(n-2) x 2*n-2)
%         B <- free-variable vector (size: 2*(n-2) x 1)
%     DESCRIPTION:
%         AD < B are the inequality absolute value constraints, where D is the
%         vector of unknown first derivatives and slack vars (size: 2n-2 x 1).
%         We wish to compute A and B to construct the absolute value
%         constraints that will be passed on to linprog().
%         There are 2*(n-2) rows in A and B because there are two absolute
%         value equations for each n-2 interior knot.
%-------------------------------------------------------------------------------
```

```
function [A,B] = abs_constr(X,Y)

n = length(Y);
A = zeros(2*(n-2),n+n-2);
B = zeros(2*(n-2),1);
for i=2:n-1
          dx1 = X(i) - X(i-1);
          dx2 = X(i+1) - X(i);

%         first absolute value constraint for knot i
          A(2*(i-2)+1,i-1)                = -1/dx1;
          A(2*(i-2)+1,i)                  = -2*(1/dx2+1/dx1);
          A(2*(i-2)+1,i+1)                = -1/dx2;
          A(2*(i-2)+1,n+i-1)              = -1;
          B(2*(i-2)+1) = 3*(Y(i-1)/dx1^2 - Y(i)*(1/dx1^2-1/dx2^2) -
                                          Y(i+1)/dx2^2);

%         second absolute value constraint for knot i
          A(2*(i-2)+2,i-1)                =  1/dx1;
          A(2*(i-2)+2,i)                  = 2*(1/dx2+1/dx1);
          A(2*(i-2)+2,i+1)                =  1/dx2;
          A(2*(i-2)+2,n+i-1)              = -1;
          B(2*(i-2)+2)= -3*(Y(i-1)/dx1^2 - Y(i)*(1/dx1^2-1/dx2^2)-
                                          Y(i+1)/dx2^2);
end;
```

B.2. File energy.m

```
%-------------------------------------------------------------------------
%
%energy.m - Compute energy measure E for a given spline.
%          Based on work described in our paper:
%          An Energy-Minimization Framework for
%          Monotonic Cubic Spline Interpolation
%
%Written by: Itzik Alfy and George Wolberg, 2000
%Copyright (C) 2000 by Itzik Alfy and George Wolberg
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%    FUNCTION:
%          e = energy(D,X,Y);
```

```
%     INPUT:
%         D <- first derivative values     (size: n x 1)
%         X <- monotonic input data vector(size: n x 1)
%         Y <- monotonic input data vector (size: n x 1)
%     OUTPUT:
%         e <- energy measure (E)
%     DESCRIPTION:
%         energy() computes the energy of the spline passing through
%         X,Y having first derivative D.
%         energy() is repeatedly called by fmincon() to update D to
          minimize e.
%
%         energy() calls MATLAB function quad8() to integrate over the
%         squared curvature.
%         We are using the Hermite representation of a spline:
%         f(x)=H_0(x)Y(i)+H_1(x)Y(i+1)+dx H_2(x)D(i)+dx H_3(x) D(i+1)
%         Therefore, the squared curvature requires parameters Y, D, and dx.
%         These parameters will be passed to quad8().
%         The full syntax for quad8() is:
%
%         quad8(    function ('curvature2'),
%                       lower limit (0),
%                       upper limit (dx),
%                       relative error tolerance (default=.001),
%                       trace (default=none),
%                       input parameters passed to function (Y(i)),
%                       input parameters passed to function (Y(i+1)),
%                       input parameters passed to function (D(i)),
%                       input parameters passed to function (D(i+1)),
%                       input parameters passed to function (dx),
% )
%-------------------------------------------------------------------------
function e = energy(D,X,Y)

e = 0;
for i=1:length(Y)-1
        dx= X(i+1) - X(i);
        e = e + quad8('curvature2',0,dx,[],[],Y(i),Y(i+1),D(i),D(i+1),dx);
end;
```

B.3. File curvature2.m

```
%-------------------------------------------------------------------------
%
%    curvature2.m -Compute the squared curvature of the spline
%        Based on work described in our paper:
%        An Energy-Minimization Framework for
%        Monotonic Cubic Spline Interpolation
%
%    Written by: Itzik Alfy and George Wolberg, 2000
%    Copyright (C) 2000 by Itzik Alfy and George Wolberg
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%    FUNCTION:
%        P = curvature2(x,Y1,Y2,D1,D2,dx)
%    INPUT:
%        x          <- position vector over an interval
%        Y1         <- data value at left end of interval
%        Y2         <- data value at right end of interval
%        D1         <- first derivative at left end of interval
%        D2         <- first derivative at right end of interval
%        dx         <- interval length
%    OUTPUT:
%        P  <- vector of squared curvature values coinciding with x
%    DESCRIPTION:
%        curvature2() computes the squared curvature of a spline interval
%        at positions given by x. The spline interval is fully specified
%        by data values Y1 and Y2, derivative values D1 and D2, and interval
%        length dx.
%        MATLAB function quad8() calls curvature2() and selects an
%        appropriate choice for x at which to compute the squared curvature.
%-------------------------------------------------------------------------
function P = curvature2(x,Y1,Y2,D1,D2,dx)

u = x / dx;
u2 = u .^ 2;
d1 = (6*(u2-u)*Y1+6*(-u2+u)*Y2+dx*(3*u2-4*u+1)*D1+dx*(3*u2-2*u)*D2) ./ dx;
d2 = ((12*u-6)*Y1 + (-12*u+6)*Y2 + dx*(6*u-4)*D1 + dx*(6*u-2)*D2) ./ dx^2;
P = (d2.^2) ./ ((1 + d1.^2).^2.5);
```

References

[1] H. Akima, A new method of interpolation and smooth curve fitting based on local procedure, J. Assoc. Comput. Mach. 17 (1970) 589–602.

[2] R.K. Beatson, H. Wolkowicz, Post-processing piecewise cubics for monotonicity, SIAM J. Numer. Anal. 26 (2) (1989) 480–502.
[3] P.L. Chebyshev, Sur les fonctions qui diffèrent le moins possible de zèro, J. Math. Pures Appl. 19 (1876) 319–346.
[4] P. Costantini, On monotone and convex spline interpolation, Math. Comput. 46 (173) (1986) 203–214.
[5] P. Costantini, Co-monotone interpolating splines of arbitrary degree: a local approach, SIAM J. Sci. Statist. Comput. 8 (6) (1987) 1026–1034.
[6] P. Costantini, An algorithm for computing shape-preserving interpolating splines of arbitrary degree, J. Comput. Appl. Math. 22 (1988) 89–136.
[7] P. Costantini, M. Rossana, An algorithm for computing shape-preserving cubic spline interpolation to data, Calcolo 21 (1984) 295–305.
[8] P. Costantini, M. Rossana, Monotone and convex cubic spline interpolation, Calcolo 21 (1984) 281–294.
[9] C. de Boor, A Practical Guide to Splines, Springer, New York, 1978.
[10] R. Delbourgo, J. Gregory, Shape preserving piecewise rational interpolation, SIAM J. Sci. Statist. Comput. 6 (1985) 967–976.
[11] S.C. Eisenstat, K.R. Jackson, J.W. Lewis, The order of monotone piecewise cubic interpolation, SIAM J. Numer. Anal. 22 (6) (1985) 1220–1237.
[12] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, Computer Graphics: Principles and Practice, 2nd Edition, Addison-Wesley, Reading, MA, 1990.
[13] F.N. Fritsch, Energy comparison of Wilson Fowler splines with other interpolating splines, in: G.E. Farin (Ed.), Geometric Modeling: Algorithms and New Trends, SIAM, Philadelphia, PA, 1987.
[14] F.N. Fritsch, J. Butland, A method for constructing local monotone piecewise cubic interpolants, SIAM J. Sci. Statist. Comput. 5 (2) (1984) 300–304.
[15] F.N. Fritsch, R.E. Carlson, Monotone piecewise cubic interpolation, SIAM J. Numer. Anal. 17 (2) (1980) 238–246.
[16] J. Gregory, R. Delbourgo, Piecewise rational quadratic interpolation to monotonic data, IMA J. Numer. Anal. 2 (1982) 123–130.
[17] T. Huynh, Accurate monotone cubic interpolation, SIAM J. Numer. Anal. 30 (1) (1993) 57–100.
[18] M.M. Malcolm, On the computation of nonlinear spline functions, SIAM J. Numer. Anal. 14 (2) (1977) 254–282.
[19] D. McAllister, E. Passow, J. Roulier, Algorithms for computing shape preserving splines interpolations to data, Math. Comput. 31 (1977) 717–725.
[20] G.M. Nielson, Some piecewise polynomial alternatives to splines under tension, in: R. Barnhill, R. Riesenfeld (Eds.), Computer Aided Geometric Design, Academic Press, New York, 1974, pp. 209–235.
[21] S. Pruess, Properties of splines in tension, J. Approx. Theory 17 (1976) 86–96.
[22] S. Pruess, Alternatives to the exponential spline in tension, Math. Comput. 33 (1979) 1273–1281.
[23] S. Pruess, Shape preserving C^2 cubic spline interpolation, IMA J. Numer. Anal. 13 (1993) 493–507.
[24] S.S. Rao, Engineering Optimization, 3rd Edition, Wiley, New York, 1996.
[25] D. Rogers, J.A. Adams, Mathematical Elements for Computer Graphics, McGraw-Hill, New York, 1990.
[26] L.L. Schumaker, On shape oreserving quadratic spline interpolation, SIAM J. Numer. Anal. 20 (1983) 854–864.
[27] D. Schweikert, An interpolation curve using splines in tension, J. Math. Phys. 45 (1966) 312–317.
[28] L.F. Shampine, R.C. Allen Jr., S. Pruess, Fundamentals of Numerical Computing, Wiley, New York, 1997.
[29] H. Späth, Exponential spline interpolation, Computing 4 (1969) 225–233.
[30] G. Wolberg, Digital Image Warping, IEEE Computer Society Press, Los Alamitos, CA, 1990.
[31] G. Wolberg, I. Alfy, Monotonic cubic spline interpolation, Proceedings of Computer Graphics International, 1999, pp. 188–195.
[32] P. Wolfe, The simplex method for quadratic programming, Econometrica 27 (1959) 382–398.