INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

NEW RESULTS ON MONOTONE CUBIC SPLINES AND HIERARCHICAL SCATTERED DATA INTERPOLATION

by

Yitzhak Alfy

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

2000

This manuscript has been read and approved by the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

9/27/00
Date

George Wolberg
Professor GEORGE WOLBERG
Chair of Examining Committee

Sept. 27, 2000
Date

Mumtaz Kassir
Dean MUMTAZ KASSIR
Executive Officer

Professor SAMIR AHMED

Professor BARRY GROSS

Professor DEMETRI TERZOPOULOS

Professor YEHOSHUA ZEEVI

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

ABSTRACT

NEW RESULTS ON MONOTONE CUBIC SPLINES AND HIERARCHICAL SCATTERED DATA INTERPOLATION

Yitzhak Alfy

Adviser: Professor George Wolberg

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) control 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.

One goal of this thesis is to determine the *smoothest* possible curve that passes through its control points while simultaneously satisfying the monotonicity constraint. We derive a novel optimization-based approach and analyze its merits against other work in the field. The following related topics have been investigated: (1) fitting monotone cubic spline to data with changing monotonicity; (2) construction of monotone and convex/concave interpolating cubic splines; (3) construction of C^2 interpolating cubic splines with extra knots inserted between each pair of data points; (4) smoothing noisy data with C^2 cubic splines; and (5) fitting monotone piecewise bicubic interpolation to scattered data and data on rectangular meshes

Surface fitting to scattered data is another important problem of great interest to many fields of science and technology. Although reconstruction from uniform samples is a well-known subject, the theory of reconstruction from nonuniform samples is not equally mature. Various local and global methods abound. In local methods, the value of the surface at some position depends only on the data points that are in its neighborhood. In global methods, the value depends upon all data points. Generally, the construction of an approximating surface by local

methods requires less computational effort than global methods, especially when dealing with large data sets. Surfaces produced by global methods, however, have better approximation properties.

In recent work, Wolberg and his colleagues introduced a multiresolution cubic B-spline approximation (MBA) algorithm. The hierarchical MBA algorithm, which is both fast and global, was demonstrated to produce high fidelity reconstruction in diverse applications such as image reconstruction, image warping, and object reconstruction. This thesis presents an analysis of key mathematical properties of the MBA algorithm. The resulting analysis suggests an alternate global hierarchical approach which we call *multiresolution filtering approximation (MFA)*. The MFA scheme is intuitive and simple to implement. Numerical experiments show that the MBA and MFA schemes yield similar quantitative performance for synthetic test functions. The applicability of the MFA algorithm for image processing applications has been examined. We applied the MFA algorithm to image reconstruction from nonuniform samples.

ACKNOWLEDGEMENTS

My deepest thanks go to advisor, Prof. George Wolberg, for his constant encouragement and patience throughout the course of my study. His guidance and support made the whole experience a truly special one for me.

I wish to thank my committee members: Professors Samir Ahmed, Barry Gross, Thao Nguyen, Demetri Terzopoulos, and Yehoshua Zeevi, for their diligence in reviewing this thesis and making valuable comments on this work.

I would like to express my gratitude and respect to my parents for all their love and support.

This work is dedicated to my beloved wife Yifat and my children Dor, Oz, and Adi. I thank them for all the love they bring to my life.

Contents

List of Figures

List of Tables

Chapter 1

MONOTONE CUBIC SPLINE INTERPOLATION

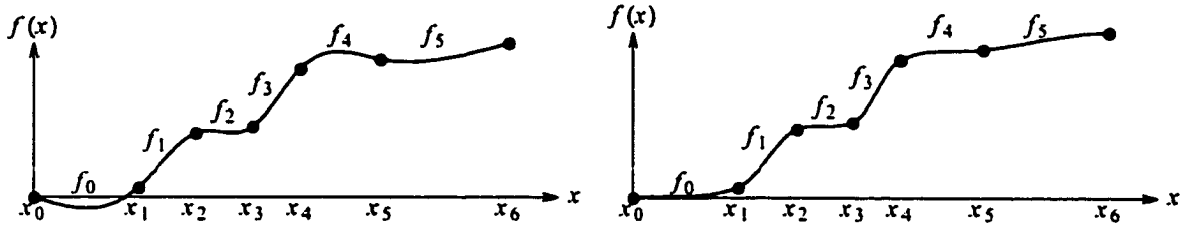
1.1 INTRODUCTION

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 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. 1.1(a). Although the seven data points are monotonically increasing in $f(x_i)$ for $0 \leq i \leq 6$, the cubic spline is not monotonic: it contains overshoots and undershoots, i.e., wiggles.

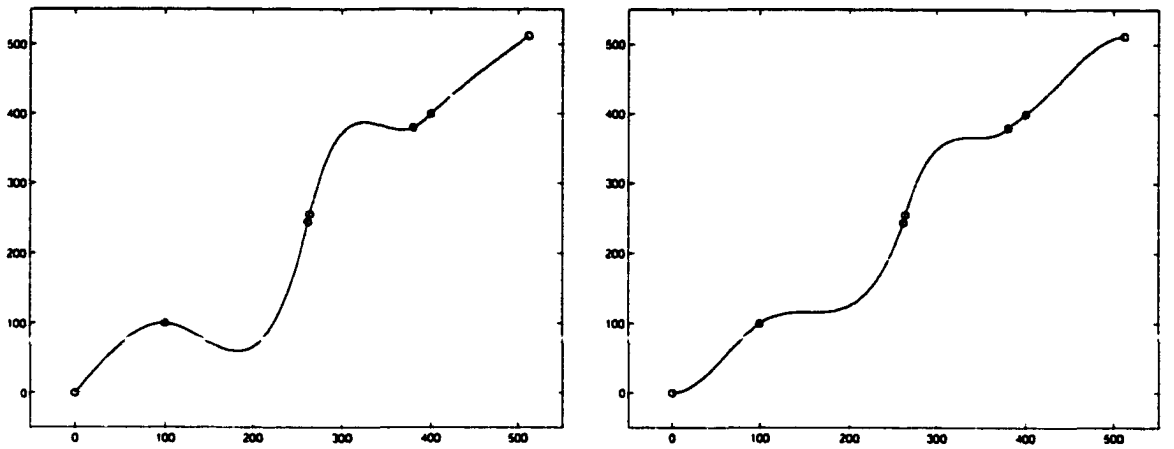


(a) non-monotone spline (b) monotone spline

Figure 1.1: Interpolating cubic splines.

Monotonicity is desirable in many applications, including image warping. Image warping is a growing branch of image processing that deals with the geometric transformation of digital images. A geometric transformation is an operation that redefines the spatial relationship between points in an image. Geometric transformations were originally introduced to invert (correct) these distortions and to allow the accurate determination of spatial relationships and scale. This requires us to first estimate the distortion model, usually by means of reference points which may be accurately marked or readily identified (e.g., road intersections and land-water interface). In the vast majority of cases, the coordinate transformation representing the distortion is modeled as a bivariate polynomial whose coefficients are obtained by minimizing an error function over the reference points. Usually, a second-order polynomial suffices, accounting for translation, scale, rotation, skew, and pincushion effects. For more local control, affine transformations and piecewise polynomial mapping functions are widely used, with transformation parameters varying from one region to another. We shall be interested in more local control whereby point-to-point correspondences between two images are supplied.

A mapping function can be derived by fitting a smooth surface through the sparse and nonuniform point constraints. The resulting analytic mapping function can then be evaluated over all pixels, allowing us to determine where each pixel maps. This is a necessary step to perform image resampling. The construction of a smooth mapping function that satisfies the point constraints may potentially introduce foldover problems if the surface is not monotonic.



(a) non-monotone mapping function (b) monotone mapping function

Figure 1.2: Mapping functions.

That is, the mapping function may cause the output pixel stream to reverse direction and cover previously computed pixels. We therefore seek to find the smoothest monotone surface that passes through the monotone point-to-point constraints. Consider the example in Fig. 1.2. Fig. 1.2(a) shows a 1-D mapping function specified by the six marked constraints. The horizontal and vertical axes respectively represent the input and output positions along an image row. A smooth cubic spline was fitted to the monotone constraints, producing the non-monotone curve in Fig. 1.2(a). Fig. 1.2(b) shows a smooth montone interpolant. For simplicity, we demonstrate the effects of these mappings functions by applying them to each row of an input image. Fig. 1.3(a) shows an image consisting of radial lines. Fig. 1.3(b) is produced by resampling the input image using the non-monotone mapping function of Fig. 1.2(a). Notice that foldover problems caused the stream of input pixels to fold back upon themselves and overwrite output pixels to produce an image replete with artifacts. We can ameliorate some of the foldover artifacts by applying a more expensive resampler that checks for the foldover problem and avoids overwriting pixels that have been previously drawn. The resulting image is shown in Fig. 1.3(c). Note, however, that real-time image resampling hardware is more cheaply built if it can assume the input stream mapping function is monotone. Fig. 1.3(d) shows the effect of applying the monotone mapping function of Fig. 1.2(b). Notice that there is less information loss and no foldover problems.

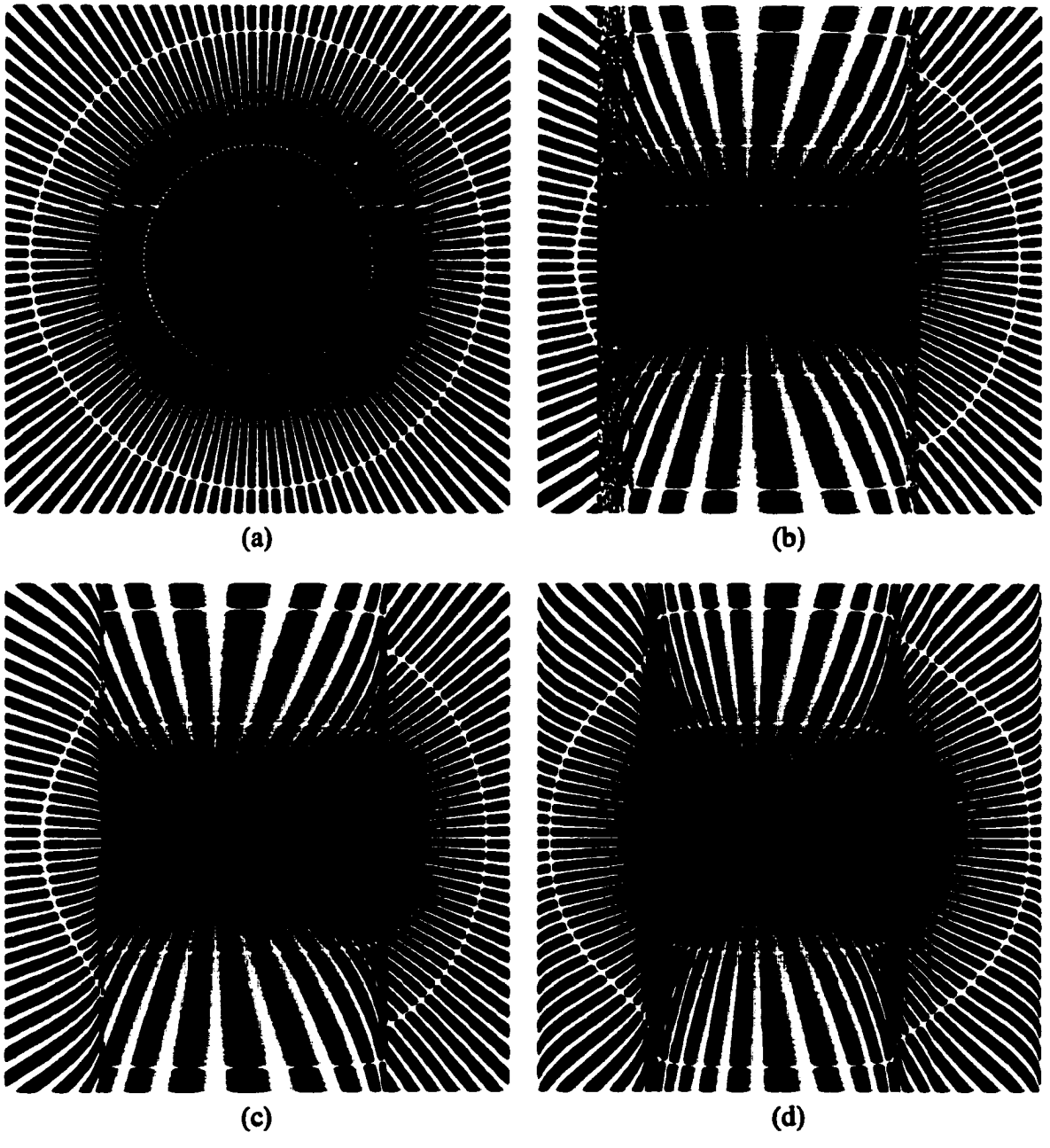


Figure 1.3: Warped images. (a) input image; (b) warped image with non-monotone mapping function. Notice foldover artifacts; (c) Same as (b) except that foldover was clamped; and (d) warped image with monotone mapping function. No foldover. More image detail is preserved.

The goal of this work is to derive the *smoothest* possible cubic spline that simultaneously interpolates the data and satisfies the monotonicity constraint. Fig. 1.1(b) shows an example of such a C^2 monotone spline overlaid on the non-monotone curve. 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 1.2 and a review of cubic spline interpolation in Section 1.3. The monotonicity constraint is discussed in Section 1.4. This thesis advances an energy minimizing framework to produce monotone curves. Optimization-based solutions central to this framework are introduced in Section 1.5. There is a large family of monotone curves that interpolate the data. Section 1.6 derives bounds on the error between any two such curves. Section 1.7 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 1.8. In addition, extensions to shape-preserving splines, knot insertion, and data smoothing are presented in Section 1.8 as well. The various methods are compared in Section 1.9. Finally, a discussion and summary of the work is presented in Section 1.10 and Section 1.11, respectively. Appendix A derives the monotonicity constraints. Appendix B lists MATLAB code to demonstrate the monotonic cubic spline interpolation algorithm.

1.2 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 [15, 7]. 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 [77]. Various other exponential and cubic spline interpolants were considered in [80, 68, 69, 70, 27]. 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 [65], 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 [28, 51], 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 [45]. 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 [34]. 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 [32].

In 1984, Fritsch and Butland proposed a modified technique to simplify the Fritsch-Carlson algorithm [44]. 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 [56]. 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" [56].

In [23, 22, 18, 19], several algorithms were proposed to compute shape preserving splines that are monotone and convex. The algorithms are based on [45] and iteratively compute a set of first derivatives that simultaneously satisfy the monotonicity and convexity constraints.

Schumaker [76] 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 [27, 70, 20, 7, 71]. 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 [71]. Additional breakpoints, however, require more storage and increased search time during evaluation [45].

This thesis presents several key enhancements beyond the work described in [87]. 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.

1.3 CUBIC SPLINES: A REVIEW

A cubic spline $f(x)$ interpolating on the partition $x_0 < x_1 < ... < x_{n-1}$ 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 $k = 1, ..., n-2$, such that $y'_k = f'(x_k)$ and $y''_k = f''(x_k)$ are continuous. Examples of cubic splines passing through $n = 7$ data points are illustrated in Fig. 1.1.

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

$$f_k(x) = a_k(x - x_k)^3 + b_k(x - x_k)^2 + c_k(x - x_k) + d_k \tag{1.1}$$

where

$$a_k = \frac{1}{\Delta x_k^2}\left(-2\frac{\Delta y_k}{\Delta x_k} + y'_k + y'_{k+1}\right) \tag{1.2a}$$

$$b_k = \frac{1}{\Delta x_k}\left(3\frac{\Delta y_k}{\Delta x_k} - 2y'_k - y'_{k+1}\right) \tag{1.2b}$$

$$c_k = y'_k \tag{1.2c}$$

$$d_k = y_k \tag{1.2d}$$

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 $k = 0, ..., n-2$.

The expressions for the cubic polynomial coefficients in Eq. (1.2) 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 [73] and [85].

The role of position data and derivatives in cubic splines can be made explicit by rewriting Eq. (1.1) as

$$f_k(x) = H_0(\frac{x - x_k}{\Delta x_k})y_k + H_1(\frac{x - x_k}{\Delta x_k})y_{k+1} + \Delta x_k H_2(\frac{x - x_k}{\Delta x_k})y'_k + \Delta x_k H_3(\frac{x - x_k}{\Delta x_k})y'_{k+1} \quad (1.3)$$

where H_0, H_1, H_2, and H_3 are the cubic Hermite basis functions [34], defined over $0 \leq u \leq 1$:

$$H_0(u) = 2u^3 - 3u^2 + 1 \quad (1.4a)$$

$$H_1(u) = -2u^3 + 3u^2 \quad (1.4b)$$

$$H_2(u) = u^3 - 2u^2 + u \quad (1.4c)$$

$$H_3(u) = u^3 - u^2 \quad (1.4d)$$

The Hermite basis functions are derived directly from Eq. (1.2) 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 curves explicitly represents the function as a linear combination of position and derivative values. In contrast, Eq. (1.1) represented the cubic curve as a linear combination of powers of x with the position and derivate values embedded in the coefficients.

Fig. 1.4 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:

$$y'_k = \alpha_k m_k \quad (1.5a)$$

$$y'_{k+1} = \beta_k m_k \quad (1.5b)$$

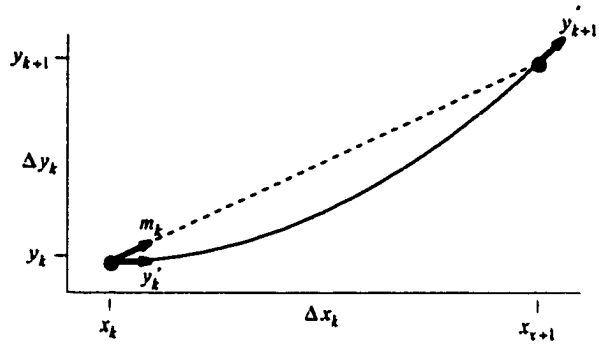


Figure 1.4: A single cubic polynomial segment.

for $\alpha_k \geq 0$ and $\beta_k \geq 0$. The cubic curve in Fig. 1.4 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. 1.5 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 $1 \leq \beta_k \leq 5$ 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.

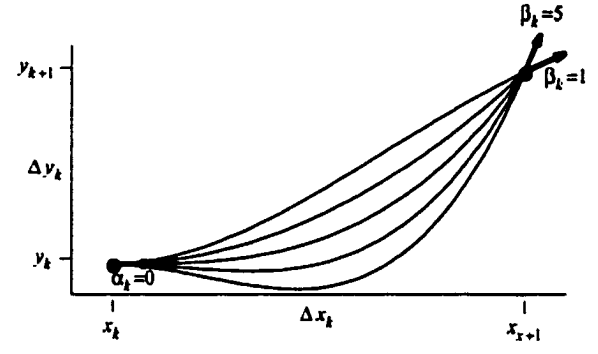


Figure 1.5: A family of interpolating cubic polynomials.

1.4 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 thesis. The conditions derived below closely follow that of [45] and are reviewed here to

make the presentation self-contained. These conditions are further simplified here to yield a fast method for determining monotonicity.

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

$$sgn(y'_k) = sgn(y'_{k+1}) = sgn(m_k) \tag{1.6}$$

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. (1.6) 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 $x_k \leq x \leq x_{k+1}$. 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. (1.1) yields the following cubic curve between (x_k, y_k) and (x_{k+1}, y_{k+1}):

$$f_k(x) = a_k x^3 + b_k x^2 + c_k x + d_k \tag{1.7}$$

where

$$a_k = -2m_k + \alpha_k m_k + \beta_k m_k \tag{1.8a}$$

$$b_k = 3m_k - 2\alpha_k m_k - \beta_k m_k \tag{1.8b}$$

$$c_k = \alpha_k m_k \tag{1.8c}$$

$$d_k = y_k \tag{1.8d}$$

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:

$$f'_k(x) = 3a_k x^2 + 2b_k x + c_k \tag{1.9}$$

$$f_k''(x) = 6a_k x + 2b_k \tag{1.10}$$

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

1.4.1 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. (1.6) is a sufficient condition for monotonicity. This case is depicted in Fig. 1.6(a).

1.4.2 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. (1.6) is satisfied, $f_k(x)$ is monotone. This case is depicted in Fig. 1.6(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. 1.6(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 $f_k(x)$.

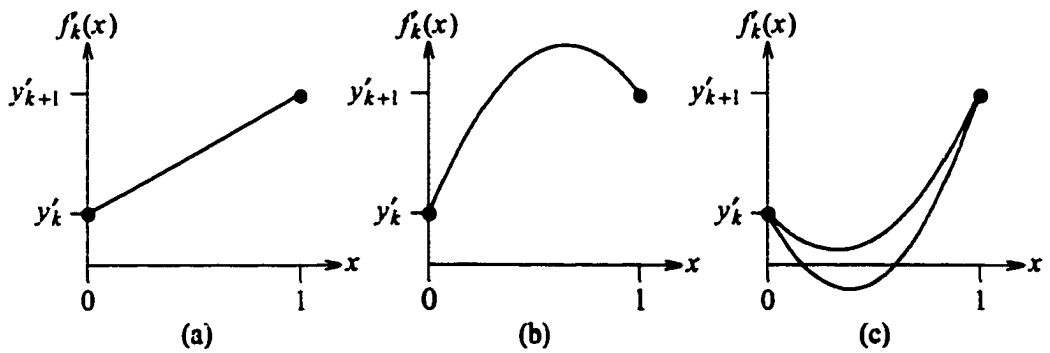


Figure 1.6: Three $f_k'(x)$ cases: (a) linear; (b) concave down; and (c) concave up.

We may distinguish between the two cases in Fig. 1.6(c) by finding the local minima of

$f_k'(x)$. This is derived by solving for x^* in $f_k''(x^*) = 0$:

$$6a_k x^* + 2b_k = 0 \tag{1.11a}$$

$$3(\alpha_k + \beta_k - 2)x^* = 2\alpha_k + \beta_k - 3 \tag{1.11b}$$

$$x^* = \frac{2\alpha_k + \beta_k - 3}{3(\alpha_k + \beta_k - 2)} \tag{1.11c}$$

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 $sgn(f_k'(x^*)) = sgn(m_k)$;

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 $2\alpha_k + \beta_k - 3 \leq 0$ and $\alpha_k + 2\beta_k - 3 \leq 0$, respectively. Condition (3) corresponds to the monotone case depicted in the strictly positive function in Fig. 1.6(c). We may write this condition as follows.

$$f_k'(x^*) = \left(\frac{3a_k(x^*)^2 + 2b_k x^* + c_k}{m_k}\right) m_k \tag{1.12}$$

$$= \left(\alpha_k - \frac{(2\alpha_k + \beta_k - 3)^2}{3(\alpha_k + \beta_k - 2)}\right) m_k \tag{1.13}$$

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

$$\alpha_k - \frac{(2\alpha_k + \beta_k - 3)^2}{3(\alpha_k + \beta_k - 2)} \geq 0 \tag{1.14}$$

Expanding Eq. (1.14) yields

$$\alpha_k^2 + \beta_k^2 - 6\alpha_k - 6\beta_k + \alpha_k\beta_k + 9 < 0$$

$$\alpha_k^2 + \alpha_k(\beta_k - 6) + (\beta_k - 3)^2 < 0 \tag{1.15}$$

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

$$\begin{aligned} f'(x) &= 3a_k x^2 + 2b_k x + c_k = 0 \\ &= Ax^2 + Bx + C = 0 \end{aligned} \tag{1.16}$$

The solution for x in the quadratic expression of Eq. (1.16) is

$$x = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \tag{1.17}$$

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

$$\begin{aligned} 2A = 0 &\rightarrow 3(\alpha_k + \beta_k) - 6 = 0 \\ &\rightarrow \alpha_k + \beta_k = 2 \end{aligned} \tag{1.18}$$

$$\begin{aligned} B^2 < 4AC &\rightarrow 16\alpha_k^2 + 4\beta_k^2 - 48\alpha_k - 24\beta_k + 16\alpha_k b_k + 36 < 12\alpha_k^2 - 24\alpha_k + 12\alpha_k\beta_k \\ &\rightarrow 4\alpha_k^2 + 4\beta_k^2 - 24\alpha_k - 24\beta_k + 4\alpha_k\beta_k + 36 < 0 \\ &\rightarrow \alpha_k^2 + \beta_k^2 - 6\alpha_k - 6\beta_k + \alpha_k\beta_k + 9 < 0 \\ &\rightarrow \alpha_k^2 + \alpha_k(\beta_k - 6) + (\beta_k - 3)^2 < 0 \end{aligned} \tag{1.19}$$

Notice that the expression in Eq. (1.18) is identical to the $a_k = 0$ case given in Section 1.4.1, and the expression in Eq. (1.19) is identical to that of Eq. (1.15). The fact that no solution exists for the above expressions implies that no local extrema is present in the [0,1] interval.

1.4.3 Monotonicity Conditions

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

1) If $\alpha_k + \beta_k - 2 \leq 0$, then $f_k(x)$ is monotone if and only if Eq. (1.6) is satisfied.

2) If $\alpha_k + 2\beta_k - 2 > 0$, then $f_k(x)$ is monotone if and only if Eq. (1.6) *and* one of the following conditions is satisfied:

   (a) $2\alpha_k + \beta_k - 3 \leq 0$

(b) $\alpha_k + 2\beta_k - 3 \leq 0$

(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. 1.7(a). The union of these regions, shown in Fig. 1.7(b), is bounded by lines $\alpha_k = 0$, $\beta_k = 0$, and the ellipse. A simple expression for this region is derived below:

$$\begin{aligned} \alpha_k^2 + \alpha_k(\beta_k - 6) + (\beta_k - 3)^2 &< 0 \\ (\alpha_k + \beta_k)^2 - 6(\alpha_k + \beta_k) + 9 - \alpha_k\beta_k &< 0 \\ [(\alpha_k + \beta_k) - 3]^2 &< \alpha_k\beta_k \\ \alpha_k + \beta_k - 3 &< \sqrt{\alpha_k\beta_k} \\ \alpha_k + \beta_k &< 3 + \sqrt{\alpha_k\beta_k} \end{aligned} \tag{1.20}$$

Eq. (1.20) defines the full ellipse of region IV. The monotonicity region M in Fig. 1.7(b) is expressed in terms of this result as follows:

$$M = \begin{cases} 0 < \alpha_k + \beta_k < 3 + s & \text{if } 0 \leq \beta_k \leq 3 \\ 3 - s < \alpha_k + \beta_k < 3 + s & \text{if } 3 < \beta_k \leq 4 \end{cases} \tag{1.21}$$

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

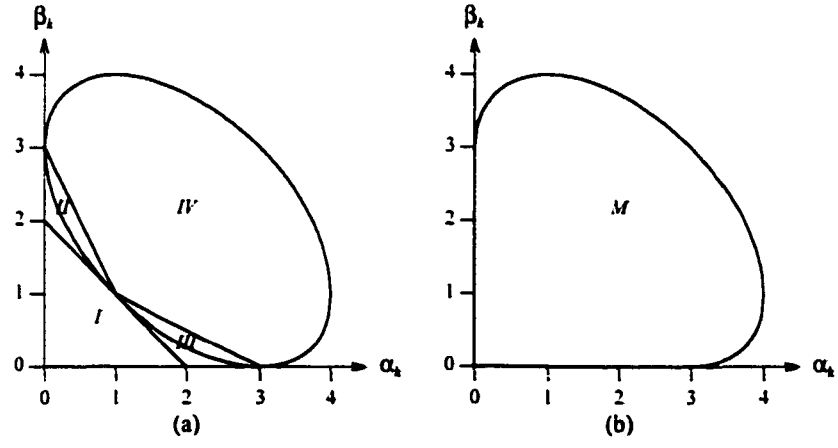


Figure 1.7: (α, β) pairs for a monotonic curve.

1.4.4 Greedy Algorithm

A reasonable approach for generating a smooth monotonic curve is to apply the standard cubic spline interpolation algorithm [85] 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. (1.21) 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. 1.7(b). To do this, we refer to the equation of an ellipse given in Eq. (1.19) 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. (1.19). Setting that quadratic expression to 0 yields the following limits for α_k:

$$\begin{aligned} \alpha_k &= \frac{-(\beta_k - 6) \pm \sqrt{(\beta_k - 6)^2 - 4(\beta_k - 3)^2}}{2} \\ &= \frac{-(\beta_k - 6) \pm \sqrt{\beta_k^2 - 12\beta_k + 36 - 4\beta_k^2 + 24\beta_k - 36}}{2} \\ &= \frac{-(\beta_k - 6) \pm \sqrt{3\beta_k(4 - \beta_k)}}{2} \end{aligned} \tag{1.22}$$

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

$$\alpha_{min} \leq \alpha_k \leq \alpha_{max} \tag{1.23}$$

where

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

and

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

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.

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.

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

$$\begin{aligned} \Omega^i_M &\subset \Omega^i \quad i = 0, 1, 2 \\ \Omega^i &\subset \Omega^{i-1} \quad i = 1, 2 \\ \Omega^i_M &\subset \Omega^{i-1}_M \quad i = 1, 2 \\ \Omega^i &\not\subset \Omega^{i-1}_M \quad i = 1, 2 \end{aligned} \tag{1.24}$$

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. 1.8 for $i = 2$.

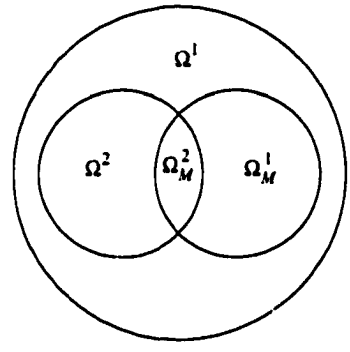


Figure 1.8: Relationship between the cubic spline domains.

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

1.5 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 1.5.1. Since the original cubic spline formulation is not guaranteed to be monotonic, we will impose the monotonicity constraint in Section 1.5.2. This will furnish a solution that may be solved

using quadratic programming. That result will be further simplified in Section 1.5.3 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 1.5.5 that addresses the extent of second derivative discontinuity in the spline. That result is further simplified in Section 1.5.6 to yield a solution that may be solved using linear programming.

1.5.1 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 [64, 27]. The strain energy of the beam is given as the integral of the curvature:

$$E = \int_{x_0}^{x_{n-1}} \frac{f''(x)^2}{(1 + [f'(x)]^2)^{5/2}} dx \tag{1.25}$$

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 [78, 27, 44, 43]:

$$E_L = \int_{x_0}^{x_{n-1}} f''(x)^2 dx \tag{1.26}$$

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. (1.25). The E_L energy measure given in Eq. (1.26) 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. (1.26) among all C^2 cubic polynomials [27, 78].

1.5.2 Linearized Energy (LE_QP)

The expression for E_L may be written in terms of the first derivatives of Eq. (1.3) as follows:

$$E_L = \sum_{k=0}^{n-2} \int_{x=x_k}^{x_{k+1}} f_k''(x)^2 dx \tag{1.27}$$

$$= \sum_{k=0}^{n-2} \int_{x=x_k}^{x_{k+1}} \frac{1}{\Delta x_k^2} \left[H_0''(\frac{x-x_k}{\Delta x_k}) y_k + H_1''(\frac{x-x_k}{\Delta x_k}) y_{k+1} + \Delta x_k H_2''(\frac{x-x_k}{\Delta x_k}) y_k' + \Delta x_k H_3''(\frac{x-x_k}{\Delta x_k}) y_{k+1}' \right]^2 dx$$

where

$$H_0''(u) = 12u - 6 \tag{1.28a}$$

$$H_1''(u) = -12u + 6 \tag{1.28b}$$

$$H_2''(u) = 6u - 4 \tag{1.28c}$$

$$H_3''(u) = 6u - 2 \tag{1.28d}$$

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. 1.7 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. 1.7 with an n-sided polygon. In the example below, we use $n = 6$ and $n = 15$ to demonstrate our approach.

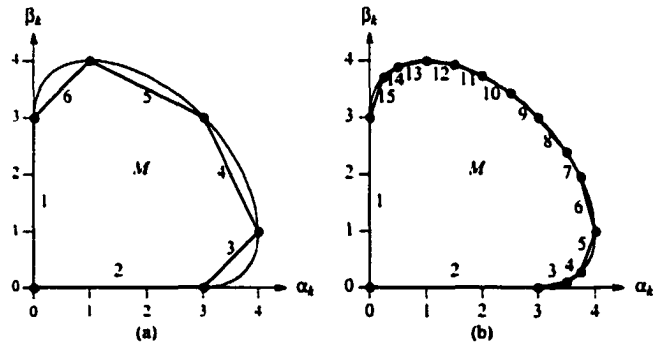


Figure 1.9: Linear approximation of region M: (a) $n = 6$; (b) $n = 15$.

Fig. 1.9 illustrates the two polygons we used to approximate region M in our work. The 6-sided and 15-sided polygons shown in Fig. 1.9 cover 90.53% and 98.79% of region M, respectively. The 6-sided polygon depicted in Fig. 1.9(a) consists of the intersection of the

following six half-planes:

$$\alpha_k \geq 0 \tag{1.29a}$$

$$\beta_k \geq 0 \tag{1.29b}$$

$$\beta_k - \alpha_k + 3 \geq 0 \tag{1.29c}$$

$$\beta_k + 2\alpha_k - 9 \leq 0 \tag{1.29d}$$

$$2\beta_k + \alpha_k - 9 \leq 0 \tag{1.29e}$$

$$\beta_k - \alpha_k - 3 \leq 0 \tag{1.29f}$$

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:

$$\text{sign}(m_k) \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & -1 \\ -1 & 1 \\ 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} y'_k \\ y'_{k+1} \end{bmatrix} \leq |m_k| \begin{bmatrix} 0 \\ 0 \\ 3 \\ 3 \\ 9 \\ 9 \end{bmatrix} \tag{1.30}$$

Note that Eq. (1.30) 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. (1.30).

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 $n = 15$ case are given below:

$$\text{sign}(m_k)\begin{bmatrix} -1.0000 & 0 \\ 0.0000 & -1 \\ 0.2088 & -1 \\ 0.7284 & -1 \\ 2.8540 & -1 \\ 3.8540 & 1 \\ 1.7284 & 1 \\ 1.2088 & 1 \\ 0.8542 & 1 \\ 0.6100 & 1 \\ 0.3900 & 1 \\ 0.1458 & 1 \\ -0.2088 & 1 \\ -0.7284 & 1 \\ -2.8540 & 1 \end{bmatrix} \begin{bmatrix} y'_k \\ y'_{k+1} \end{bmatrix} \leq |m_k| \begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.6264 \\ 2.4450 \\ 10.4160 \\ 16.4160 \\ 8.4450 \\ 6.6264 \\ 5.5626 \\ 4.9521 \\ 4.5121 \\ 4.1458 \\ 3.7912 \\ 3.5314 \\ 3.0000 \end{bmatrix} \tag{1.31}$$

Since E_L is quadratic with respect to y'_k, and Eq. (1.30) and Eq. (1.31) are linear with respect to y'_k, we can use quadratic programming to minimize E_L subject to the following constraints:

1. 2nd derivative continuity $f''(x_k^+) = f''(x_k^-)$ (Eq. (1.32))
2. Monotonicity constraints Eq. (1.30) or Eq. (1.31)

The second derivative continuity constraint can be expressed as follows:

$$-y'_{k-1}\frac{1}{\Delta x_{k-1}} - y'_k\left[\frac{2}{\Delta x_k} + \frac{2}{\Delta x_{k-1}}\right] - y'_{k+1}\frac{1}{\Delta x_k} - y_{k-1}\frac{3}{\Delta x_{k-1}^2} + y_k\left[\frac{3}{\Delta x_{k-1}^2} - \frac{3}{\Delta x_k^2}\right] + y_{k+1}\frac{3}{\Delta x_k^2} = 0 \tag{1.32}$$

Interpolation and first derivative continuity constraints are not required since they are implicit in the cubic Hermite form of Eq. (1.3).

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.

1.5.3 Modified Linearized Energy (LE_LP)

Quadratic programming can be solved by using linear programming. 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 [72, 88]. As a result, we simplify the linearized energy measure to be linear with 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. (1.27) using linear programming, we approximate it with the following expression:

$$\tilde{E}_L = \sum_{k=0}^{n-2} \int_{x_k}^{x_{k+1}} |f_k''(x)| dx \tag{1.33}$$

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 $f_k''(x) + K \geq 0$ everywhere. This yields $\overline{E}_L$, the objective function for our linear programming solution:

$$\overline{E}_L = \sum_{k=0}^{n-2} \int_{x_k}^{x_{k+1}} (f_k''(x) + K) dx \tag{1.34}$$

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

$$f_k''(0) + K = \frac{1}{\Delta x_k^2}(-3y_k + 3y_{k+1} - 2\Delta x_k y_k' - \Delta x_k y_{k+1}') + K \geq 0 \tag{1.35}$$

$$f_k''(\Delta x_k) + K = \frac{1}{\Delta x_k^2}(3y_k - 3y_{k+1} + \Delta x_k y_k' + 2\Delta x_k y_{k+1}') + K \geq 0 \tag{1.36}$$

where $0 \leq x \leq \Delta x_k$ for all k. Eq. (1.34) can therefore be rewritten as

$$\overline{E}_L = \sum_{k=0}^{n-2} \int_{x=0}^{\Delta x_k} (f_k''(x) + K)dx = \sum_{k=0}^{n-2} y'_{k+1} - y'_k + K\Delta x_k = y'_{n-1} - y'_0 + K(x_{n-1} - x_0) \quad (1.37)$$

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

1. 2nd derivative continuity $f''(x_k^+) = f''(x_k^-)$ (Eq. (1.32))
2. 2nd derivative on left side $f_k''(0) + K \geq 0$ (Eq. (1.35))
3. 2nd derivative on right side $f_k''(\Delta x_k) + K \geq 0$ (Eq. (1.36))
4. Monotonicity constraints Eq. (1.30) or Eq. (1.31)

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.

1.5.4 Linearized Energy Properties

In this section, we show that E_L is not a good measure for smoothness. We begin by noting that the E_L values should reflect the curve smoothness. Therefore, we would expect that C^2 solutions should be associated with lower E_L values than C^0 solutions. However, we know that the solution for C^0 monotone constraints is a linear interpolant with $E_L = 0$. This suggests that the E_L measure favors a C^0 solution over any possible C^2 solution, making it an ineffective smoothness measure. This can be understood by noting that the domains of C^0 and C^1 solutions are supersets of the C^2 solution domain (see Fig. 1.8). Furthermore, E_L is based on the (possibly false) assumption that $f'(x)^2 \ll 1$ everywhere. The following example demonstrates this argument. Consider the data set:

$$\begin{aligned} \mathbf{X} &= [0 \quad 1 \quad 2 \quad 3] \\ \mathbf{Y} &= [0 \quad 400 \quad 400 \quad 800] \end{aligned}$$

Fig. 1.10(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. Figures

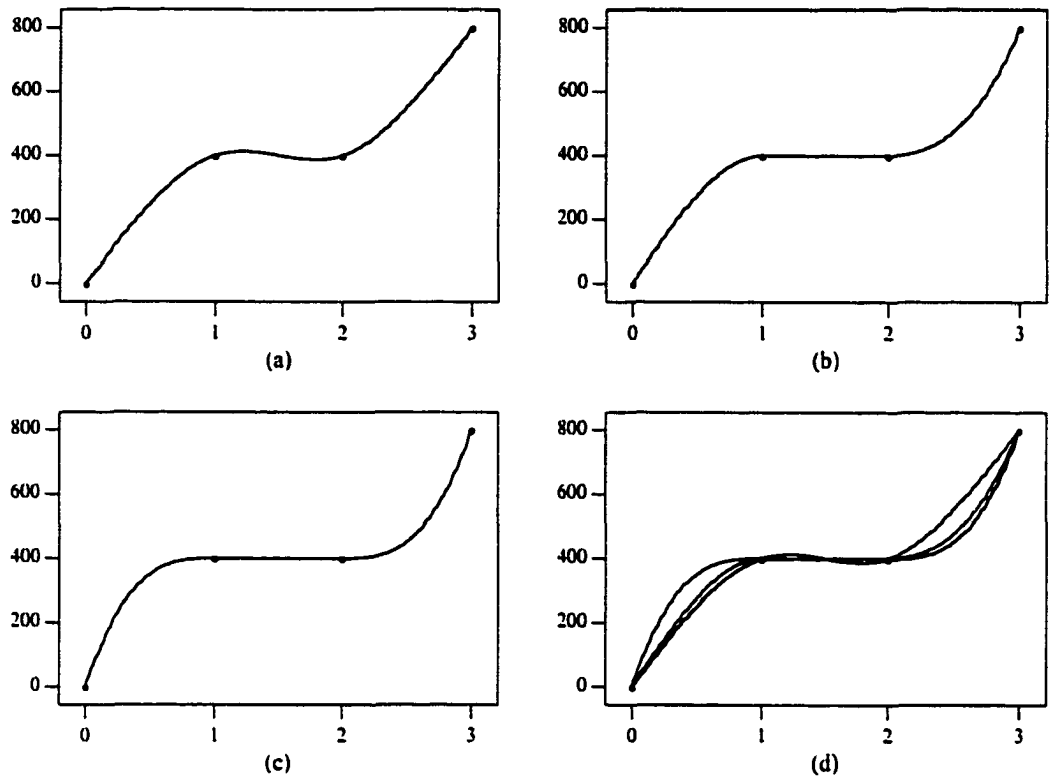


Figure 1.10: (a) FE; (b) C^1 LE_QP; (c) C^2 LE_QP; (d) overlay.

1.10(b) and 1.10(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. 1.10(b) at (1,400) is high and visually prominent.

Table 1.1 summarizes the energy measures for Fig. 1.10. Note that although the C^2 LE_QP curve in Fig. 1.10(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. 1.9(a).

Method	E	E_L
FE	1231.66	640000
LE_QP (C^1)	1599.34	960000
LE_QP (C^2)	58.70	3840000

Table 1.1: Energy measures for Fig. 1.10

1.5.5 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, thereby producing a more natural looking curve. This process requires us to introduce an energy measure based on the second derivative discontinuities:

$$E_D = \sum_{k=1}^{n-2}(f''(x_k^-) - f''(x_k^+))^2 \tag{1.38}$$
$$= \sum_{k=1}^{n-2}\left(\frac{1}{\Delta x_k^2}[-3y_k + 3y_{k+1} - 2\Delta x_k y_k' - \Delta x_k y_{k+1}'] - \frac{1}{\Delta x_{k-1}^2}[3y_{k-1} - 3y_k + \Delta x_{k-1} y_{k-1}' + 2\Delta x_{k-1} y_k']\right)^2$$

A similar objective function was suggested by Nielson in his work on ν-splines [68, 43].

The energy measure E_D can be minimized by using quadratic programming subject to the monotonicity constraints given in Eq. (1.30) or Eq. (1.31). In the minimization formulations for the E_L and $\overline{E}_L$ 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 $\overline{E}_L$ 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.

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

$$\begin{array}{lllllll} \mathbf{X} & = & [0.00 & 1.00 & 1.50 & 2.05 & 2.90] \\ \mathbf{Y} & = & [0.00 & 350.00 & 354.65 & 428.00 & 650.00] \end{array}$$

Method	E	E_L	E_D
FE	855.84	343408.02	0
SDDE_QP	845.22	1601043.30	0.70
LE_QP (C^1)	8.28	392688.93	666782.99
CSE	27.15	1504779.88	0

Table 1.2: Energy measures for Fig. 1.11

Fig. 1.11(a) shows the spline satisfying the free-end condition. Notice that although the data is monotone, the curve is not monotonic. Figs. 1.11(b) and 1.11(c) show the C^1 curve that

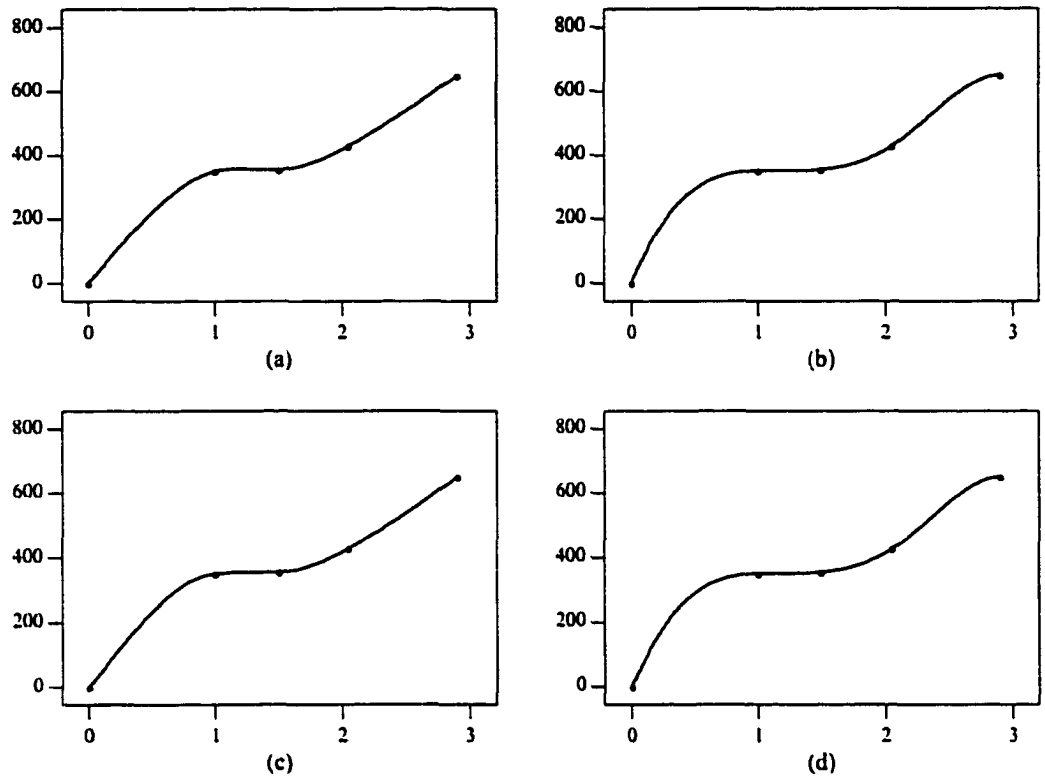


Figure 1.11: (a) FE; (b) SDDE_QP (c) C^1 LE_QP (d) CSE.

solves the quadratic programming problems required in minimizing E_L and E_D, respectively, using the 6-sided polygonal approximation to region M shown in Fig. 1.9(a). Fig. 1.11(d) shows the cubic spline elastica (CSE) obtained by minimizing Eq. (1.25). Table 1.2 summarizes the energy measures for Fig. 1.11. 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.

1.5.6 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 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.

$$\begin{aligned}\widehat{E}_D &= \sum_{k=1}^{n-2} f''(x_k^-) - f''(x_k^+) + K \qquad (1.39)\\ &= \sum_{k=1}^{n-2} \frac{1}{\Delta x_k^2}[-3y_k + 3y_{k+1} - 2\Delta x_k y_k' - \Delta x_k y_{k+1}'] - \frac{1}{\Delta x_{k-1}^2}[3y_{k-1} - 3y_k + \Delta x_{k-1} y_{k-1}' + 2\Delta x_{k-1} y_k'] + K\end{aligned}$$

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

1. Positivity constraint $f''(x_k^-) - f''(x_k^+) + K \geq 0$
2. Monotonicity constraints Eq. (1.30) or Eq. (1.31)

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

$$\overline{E}_D = \sum_{k=1}^{n-2} |f''(x_k^-) - f''(x_k^+)| \qquad (1.40)$$

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:

$$f''(x_k^-) - f''(x_k^+) \leq s_k \qquad (1.41)$$

$$-[f''(x_k^-) - f''(x_k^+)] \leq s_k \qquad (1.42)$$

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

$$\overline{E}_D = \sum_{k=1}^{n-2} s_k \qquad (1.43)$$

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

1. Absolute value constraints Eq. (1.41) and Eq. (1.42)
2. Monotonicity constraints Eq. (1.30) or Eq. (1.31)

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

1.5.7 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:

$$\begin{aligned}\tilde{E}_D &= \text{MAX}\{|f''(x_k^-) - f''(x_k^+)|\} \qquad (1.44)\\ &= \text{MAX}\left\{\left|\frac{1}{\Delta x_k^2}[-3y_k + 3y_{k+1} - 2\Delta x_k y'_k - \Delta x_k y'_{k+1}] - \frac{1}{\Delta x_{k-1}^2}[3y_{k-1} - 3y_k + \Delta x_{k-1} y'_{k-1} + 2\Delta x_{k-1} y'_k]\right|\right\}\end{aligned}$$

The energy measure $\tilde{E}_D$ can be minimized by using linear programming subject to the monotonicity constraints given in Eq. (1.30) or Eq. (1.31). For each discontinuity point we define a slack variable s_k whose value is forced to be the absolute value of the discontinuity, using Eq. (1.41) and Eq. (1.42). 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:

$$s_k \leq S \qquad (1.45)$$

$\tilde{E}_D$ 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 — Eq. (1.41) and Eq. (1.42)
2. Maximum constraint — Eq. (1.45)
3. Monotonicity constraints — Eq. (1.30) or Eq. (1.31)

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

1.6 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 $e_k(x) \leq 0.866|\Delta y_k|$.*

Proof. Evaluating $e_k(x)$ using the Hermite basis functions (Eq. (1.3)) yields

$$\begin{aligned} e_k(x) &= |\Delta x_k H_2(\frac{x - x_k}{\Delta x_k})(f'(x_k) - g'(x_k)) \\ &\quad + \Delta x_k H_3(\frac{x - x_k}{\Delta x_k})(f'(x_{k+1}) - g'(x_{k+1}))| \\ &= \left|\Delta y_k \left(H_2(\frac{x - x_k}{\Delta x_k})(\alpha_k^f - \alpha_k^g) + H_3(\frac{x - x_k}{\Delta x_k})(\beta_k^f - \beta_k^g)\right)\right| \end{aligned} \tag{1.46}$$

where

$$\begin{aligned} f'(x_k) &= \alpha_k^f \Delta m_k \\ f'(x_{k+1}) &= \beta_k^f \Delta m_k \\ g'(x_k) &= \alpha_k^g \Delta m_k \\ g'(x_{k+1}) &= \beta_k^g \Delta m_k \end{aligned}$$

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

$$MAX\widetilde{e_k}(x) = (u^3 - 2u^2 + u)(\alpha_k^f - \alpha_k^g) + (u^3 - u^2)(\beta_k^f - \beta_k^g) \tag{1.47}$$

subject to:

$$\begin{aligned} 0 &\leq u \leq 1 \\ (\alpha_k^f, \beta_k^f) &\in M \\ (\alpha_k^g, \beta_k^g) &\in M \end{aligned}$$

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:

$$u = 0.5$$

$$
\begin{aligned}
(\alpha_k^f, \beta_k^f) &= (3.7320, 0.2679) \\
(\alpha_k^g, \beta_k^g) &= (0.2679, 3.7320) \\
\widetilde{e_k(x)} &= 0.866
\end{aligned}
$$

Combining this result with Eq. (1.46) gives $e_k(x) \leq 0.866|\Delta y_k|$. ■

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 $(\alpha_k^f, \beta_k^f) \in \widetilde{M} \subset M$ then*

$$
If(\alpha_k^g, \beta_k^g) \notin \widetilde{M} \quad MAXMIN e_k(x) = \begin{cases} 0 & \widetilde{M} = M \\ 0.058|\Delta y_k| & \widetilde{M} = \textit{six-sided polygon} \\ 0.1481|\Delta y_k| & \widetilde{M} = \textit{square} \end{cases} \tag{1.48}
$$

(1.49)

Proof. If $(\alpha_k^g, \beta_k^g) \in \widetilde{M}$ 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 $(\alpha_k^g, \beta_k^g) \notin \widetilde{M}$ 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:

$$
MAX\widetilde{e_k(x)} = (u^3 - 2u^2 + u)(\alpha_k^f - \alpha_k^g) + (u^3 - u^2)(\beta_k^f - \beta_k^g) \tag{1.50}
$$

subject to:

$$
\begin{aligned}
0 &\leq u \leq 1 \\
(\alpha_k^f, \beta_k^f) &\in \widetilde{M} \\
(\alpha_k^g, \beta_k^g) &\in M - \widetilde{M}
\end{aligned}
$$

There are two solutions to Eq. (1.50) for the case where $\widetilde{M}$ is the 3×3 square used in [44].

$$
[G_1, F_1, u_1] = \left[(\alpha_k^g, \beta_k^g), (\alpha_k^f, \beta_k^f), u_1\right] = [(4, 1), (3, 1), 0.66] \tag{1.51a}
$$

$$[G_2, F_2, u_2] = 2\left[(\alpha_k^g, \beta_k^g), (\alpha_k^f, \beta_k^f), u_2\right] = [(1,4),(1,3),0.33] \quad (1.51b)$$

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

There are two solutions to Eq. (1.50) for the case where $\widetilde{M}$ is a six-sided polygon:

$$[G_1, F_1, u_1] = \left[(\alpha_k^g, \beta_k^g), (\alpha_k^f, \beta_k^f), u_1\right] = [(3.7320, 0.2679), (3.5, 0.5), 0.5] \quad (1.52a)$$

$$[G_2, F_2, u_2] = \left[(\alpha_k^g, \beta_k^g), (\alpha_k^f, \beta_k^f), u_2\right] = [(0.2679, 3.7320), (0.5, 3.5), 0.5] \quad (1.52b)$$

The error associated with either of these point pairs is $e_k(x) = 0.058|\Delta y_k|$. Fig. 1.12 shows the location of these two solutions for both cases.

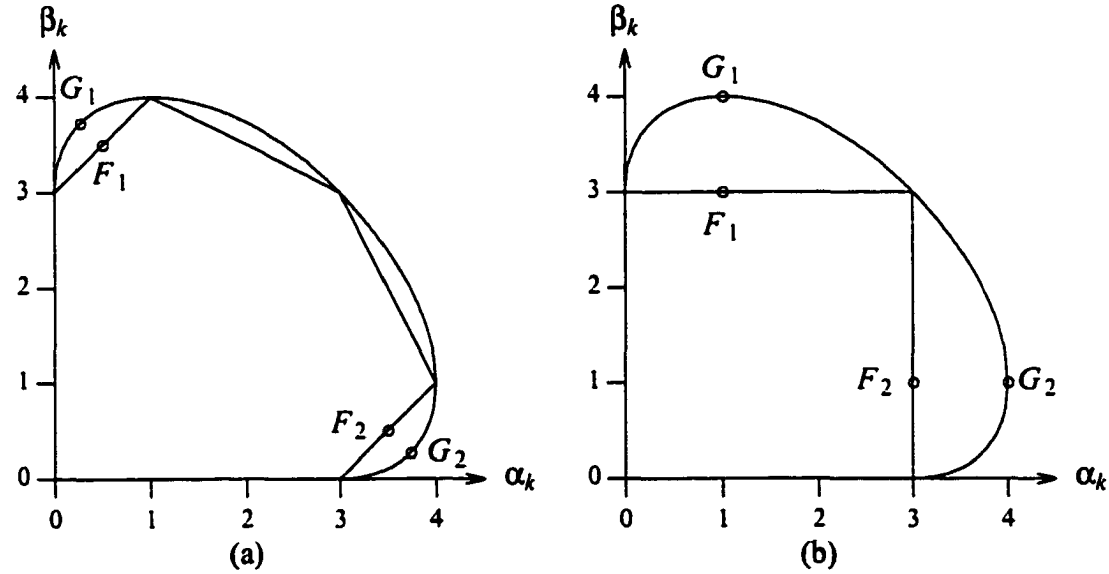


Figure 1.12: MAXMIN solution for (a) six-sided polygon, and (b) square.

■

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 $\widetilde{M}$ we can improve the approximation of $g(x)$.

Example 3 *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. 1.13 shows the approximation error,* $e(x)$, *for the MCSE (Monotone Cubic Spline Elastica) and FB splines, respectively. The FB algorithm utilizes the* 3×3 *square in* M. *The MCSE splines are obtained by minimizing Eq. (1.25) subject to the six*

monotonicity constraints. The MCSE interpolates exactly, i.e. $e(x) = 0$, *in the intervals from 0.2 to 0.7, while the FB curve has approximation error everywhere (except at the data points).*

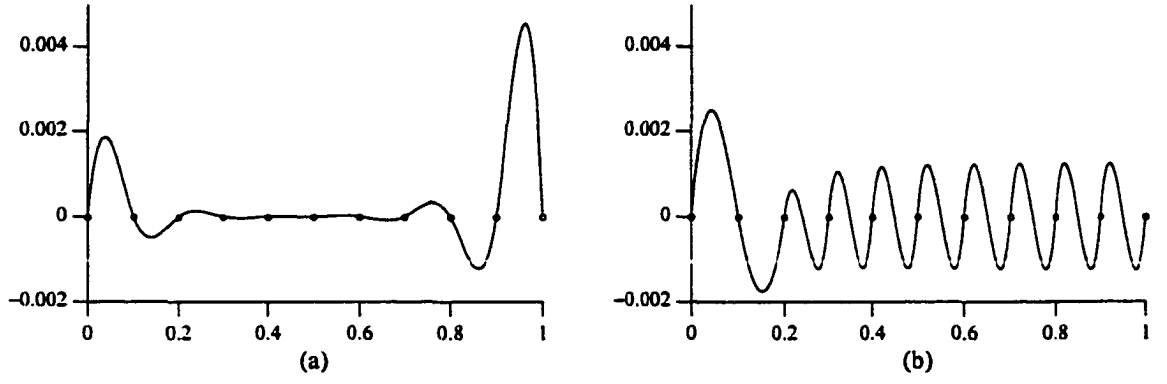


Figure 1.13: Approximation error. (a) MCSE; (b) FB.

1.7 RESULTS

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

1. Cubic Spline Elastica (CSE) - The cubic spline coefficients are obtained by minimizing Eq. (1.25) subject to the constraint that $f(x)$ is a C^2 cubic function. Since the energy given in Eq. (1.25) 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. (1.26), 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 & Butland (FB): The algorithm is described in [44] and implemented in PCHIM.FOR, NETLIB's PCHIPD package for piecewise cubic hermite interpolation by F.N. Fritsch. The first derivatives are calculated using Brodlie's formula with $\alpha = \frac{\Delta x_{k-1}+2\Delta x_k}{3(\Delta x_{k-1}+\Delta x_k)}$:

$$f'(x_k) = \frac{m_{k-1}m_k}{\alpha m_k + (1-\alpha)m_{k-1}} \tag{1.53}$$

5. Second derivative discontinuity energy (SDDE_QP) - Minimize E_D

6. Modified discontinuity energy (SDDE_LP) - Minimize $\overline{E}_D$

7. Minmax discontinuity energy (SDDE_MM) - Minimize $\tilde{E}_D$

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:

$$\begin{array}{lllllllllllll} \mathbf{X} & = & [0 & 1 & 2 & 3 & 4 & 4.5 & 6 & 7 & 7.3 & 9 & 10 & 11] \\ \mathbf{Y} & = & [0 & 1 & 4.8 & 6 & 8 & 13 & 14 & 15.5 & 18 & 19 & 23 & 24.1] \end{array}$$

Figures 1.14 to 1.16 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 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 1.3. Note also that for C^2 curves, e.g., $E_D = 0$, 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

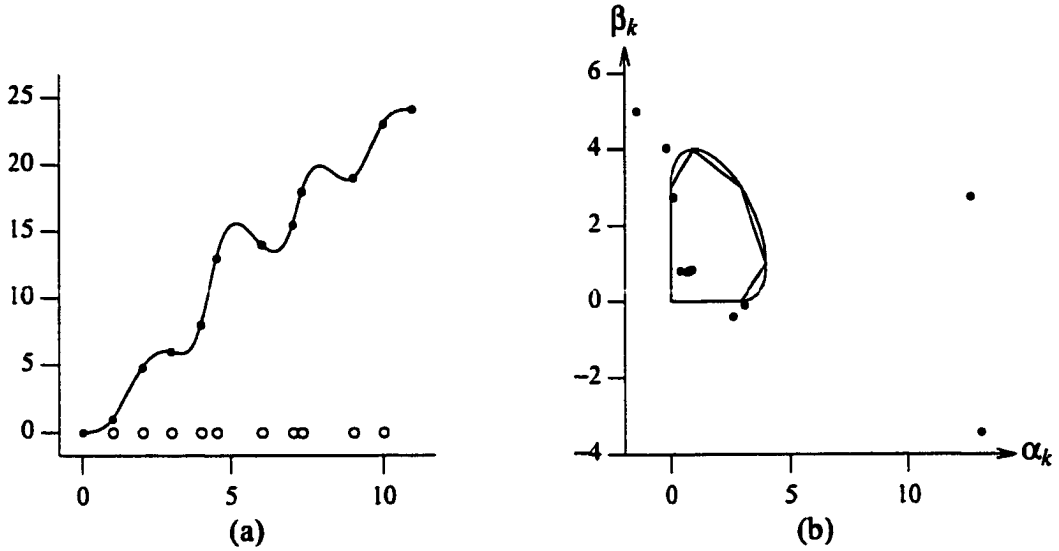


Figure 1.14: Free end (FE): (a) interpolating spline; (b) $\{\alpha, \beta\}$ points

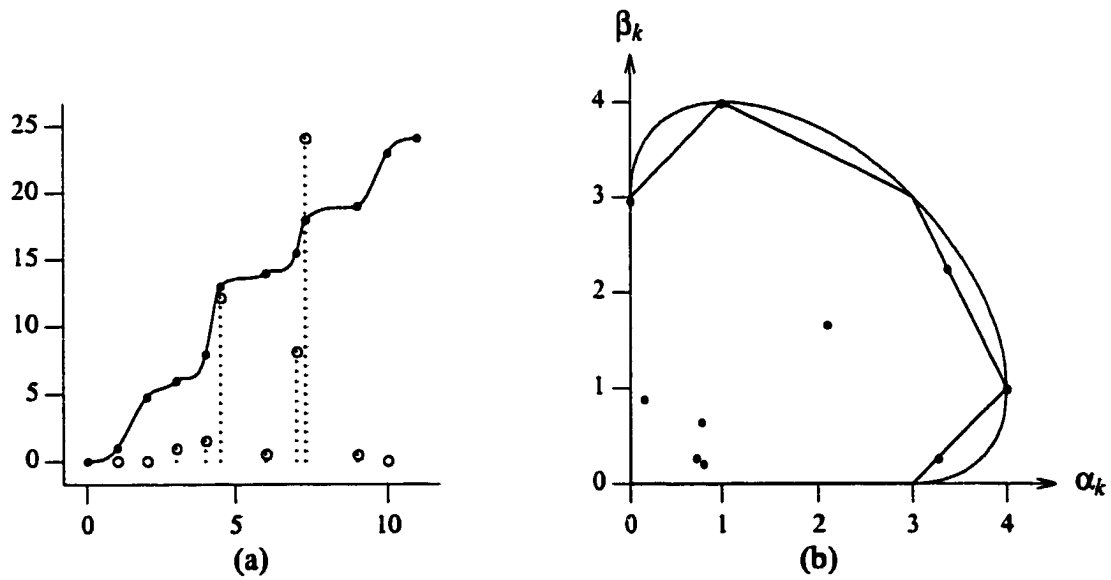


Figure 1.15: Second derivative discontinuity energy (SDDE_LP): (a) interpolating spline; (b) $\{\alpha, \beta\}$ points

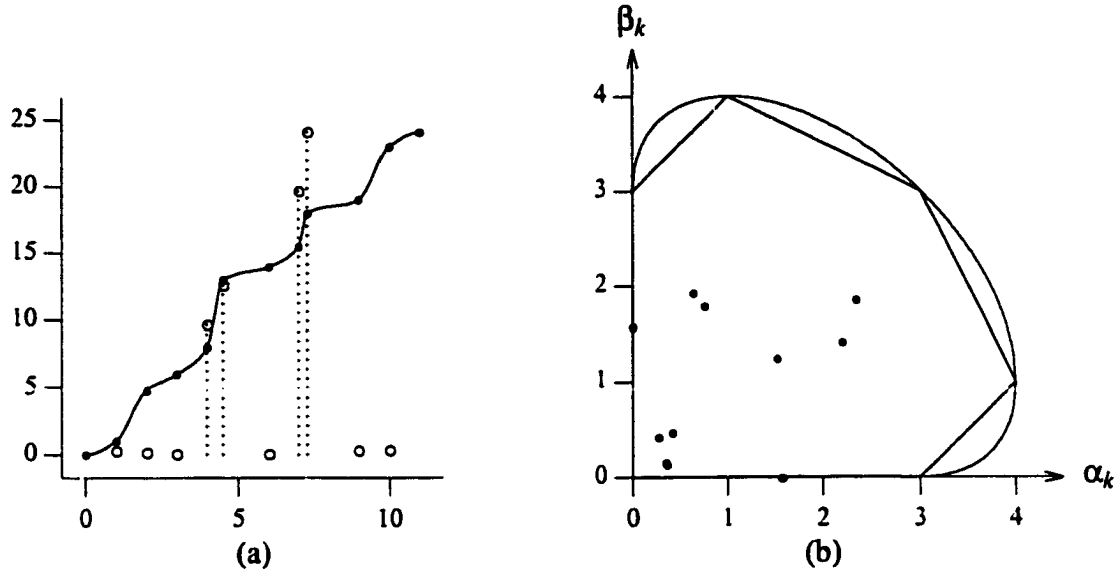


Figure 1.16: Fritsch-Butland (FB): (a) interpolating spline; (b) $\{\alpha, \beta\}$ points

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 1.3 summarizes these results.

Method	E	E_D	$\tilde{E}_D$
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

Table 1.3: Energy measures.

The second example is the third set used by Akima [3].

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

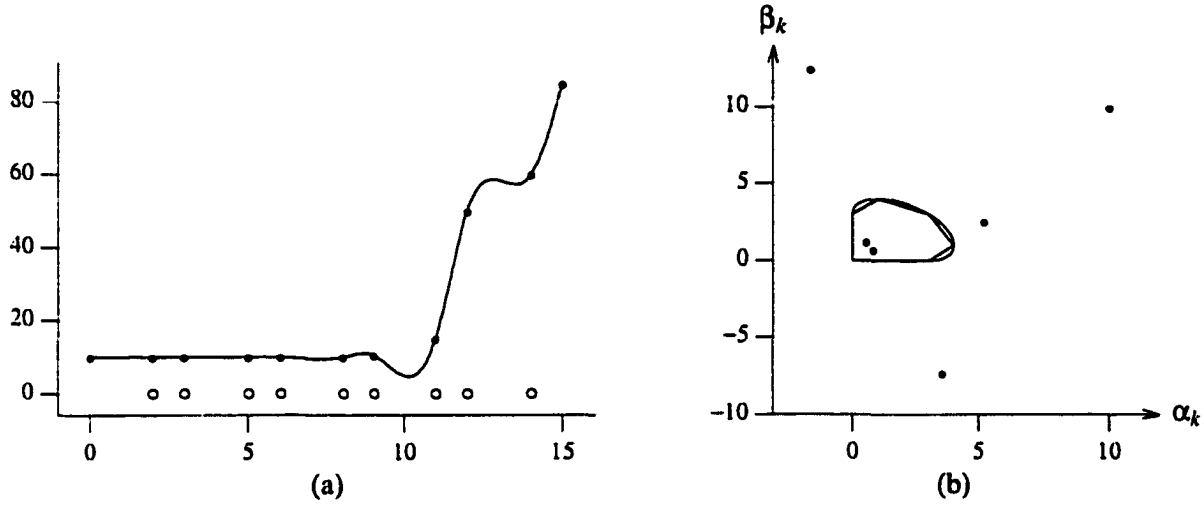


Figure 1.17: Free end (FE): (a) interpolating spline; (b) $\{\alpha, \beta\}$ points

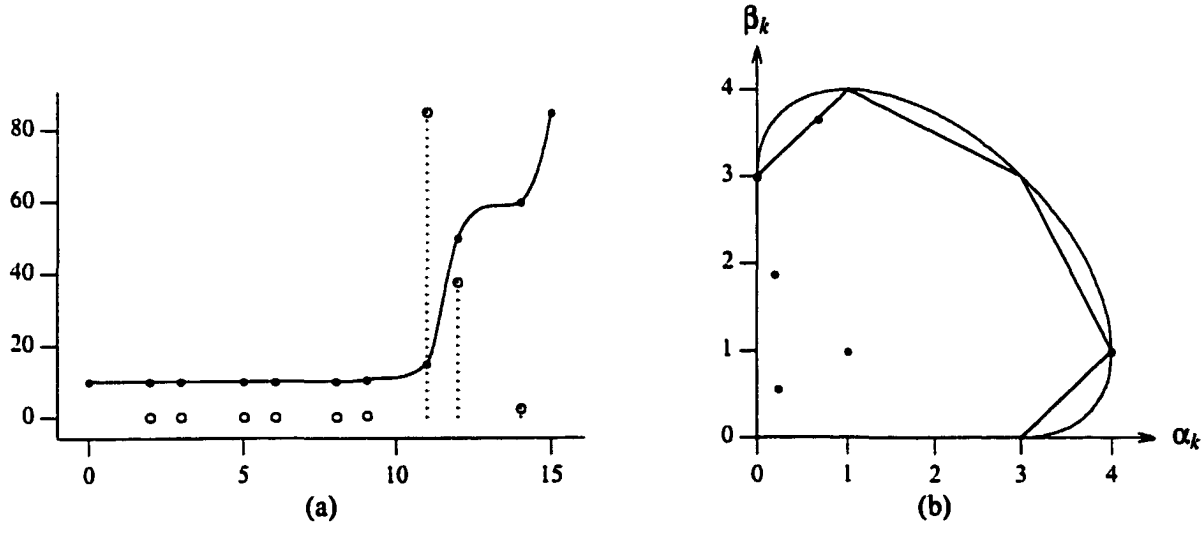


Figure 1.18: SDDE_LP method: (a) interpolating spline; (b) $\{\alpha, \beta\}$ points

Figures 1.17 to 1.19 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

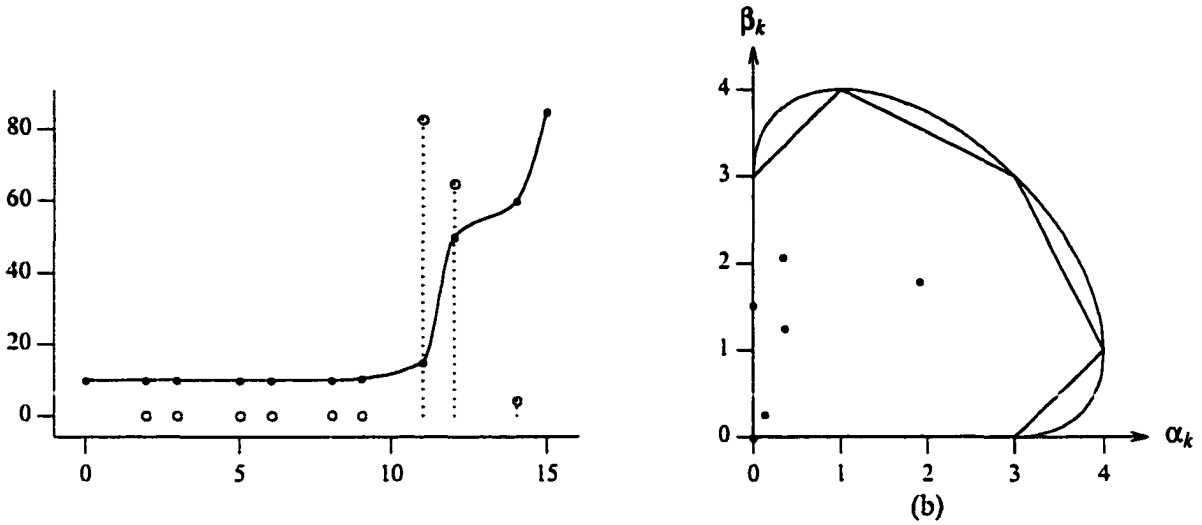


Figure 1.19: Fritsch-Butland (FB): (a) interpolating spline; (b) $\{\alpha, \beta\}$ points

be explained by the $\{\alpha, \beta\}$ pairs that are spread more widely across the M region. Table 1.4 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.

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

Table 1.4: Energy measures for Akima data interpolants.

1.8 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

1.8.1 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.

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 intervals. Details are provided in the supplied MATLAB code in Appendix B.

Figs. 1.20-1.22 show the free-end, Fritsch-Butland (FB), and SDDE methods applied to data having three local extrema. 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. This accounts for the overshoot before the first extrema. Similarly, the extrema on the right side of the figure demonstrates the smoother SDDE_LP fit. Fig. 1.23 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 the whole monotonicity region.

1.8.2 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 [23, 22, 18, 19]. The algorithms are based on [45] and iteratively compute a set of first derivatives that simultaneously satisfy the monotonicity and convexity constraints to produce

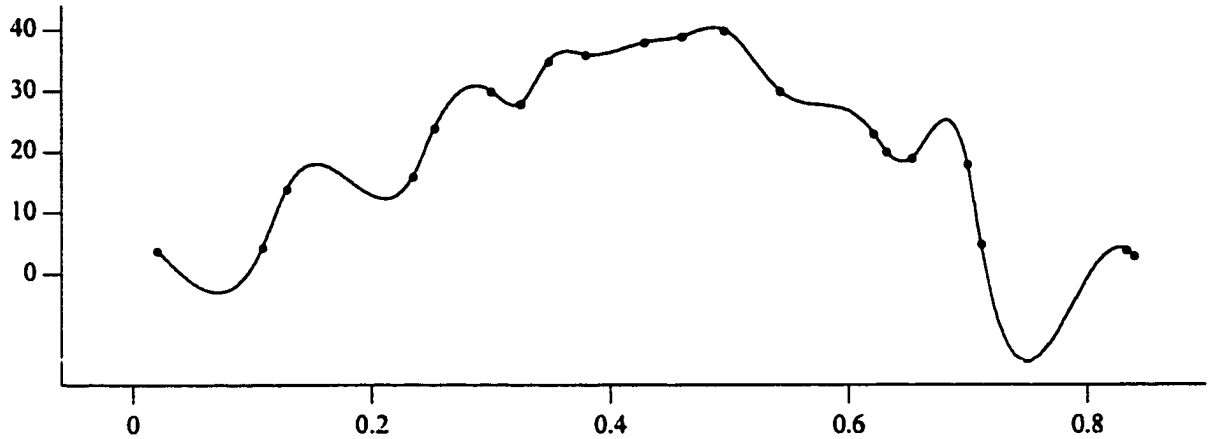


Figure 1.20: Data fitting with natural spline (free end condition).

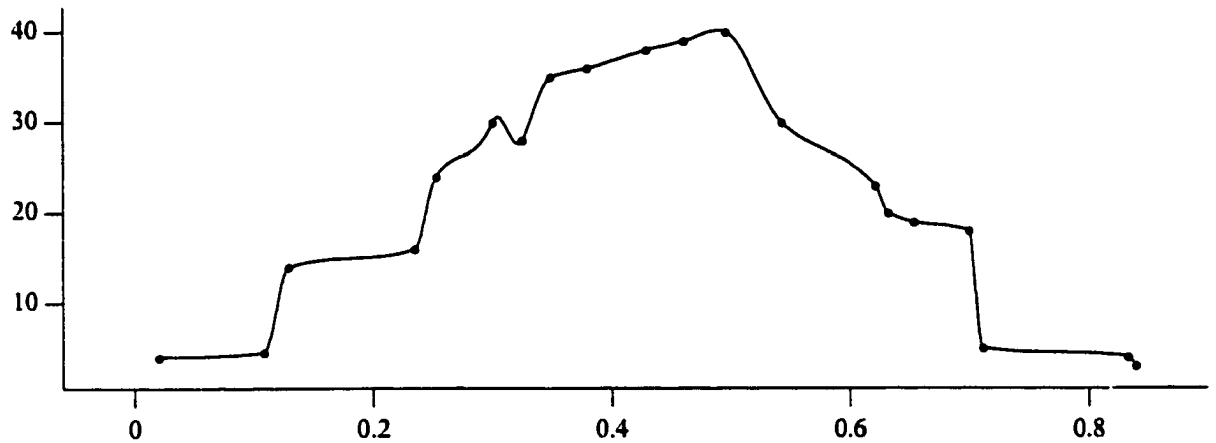


Figure 1.21: Data fitting with Fritsch-Butland (FB) algorithm.

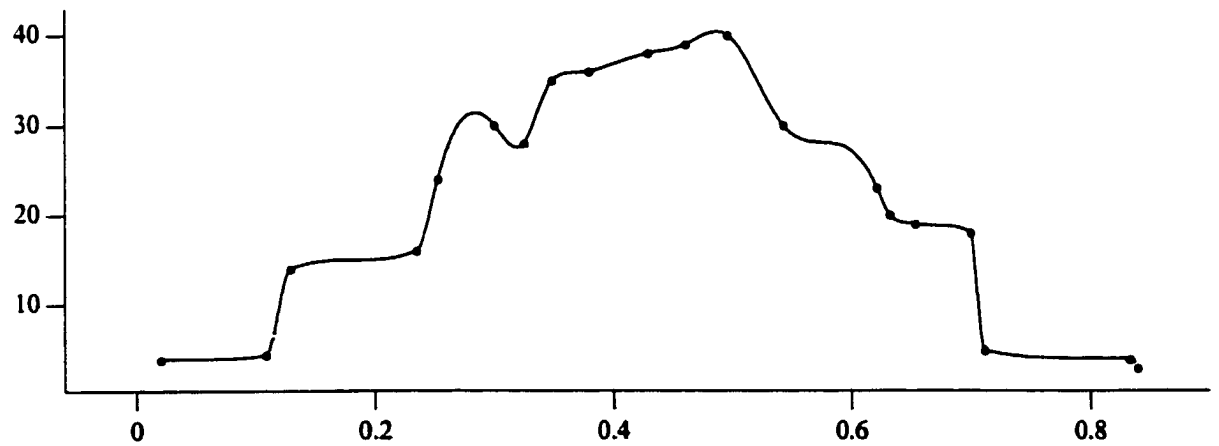


Figure 1.22: Data fitting with SDDE_LP method.

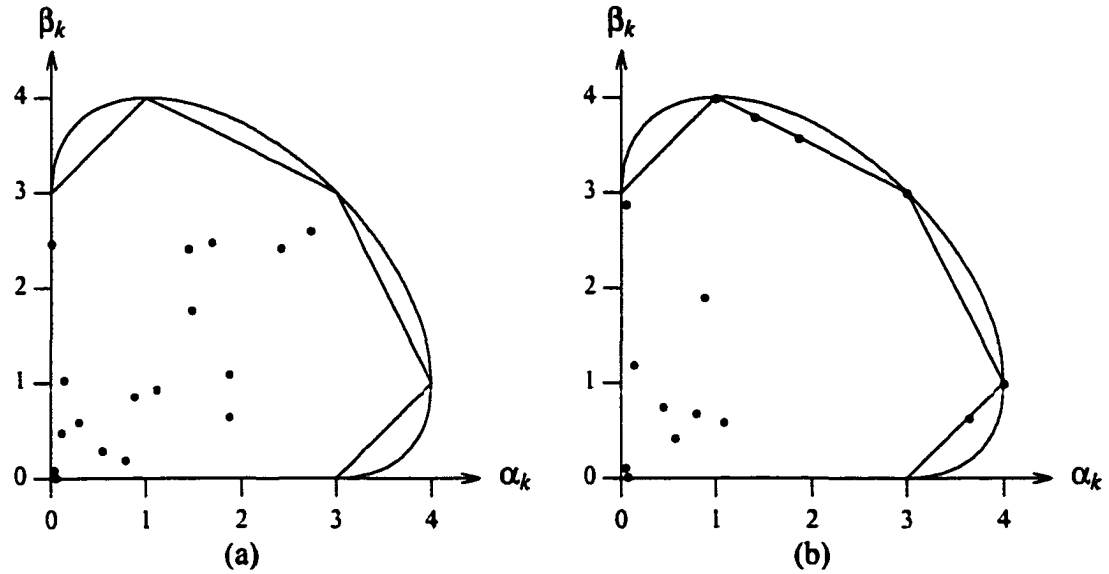


Figure 1.23: $\{\alpha, \beta\}$ points for (a) FB; (b) SDDE_LP.

a C^1 spline. We seek to introduce shape preserving constraints in our optimization framework to produce a C^2 solution, if it exists.

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

$$f'(x) \geq 0 \qquad x \in [x_k, x_{k+1}] \tag{1.54}$$

and

$$f''(x) \geq 0 \qquad x \in [x_k, x_{k+1}] \tag{1.55}$$

The increasing condition of Eq. (1.54) is a monotonicity requirement discussed in Section 1.4.

Since the second derivative of $f(x)$ is linear with 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 $f_k''(x_k) \geq 0$ and $f_k''(x_{k+1}) \geq 0$. This yields the following convexity constraints:

$$2y_k' + y_{k+1}' \leq 3m_k \tag{1.56a}$$

$$-y_k' - 2y_{k+1}' \leq -3m_k \tag{1.56b}$$

Fig. 1.24 shows the convexity region C embedded in monotonicity region M. The intersection

of both regions yields the shape preserving constraints:

$$2y'_k + y'_{k+1} \leq 3m_k \tag{1.57a}$$

$$-y'_k - 2y'_{k+1} \leq -3m_k \tag{1.57b}$$

$$y'_k \geq 0 \tag{1.57c}$$

$$y'_{k+1} \geq 0 \tag{1.57d}$$

These shape preserving constraints can be used in place of the monotonicity constraints (Eq. (1.30) or Eq. (1.31)) for minimizing E_L, $\overline{E}_L$, E_D, and $\overline{E}_D$.

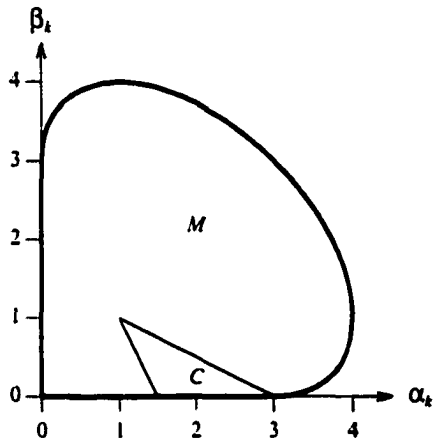


Figure 1.24: Convexity region C is embedded in monotonicity region M.

1.8.3 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 $\tau \leq \frac{\Delta x_k}{3}$. It has been shown that this knot insertion process will guarantee the existence of a monotone C^2 cubic spline interpolant [71].

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 $P = D \bigcup K$. 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, $\overline{E}_L$, E_D, and $\overline{E}_D$. 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 $\{Y_m\}$ and $\{Y'_m\}$ to be unknown. The original data values $\{y_k\}$ are 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. (1.32))
3.	Monotonicity constraints	Eq. (1.30) or Eq. (1.31)

Note that m_k in Eq. (1.30) and Eq. (1.31) 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 [71]. This time we are guaranteed to have a feasible C^2 solution.

1.8.4 Smoothing

In case the data values $\{y_k\}$ are 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 for non-monotone data.

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:

$$E_S = \sum_{k=0}^{n-1} (y_k - f_k)^2 \tag{1.58}$$

The following are the constraints for the optimization problem:

1.	2nd derivative continuity	$f''(x_k^+) = f''(x_k^-)$ (Eq. (1.32))
2.	Monotonicity constraints	Eq. (1.30) or Eq. (1.31)

Since we are performing approximation rather than interpolation, we use f_k rather than y_k in Eq. (1.32). Also, note that m_k in Eq. (1.30) and Eq. (1.31) refers to $(f_{k+1} - f_k)/\Delta x_k$ in this case. Finally, the E_S energy measure can be minimized by using quadratic programming subject to the constraints given above.

1.9 COMPARISON

In this section, we compare the various techniques. We consider the CSE, SDDE_QP, SDDE_LP, LE_QP, LE_LP, and FB methods. Note that this order corresponds to the quality of the splines produced, beginning with the CSE's optimal C^2 spline. These methods are outlined in Table 1.5, where LP and QP are used to refer to linear and quadratic programming, respectively.

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	$\overline{E}_D$	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	$\overline{E}_L$	LP	suboptimal C^2	poor C^1
FB	None	None	fast	C^1, even if C^2 exists

Table 1.5: Comparison of proposed methods.

- The CSE 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 $\overline{E}_D$. 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 $\overline{E}_L$, 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.

1.10 DISCUSSION

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

1. This thesis presents several key enhancements beyond the work described in [87]. 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. 1.11, higher E for the free-end (FE) curve does not translate to higher E_L (see Table 1.2).

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 thesis 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 98.79% 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 1.8 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 [72]. 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 [88] proposed a method for converting a quadratic programming problem into a simpler 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 1.9 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.

1.11 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 thesis, 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 thesis. Modifications were introduced to simplify the problem and facilitate the use of linear programming. The interpolation methods considered in this thesis 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 [44]. 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 in Appendix B to demonstrate the monotonic cubic spline interpolation algorithm.

Chapter 2

MONOTONE SURFACE INTERPOLATION

2.1 INTRODUCTION

In the previous chapter, we developed a univariate piecewise cubic interpolation algorithm which produces a smooth monotone interpolant to monotone data. We now extend these results to produce smooth monotone bivariate interpolants based on tensor product surfaces. A bicubic interpolant is determined by the first partial derivatives and first mixed partials at the mesh nodes. We first derive a set of constraints on these derivatives such that the resulting bicubic polynomial is monotone on a single rectangular element. These constraints are collected into a system of linear inequalities which can be solved using nonlinear programming. We finally extend the approach to handle scattered data. The algorithm is based on fitting the domain with a rectangular grid and searching for the coefficients of the monotone interpolating bicubic polynomials. We also present an algorithm for producing a monotone C^2 interpolant for scattered data that is based on an iterative refine-interpolate-adjust process.

All existing approaches for solving the monotone surface interpolation problem fit piecewise polynomials over triangular or rectangular grids [8, 13, 46, 14, 16, 21, 52, 5, 36, 37]. Carlson and Fritsch [13, 46, 14] developed sufficient conditions on the Hermite derivatives for monotone bicubic functions. Their work, limited to gridded data, is used to determine the coefficients of the bicubic functions forming a monotone C^1 interpolant. We show how to extend their results for scattered data by constructing a system of linear constraints to produce the desired

coefficients.

Han and Schumaker recently presented a method for solving the monotone C^1 interpolation problem for scattered data, as well as gridded data [52]. They derived sufficient conditions for Bernstein-Bezier polynomials defined on a triangle to insure that the corresponding surface is monotone. Their method is based on the simplifying assumption that the first derivatives along the edges are linear. This produces undesirable artifacts when the data distribution is sparse and clustered.

In this chapter, we introduce the Mono-SDI algorithm for producing a monotone C^2 surface. It makes use of an iterative grid refinement process that may be coupled to any scattered data interpolation procedure. Thin plate splines [29] and multilevel B-spline interpolation [60] are considered for the underlying interpolation methods.

2.2 MONOTONE BICUBIC FUNCTIONS

Consider a bivariate function $f(x, y)$ with domain $\Omega = [a, b] \times [c, d]$, as shown in Fig. 2.1. We define $f(x, y)$ to be monotone if it is monotone in both x and y [13]. Therefore, the following two conditions must hold:

- For each $x, y_1 \leq y_2$ and $s \cdot f(x, y_1) \leq s \cdot f(x, y_2)$, where $s = +1$ for monotone increasing functions and $s = -1$ for monotone decreasing functions along y.
- For each $y, x_1 \leq x_2$ and $s \cdot f(x_1, y) \leq s \cdot f(x_2, y)$, where $s = +1$ for monotone increasing functions and $s = -1$ for monotone decreasing functions along x.

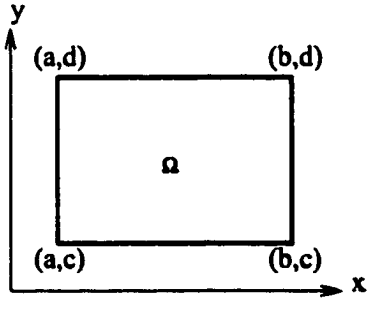


Figure 2.1: Bicubic function domain Ω.

A bicubic polynomial $f(x,y)$, in $\Omega = [a,b] \times [c,d]$ is determined by the first partial derivatives and first mixed partials at the corners:

$$\begin{aligned} f(x,y) = {} & f(a,c)H_1(x)H_1(y) + f(b,c)H_2(x)H_1(y) + \\ & f(a,d)H_1(x)H_2(y) + f(b,d)H_2(x)H_2(y) + \\ & f_x(a,c)H_3(x)H_1(y) + f_x(b,c)H_4(x)H_1(y) + \\ & f_x(a,d)H_3(x)H_2(y) + f_x(b,d)H_4(x)H_2(y) + \\ & f_y(a,c)H_1(x)H_3(y) + f_y(b,c)H_2(x)H_3(y) + \\ & f_y(a,d)H_1(x)H_4(y) + f_y(b,d)H_2(x)H_4(y) + \\ & f_{xy}(a,c)H_3(x)H_3(y) + f_{xy}(b,c)H_4(x)H_3(y) + \\ & f_{xy}(a,d)H_3(x)H_4(y) + f_{xy}(b,d)H_4(x)H_4(y) \end{aligned} \tag{2.1}$$

where $H_1(t), H_2(t), H_3(t), H_4(t)$ are the cubic Hermite basis functions defined on the interval $[0,h]$. Therefore, the x and y indices to H must be normalized to lie in the $[0, b-a]$ and $[0, d-c]$ ranges, respectively. The basis functions, defined below, are depicted in Fig. 2.2.

$$\begin{aligned} H_1(t) &= \frac{1}{h^3}(2t^3 - 3ht^2 + h^3) \\ H_2(t) &= \frac{1}{h^3}(-2t^3 + 3ht^2) \\ H_3(t) &= \frac{1}{h^2}(t^3 - 2ht^2 + h^2t) \\ H_4(t) &= \frac{1}{h^2}(t^3 - ht^2) \end{aligned}$$

The conditions for bicubic function $f(x,y)$ to be monotone over Ω in both x and y are analogous to the univariate case:

$$(\alpha(x), \beta(x)) \in M \quad \text{for all } x \text{ in } [a,b] \tag{2.2a}$$

and $f(x,y)$ is monotonically increasing along x in Ω if

$$(\alpha(y), \beta(y)) \in M \quad \text{for all } y \text{ in } [c,d] \tag{2.3a}$$

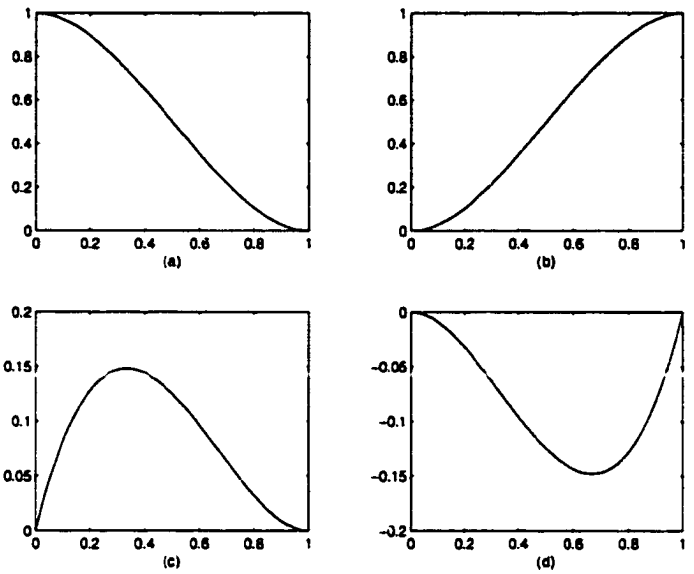


Figure 2.2: Hermite basis functions. (a) H_1; (b) H_2; (c) H_3; (d) H_4.

where M is the monotone region given in Fig. 1.7 and

$$\begin{aligned}
\alpha(x) &= \tfrac{f_y(x,c)}{m_y(x)}, & \alpha(y) &= \tfrac{f_x(a,y)}{m_x(y)} \\
\beta(x) &= \tfrac{f_y(x,d)}{m_y(x)}, & \beta(y) &= \tfrac{f_x(b,y)}{m_x(y)} \\
m_y(x) &= \tfrac{f(x,d)-f(x,c)}{d-c}, & m_x(y) &= \tfrac{f(b,y)-f(a,y)}{b-a}
\end{aligned}$$

and

$$\begin{aligned}
f(a,y) &= f(a,c)H_1(x) + f(a,d)H_2(x) + f_y(a,c)H_3(x) + f_x(a,d)H_4(x) \\
f(b,y) &= f(b,c)H_1(x) + f(b,d)H_2(x) + f_y(b,c)H_3(x) + f_x(b,d)H_4(x) \\
f(x,c) &= f(a,c)H_1(x) + f(b,c)H_2(x) + f_x(a,c)H_3(x) + f_x(b,c)H_4(x) \\
f(x,d) &= f(a,d)H_1(x) + f(b,d)H_2(x) + f_x(a,d)H_3(x) + f_x(b,d)H_4(x) \\
f_x(a,y) &= f_x(a,c)H_1(x) + f_x(a,d)H_2(x) + f_{xy}(a,c)H_3(x) + f_{xy}(a,d)H_4(x) \\
f_x(b,y) &= f_x(b,c)H_1(x) + f_x(b,d)H_2(x) + f_{xy}(b,c)H_3(x) + f_{xy}(b,d)H_4(x) \\
f_y(x,c) &= f_y(a,c)H_1(x) + f_y(b,c)H_2(x) + f_{xy}(a,c)H_3(x) + f_{xy}(b,c)H_4(x) \\
f_y(x,d) &= f_y(a,d)H_1(x) + f_y(b,d)H_2(x) + f_{xy}(a,d)H_3(x) + f_{xy}(b,d)H_4(x)
\end{aligned}$$

The expressions for α and β given in Eq. (2.3) represent the monotonicity conditions for a single rectangular element. We seek to find a C^2 solution over a grid of data. This is a global problem that requires optimization to determine the unknown derivative values at the data points to yield the smoothest C^2 monotone bicubic surface. An energy measure must be defined in order to quantify smoothness. In the next section, we introduce an energy measure based on the physical characteristics of a thin plate under small deflections.

2.3 SURFACE ENERGY

The energy of a thin plate of arbitrary shape is given by

$$\tilde{E} = \int [(f_{xx} + f_{yy})^2 - 2(1 - v)(f_{xx}f_{yy} - f_{xy}^2)]dxdy$$

where the parameter v is a constant depending on the material at hand [33, 83]. If we let $v = 0$, then a simpler energy measure can be used [33, 29]:

$$E = \int [f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2]dxdy$$

Although $v = 0$ furnishes a simple expression, $v = 0.3$ has been used to correspond to aluminium or steel [33]. It has been shown that the thin plate spline (TPS) minimizing E has the form [29]

$$f(x, y) = \sum_k \alpha_k r_k^2 \log r_k + ax + by + c$$

Note that $f(x, y)$, represented as the sum of radial basis functions, is not guaranteed to preserve monotonicity.

Energy measure E is applicable only for surfaces with piecewise continuous second derivatives f_{xx}, f_{yy}, f_{xy}. When no C^2 monotone surface exists, the second derivative discontinuity may be visually prominent at the joints of the rectangular elements. Therefore, we solve for the closest C^2 surface in an attempt to produce a more natural looking surface. This process requires us to introduce an energy measure based on the second derivative discontinuity, E_D.

Consider a rectangular mesh consisting of data values at the nodes. Let B be the set of all

shared edges (borders) b in the mesh. We may define energy measure E_D as the sum of the discontinuity energy measured along each of the shared edges:

$$E_D = \sum_{b \in B} \int_b \left\{ \left[f_{xx}(b^+) - f_{xx}(b^-) \right]^2 + \left[f_{yy}(b^+) - f_{yy}(b^-) \right]^2 + \left[f_{xy}(b^+) - f_{xy}(b^-) \right]^2 \right\} db \tag{2.4}$$

Note that E_D requires us to compute second derivatives along both sides of a shared edge. The second derivative of a bicubic function along an edge is a cubic polynomial. This is demonstrated below for $f_{yy}(x, c)$.

$$\begin{aligned} f_{yy}(x,c) \;=\; & H_1(x)[f(a,c)H_1''(0) + f(a,d)H_2''(0) + f_y(a,c)H_3''(0) + f_y(a,d)H_4''(0)] + \\ & H_2(x)[f(b,c)H_1''(0) + f(b,d)H_2''(0) + f_y(b,c)H_3''(0) + f_y(b,d)H_4''(0)] + \\ & H_3(x)[f_x(a,c)H_1''(0) + f_x(a,d)H_2''(0) + f_{xy}(a,c)H_3''(0) + f_{xy}(a,d)H_4''(0)] + \\ & H_4(x)[f_x(b,c)H_1''(0) + f_x(b,d)H_2''(0) + f_{xy}(b,c)H_3''(0) + f_{xy}(b,d)H_4''(0)] \end{aligned}$$

The three terms within the integral in Eq. (2.4) are each of degree six. This make the minimization of E_D computationally unfeasible.

2.4 MONOTONICITY CONSTRAINTS

Monotonicity constraints for univariate cubic spline were presented in previous papers [87, 45]. If the monotonicity region is approximated using a $[0,3] \times [0,3]$ square, then the monotonicity constraints for function $g(t)$ are $0 \leq \frac{g'_k}{m_k} \leq 3$ and $0 \leq \frac{g'_{k+1}}{m_k} \leq 3$ where g'_k and g'_{k+1} are the first derivatives at the right and left side of the interval, respectively, and $m_k = \frac{\Delta g}{\Delta t}$. Writing these conditions in matrix form yields:

$$sign(m_k) \cdot A \cdot \begin{bmatrix} g'_k \\ g'_{k+1} \\ m_k \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.5}$$

where

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 0 & -3 \\ 0 & 1 & -3 \end{bmatrix}$$

If we consider the case of increasing monotonicity in y, we use the following expressions:

$$\begin{aligned} sign(m_k) &= 1 \\ g'_k &= f_y(x,c) \\ g'_{k+1} &= f_y(x,d) \\ m_k &= m_y(x) \end{aligned}$$

By direct substitution into the four constraints in Eq. (2.5), we have

$$\begin{aligned} 0 \geq & \left[A_{i,1} f_y(a,c) + A_{i,2} f_y(a,d) + A_{i,3} \frac{f(a,d) - f(a,c)}{\Delta_y}\right] H_1(x) + \\ & \left[A_{i,1} f_y(b,c) + A_{i,2} f_y(b,d) + A_{i,3} \frac{f(b,d) - f(b,c)}{\Delta_y}\right] H_2(x) + \\ & \left[A_{i,1} f_{xy}(a,c) + A_{i,2} f_{xy}(a,d) + A_{i,3} \frac{f_x(a,d) - f_x(a,c)}{\Delta_y}\right] H_3(x) + \\ & \left[A_{i,1} f_{xy}(b,c) + A_{i,2} f_{xy}(b,d) + A_{i,3} \frac{f_x(b,d) - f_x(b,c)}{\Delta_y}\right] H_4(x) \end{aligned} \tag{2.6}$$

where $i = [1,4]$, $\Delta_x = b - a$, and $\Delta_y = d - c$. A similar result is obtained for increasing monotonicity in x, whereby we use the following expressions:

$$\begin{aligned} sign(m_k) &= 1 \\ g'_k &= f_x(a,y) \\ g'_{k+1} &= f_x(b,y) \\ m_k &= m_x(y) \end{aligned}$$

This yields

$$
\begin{aligned}
0 \geq & \left[A_{i,1}f_x(a,c) + A_{i,2}f_x(b,c) + A_{i,3}\frac{f(b,c)-f(a,c)}{\Delta_x}\right]H_1(x) + \\
& \left[A_{i,1}f_x(a,d) + A_{i,2}f_x(b,d) + A_{i,3}\frac{f(b,d)-f(a,d)}{\Delta_x}\right]H_2(x) + \\
& \left[A_{i,1}f_{xy}(a,c) + A_{i,2}f_{xy}(b,c) + A_{i,3}\frac{f_y(b,c)-f_y(a,c)}{\Delta_x}\right]H_3(x) + \\
& \left[A_{i,1}f_{xy}(a,d) + A_{i,2}f_{xy}(b,d) + A_{i,3}\frac{f_y(b,d)-f_y(a,d)}{\Delta_x}\right]H_4(x)
\end{aligned} \tag{2.7}
$$

Note that the righthand side of the inequality term is a cubic polynomial, and the unknowns are the first and mixed derivatives at the corners of Ω. These inequalities must hold for every $x \in [a,b]$ and $y \in [c,d]$. We therefore need to eliminate the dependency of the inequalities with respect to x and y. This is achieved through the following lemma [13].

Lemma 4 *Let $q(t)$ be a cubic polynomial on $[0,h]$*

$$q(t) = q(0)H_1(t) + q(h)H_2(t) + q'(0)H_3(t) + q'(h)H_4(t)$$

where $q(0)$ and $q(h)$ are of the same sign. $q(t)$ will be negative throughout the interval $[0,h]$ if:

- *If $q(0) = q(h) = 0$, then $sign[q'(0)] = -1$ and $sign[q'(h)] = 1$*
- *If $q(0) \leq 0$ and $q(h) \leq 0$, then $3q(0) + hq'(0) \leq 0$ and $3q(h) - hq'(h) \leq 0$*

Proof. If $q(0) = q(h) = 0$, then $q(t) = q'(0)H_3(t) + q'(h)H_4(t)$. We know from Fig. 2.2 that $H_3(t) > 0$ and $H_4 < 0$ for $t \in [0,h]$. Therefore, if $q(0) = q(h) = 0$, then $q(t)$ is always negative in $[0,h]$. Otherwise, if we let $q(0) \leq 0$ and $q(h) \leq 0$, then we construct two cubic polynomials as follows

$$
\begin{aligned}
r(t) &= q(0)H_1(t) + \frac{-3q(0)}{h}H_3(t) \\
s(t) &= q(h)H_2(t) + \frac{3q(h)}{h}H_4(t)
\end{aligned}
$$

$r(t)$ and $s(t)$ are monotone because their (α, β) lie in the monotone region:

$$\begin{aligned} \alpha_r &= \frac{r'(0)}{m_r} = 3, & \alpha_s &= \frac{s'(0)}{m_s} = 0 \\ \beta_r &= \frac{r'(h)}{m_r} = 0, & \beta_s &= \frac{s'(h)}{m_s} = 3 \\ m_r &= \frac{r(h)-r(0)}{h} = -\frac{q(0)}{h}, & m_s &= \frac{s(h)-v(0)}{h} = \frac{q(h)}{h} \end{aligned}$$

Note that $r(t)$ is monotone increasing because $\frac{-3q(0)}{h} > 0$. Also, note that $r(0) \leq 0$ and $r(h) = 0$. This implies that $r(t)$ is always nonpositive because $r(t)$ is monotone increasing between a negative r(0) and $r(h) = 0$. A similar argument holds for nonpositive $s(t)$. In this case, $s(t)$ is monotone decreasing because $\frac{3q(h)}{h} > 0$. Also, $s(0) = 0$ and $s(h) \leq 0$. This implies that $s(t)$ is always nonpositive. Therefore, $r(t)$, $s(t)$ and $[r(t) + s(t)]$ are nonpositive on $[0, h]$. Given these results, we now show that $q(t) < 0$:

$$q(t) - [r(t) + v(t)] = \left[q'(0) + \frac{3q(0)}{h}\right] H_3(t) + \left[q'(h) - \frac{3q(h)}{h}\right] H_4(t) \tag{2.8}$$

The bracketed terms in Eq. (2.8) refer to the lemma conditions: $q'(0) + \frac{3q(0)}{h} \leq 0$ and $q'(h) - \frac{3q(h)}{h} \geq 0$. From Fig. 2.2 we know that H_3 is nonpositive and H_4 is nonnegative. Therefore, the expression in Eq. (2.8) is nonpositive. Using the fact that $[r(t) + s(t)] < 0$, we get $q(t) \leq r(t) + s(t) < 0$ for $t \in [0, h]$. ■

We use the lemma to translate each monotonicity constraint in Eq. (2.6) into four inequalities that are independent of x:

$$0 \geq A_{i,1} f_y(a,c) + A_{i,2} f_y(a,d) + A_{i,3} \frac{f(a,d) - f(a,c)}{\Delta_y} \tag{2.9a}$$

$$0 \geq A_{i,1} f_y(b,c) + A_{i,2} f_y(b,d) + A_{i,3} \frac{f(b,d) - f(b,c)}{\Delta_y} \tag{2.9b}$$

$$\begin{aligned} 0 \geq\ & 3\left[A_{i,1} f_y(a,c) + A_{i,2} f_y(a,d) + A_{i,3} \frac{f(a,d) - f(a,c)}{\Delta_y}\right] + \\ & \Delta_x \left[A_{i,1} f_{xy}(a,c) + A_{i,2} f_{xy}(a,d) + A_{i,3} \frac{f_x(a,d) - f_x(a,c)}{\Delta_y}\right] \end{aligned} \tag{2.9c}$$

$$\begin{aligned} 0 \geq\ & 3\left[A_{i,1} f_y(b,c) + A_{i,2} f_y(b,d) + A_{i,3} \frac{f(b,d) - f(b,c)}{\Delta_y}\right] - \\ & \Delta_x \left[A_{i,1} f_{xy}(b,c) + A_{i,2} f_{xy}(b,d) + A_{i,3} \frac{f_x(b,d) - f_x(b,c)}{\Delta_y}\right] \end{aligned} \tag{2.9d}$$

Similarly, we use the lemma to translate each monotonicity constraint in Eq. (2.7) into four inequalities that are independent of y:

$$0 \geq A_{i,1} f_x(a,c) + A_{i,2} f_x(b,c) + A_{i,3} \frac{f(b,c) - f(a,c)}{\Delta_x} \tag{2.10a}$$

$$0 \geq A_{i,1} f_x(a,d) + A_{i,2} f_x(b,d) + A_{i,3} \frac{f(b,d) - f(a,d)}{\Delta_x} \tag{2.10b}$$

$$0 \geq 3\left[A_{i,1} f_x(a,c) + A_{i,2} f_x(b,c) + A_{i,3} \frac{f(b,c) - f(a,c)}{\Delta_x}\right] + \Delta_y \left[A_{i,1} f_{xy}(a,c) + A_{i,2} f_{xy}(b,c) + A_{i,3} \frac{f_y(b,c) - f_y(a,c)}{\Delta_x}\right] \tag{2.10c}$$

$$0 \geq 3\left[A_{i,1} f_x(a,d) + A_{i,2} f_x(b,d) + A_{i,3} \frac{f(b,d) - f(a,d)}{\Delta_x}\right] - \Delta_y \left[A_{i,1} f_{xy}(a,d) + A_{i,2} f_{xy}(b,d) + A_{i,3} \frac{f_y(b,d) - f_y(a,d)}{\Delta_x}\right] \tag{2.10d}$$

An optimal solution can be derived only at great computational expense, as discussed in Section 2.3. Instead, we use linear programming to find any feasible solution. We therefore express the inequality constraints in matrix form so that they are suitable for linear programming. Each one of the four monotonicity constraints in Eq. (2.5) expands to four inequalities using Eq. (2.9) and Eq. (2.10), for a total of 32 inequalities, i.e., 16 for monotonicity along each axis:

$$B_x \bar{u} \leq \bar{b}_x \tag{2.11}$$

$$B_y \bar{u} \leq \bar{b}_y \tag{2.12}$$

where

$$B_y = \begin{bmatrix}
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & -\Delta_x & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & \Delta_x & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & -\Delta_x & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & \Delta_x \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
3\frac{\Delta_x}{\Delta_y} & 0 & -3\frac{\Delta_x}{\Delta_y} & 0 & 3 & 0 & 0 & 0 & \Delta_x & 0 & 0 & 0 \\
0 & -3\frac{\Delta_x}{\Delta_y} & 0 & 3\frac{\Delta_x}{\Delta_y} & 0 & 3 & 0 & 0 & 0 & -\Delta_x & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
3\frac{\Delta_x}{\Delta_y} & 0 & -3\frac{\Delta_x}{\Delta_y} & 0 & 0 & 0 & 3 & 0 & 0 & 0 & \Delta_x & 0 \\
0 & -3\frac{\Delta_x}{\Delta_y} & 0 & 3\frac{\Delta_x}{\Delta_y} & 0 & 0 & 0 & 3 & 0 & 0 & 0 & -\Delta_x
\end{bmatrix}$$

$$B_x = \begin{bmatrix}
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Delta_y & 0 & 0 & 0 \\
0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta_y & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Delta_y & 0 & 0 \\
0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta_y \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 3\frac{\Delta_x}{\Delta_x} & -3\frac{\Delta_y}{\Delta_x} & 0 & 0 & \Delta_y & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & 0 & -3\frac{\Delta_y}{\Delta_x} & 3\frac{\Delta_y}{\Delta_x} & 0 & 0 & -\Delta_y & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 3\frac{\Delta_y}{\Delta_x} & -3\frac{\Delta_y}{\Delta_x} & 0 & 0 & 0 & \Delta_y & 0 & 0 \\
0 & 0 & 0 & 3 & 0 & 0 & -3\frac{\Delta_y}{\Delta_x} & 3\frac{\Delta_y}{\Delta_x} & 0 & 0 & 0 & -\Delta_y
\end{bmatrix}$$

$$\begin{aligned}
\bar{u} = [\ & f_x(a,c) \quad f_x(b,c) \quad f_x(a,d) \quad f_x(b,d) \\
& f_y(a,c) \quad f_y(b,c) \quad f_y(a,d) \quad f_y(b,d) \\
& f_{xy}(a,c) \quad f_{xy}(b,c) \quad f_{xy}(a,d) \quad f_{xy}(b,d)\ ]^T
\end{aligned}$$

$$\bar{b}_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_1 & k_1 & 3k_1 & 3k_1 & k_2 & k_2 & 3k_2 & 3k_2 \end{bmatrix}^T$$

$$\bar{b}_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_3 & k_3 & 3k_3 & 3k_3 & k_4 & k_4 & 3k_4 & 3k_4 \end{bmatrix}^T$$

$$k_1 = 3\frac{f(a,d)-f(a,c)}{\Delta_y} \quad k_2 = 3\frac{f(b,d)-f(b,c)}{\Delta_y}$$
$$k_3 = 3\frac{f(b,c)-f(a,c)}{\Delta_x} \quad k_4 = 3\frac{f(b,d)-f(a,d)}{\Delta_x}$$

2.4.1 Discussion

Eq. (2.11) and Eq. (2.12) define the monotonicity constraints for a single rectangular element in the x and y directions, respectively. For gridded data, we must concatenate the constraints of each individual rectangular element into one large system of equations. We solve for the smoothest monotone bicubic surface by determining the unknown derivatives at the grid nodes using the linear programming method. We can achieve performance gains by exploiting redundancy in Eq. (2.11) and Eq. (2.12), and thereby reducing the large system of equations.

For convenience, we shall confine our discussion to monotonicity in the x direction. Eq. (2.11) makes use of matrix B_x to establish 16 constraints for monotonicity in the x direction. B_x contains eight rows consisting of a single non-zero element. They represent upper and lower bound constraints on the first derivatives at each of the four corners of the rectangular element. Since we are assuming that the (α, β) pair along the mesh lines lie in the $[0, 3]$ square, the lower bounds are zero and the upper bounds are $3m$, where m is the slope of that border segment. A corner point c may be shared among one, two, or four rectangular elements. Each rectangular element r assigns an upper bound ub_r to the first derivative at c. The actual upper bound at c is then taken to be the minimum of the ub_r values. This reduces the number of monotonicity constraints in the x direction for an $m \times n$ mesh from $16(m-1)(n-1)$ to $8(m-1)(n-1)+2mn$. Note that the $8(m-1)(n-1)$ represents the number of constraints that could not be reduced further, and $2mn$ represents the actual lower and upper bounds at each mesh node. A similar argument holds for the reduction of Eq. (2.12).

Reducing the number of constraints influences the convergence rate as well as the optimization method that is best suited: simplex or the primal-dual interior point method [62]. The size of the problem determines which method is more appropriate. Any problem where there are fewer than 2,000 rows and columns (combined) is considered small and is best solved using the simplex method. For moderate problems, where there are fewer than a combined total of 10,000 rows and columns, neither method shows a clear advantage. For larger problems, the primal-dual interior points methods is far more efficient than the simplex method [62].

It is important to note that Carlson and Fritsch used a similar method for evaluating the upper bounds from neighboring rectangular elements [13]. However, they did so for different purposes. They are interested in determining an upper bound for the first derivatives at each mesh node. Their local algorithm uses these bounds to directly solve for the coefficients of the monotone C^1 bicubic interpolant. The method outlined in this section, however, uses this method for determining upper bounds to reduce the total number of constraints we pass to a global optimization procedure. Since the Carlson-Fritsch method does not perform optimization, their method is faster than the optimization-based method outlined here. Our purpose in introducing optimization, however, is to motivate its use for the scattered data problem described in Section 2.5.

2.4.2 Example

The following example demonstrates this approach. Consider a sigmoidal function $G(x, y)$ that is uniformly sampled on a 6×6 grid defined on the unit square [52, 8]:

$$G(x, y) = \sqrt{1 + 2e^{-3(9r-6.7)}}$$

where $r = \sqrt{x^2 + y^2}$. Due to the intensive computational effort that is required to evaluate E_D, the following results are obtained by applying only the first phase of the linear programming procedure. The first phase uses the simplex algorithm to find whether the problem has a feasible solution. If a feasible solution exists, it provides a feasible solution to initiate the second phase which, in turn, uses the simplex algorithm to find a feasible solution that is optimal [72].

Consider the surface shown in Fig. 2.3(a). Its level curves are shown in Fig. 2.3(b). Figs. 2.3(c,e,g) respectively illustrate perspective views of the reconstructed surface produced by the TPS method, Han-Schumaker algorithm, and the feasible solution approach described in this section. Figs. 2.3(d,f,h) respectively illustrate the level curves of those surfaces. It is evident that the TPS surface is not monotone. The Han-Schumaker method, which uses cubic Bernstein-Bezier polynomials, also produces a surface that is not monotone. As we will discuss later, their method is flawed and this example highlights that problem.

We computed the error on a 33×33 uniform grid. The mean error for the thin plate

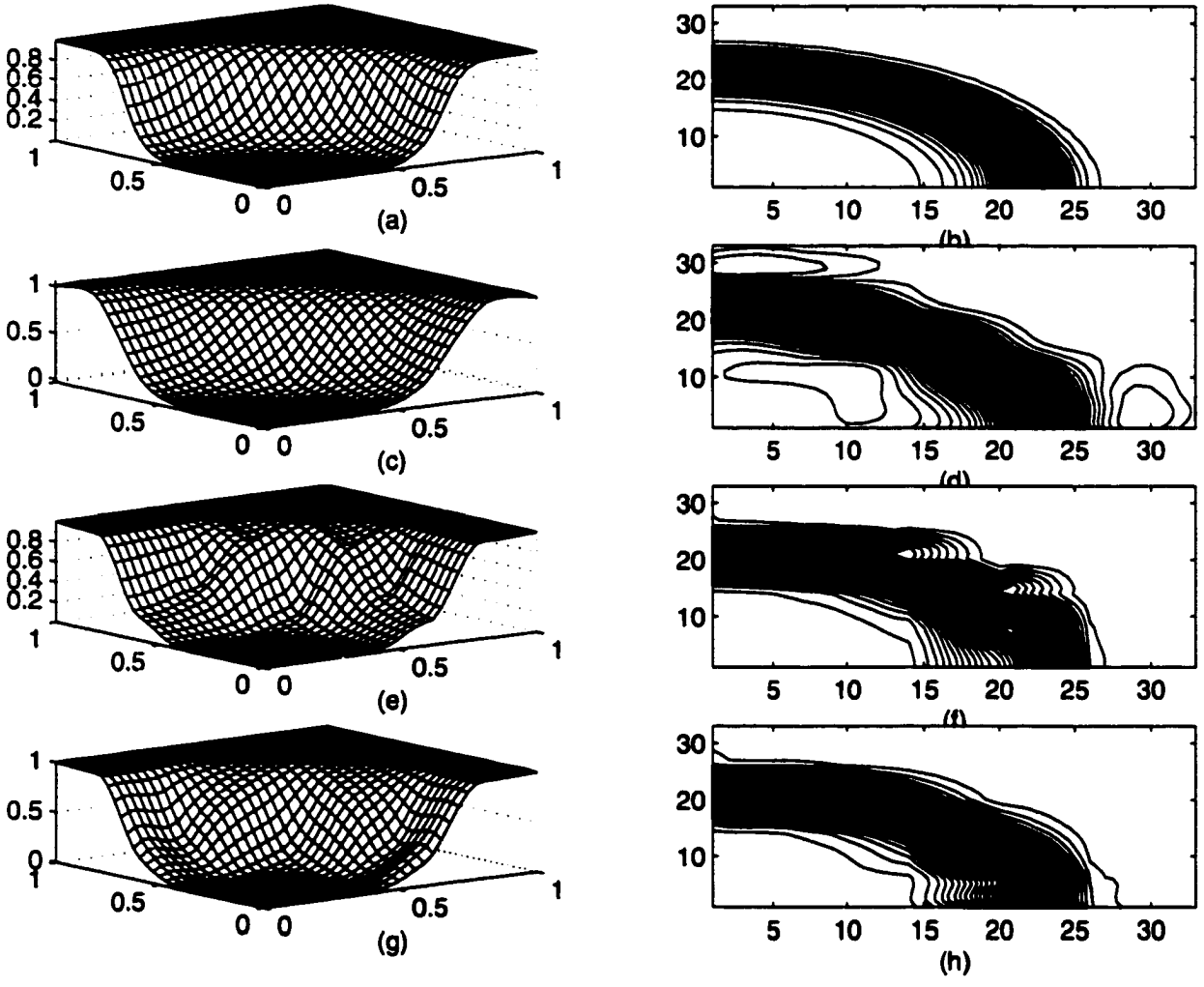


Figure 2.3: Perspective view and level curves for gridded data. (a,b) $G(x, y)$; (c,d) thin plate spline; (e,f) Han-Schumaker; (g,h) monotone bicubic interpolant.

spline, Han-Schumaker, and the bicubic monotone surfaces are 0.0246, 0.0263, and 0.0204, respectively. Thus, we produced a smaller error than the Han-Schumaker method and thin plate splines.

2.5 MONOTONE BICUBIC INTERPOLATION FOR SCATTERED DATA

The method presented thus far has been developed for gridded data sets. In this section we present three alternatives for solving the monotone interpolation problem for scattered data. It is important to note that the purpose of the gridded data presentation is to establish the framework for the scattered data case.

2.5.1 Fitting the Domain with a Grid

Let $\Omega = \{(x,y)|0 \leq x \leq A, 0 \leq y \leq B\}$ be a rectangular domain in the xy-plane. Consider a set of monotone scattered points $P = \{(x_i, y_i, z_i)\}_{i=1}^{N}$ where (x_i, y_i) is a point in Ω. We overlay on the domain Ω a grid with n_x vertical lines and n_y horizontal lines (Fig. 2.4). We denote the corresponding grid points by $G = \{(X_i, Y_j, Z_i)\}_{i=1,j=1}^{n_x,n_y}$ and redefine the unknown vector $\tilde{u}$ to include the function values at the grid points:

$$\tilde{u} = [\ f(a,c)\ \ f(b,c)\ \ f(a,d)\ \ f(b,d)\ \ f_x(a,c)\ \ f_x(b,c)\ \ f_x(a,d)\ \ f_x(b,d)$$
$$f_y(a,c)\ \ f_y(b,c)\ \ f_y(a,d)\ \ f_y(b,d)\ \ f_{xy}(a,c)\ \ f_{xy}(b,c)\ \ f_{xy}(a,d)\ \ f_{xy}(b,d)\ ]^T$$

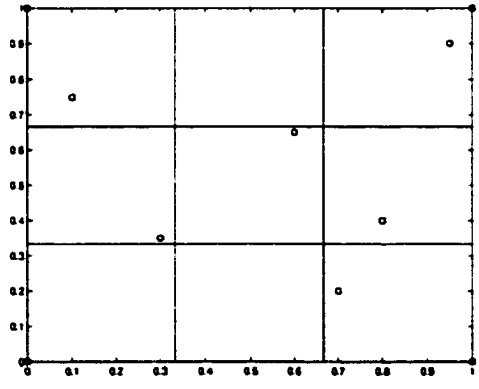


Figure 2.4: Ten scattered data points with a 4 × 4 uniform grid

The scattered data problem is to minimize E_D subject to the following constraints:

1. Interpolation (Eq. (2.1))

2. Monotonicity (Eqs. 2.9 and 2.10)

Fig. 2.5 shows the surface produced by solving the bicubic problem for the data points in Fig. 2.4. The optimization procedure searches only for a feasible solution, not one that necessarily minimizes E_D. For comparison, we computed the thin plate spline (TPS) and the Han-Schumaker (HS) surfaces which are clearly not monotone. We computed the error on a 33 × 33 uniform grid. The mean error for the TPS and HS surfaces are 0.08920 and 0.09157, respectively. The bicubic monotone interpolant produces a surface with a mean error of 0.07109. Thus we not only managed to create a monotone surface from the scattered data but we also produced one with lower error.

The main drawback of this method is that the minimal grid density is unknown. Note that if we force a horizontal and vertical grid line through each point, we are guaranteed to find a solution. Too many lines also guarantees a solution, but is wasteful. Too few lines leads to an overdetermined system. Empirically, we require each data point to be in a separate rectangular patch.

2.5.2 Reducing Scattered Data to Gridded Data

Consider monotone scattered data $P = \{(x_i, y_i, z_i)\}_{i=1}^{N}$, let $D = \{(x_i, y_i)\}_{i=1}^{N}$. An alternate approach for computing a monotone surface from P is the Han-Schumaker method [52]. They map scattered data onto a rectangular grid whereby a horizontal and vertical grid line is made to pass through all points $(x_i, y_i) \in D$, as depicted in Fig. 2.6. Consider n_x vertical lines and n_y horizontal lines. Let corresponding grid points be denoted by $G = \{(X_i, Y_j)\}_{i=1,j=1}^{n_x,n_y}$. They showed how to construct data values Z_{ij}^M for $1 \leq i \leq n_x$ and $1 \leq j \leq n_y$ that are monotone and consistent with the data in P. Consistent data means that those grid points that coincide with points in P must share their same z values.

To construct $\{Z_{ij}^M\}$ we begin with an arbitrary interpolation method for the scattered data P (see the next chapter for a survey of scattered data interpolation methods). This yields $\{Z_{ij}\}$ which is not necessarily monotone. They showed how to adjust Z_{ij} to produce a monotone

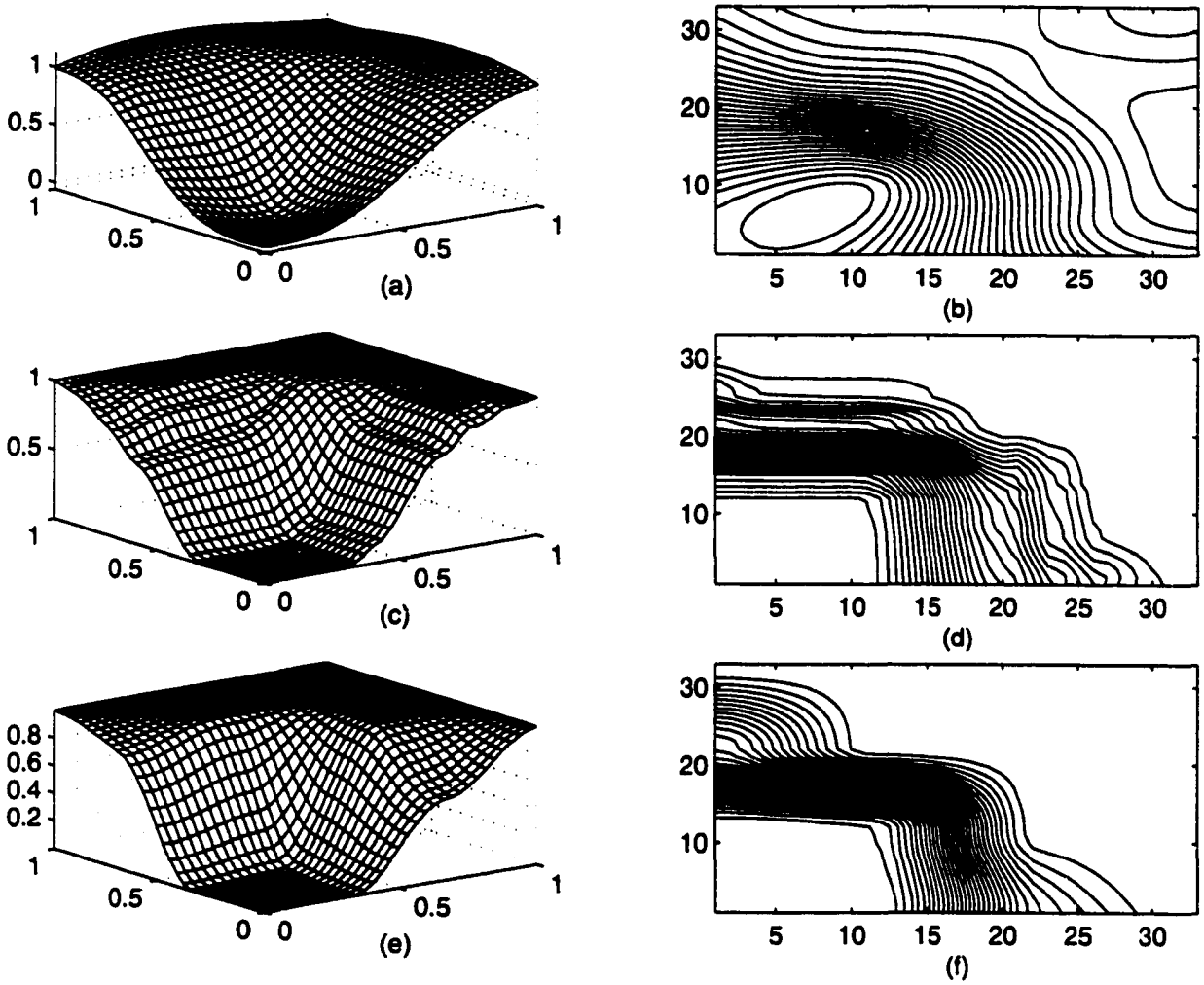


Figure 2.5: Perspective view and level curves for scattered data. (a,b) thin plate spline; (c,d) Han-Schumaker surface; (e,f) feasible solution for monotone bicubic interpolant.

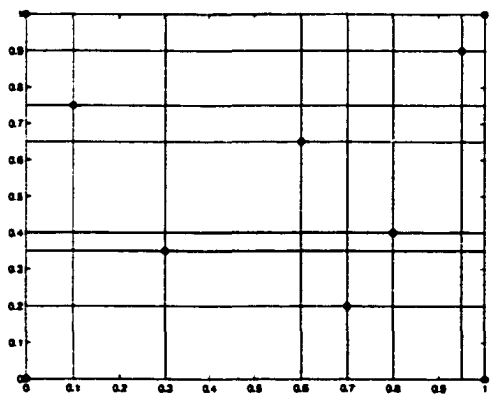


Figure 2.6: Ten scattered data points with an 8 × 8 nonuniform grid

11	7	4	2	1
15	12	8	5	3
18	16	13	9	6
20	19	17	14	10

Figure 2.7: Order of Z_{ij} adjustments.

gridded data set that remains consistent with the original data. The idea is to adjust the Z_{ij} values for those points not in P, starting in the upper-right corner, and scanning in a zigzag direction toward the lower-left corner. The order in which we make the adjustment is indicated by the numbering of the vertices shown in Fig. 2.7.

Suppose that for some (i,j) we want to adjust the value of Z_{ij} corresponding to (x_i, y_j), having done all the previous points. Let

$$\begin{aligned} I_{ij}^{+} &= \{(l,k): && l \geq i, k \geq j\} \\ I_{ij}^{-} &= \{(l,k): && (l,k) \in I : l \leq i, k \leq j\} \\ m_{ij}^{-} &= \max\{Z_{lk}: && (l,k) \in I_{ij}^{-}\} \\ m_{ij}^{+} &= \min\{Z_{lk}: && (l,k) \in I_{ij}^{+}\} \end{aligned}$$

Since the original data is monotone, and the adjustment process maintains monotonicity, at every step of the process, $m_{ij}^{-} \leq m_{ij}^{+}$. Now we define

$$Z_{ij}^{M} = \begin{cases} m_{ij}^{+} & \text{if } Z_{ij} > m_{ij}^{+} \\ m_{ij}^{-} & \text{if } Z_{ij} < m_{ij}^{-} \\ Z_{ij} & otherwise \end{cases}$$

This adjustment process does not change Z_{ij} for points $i,j \in P$, and so the final values are still consistent with the given data. Figs. 2.8(a)-(d) demonstrates this process. Fig. 2.8(a) shows the 8×8 grid G corresponding to ten scattered data points. Fig. 2.8(b) shows the surface produced by interpolating the grid nodes using the TPS method. Fig. 2.8(c) shows the resulting surface produced after adjusting the grid nodes to enforce monotonicity. Finally, Fig. 2.8(d) shows the final surface obtained using the Han-Schumaker algorithm.

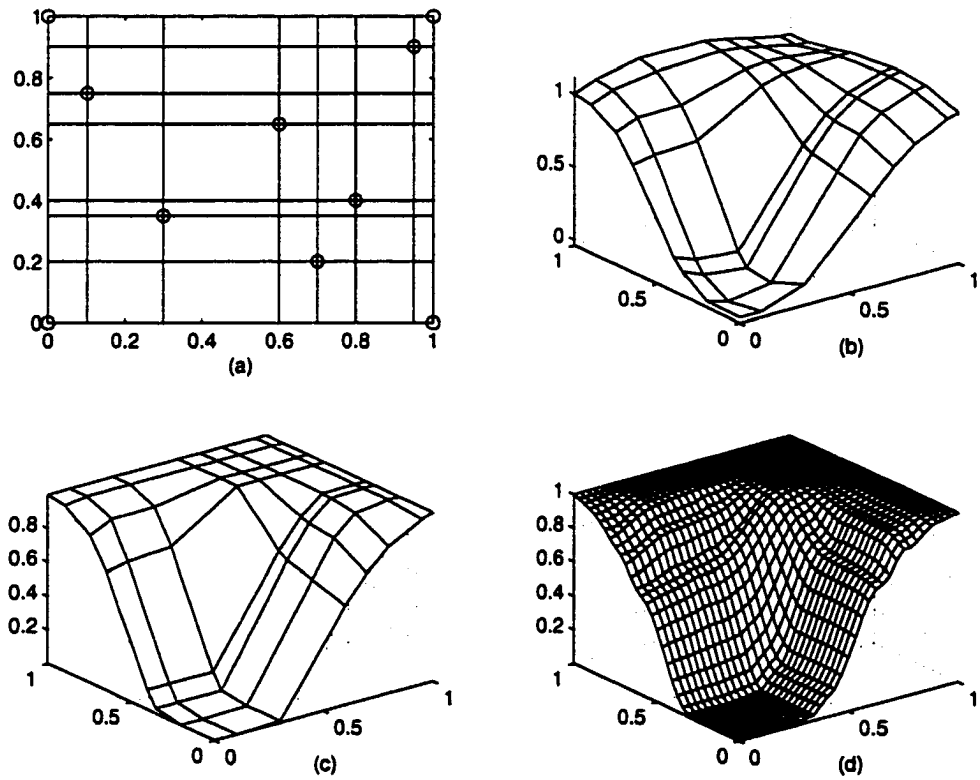


Figure 2.8: (a) Ten scattered data points and grid; (b) TPS interpolation; (c) monotone gridded data (d) final Han-Schumaker surface.

The main drawback of the Han-Schumaker method is that N scattered data points may produce a dense grid having $O(N^2)$ rectangles. Furthermore, some of the rectangles may be very small in one or both directions. We will show how to circumvent this problem using an iterative grid refinement process.

It is important to make several comments about the Han-Schumaker algorithm. The conceptual basis of their work stems from the Carlson-Fritsch algorithm, which described how to create a monotone C^1 surface from monotone gridded data. The contribution of Han and Schumaker's work was to describe how to convert scattered data to gridded data. First, they applied an arbitrary scattered data interpolation method (e.g., TPS, Hardy multiquadratics) to produce gridded data. They then adjusted those interpolated values to ensure that they are left with a monotone grid. At this point, the Carlson-Fritsch method can be employed to construct the final C^1 surface. Then, they converted the rectangular grid into a triangular grid and applied Bernstein-Bezier polynomials to construct the C^1 surface. They developed monotonicity constraints on these polynomials. It is not clear what advantages these particular polynomials hold that warranted this approach and the development of the associated monotonicity constraints. They did not compare their results with the Carlson-Fritsch method once the monotone gridded data was constructed.

There is a flaw in the Han-Schumaker algorithm that deals with updating the first derivatives at the grid nodes, e.g., Z_{ij}^x and Z_{ij}^y. The problem lies in the fact that their updates of Z_{ij}^x are actually dependent on the initial estimates of Z_{ij}^y. If Z_{ij}^y had to be updated themselves, then the evaluated Z_{ij}^x would be wrong. We have, in fact, confirmed this error on some data sets. See Fig. 2.3 for an example.

2.5.3 Mono-SDI Algorithm for Monotone C^2 Scattered Data

Empirical results with monotone gridded sets show that for sufficiently fine grids an unconstrained scattered data interpolation method may produce a monotone surface. Based on this observation we propose an algorithm for monotone C^2 scattered data interpolation.

The formulation of the algorithm uses the following notation:

- $G = GRID(P)$ is a procedure that creates a grid that passes through the (x, y) positions of the scattered data.

- $I = SDI(P, G)$ is a procedure that interpolates data P, and evaluates the surface at grid locations G.
- $I_M = ADJUST(I)$ is a procedure that adjusts gridded data to be monotone as described in the previous section.
- $G_{new} = REFINE(G)$ is a procedure that refines a grid G. The procedure finds the largest grid step and adds a grid line to bisect that space.

The following procedure outlines the method, which we denote as the Mono-SDI algorithm.

Mono-SDI Algorithm

Input: Monotone scattered data $P = \{(x_i, y_i, z_i)\}_{i=1}^{N}$

Output: I - C^2 monotone interpolant defined over grid G

$G_0 = GRID(P)$

$\tilde{I} = SDI(P, G_0)$

$\tilde{I} = ADJUST(\tilde{I})$

$I = SDI([G_0, \tilde{I}], G)$

$i = 0$

while I is not monotone **do**

$G_{i+1} = REFINE(G_i)$

$\tilde{I} = SDI(P, G_{i+1})$

$\tilde{I} = ADJUST(\tilde{I})$

$I = SDI([G_{i+1}, \tilde{I}], G)$

$i = i + 1$

end

Note that $[G_i, \tilde{I}]$ refers to the set of data values produced by concatenating the (x, y) positions in G_i with the z values of $\tilde{I}$. The refine-interpolate-adjust process outlined above does not change the Z_{ij} for points in P and so the final values are still consistent with the given data.

Fig. 2.9 demonstrates the Mono-SDI algorithm using thin plate splines to perform scattered data interpolation. Fig. 2.9(a) shows an 8×8 grid G corresponding to ten scattered data points.

Fig. 2.9(b) depicts the surface produced by interpolating the grid nodes using the TPS method. Fig. 2.9(d) illustrates the 17 × 18 grid obtained after the refinement process. Finally, Fig. 2.9(e) shows the final surface. Evaluations on a 100 × 100 grid were used to verify the surface monotonicity. The mean and maximum error is 0.06156 and 0.38712, respectively.

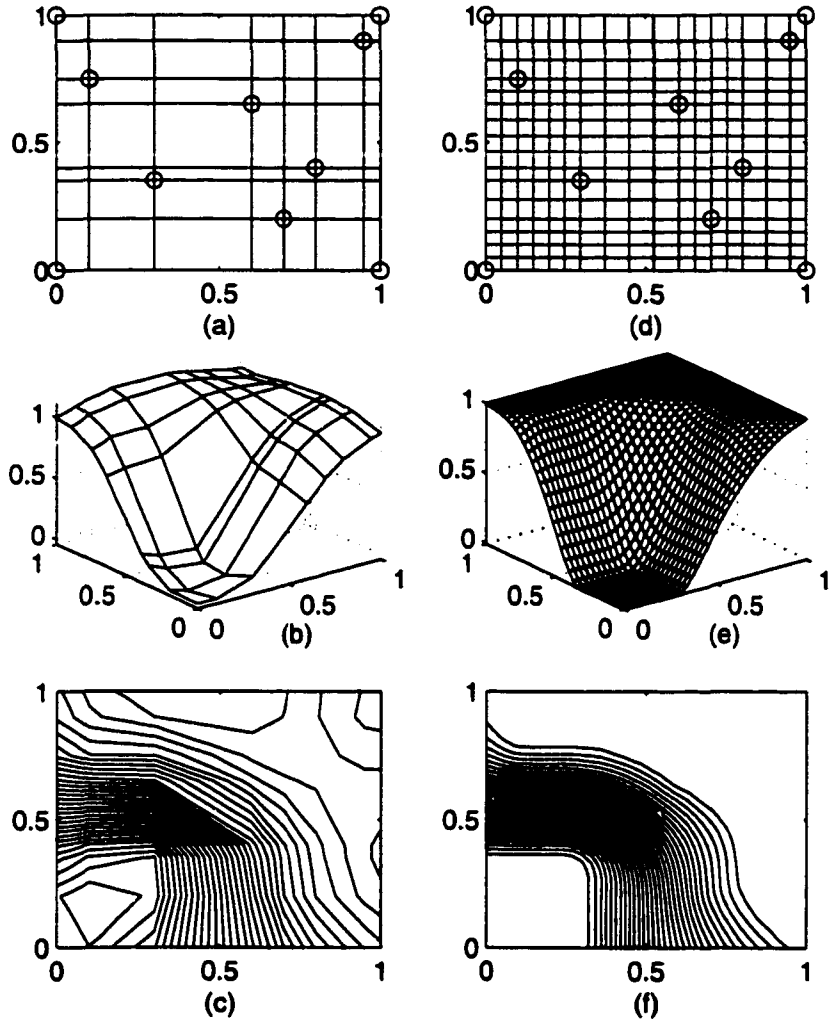


Figure 2.9: (a) Ten scattered data points and grid; (b) TPS interpolation; (c) level curves of (b); (d) refinement; (e) final surface; (f) level curves of (e).

Fig. 2.10 demonstrates the Mono-SDI algorithm with the MBA method used to perform the scattered data interpolation. Fig. 2.10(a) shows an 8 × 8 grid G corresponding to ten scattered data points. Fig. 2.10(b) depicts the surface produced by interpolating the grid nodes using the MBA algorithm. Fig. 2.10(d) illustrates the 17 × 16 grid obtained after the refinement process. Finally, Fig. 2.10(e) shows the final surface. The mean and maximum error is 0.06432 and 0.44111, respectively.

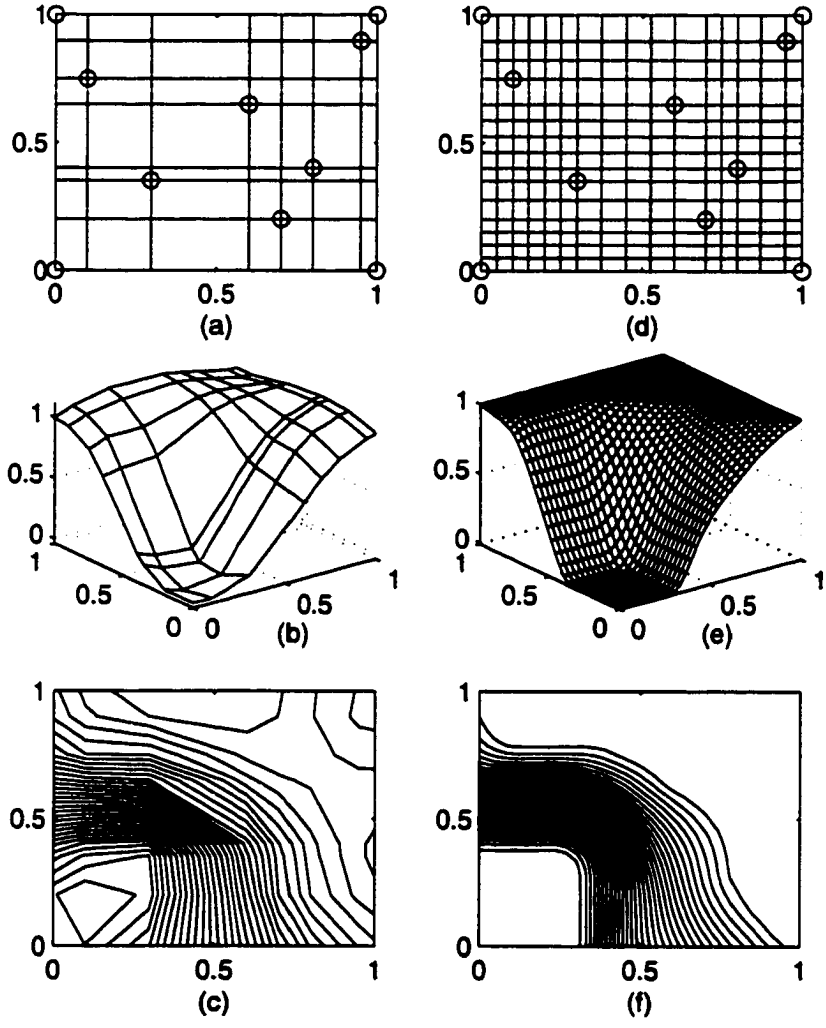


Figure 2.10: (a) Ten scattered data points and grid; (b) MBA interpolation; (c) level curves of (b); (d) refinement; (e) final surface; (f) level curves of (e).

2.6 COMPARISON

In this section, we compare the various techniques for gridded and scattered data. We consider the Mono-SDI, energy minimization, Han-Schumaker, Carlson-Fritsch, and feasible solution methods. These methods are ranked in Table 2.1 and Table 2.2 for the gridded and scattered data cases, respectively, beginning with the monotone C^2-MBA surface.

Method	Continuity	Speed
Mono-SDI	C^2	fast
energy minimization	C^2 / C^1	slow
Han-Schumaker	C^1	fast
Carlson-Fritsch	C^1	fast
feasible solution	C^1	slow

Table 2.1: Comparison of proposed methods for gridded data.

Method	Continuity	Speed
Mono-SDI	C^2	fast
energy minimization	C^2 / C^1	slow
feasible solution	C^1	slow
Han-Schumaker	C^1	fast

Table 2.2: Comparison of proposed methods for scattered data.

2.7 SUMMARY

This chapter described extensions to previous work in the area of monotone surface interpolation. The bulk of the literature is devoted to the problem of fitting a monotone surface through gridded data. The early work of Carlson and Fritsch [13, 46, 14] developed sufficient conditions on the Hermite derivatives for monotone bicubic functions. Their work, limited to gridded data, determined the coefficients of the bicubic functions forming a monotone C^1 interpolant.

We showed how to formulate the solution for scattered data by constructing a system of linear constraints to produce the desired coefficients by using linear programming to solve for a feasible, albeit not optimal, solution. Depending on the size of the problem, either the simplex

method or the primal-dual interior point method is used to solve for the unknown derivatives. This approach has two problems: it doesn't guarantee a C^2 solution and it is slow.

To overcome these problems, we introduced an iterative grid refinement process that may be coupled to any scattered data interpolation procedure to produce a monotone C^2 surface. Thin plate splines [29] and multilevel B-spline interpolation [60] were considered for the underlying interpolation methods. The iterative grid refinement process introduced in this thesis is our method of choice for producing monotone C^2 interpolants through either gridded or scattered data.

Chapter 3

SCATTERED DATA INTERPOLATION

3.1 INTRODUCTION

Scattered data interpolation refers to the process of fitting a smooth surface through scattered data. This process is of great interest to many fields of science and engineering. We therefore seek to construct a smooth bivariate function $z = f(x, y)$ that may be evaluated at any position in the problem domain Ω. The function is considered smooth if it is at least C^1 continuous.

There are three principal sources of scattered data: measured values of physical quantities, experimental results, and computational values [42]. For example, nonuniform measurements of physical quantities are collected in geology, meteorology, oceanography and mining. Scattered experimental data is produced in chemistry, physics and engineering. Nonuniformly spaced computational values arise in various applications in computer graphics and computer vision.

These fields require scattered data interpolation to determine values at arbitrary positions, not just those at which the data is available. This facilitates many useful operations for visualizing scattered multidimensional data. For instance, in medical imaging, scattered data interpolation is essential to construct a closed surface from CT and MRI images of human organs. In geological applications, the derived interpolation function facilitates a contour map. Computer vision utilizes scattered data interpolation to perform visual surface reconstruction on sparse measurements. In image morphing, scattered data interpolation is useful for deriving a smooth mapping function from the correspondence of feature points between a pair of images.

Despite a flurry of activity in this area, scattered data interpolation remains a difficult

and computationaly expensive problem. The vast literature devoted to this subject documents various approaches, many of which suffer from limitations in smoothness, time complexity or allowable data distributions [42].

The *multiresolution B-spline approximation (MBA)* technique was introduced in [60, 86] for image morphing. The MBA is a fast algorithm to compute C^2 interpolating surfaces. It makes use of a coarse-to-fine hierarchy of control lattices to generate a sequence of bicubic B-spline functions whose sum approaches the desired interpolation function. The presentation of the MBA in [60, 86] is application-oriented and was demonstrated to produce high fidelity results in image reconstruction, image warping and object reconstruction.

In this work, we investigate some key properties of the MBA algorithm, including its basis functions, kernel for uniform data, and shift-variant property. Based on this analysis, we propose a simpler and more intuitive method – *Multiresolution Filtering Approximation (MFA)*. Interpolation is achieved by successively improving the approximation by applying a hierarchical set of low-pass filters.

We conducted a performance comparison of MBA, MFA and TPS schemes. The comparisons are based on the work in [41]. The TPS algorithm is Duchon's thin plate spline interpolation function [29]. It is a global interpolating function which was among the most successful interpolation schemes tested in [41].

3.2 PREVIOUS WORK

There is a vast literature devoted to scattered data interpolation. Fine surveys can be found in [42, 54, 75, 6]. In this section, we review several dominant approaches based on Shepard's method, radial basis functions, thin plate splines, and finite element methods. We also consider related research in image processing and geometric modeling, including multiresolution filtering, direct manipulation techniques, and hierarchical refinement.

One of the earliest algorithms in this field was based on inverse distance weighting of data developed by meteorologists and geologists [24] [25], it has become known as Shepard's method [79]. Shepard defined the interpolation function as the weighted average of the data,

with the weights being inversely proportional to distance.

$$f(x,y) = \frac{\sum_{k=1}^{N} w_k(x,y) z_k}{\sum_{k=1}^{N} w_k(x,y)}$$

where

$$w_k = \frac{1}{(x-x_k)^2 + (y-y_k)^2}$$

and $\{(x_k, y_k, z_k)\}$ are the given scattered data. This technique suffers from flat spots at the data points, as well as undue influence of points which are far away. Furthermore, it is a global method requiring all the weights to be recomputed if any data point is added, removed, or modified. Franke and Nielson introduced the modified quadratic Shepard's method [79] to address these deficiencies and produce C^1 interpolation.

Another popular approach to scattered data interpolation is to define the interpolation function as a linear combination of radially symmetric basis function, each centered at a data point.

$$f(x,y) = \sum_{k=1}^{N} a_k G_k(r_k) + p_m(x,y)$$

where

$$r_k = \sqrt{(x-x_k)^2 + (y-y_k)^2}$$

and $p_m(x,y)$ is a bivariate polynomial of degree m. The unknown coefficients for the basis functions are determined by solving a linear system of equations to interpolate the data. The coefficient matrix is always full and, for large data sets, it may become poorly conditioned and require preconditioning [31][30]. Popular choices for the basis functions include the Gaussian $\left(G_k = e^{-\frac{r_k}{R^2}}\right)$, Hardy's Multiquadratic $\left(G_k = \sqrt{r_k^2 + R^2}\right)$ [53], and Shifted Log $\left(G_k = \log(r_k^2 + R^2\right)$ [41, 30, 42, 54] where R is a parameter.

Thin plate splines (TPS) are derived by minimizing the roughness R of the surface

$$R = \int_{R^2} \left(\frac{\partial^2 f}{\partial x^2} + 2\frac{\partial^2 f}{\partial x \partial y} + \frac{\partial^2 f}{\partial y^2} \right)^2 d_x d_y$$

among all interpolating functions. The solution is of the form

$$f(x,y) = \sum \alpha_k r_k^2 \log r_k + ax + by + c$$

and the system of equations that is solved is

$$\begin{aligned} f(x_k, y_k) &= f_k \\ \sum_{k=1}^{N} \alpha_k &= \sum_{k=1}^{N} \alpha_k x_k = \sum_{k=1}^{N} \alpha_k y_k = 0 \end{aligned}$$

TPS are widely used due to their visually pleasing results and stability for large data sets. Although they are usually formulated as the solution of a variational problem, Duchon has shown thin plate splines to be derived from radial basis functions [29]. The numerical solution of thin plate splines can be accelerated by using another approach based on multigrid relaxation techniques [10, 11, 82]. An alternative to relaxation has been presented that uses hierarchical basis functions or wavelets to accelerate the convergence of an iterative technique such as conjugate descent [81, 49]. Nevertheless, the numerical solution remains computationally expensive when the interpolation function is computed on a large grid. Thin plate splines have been used to generate smooth warp functions for image warping and morphing [61, 58, 59].

C^2 bicubic surfaces for gridded data were introduced in [26, 9]. This function minimizes $\int\int \left[\frac{\partial^2 f(x,y)}{\partial^2 x \partial^2 y}\right]^2 dxdy$ [75, 2]. Ahlberg, Nilson, and Walsh [1, 2] generalized the interpolation problem to minimize $\int\int \left[\frac{\partial^{m+n} f(x,y)}{\partial^m x \partial^n y}\right]^2 dxdy$, for even m and n, using higher order polynomial splines.

Another class of solutions to scattered data interpolation is due to finite element methods. These methods are based on the concept of using C^1 finite element functions on rectangular grids or on a triangulation of the point set. The latter approach requires a scheme for estimating some derivatives at the data points. It involves creating an optimal triangulation on the set of data points to delimit local neighborhoods over which surface patches are defined. These patches are constrained to interpolate the original data. There are several criteria suggested by Lawson [57] to derive optimal triangulation in which long thin triangles with small angles are avoided. Piecewise linear approximation over the triangulation is not smooth, achieving only

C^0 continuity. The most common C^1 method uses the Clough-Tocher triangular interpolant [17, 6, 50]. A related technique was proposed in [66]. Triangulation methods however, are sensitive to data distribution, i.e., long thin triangles cannot be always be avoided.

Schumaker [75] proposed a two-stage method that first generates a grid of data using any method for scattered data interpolation. The second stage applies a standard tensor product approximation on the grid. The resulting approximation may be made to interpolate the original data through the use of an iterative process called the delta iteration, developed by Foley and Nielson [35]. Arge *et al.* proposed an approximation scheme consisting of three steps: regularization, local approximation and extrapolation [4]. They first determine the approximation function by a local method at a subset of the grid points where the data density is high. That function is then extrapolated to the entire grid by a global method.

In the image processing community, scattered data interpolation is necessary to perform reconstruction among uniform samples. A fine survey of nonuniform reconstruction techniques can be found in [47]. A trend in recent algorithms has been the use of hierarchical, or multiresolution, filtering to extend onto all positions the information known only at sparse and irregular samples. Burt proposed hierarchical polynomial fit filtering to yield a multiresolution set of low-pass filtered images that can be combined to form a smooth surface passing through the original data [12]. Mitchell proposed multistage filtering to handle highly variable sample density [67]. In that work, weighted-average filters are repeatedly applied with ever-narrowing low pass cutoff, until the proper bandwidth for the displayed is reached.

Recent work in geometric modeling has addressed scattered data interpolation. A B-spline approximation technique for scattered data was introduced to directly manipulate an object modeled by free-form deformation [55]. That technique calculates the pseudoinverse of a matrix containing B-spline basis function values to minimize the approximation error. Welch and Witkin proposed a variational approach to directly manipulate B-spline surfaces with scattered points or curves [84]. The control lattice is determined by minimizing the sum of energy functional to derive a smooth interpolating surface. The high computational cost of pseudoinverse and energy minimization solutions render these methods prohibitively expensive for large data sets.

Forsey and Bartlet proposed hierarchical B-spline refinement for object modeling [38].

They augmented B-spline approximation with that technique to interpolate a grid of data using a control lattice hierarchy [39][40]. Interpolation is achieved by successively improving the approximation at a coarse level with a correction term from the next finer level. Although their method is similar in spirit to the multilevel B-spline approximation presented in [60], their method cannot handle scattered data.

3.3 MULTIRESOLUTION B-SPLINE APPROXIMATION

In [60, 86] a multiresolution B-spline approximation algorithm for scattered data was presented. The algorithm produces a hierarchy of control lattices using the BA algorithm. Each control lattice generates a bicubic B-spline function. The resulting approximation surface is the sum of the functions corresponding to all of the lattices. We first present the BA and MBA algorithms and then discuss some key properties of the approximation.

3.3.1 B-spline approximation - BA algorithm

The BA algorithm is defined using the following notations and definitions:

$\Omega = \{(x,y)|0 \leq x < a, 0 \leq y < b\}$ is a rectangular region in R^2.

$P = \{(x_c, y_c, z_c)\}_{c=1}^{N}$ are N distinct data points and $\{(x_c, y_c)\}$ are in Ω.

$m, \Delta x$ partition of $x - axis$ such that $m\Delta x = a$

$n, \Delta y$ partition of $y - axis$ such that $n\Delta y = b$

$\tilde{\Omega} = \{(x,y)| - \Delta x \leq x < a + \Delta x, -\Delta y \leq y < b + \Delta y\}$ is a rectangular region in R^2

Φ is an $m \times n$ lattice overlaid on $\tilde{\Omega}$ with spacings Δx and Δy

$\{(x_i, y_j)\} = \{i\Delta x, j\Delta y\}$, $i = [-1, m+1]$ and $j = [-1, n+1]$ locations of lattice crossings

$\{\phi_{ij}\}$ control points on Φ located at $\{(x_i, y_j)\}$

Fig. 3.1 shows the configuration of Φ in $\tilde{\Omega}$.

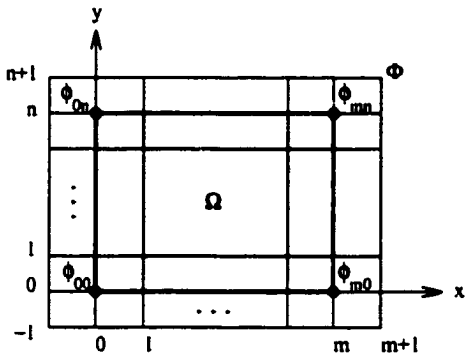


Figure 3.1: The configuration of control lattice Φ.

Definition 5 *The bicubic B-spline function is*

$$f(x,y) = \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(s)B_l(t)\phi_{(i+k)(j+l)} \tag{3.1}$$

where

$$\begin{aligned} i &= \left\lfloor \frac{x}{\Delta x} \right\rfloor - 1 \\ j &= \left\lfloor \frac{y}{\Delta y} \right\rfloor - 1 \\ s &= \frac{x - i\Delta x}{\Delta x} \\ t &= \frac{y - j\Delta y}{\Delta y} \end{aligned}$$

and $\{B_i(t)\}_{i=0}^{3}$ *are the B-spline basis functions defined on the [0,1] interval:*

$$\begin{aligned} B_0(t) &= (1-t)^3/6 \\ B_1(t) &= (3t^3 - 6t^2 + 4)/6 \\ B_2(t) &= (-3t^3 + 3t^2 + 3t + 1)/6 \\ B_3(t) &= t^3/6 \end{aligned} \tag{3.2}$$

Definition 6 *The proximity* P_{ij} *of* ϕ_{ij} *are data points in P that lie in the square region:*

$$(i-2)\Delta x \leq x < (i+2)\Delta x$$

$$(j-2)\Delta y \leq y < (j+2)\Delta y$$

The approximation function is the bicubic B-spline of Eq. (3.1) where the control points $\{\phi_{ij}\}$, are obtained by a weighted sum of their proximity as follows:

$$\begin{aligned} \phi_{ij} &= \sum_{c \in P_{ij}} z_c \xi_c \\ \xi_c &= \frac{[B_{k_c}(s_c)B_{l_c}(t_c)]^3}{\left[\sum_{a=0}^{3}\sum_{b=0}^{3}[B_a(s_c)B_b(t_c)]^2\right]\left[\sum_{c \in P_{ij}}[B_{k_c}(s_c)B_{l_c}(t_c)]^2\right]} \end{aligned} \tag{3.3}$$

where

$$\begin{aligned} i_c &= \left\lfloor \frac{x_c}{\Delta x} \right\rfloor - 1 \\ j_c &= \left\lfloor \frac{y_c}{\Delta y} \right\rfloor - 1 \\ k_c &= i - i_c + 1 \\ l_c &= j - j_c + 1 \\ s_c &= \frac{x_c - i_c \Delta x}{\Delta x} \\ t_c &= \frac{y_c - j_c \Delta y}{\Delta y} \end{aligned}$$

To make the presentation complete we describe the two considerations that lead to the above function. Assuming one data point at (x_c, y_c, z_c), there exists more then one set of control points that yield an interpolating bicubic function. The authors chose the set that minimizes the sum $\sum_{k=0}^{3}\sum_{l=0}^{3}\phi_{kl}^2$. The solution of the constraint minimization problem is

$$\phi_{kl} = z_c \frac{B_{k_c}(s_c)B_{l_c}(t_c)}{\sum_{a=0}^{3}\sum_{b=0}^{3}[B_a(s_c)B_b(t_c)]^2}$$

To solve for the general case where the proximity of a control point contains more then one data point, or in other words, more then one data point shares the same control point, the authors chose the final value of ϕ_{ij} to be the one that minimizes

$$e = \sum_c [B_{k_c}(s_c)B_{l_c}(t_c)\phi_{ij} - B_{k_c}(s_c)B_{l_c}(t_c)\phi_c]^2$$

where ϕ_c is the value of the control point assuming only (x_c, y_c, z_c) exist in its proximity.

The term $B_{k_c}(s_c)B_{l_c}(t_c)\phi_{ij} - B_{k_c}(s_c)B_{l_c}(t_c)\phi_c$ is the difference between the real and expected contribution of ϕ_{ij} to function f at (x_c, y_c). The solution of that minimization problem is the expression in Eq. (3.3).

3.3.2 MBA algorithm

The MBA algorithm makes use of a hierarchy of control lattices $\Phi_0, ..., \Phi_h$ with decreasing spacing $\Delta x_0 > \Delta x_1 ... > \Delta x_h$ and $\Delta y_0 > \Delta y_1 ... > \Delta y_h$. The algorithm sequentially applies the BA procedure to the residual, beginning with Φ_0 and $\{(x_c, y_c, z_c)\}_{c=1}^{N}$ as the initial residual. The following pseudocode describes the algorithm:

Input: (a) Scattered data P (b) Lattices spacings $\{\Delta x_i, \Delta y_i\}_0^h$,

Output: (a) Control lattices $\{\Phi_i\}_{i=0}^h$ (b) Approximating function $f(x, y)$

let $k = 0$

let $P_0 = P$

while $k \leq h$ do

compute Φ_k from $P_k, \Delta x_k, \Delta y_k$ using BA algorithm

compute $f_k(x, y)$

compute $P_{k+1} = \{(x_c, y_c, z_c - \sum_{i=0}^{k} f_i(x_c, y_c))\}$

let $k = k + 1$

end

compute $f(x, y) = \sum_{i=0}^{h} f_k(x, y)$

3.3.3 Discussion

In this section, we review several key points about the BA and MBA algorithms.

Basis functions Basis functions are defined as the contribution of a point in P to the reconstruction function. We describe the BA basis functions and show that $f(x, y)$ is a weighted sum (linear combination) of its basis functions.

Theorem 7 *The control points produced by the BA algorithm can be obtained by solving an equivalent linear problem. Define P^v to be the set of scattered data where the values are all*

zero except at location v:

$$P^v = \begin{cases} (x_i, y_i, 0) & i \neq v \\ (x_i, y_i, z_i) & i = v \end{cases}$$

Let ϕ_{ij}^v be the control point obtained by solving the BA equations for data P^v. The value of control point ϕ_{ij} is:

$$\phi_{ij} = \sum_{v=1}^{N} \phi_{ij}^v \tag{3.4}$$

Proof. Let $\tilde{\phi}_{ij}$ be the sum in Eq. (3.4). Since ξ_c is independent of z we get:

$$\begin{aligned} \widetilde{\phi_{ij}} &= \sum_{v=1}^{N} \phi_{ij}^v = \sum_{v=1}^{N} \sum_{c \in P_{ij}} z_c \xi_c \\ &= \sum_{[k,\ldots,c,\ldots,l] \in P_{ij}} (0 \cdot \xi_k + \ldots + z_c \cdot \xi_c + \ldots + 0 \cdot \xi_l) \qquad (3.5) \\ &= \sum_{c \in P_{ij}} z_c \xi_c = \phi_{ij} \qquad (3.6) \end{aligned}$$

■

A direct result of the above theorem is that $BA(P) = \sum BA(P^v)$ and that $MBA(P) = \sum MBA(P^v)$. This is demonstrated in the following example. Consider P, P_1, P_2, P_3, P_4 to be a set of points, as shown in Table 3.1. Figures 3.2(a)-3.2(e) show the curves obtained by applying MBA to P, P_1, P_2, P_3, P_4 respectively, with $\{\Delta x_i\} = \{\Delta y_i\} = \{2^{-i}\}_{i=0}^{5}$ at $y = 0.5$. Figure 3.2(f) is the sum of the four curves in Figures 3.2(b)-3.2(e). Notice that it is identical to Fig. 3.2(a). In all cases, the support of each basis function is the domain Ω.

$P =$ {	(0.1,0.5,4)	(0.3,0.5,1)	(0.5,0.5,6)	(0.6,0.5,3) }
$P_1 =$ {	(0.1,0.5,4)	(0.3,0.5,0)	(0.5,0.5,0)	(0.6,0.5,0) }
$P_2 =$ {	(0.1,0.5,0)	(0.3,0.5,1)	(0.5,0.5,0)	(0.6,0.5,0) }
$P_3 =$ {	(0.1,0.5,0)	(0.3,0.5,0)	(0.5,0.5,6)	(0.6,0.5,0) }
$P_4 =$ {	(0.1,0.5,0)	(0.3,0.5,0)	(0.5,0.5,0)	(0.6,0.5,3) }

Table 3.1: Data points

MBA kernel Sampling theory for uniform sampling establishes that the sinc function is the ideal interpolation kernel. We derived the MBA kernel by applying the MBA algorithm to an

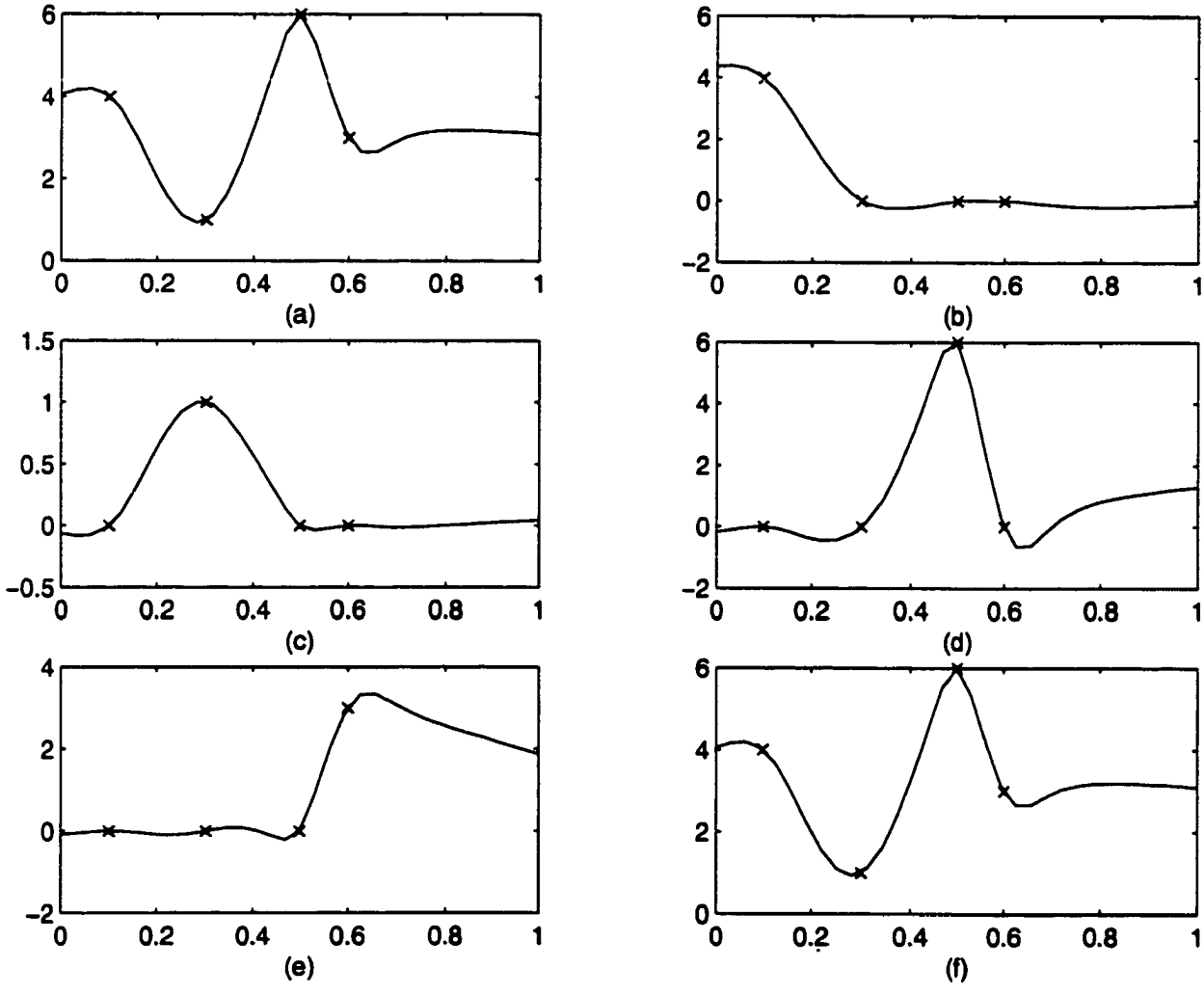


Figure 3.2: (a) MBA(P) (b) MBA(P_1) (c) MBA(P_2) (d) MBA(P_3) (e) MBA(P_4) (f) $\sum_{i=1}^{4}$MBA(P_i)

impulse:

$$\begin{aligned} P &= \{(0.5, 0.5, 1.0), (0.1 \cdot i, 0.1 \cdot j, 0.0)\} \\ i, j &= 0, 1, 2, 3, 4, 6, 7, 8, 9, 10 \end{aligned}$$

Fig. 3.3 shows the kernel and its spectrum. The support of the kernel is the domain Ω. The transition of the spectrum is not sharp, compared to the ideal reconstruction filter, i.e., a box.

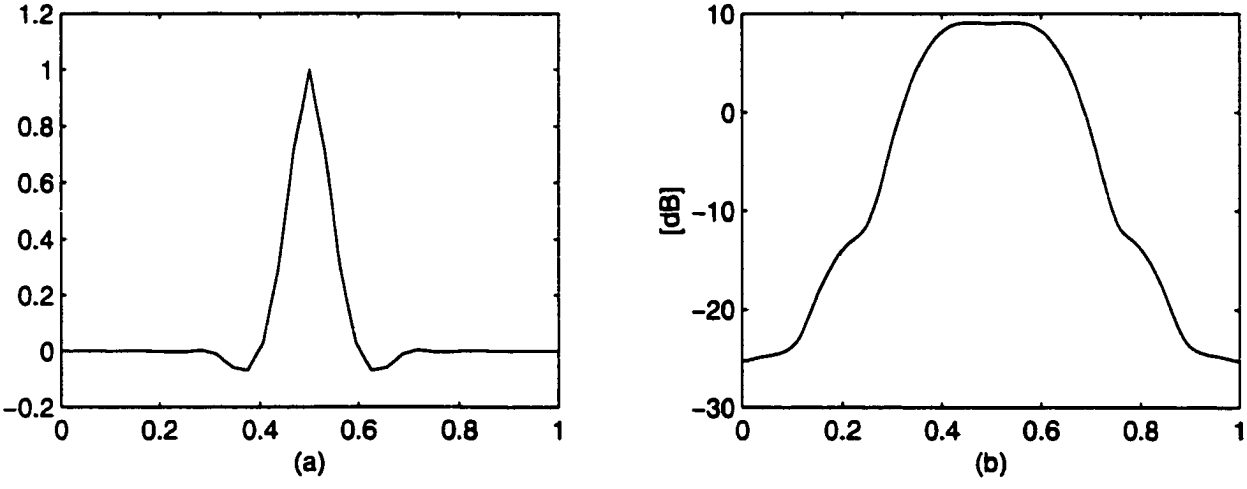


Figure 3.3: (a) MBA kernel; (b) Spectrum.

Note that the kernel will change shape for different impulse locations.

Shift-variance A desirable mathematical property of interpolation and approximation algorithms is shift-invariance, i.e., a shift in the input produces a shift in the output. The BA algorithm is a shift-variant functional and produces different surfaces for shifted data points.

$$f_{BA}(x, y)|_{\{(x_c, y_c, z_c)\}} \neq f_{BA}(x + \Delta x, y + \Delta y)|_{\{(x_c + \Delta x, y_c + \Delta y, z_c)\}}$$

This is due to the nonlinear floor operation of Eq. (3.3). The following example demonstrates that undesired property. Consider the following two impulse signals $P_1 = (0.5, 0.5, 1)$ and $P_2 = (0.8125, 0.5, 1)$. Figures 3.4(a) and 3.4(b) show curves $f_1(x, 0.5)$ and $f_2(x, 0.5)$ produced with the BA algorithm using a grid spacing of $\Delta x = \Delta y = 0.5$, for P_1 and P_2, respectively. Fig. 3.4(c) shows the overlay of $f_1(x, 0.5)$ and $f_2(x - 0.3125, 0.5)$.

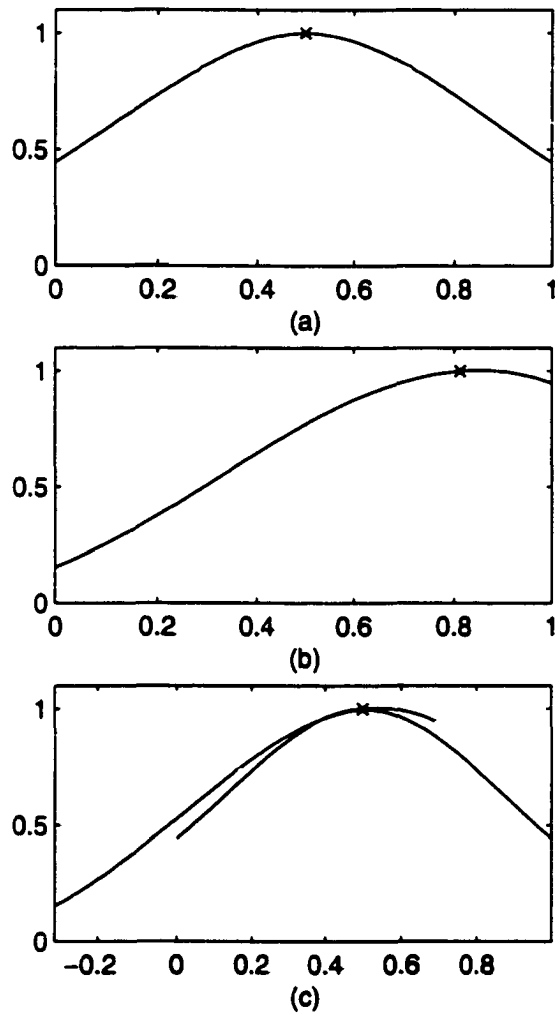


Figure 3.4: Shift-variant property of MBA. (a) $f_1(x, 0.5)$ (b) $f_2(x, 0.5)$ (c) overlay $f_1(x, 0.5), f_2(x - 0.3125, 0.5)$

Interpolation condition The MBA algorithm generates an approximation function f that passes near the data points P, but not necessarily through them. The following sufficient conditions for a control lattice generate an interpolating function with zero residual:

$$\begin{aligned} MIN(dx_k) &\geq 4\Delta x \\ MIN(dy_k) &\geq 4\Delta y \end{aligned}$$

where dx_k (dy_k) is the horizontal (vertical) distance to the closest point from (x_k, y_k, z_k). In other words, the interpolating condition is that no two data points share a control point in their 4×4 neighborhoods [60].

Initial linear approximation If the coarsest control lattice is very fine, many control points may have empty proximity, and the final approximation will contain local peaks near the data points, which may be unsatisfactory for many applications. The authors in [60] suggested to use an initial linear fit to the data $f_{-1}(x, y) = a_1 x + a_2 y + a_3$ and solve for the parameters a_1, a_2, a_3 that minimize the residual energy $\sum_c [z_c - f_{-1}(x_c, y_c)]^2$. The result is

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = (A^T A)^{-1} a^T b \tag{3.7}$$

where

$$A = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_N & Y_N & 1 \end{pmatrix}, \quad b = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{pmatrix} \tag{3.8}$$

3.4 MULTIRESOLUTION FILTERING APPROXIMATION

We showed in the previous section that the interpolation surface produced by the MBA algorithm is a nonlinear shift-variant function of the data points. It is also evident that the resulting function is a weighted sum of the points, where the weights are obtained through a nonlinear process. The approximation process of the BA algorithm can be generalized as a two-phase mapping:

$$P \rightarrow \Phi \rightarrow f(x,y)$$

The first phase $P \rightarrow \Phi$ is performed using

$$\begin{aligned} \phi_{ij} &= \sum_{c \in P_{ij}} z_c \xi_c \\ \xi_c &= \frac{[B_{k_c}(s_c) B_{l_c}(t_c)]^3}{\left[\sum_{a=0}^{3} \sum_{b=0}^{3} [B_a(s_c) B_b(t_c)]^2\right] \left[\sum_{c \in P_{ij}} [B_{k_c}(s_c) B_{l_c}(t_c)]^2\right]} \end{aligned}$$

and the second phase $\Phi \rightarrow f(x,y)$ is performed using

$$f(x,y) = \sum_{k=0}^{3} \sum_{l=0}^{3} B_k(s) B_l(t) \phi_{(i+k)(j+l)}$$

We showed in the previous section that the MBA algorithm interpolates a single point using a bell-shaped function. The width (support) of the bell-shaped function is $8\Delta x \times 8\Delta y$. Suppose we apply the BA algorithm to a single point using lattices $\Phi_0, \ldots, \Phi_h$ with decreasing spacing $\Delta x_0 > \Delta x_1 \ldots > \Delta x_h$ and $\Delta y_0 > \Delta y_1 \ldots > \Delta y_h$. This will result in a hierarchy of bell-shaped functions with decreasing width.

We use that property to suggest a one-phase approximation algorithm $P \rightarrow f(x,y)$, which we call *multiresolution filtering approximation (MFA)*. In the filtering approximation (FA) algorithm we apply a reconstruction filter directly to the points by centering the impulse response at the points. The MFA algorithm applies the FA scheme to the residual with a decreasing filter width. We will now show that the advantages of the MFA are its simplicity and linearity.

3.4.1 Filtering Approximation - FA algorithm

The FA algorithm is defined using the following notations and definitions:

$\Omega = \{(x, y) | 0 \leq x < a, 0 \leq y < b\}$ is a rectangular region in R^2.

$P = \{(x_c, y_c, z_c)\}_{c=1}^{N}$ are N distinct data points and $\{(x_c, y_c)\}$ are in Ω.

$h(t)$ is the cubic B-spline kernel [85] with support of 4

$$h[t] = \begin{cases} 3|t|^3 - 6|t|^2 + 4 & 0 \leq |t| < 1 \\ -|t|^3 + 6|t|^2 - 12|t| + 8 & 1 \leq |t| < 2 \\ 0 & 2 \leq |t| \end{cases}$$

$A = \{(x_c, y_c, a_c)\}_{c=1}^{N}$ is a set of impulses that coincide with the data points in P.

$\Delta x, \Delta y$ are scaling factors for the cubic B-spline kernel.

Definition 8 *Bicubic B-spline function*

$$\begin{aligned} g(x, y) &= \sum_{c \in P} a_c h[r_c(x, y)] \\ r_c(x, y) &= \sqrt{\left(\frac{x - x_c}{\Delta x}\right)^2 + \left(\frac{y - y_c}{\Delta y}\right)^2} \end{aligned} \tag{3.9}$$

Definition 9 *The proximity Q_c of (x_c, y_c, z_c) are data points in P that rely in the square region:*

$$\begin{aligned} x_c - 2\Delta x &\leq x < x_c + 2\Delta x \\ y_c - 2\Delta y &\leq y < y_c + 2\Delta y \end{aligned}$$

The FA algorithm seeks to find amplitudes $\{a_c\}$ in Eq. (3.9) that produce a good approximation. In order to derive an optimal solution, we must solve the MSE problem

$$MIN \sum_{i=1}^{N} \left[z_i - \sum_{c=1}^{N} a_c h[r_c(x_i, y_i)] \right]^2$$

Although the solution is linear with the amplitudes, it requires us to solve an $N \times N$ matrix. This is prohibitively expensive for large data sets. Alternatively, we may define e_c to be the absolute residual at (x_c, y_c) assuming that all the data points have the same amplitudes a_c:

$$e_c = \left| z_c - a_c \sum_{i=1}^{N} h[r_c(x_i, y_i)] \right|$$

The minimum residual $e_c = 0$ can be obtained with:

$$a_c = \frac{z_c}{\sum_{i=1}^{N} h[r_c(x_i, y_i)]} \tag{3.10}$$

The result is the ratio of z_c over the sum of the contributions of the cubic kernels centered at the data points with unit amplitude. Note that only points in Q_c will affect a_c.

3.4.2 MFA algorithm

The MFA algorithm makes use of a hierarchy of cubic B-spline kernels $h_0(t), ..., h_h(t)$ with decreasing support $\Delta x_0 > \Delta x_1 ... > \Delta x_h$ and $\Delta y_0 > \Delta y_1 ... > \Delta y_h$. The algorithm sequentially applies the FA procedure to the residual R_{k-1}, where $R_0 = \{(x_c, y_c, z_c)\}_{c=1}^{N}$. The following pseudocode outlines the algorithm: Note that we apply several iterations of the FA algorithm at each level to improve the approximation accuracy of the coefficients in Eq. (3.10).

MFA Algorithm

Input: scattered data $P = \{(x_c, y_c, z_c)\}$

Output: approximating function $g(x, y)$

let $i = k = 1$

while $k \leq h$ **do**

  while $i \leq itrs$ **do**

    let $P_k = \{(x_c, y_c, \Delta^k z_c)\}$

    update $\{a_{k,c}\}$ from P_k by the FA algorithm

    update $\Delta^k z_c = \Delta^k z_c - g_k(x_c, y_c)$ for all c

    let $i = i + 1$

  end

let $\Delta^{k+1} z_c = \Delta^k z_c$

let $k = k + 1$

end

compute $g(x, y) = \sum_{k=1}^{h} \sum_c a_{k,c} h_k[r_c(x, y)]$

3.4.3 Discussion

In this section, we review several key points about the FA and MFA algorithms.

Basis functions We describe the FA basis function and show that $g(x, y)$ is a weighted sum (linear combination) of its basis function.

Theorem 10 *The amplitudes produced by the FA algorithm can be obtained by solving an equivalent linear problem. Define P^v to be the set of scattered data where the values are all zero except at location v:*

$$P^v = \begin{cases} (x_i, y_i, 0) & i \neq v \\ (x_i, y_i, z_i) & i = v \end{cases}$$

We also define a_i^v to be the i-th point amplitude obtained by solving the FA equations for data sets P^v. Then,

$$g(x, y) = FA(P) = \sum_{v=1}^{N} FA(P^v) \tag{3.11}$$

Proof.

$$\begin{aligned} g(\widetilde{x, y}) &= \sum_{v=1}^{N} FA(P^v) = \sum_{v=1}^{N} \sum_{i=1}^{N} a_i^v h[r_c(x, y)] \\ &= \sum_{i=1}^{N} a_i^v h[r_c(x, y)] = \sum_{i=1}^{N} a_i h[r_c(x, y)] \\ &= FA(P) = g(x, y) \end{aligned}$$

The third equality is due to the fact that $a_i^v = 0$ for $i \neq v$ and the fourth equality is true because the amplitude depends only on z_i and the locations of other data points (through $r_c(x, y)$). ■

The following example demonstrates the above theorem. Consider P, P_1, P_2, P_3, P_4 to be a set of points, as shown in Table 3.1. Figures 3.5(a)-3.5(e) show the curves obtained by applying

MFA to P, P_1, P_2, P_3, P_4 respectively, with $\{\Delta x_i\} = \{\Delta y_i\} = \{2^{-i}\}_{i=-1}^{5}$ at $y = 0.5$. Fig. 3.5(f) is the sum of the four curves in Figures 3.5(b)-3.5(e). Notice that it is identical to Fig. 3.5(a).

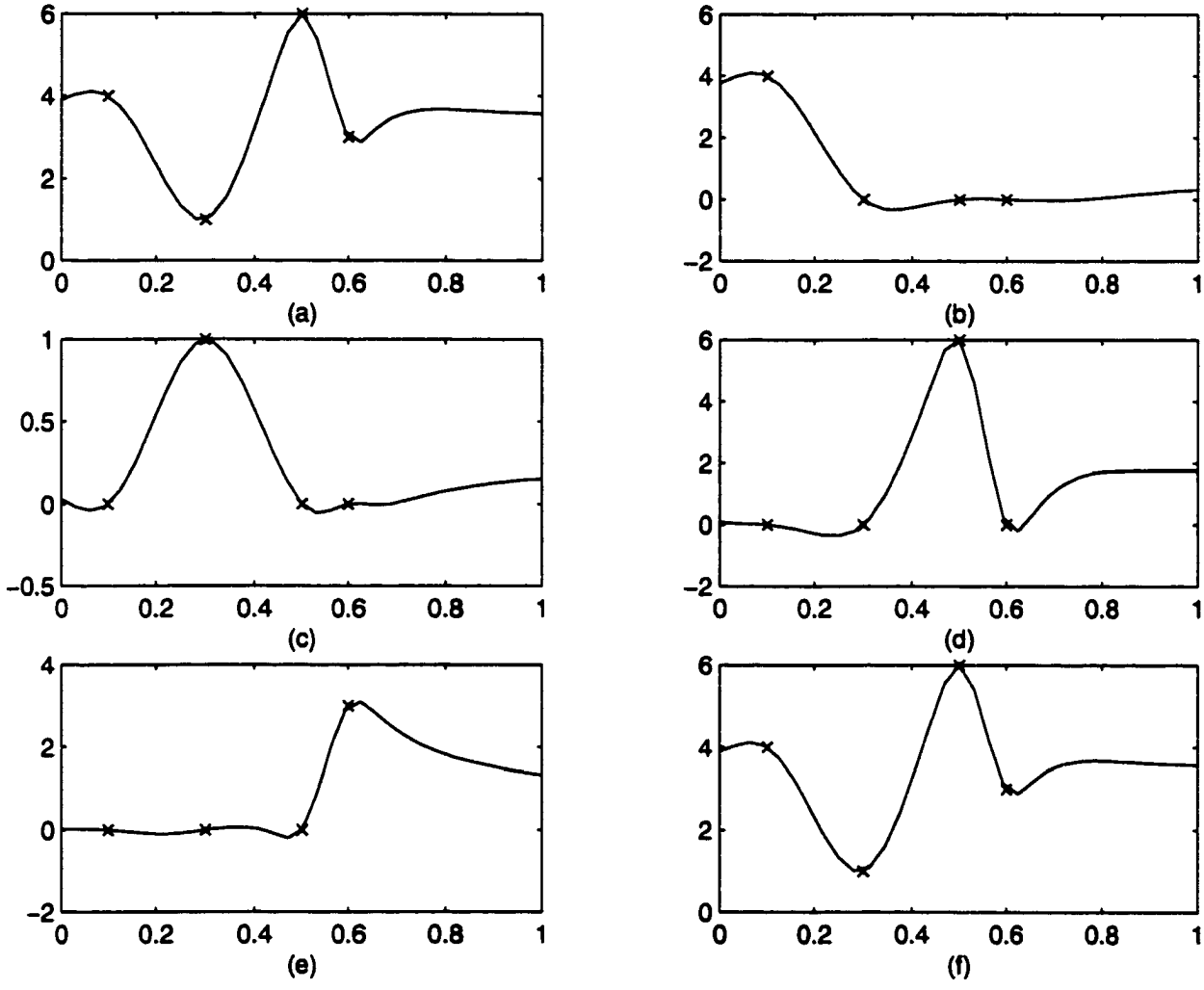


Figure 3.5: (a) MFA(P); (b) MFA(P_1); (c) MFA(P_2); (d) MFA(P_3); (e) MFA(P_4); (f) $\sum_{i=1}^{4}$MFA(P_i).

Note that the support of the basis function is the domain Ω.

MFA kernel We derived the MFA kernel by applying the MFA algorithm to an impulse:

$$P = \{(0.5, 0.5, 1.0), (0.1 \cdot i, 0.1 \cdot j, 0.0)\}$$
$$i, j = 0, 1, 2, 3, 4, 6, 7, 8, 9, 10$$

Fig. 3.6 shows the kernel and its spectrum. The support of the kernel is the domain Ω. The transition of the spectrum is sharper then the spectrum of the MBA kernel. Note that the kernel will change for different impulse locations.

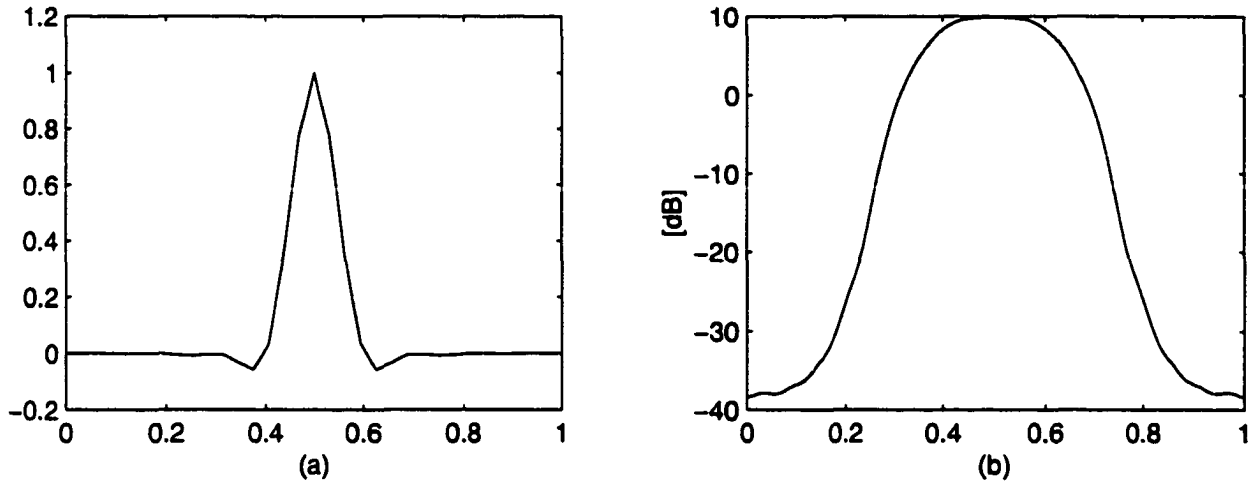


Figure 3.6: (a) MFA kernel; (b) Spectrum.

Shift-variant In contrast to the BA algorithm, the FA algorithm is a shift-invariant functional and produces identical approximants for shifted data points

$$g_{FA}(x,y)|_{\{(x_c,y_c,z_c)\}} = g_{FA}(x+\Delta x, y+\Delta y)|_{\{(x_c+\Delta x, y_c+\Delta y, z_c)\}}$$

This is due to the shift-invariant property of the distance measure $r_c(x,y)$ (Eq. (3.9))

Interpolation condition The MFA algorithm generates an approximation function $f(x,y)$ that passes near the data points P, but not necessarily through them. A sufficient condition to generate an interpolating function with zero residual is

$$MIN(dx_k) \geq 4\Delta x$$
$$MIN(dy_k) \geq 4\Delta y$$

where dx_k (dy_k) is the horizontal (vertical) distance to the closest point from (x_k, y_k, z_k). In other words, the interpolating condition is that no two data points lie in the same $4\Delta x \times 4\Delta y$ neighborhood.

Initial linear approximation Due to the similarity in the behavior of the BA and FA algorithms, we used an identical initial linear approximation procedure described in Section 3.3.

3.5 NUMERICAL EXPERIMENTS

Two sets of data locations taken from [41] were used in our numerical experiments. The first set contains 100 data points distributed uniformly over the unit square, while the second set, with 33 data points, was designed with larger variations in the density of the data points. See Appendix C for more details.

The reconstructed values $\{z_i\}$ were obtained by evaluating nine test functions at the data points. All of the functions are defined on the unit square. See appendix D for more details.

We constructed the approximating surface $f(x, y)$ using the MBA, MFA, and the TPS algorithms. The errors between $f(x, y)$ and test function $F^{(i)}$ were computed on a grid of 33×33 nodes, uniformly placed over the unit square. This grid size was taken from the experiments in [41] and was found satisfactory by comparison with finer grids. Only points inside the convex hull were included in the following results. The tables of the numerical testing show the means of these errors.

The TPS algorithm is Duchon's thin plate spline interpolating function [29]. It is a global interpolating function which was among the most successful interpolation schemes tested in [41].

As expected, the results show very high correlation between the performance of the MBA and MFA schemes. In all cases, the TPS scheme produces superior results, although at a substantially higher cost.

3.6 IMAGE RECONSTRUCTION FROM NONUNIFORM SAMPLES

Consider a set of nonuniform samples taken from an image. Reconstruction refers to the interpolation necessary to recover the image from its samples. We address this problem by

	Mean Deviation			Max Deviation		
	MBA	MFA	TPS	MBA	MFA	TPS
$F^{(1)}$	0.01602	0.01138	0.00520	0.17814	0.12976	0.05347
$F^{(2)}$	0.00494	0.00426	0.00207	0.04654	0.04661	0.03561
$F^{(3)}$	0.00192	0.00231	0.00051	0.02272	0.02304	0.00607
$F^{(4)}$	0.00429	0.00202	0.00017	0.12856	0.01518	0.00294
$F^{(5)}$	0.00380	0.00284	0.00087	0.05313	0.04106	0.01768
$F^{(6)}$	0.00768	0.00421	0.00055	0.24451	0.07019	0.01692
$F^{(7)}$	0.03369	0.03280	0.02203	0.35876	0.46240	0.28652
$F^{(8)}$	0.04581	0.04281	0.02369	0.30000	0.30706	0.25444
$F^{(9)}$	0.01904	0.00973	0.00325	0.59526	0.20971	0.11246

Table 3.2: Mean and Maximum error for 100 data points.

	Mean Deviation			Max Deviation		
	MBA	MFA	TPS	MBA	MFA	TPS
$F^{(1)}$	0.04098	0.03974	0.02928	0.18789	0.18700	0.15345
$F^{(2)}$	0.01264	0.01401	0.00778	0.05902	0.06403	0.05259
$F^{(3)}$	0.01470	0.01617	0.00912	0.11209	0.12170	0.05742
$F^{(4)}$	0.01489	0.01704	0.00415	0.08240	0.09859	0.02587
$F^{(5)}$	0.01976	0.02272	0.01296	0.21961	0.24100	0.14913
$F^{(6)}$	0.00980	0.01092	0.00315	0.03689	0.05227	0.02322
$F^{(7)}$	0.08285	0.08851	0.08527	0.65047	0.67940	0.55796
$F^{(8)}$	0.13897	0.14886	0.09733	0.66337	0.72230	0.57643
$F^{(9)}$	0.03023	0.04044	0.02743	0.23065	0.28282	0.19135

Table 3.3: Mean and Maximum error for 33 data points

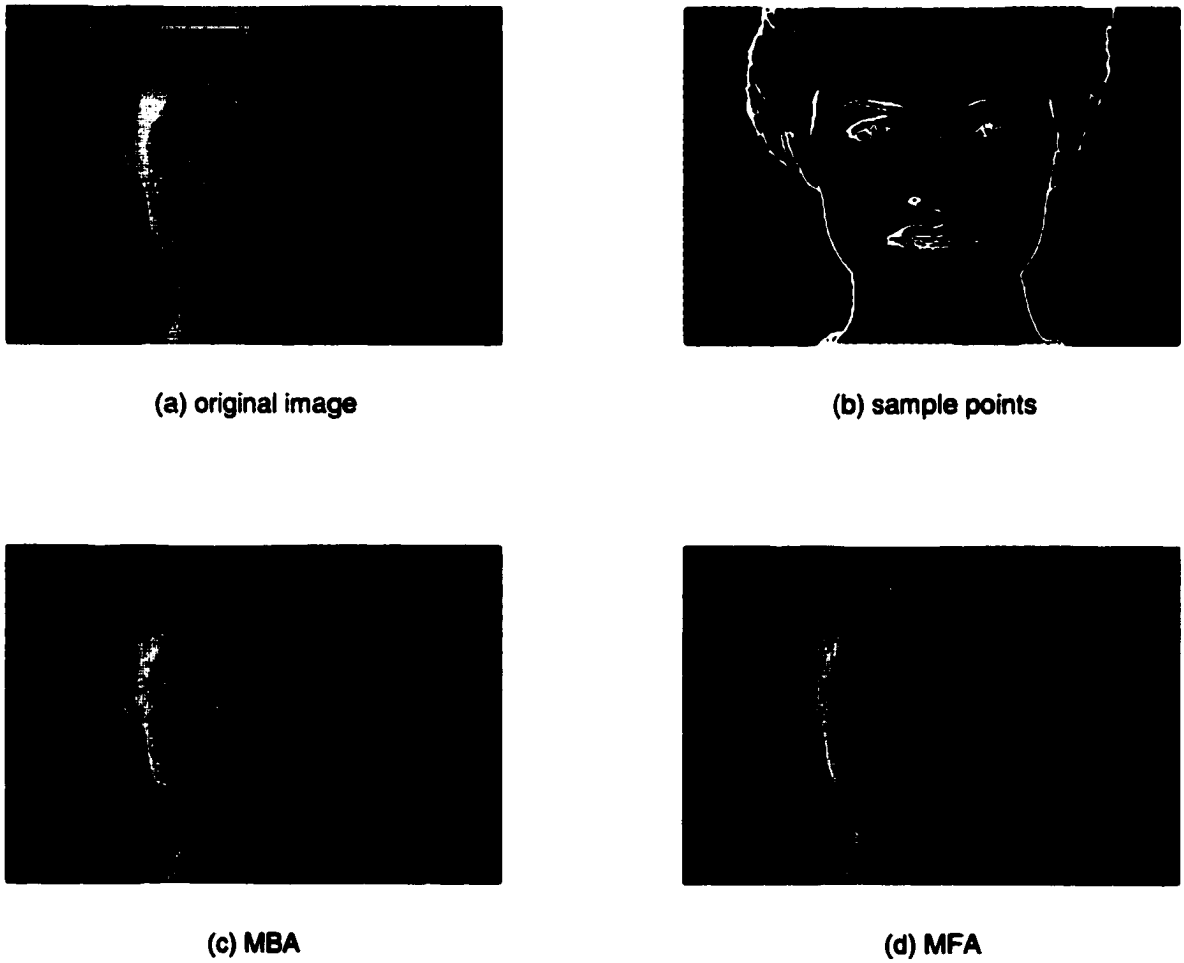
interpolating a grayscale image as a surface, where the value of each pixel represents its height. Image reconstruction from nonuniform samples is then cast into a surface fitting framework that can be solved with the MBA and MFA algorithms.

Consider the image in Fig. 3.7(a). To derive a set of nonuniform samples, we first apply the Sobel operator [48] to the image and threshold the result to identify edges. These edges consist of an important set of pixels that capture the visual details of the image. Due to their rather arbitrary definition and distribution, edges alone are not sufficient to recover the image. Therefore, we also sample the image on a coarse regular grid to properly reconstruct the parts of the image where there are no nearby edge points. The positions of the sampled pixels are shown in Fig. 3.7(b). For the purpose of this example, we used a uniform sampling rate of one sample per six pixels along each direction and an edge threshold of 25. The MBA and MFA algorithms are applied to the samples to reconstruct the images in Fig. 3.7(c) and Fig. 3.7(d)) respectively. The diameters of the MFA kernels are 23, 11, 5 and 3. The mean and maximum error for the MBA reconstruction is 0.00993 and 0.15273, respectively. The mean and maximum error for the MFA reconstruction is 0.01036 and 0.14089, respectively.

Large data reduction is achieved here. The dimensions of the original image in Fig. 3.7 is 360×243. The number of sample points in Fig. 3.7(b) is 7652, of which 5359 lie upon the edge contours. Therefore, sample points constitute only 8.75% of the total number of pixels in the original image.

Consider the image in Fig. 3.8(a). We used the same approach to generate the nonuniform image. The positions of the sampled pixels are shown in Fig. 3.8(b). For the purpose of this example, we used a uniform sampling rate of one sample per six pixels along each direction and an edge threshold of 50. The MBA and MFA algorithms are applied to the samples to reconstruct the images in Fig. 3.8(c) and Fig. 3.8(d), respectively. The diameters of the MFA kernels are 22, 11, 5 and 3. The mean and maximum error for the MBA reconstruction is 0.00914 and 0.18792, respectively. The mean and maximum error for the MFA reconstruction is 0.01034 and 0.17060, respectively.

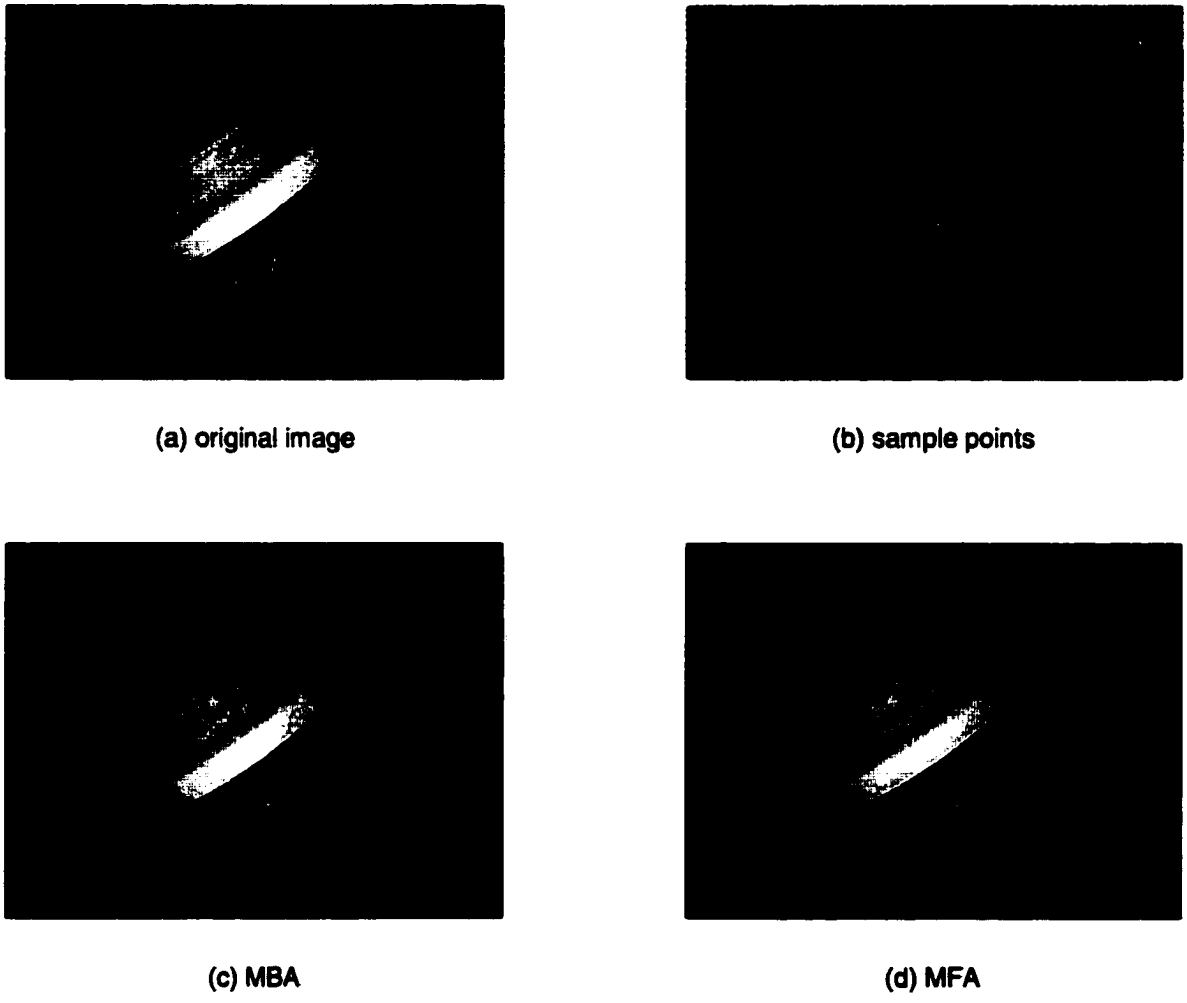
Large data reduction is achieved here. The dimensions of the original image in Fig. 3.8 is 328×438. The number of sample points in Fig. 3.8(b) is 6406, of which 2464 lie upon the edge contours. Therefore, sample points constitute only 4.46% of the total number of pixels in



(a) original image (b) sample points

(c) MBA (d) MFA

Figure 3.7: Image reconstruction example.



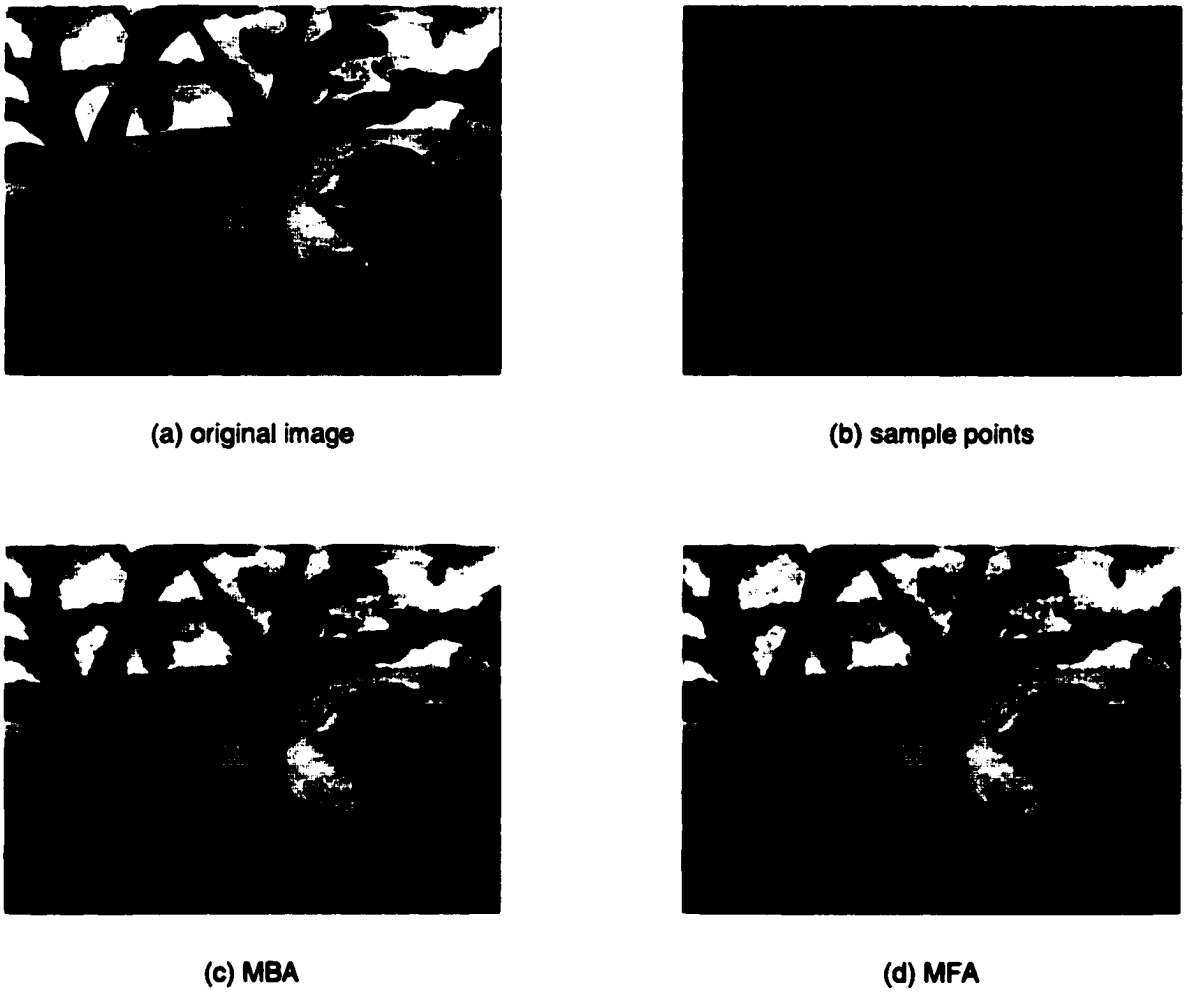
(a) original image (b) sample points

(c) MBA (d) MFA

Figure 3.8: Image reconstruction example.

the original image.

Finally, consider the image in Fig. 3.9(a). To derive a set of nonuniform samples, we randomly sampled the image. The positions of the sampled pixels are shown in Fig. 3.9(b). The MBA and MFA algorithms are applied to the samples to reconstruct the images in Fig. 3.9(c) and Fig. 3.9(d), respectively. The diameters of the MFA kernels are 28, 15, 7 and 3. The mean and maximum error for the MBA reconstruction is 0.04400 and 0.61287, respectively. The mean and maximum error for the MFA reconstruction is 0.04438 and 0.59883, respectively.



(a) original image (b) sample points

(c) MBA (d) MFA

Figure 3.9: Image reconstruction example.

The dimensions of the original image in Fig. 3.9 is 258×350. The number of sample points in Fig. 3.9(b) is 12613. Therefore, sample points constitute only 13.97% of the total number of pixels in the original image.

3.7 COMPLEXITY

In this section, we compare the complexity of the MBA, MFA, and thin plate spline (TPS) algorithms. We refer to the complexity in terms of the number of multiply operations that are required to evaluate S surface points, given N scattered data points. It is shown that the method of choice depends on the size of S. For small S, the MFA algorithm is faster while for large S the MBA algorithm is faster.

3.7.1 TPS

Thin plate splines require a global algorithm that first computes $N + 3$ coefficients by solving a set of linear equations:

$$\begin{aligned} f(x_k, y_k) &= z_k \\ \sum_{k=1}^{N} \alpha_k &= \sum_{k=1}^{N} \alpha_k x_k = \sum_{k=1}^{N} \alpha_k y_k = 0 \end{aligned}$$

where $\{(x_k, y_k, z_k)\}$ is the set of scattered points,

$$r_k = \sqrt{(x - x_k)^2 + (y - y_k)^2}$$

and

$$f(x, y) = \sum \alpha_k r_k^2 \log r_k + ax + by + c \tag{3.12}$$

The evaluation of each surface point requires an additional $4N$ multiply operations. The complexity of the algorithm is $Q_{TPS}(s) = (N + 3)^3 + N^2 + 4NS$. Note that the three terms in Q_{TPS} refer to the cost of inverting a matrix, computing the coefficients, and evaluating S surface points, respectively.

3.7.2 MBA

Consider the use of h levels with the optimal MBA algorithm. Table 3.4 presents the complexity at level i, which corresponds to grid size $m_i \times l_i$:

For each surface point, it requires additional 32 multiply operations. The complexity of the algorithm is $Q_{MBA}(S) = 32S + \sum_{i=0}^{h}[112N + 9m_i l_i]$

BA	$80N + m_i n_i$
residual	$32N$
refinement	$8(m_i - 1)(l_i - 1)$

Table 3.4: Multiply operations for the MBA algorithm.

3.7.3 MFA

Consider N data points and h levels with the MFA algorithm. Also, consider an $S_1 \times S_2$ output surface. For each level k, a kernel with radius r_i is applied n times. The complexity of evaluating the amplitudes is nkN. For S surface points, the MFA algorithm requires q multiply operations, where $q = \sum_{i=0}^{h-1} \min\{S_1 S_2 + 2S_1 S_2 \log(S_1 S_2), \pi r_i^2 S\}$ operations. Note that the first term in the min function applies when we are evaluating the surface points with the Fast Fourier transform. The complexity of the algorithm is $Q_{MFA}(S) = nkN + \sum_{i=0}^{h-1} \min\{S_1 S_2 + 2S_1 S_2 \log(S_1 S_2), \pi r_i^2 S\}$.

3.7.4 Discussion

The above discussion indicates that the complexity of TPS is $O(N^3)$ and it is independent of S. The major complexity of the MBA algorithm is the computation of the control lattices, while only 32 multiply operations are required to evaluate any surface point. The computation of the MFA coefficients requires only nkN multiply operations, and an additional q multiply operations. This indicates that the MBA is more efficient for surface reconstruction, while MFA is more efficient for applications where it must evaluate the surface locally.

Table 3.7.4 lists the complexity of the MBA and MFA algorithms for the reconstruction of each one of the three images shown above.

	MBA	MFA
face	24	18.6
saturn	24.3	38.0
trees	29.6	38.3

Table 3.5: Complexity comparison (in millions of multiply operations).

Example 11 *Suppose that a surface has to be evaluated at* 128×128 *grid points. Fig. 3.10 shows the graphs in logarithmic vertical scale. We applied* $h = 8$ *levels for the MBA algorithm.*

For the MFA algorithm, we iterated four times on each of three levels, using kernel sizes of 17, 7, and 3.

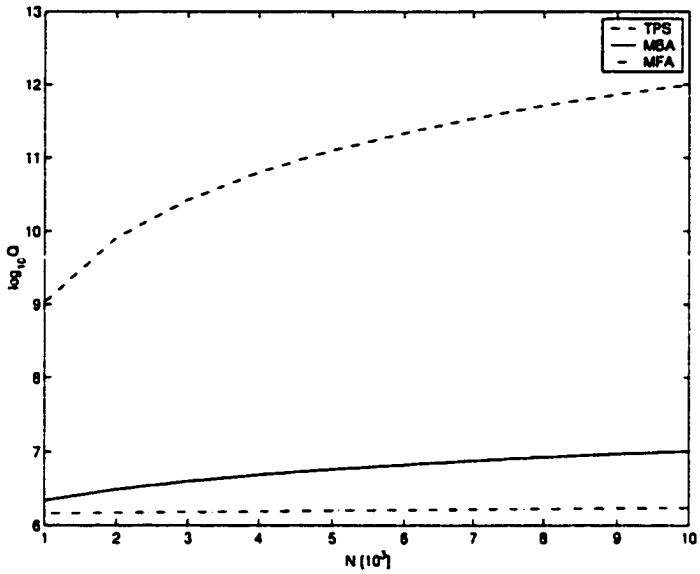


Figure 3.10: Complexity of computing a 128 × 128 surface.

Example 12 *Suppose that S points must be evaluated, given N = 1000. Figure (3.11) shows the graph in logarithmic vertical scale. We applied h = 8 levels for MBA algorithm. For the MFA algorithm, we iterated four times on each of three levels, using kernel sizes of 17, 7, and 3.*

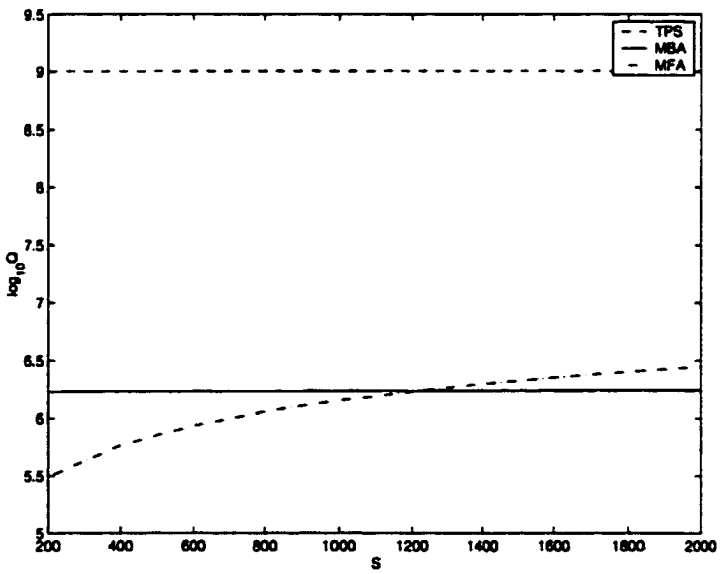


Figure 3.11: Complexity of computing S surface points.

Chapter 4

CONCLUSIONS

4.1 CONTRIBUTIONS

One 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. 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 thesis. Modifications were introduced to simplify the problem and facilitate the use of linear programming. The interpolation methods considered in this thesis 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 [44]. We suggested that the SDDE_LP energy measure be used as an optimization objective since it permits us to minimize the second derivative discontinuity energy using linear programming rather than the slower quadratic programming. The SDDE_LP objective function yields 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. We also 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.

In order to produce smooth monotone bivariate interpolants, we extended the energy minimization framework to work with 2-D surfaces in addition to 1-D curves. We presented a set of constraints on the bicubic function coefficients such that the resulting bicubic polynomial is monotone on a single rectangular element. These constraints were collected into a system of linear inequalities which can be solved using linear programming. We extended the approach to handle scattered data. The algorithm is based on fitting the domain with a rectangular grid and searching for the coefficients of the monotone interpolating bicubic polynomials. We also introduced an iterative grid refinement process that may be coupled to any scattered data interpolation procedure to produce a monotone C^2 surface. Thin plate splines and multilevel B-spline interpolation were considered for the underlying interpolation methods. The iterative grid refinement algorithm, which we refer to as Mono-SDI, is our method of choice for producing monotone C^2 interpolants through either gridded or scattered data.

In recent work, Wolberg and his colleagues introduced a multiresolution cubic B-spline approximation (MBA) algorithm. The hierarchical MBA algorithm, which is both fast and global, was demonstrated to produce high fidelity reconstruction in diverse applications such as image reconstruction, image warping, and object reconstruction. This thesis presents an analysis of key mathematical properties of the MBA algorithm. The resulting analysis suggests an alternate global hierarchical approach which we call *multiresolution filtering approximation (MFA)*. The MFA scheme is intuitive and simple to implement. Although MBA is more efficient for surface reconstruction, MFA is more efficient for applications where it must evaluate the surface locally. Numerical experiments show that the MBA and MFA schemes yield similar quantitative performance for synthetic test functions.

The MFA and MBA algorithms are closely related to second generation wavelets [74]. The basis functions for the MBA and MFA, developed in Sections 3.3 and 3.4, are equivalent to their polynomial subdivision scaling functions. Note that the main difference between the first and second generation cases is that the scaling functions are not necessarily translates and dilates of each other due to the nonuniform distribution of the data. Their method uses a lifting scheme to perform the interpolation while the MFA method assembles the scaling functions

from multiresolution radial kernels. This enables us to use the Fast Fourier Transform, despite the irregular distribution of the data.

The applicability of the MFA algorithm for image processing applications has been examined. We applied the MFA algorithm to the problem of image reconstruction from nonuniform samples.

4.2 FUTURE WORK

There are several points which may be investigated in order to extend and improve the results presented in this thesis.

- The energy minimization framework developed for monotone cubic spline interpolation enables us to solve for new knots that are inserted between the data points. Potentially significant reduction in the number of constraints can be achieved by determining which intervals requires us to introduce additional knots, the minimum number of knots, and their locations.

- The Mono-SDI algorithm introduces new knots through an iterative refinement-adjust process. The refinement process is global and seeks to reduce the largest grid step, without considering the behavior of the surface. We may reduce the number of iterations by attempting to determine where new knots should be inserted.

- We solved the monotonicity problem for the 1-D and 2-D cases. It is of great interest to augment these results for data of changing monotonicity in 2-D. Also, we may extend these results to 3-D data, i.e., construct a shape preserving surface for scattered 3-D data.

- The MBA algorithm demonstrates high quality reconstruction from scattered data. An interesting and valuable problem is to investigate how to select a minimal number of sparse and irregular samples necessary to achieve high quality reconstruction within a specified error tolerance.

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 $0 \leq i < n$. Let s_i be the slopes of the polygon sides:

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

The equation of the i-th 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:

$$\begin{aligned} \textbf{IF } s_i\alpha_{i-1} - \beta_{i-1} > 0 \ \textbf{ THEN } \ & \beta - s_i\alpha + s_i\alpha_{i-1} - \beta_{i-1} \geq 0 \\ \textbf{ELSE } \ & \beta - s_i\alpha + s_i\alpha_{i-1} - \beta_{i-1} \leq 0. \end{aligned}$$

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 $m_k > 0$:

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

$$\textbf{ELSE} \quad y'_{k+1} - s_i y'_k + m_k(s_i\alpha_{i-1} - \beta_{i-1}) \leq 0$$

When $m_k < 0$, we have:

$$\textbf{IF } s_i\alpha_{i-1} - \beta_{i-1} > 0 \textbf{ THEN } \quad y'_{k+1} - s_i y'_k + m_k(s_i\alpha_{i-1} - \beta_{i-1}) \leq 0$$
$$\textbf{ELSE} \quad y'_{k+1} - s_i y'_k + m_k(s_i\alpha_{i-1} - \beta_{i-1}) \geq 0$$

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 A.1 and A.2 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. 1.30 and 1.31.

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

Table A.1: Approximating monotonicity region M with a 6-sided polygon.

								Inequality direction	
α_{i-1}	β_{i-1}	α_i	β_i	s_i	y'_k	y'_{k+1}	m_k	$m_k > 0$	$m_k < 0$
0	3	0	0	∞	0	1	0	≥ 0	≤ 0
0	0	3	0	0	1	0	0	≥ 0	≤ 0
3	0	3.5	0.1044	0.2088	-0.2088	1	0.6264	≥ 0	≤ 0
3.5	0.1044	3.75	0.2865	0.7284	-0.7284	1	2.4450	≥ 0	≤ 0
3.75	0.2865	4	1	2.8540	-2.8540	1	10.4160	≥ 0	≤ 0
4	1	3.75	1.9635	-3.8540	3.8540	1	-16.4160	≤ 0	≥ 0
3.75	1.9635	0	2.3956	-1.7284	1.7284	1	-8.4450	≤ 0	≥ 0
3.5	2.3956	0	3	-1.2088	1.2088	1	-6.6264	≤ 0	≥ 0
3	3	0	3.4271	-0.8542	0.8542	1	-5.5626	≤ 0	≥ 0
2.5	3.4271	2	3.7321	-0.6100	0.6100	1	-4.9521	≤ 0	≥ 0
2	3.7321	1.5	3.9271	-0.3900	0.3900	1	-4.5121	≤ 0	≥ 0
1.5	3.9271	1	4	-0.1458	0.1458	1	-4.1458	≤ 0	≥ 0
1	4	0.5	3.8956	0.2088	-0.2088	1	-3.7912	≤ 0	≥ 0
0.5	3.9856	0.25	3.7135	0.7284	-0.7284	1	-3.5314	≤ 0	≥ 0
0.25	3.7135	0	3	2.8540	-2.8540	1	-3.0000	≤ 0	≥ 0

Table A.2: Approximating monotonicity region M with a 15-sided polygon.

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.

B.1 File monotone.m

```
% ------------------------------------------------------------------------
%
% montone.m -   Compute interpolating monotonic cubic spline.
%
% 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 (0),
%               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
        [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',zeros(1,2*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.
%
% 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
%
% 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);
```

Appendix C

SCATTERED DATA SETS

Two sets of data locations taken from [41] were used in our numerical experiments. The first set consists of 100 points generated by a pseudorandom number generator, one point in each square of side $\frac{1}{9}$ centered at $\left(\frac{i}{9}, \frac{j}{9}\right)$ for $i, j = 1, ..., 10$. This yields a set of scattered points forced to have uniform density. The points are listed in Table C.2, and shown in Fig. C.1.

The second set consists of 33 points distributed such that some areas are sparsely populated by points while other areas are not. This set of points is listed in Table C.1 and shown in Fig. C.2.

x	y	x	y	x	y	x	y	x	y
0.00	0.00	1.00	0.00	0.80	0.40	0.85	0.25	0.05	0.45
0.00	1.00	1.00	1.00	0.70	0.20	0.80	0.65		
0.00	0.50	0.50	0.00	0.95	0.90	0.75	0.85		
0.50	1.00	1.00	0.50	0.60	0.65	0.70	0.90		
0.10	0.15	0.20	0.10	0.65	0.70	0.70	0.65		
0.15	0.30	0.25	0.20	0.35	0.85	0.75	0.10		
0.30	0.35	0.60	0.25	0.60	0.85	0.75	0.35		
0.10	0.75	0.90	0.35	0.9	0.80	0.55	0.95		

Table C.1: 33 sparse points.

x	y	x	y	x	y	x	y
0.022703	−0.031021	0.479258	0.632425	0.053989	0.158674	0.573008	0.127243
0.021701	0.257692	0.397776	0.848971	0.017513	0.341401	0.501389	0.347773
0.001903	0.494360	0.584869	−0.027195	−0.050968	0.578258	0.610695	0.608471
0.039541	0.699342	0.606389	0.270927	−0.048706	0.747019	0.538062	0.723524
0.031583	0.910765	0.574131	0.425942	−0.041878	0.996289	0.502619	1.030876
0.132419	0.50133	0.599010	0.673878	0.109027	0.091855	0.642784	0.070783
0.125444	0.259297	0.609697	0.924241	0.093454	0.338169	0.670398	0.325984
0.076758	0.417112	0.661693	0.025596	0.145187	0.561556	0.633359	0.509632
0.062649	0.655223	0.639647	0.200834	0.145273	0.752407	0.689564	0.775957
0.095867	0.914652	0.700118	0.439070	0.069556	0.963242	0.683767	1.006451
0.264560	0.029294	0.690895	0.669788	0.239164	0.060230	0.763533	0.102140
0.208899	0.266878	0.671889	0.936616	0.276733	0.369604	0.825898	0.323577
0.171473	0.480174	0.773694	0.028537	0.226678	0.594059	0.808661	0.609159
0.190921	0.687880	0.741042	0.193658	0.186765	0.818558	0.729064	0.302281
0.230463	0.904651	0.730603	0.471423	0.242622	0.980541	0.817095	1.051236
0.366317	0.039695	0.821453	0.668505	0.885766	0.068448	0.868405	0.090205
0.383239	0.238955	0.807664	0.847679	0.317909	0.312413	0.841846	0.331849
0.346632	0.490299	0.842457	0.038050	0.377659	0.519980	0.859958	0.591014
0.387316	0.644523	0.836692	0.208309	0.381292	0.820379	0.859633	0.814484
0.379536	0.893808	0.847812	0.433563	0.280351	0.971172	0.851280	0.969608
0.414977	−0.028462	0.917570	0.630738	0.427768	0.156096	0.967063	0.133411
0.420001	0.226247	0.927987	0.904231	0.466363	0.317509	0.967631	0.379528
0.485566	0.389142	1.044982	−0.012090	0.409203	0.508495	0.965704	0.504442
1.001985	0.694152	0.985788	0.269584	0.481228	0.751101	1.035930	0.745992
1.041468	0.868208	1.012929	0.439605	0.402732	0.997873	0.947151	0.980141

Table C.2: 100 scattered points.

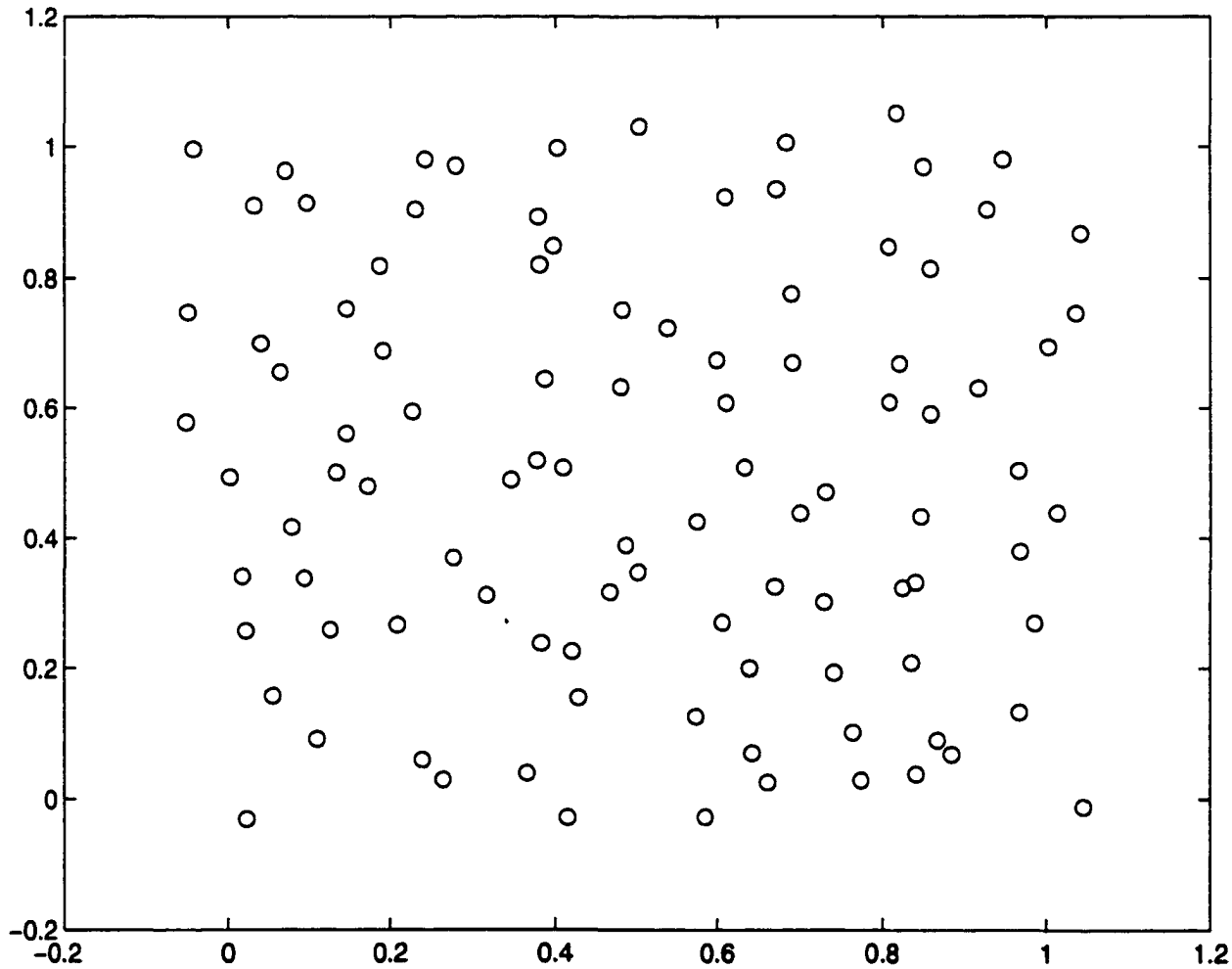


Figure C.1: 100 scattered points.

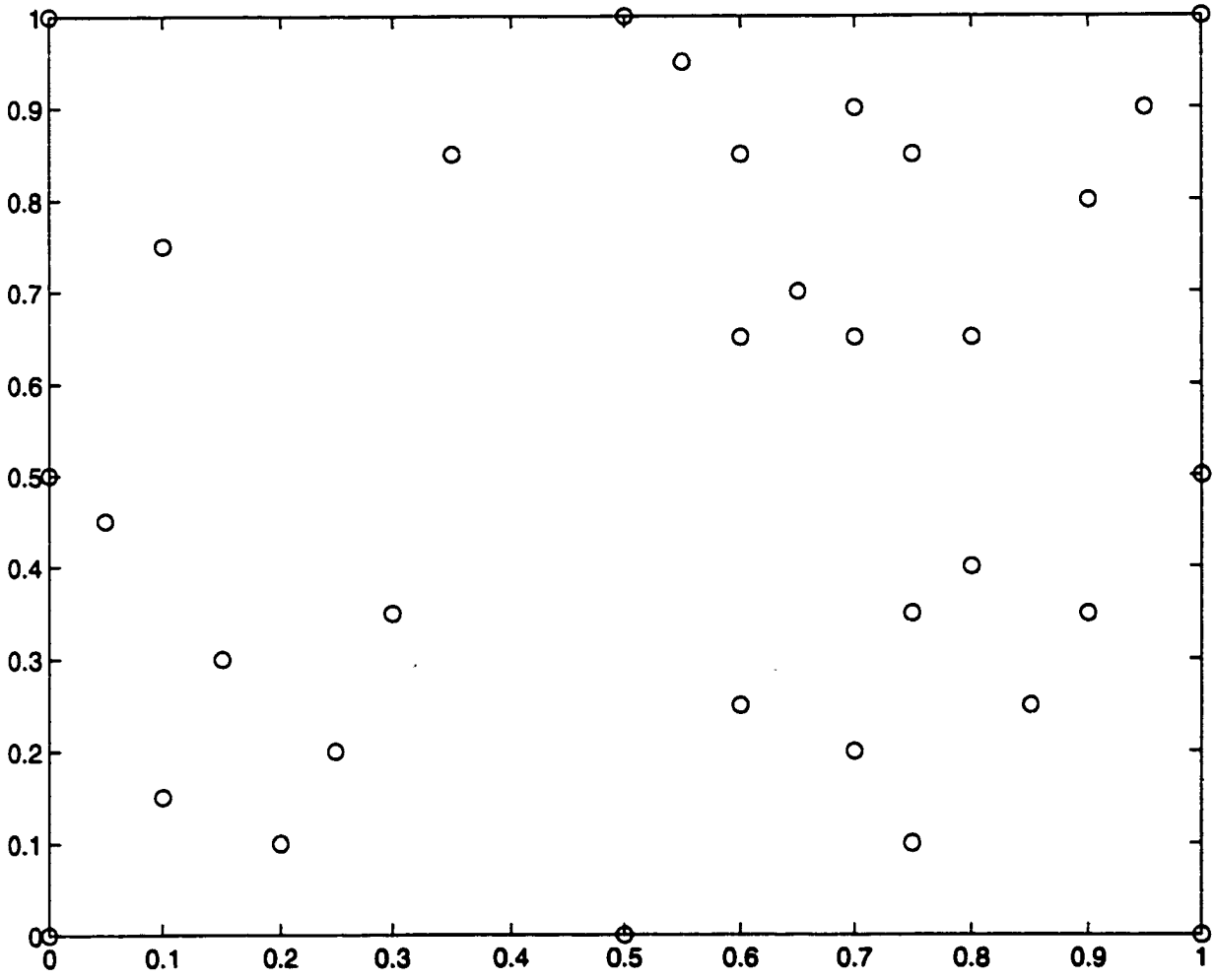


Figure C.2: 33 sparse points.

Appendix D

TEST FUNCTIONS

The scattered data positions in Appendix C are made to assume values $\{z_k\}$ by evaluating various test functions over the unit square. The first six test functions were taken from [41].

$$
\begin{aligned}
F^{(1)} &= \frac{3}{4}\exp\left[-\frac{(9x-2)^2+(9y-2)^2}{4}\right] + \frac{3}{4}\exp\left[-\frac{(9x+1)^2}{49} - \frac{9y+1}{10}\right] \\
&\quad + \frac{1}{2}\exp\left[-\frac{(9x-7)^2+(9y-3)^2}{4}\right] - \frac{1}{5}\exp\left[-(9x-4)^2-(9y-7)^2\right] \\
F^{(2)} &= \frac{\tanh(9y-9x)+1}{9} \\
F^{(3)} &= \frac{1.25+\cos(5.4y)}{6[1+(3x-1)^2]} \\
F^{(4)} &= \frac{1}{3}\exp\left\{-\frac{81}{16}[(x-0.5)^2+(y-0.5)^2]\right\} \\
F^{(5)} &= \frac{1}{3}\exp\left\{-\frac{81}{4}[(x-0.5)^2+(y-0.5)^2]\right\} \\
F^{(6)} &= \frac{1}{9}\left\{64-81[(x-0.5)^2+(y-0.5)^2]\right\}^{0.5} - 0.5
\end{aligned}
$$

The function $F^{(1)}$ is composed of two Gaussian dips (Fig. D.1). The function $F^{(2)}$ simulates a sharp rise, running diagonally across the unit square (Fig. D.2). Function $F^{(3)}$ is saddle-shaped, $F^{(4)}$ is a gentle Gaussian hill, $F^{(5)}$ is a steep Gaussian hill, and $F^{(6)}$ represents a part of a sphere

above the unit square. The seventh test function is taken from [63]

$$F^{(7)} = \left\{ \begin{array}{cc} 1 & y-\xi \geq 0.5 \\ 2(y-\xi) & 0 \leq y-\xi \leq 0.5 \\ 0.5\{\cos(4\pi r(\xi,y)] + 1\} & r(\xi,y) \leq 0.25 \\ 0 & otherwise \end{array} \right\}$$

where

$$\begin{aligned} r(\xi,y) &= [(\xi-1.5)^2 + (y-0.5)^2]^{0.5} \\ \xi &= 2.1x - 0.1 \end{aligned}$$

This function represents a "mountain" on a plane, and a ramp leading to another plane (Fig. D.7). It is a function with discontinuous first derivatives. The last two test functions were taken from [63]

$$\begin{aligned} F^{(8)} &= \tanh(-3g(x,y)) + 1 \\ F^{(9)} &= \left(1-\frac{x}{2}\right)^6 \left(1-\frac{y}{2}\right)^6 + 1000(1-x)^3x^3(1-y)^3y^3 + x^6\left(1-\frac{y}{2}\right)^6 \end{aligned}$$

where

$$g(x,y) = 0.595576(y+3.79762)^2 - x - 10$$

Function $F^{(8)}$ resembles $F^{(2)}$. Its contour lines are parabolas $g(x,y) = const$ while those of $F^{(2)}$ are the straight lines $y - x = const$. Function $F^{(9)}$ is a polynomial surface of degree 12.

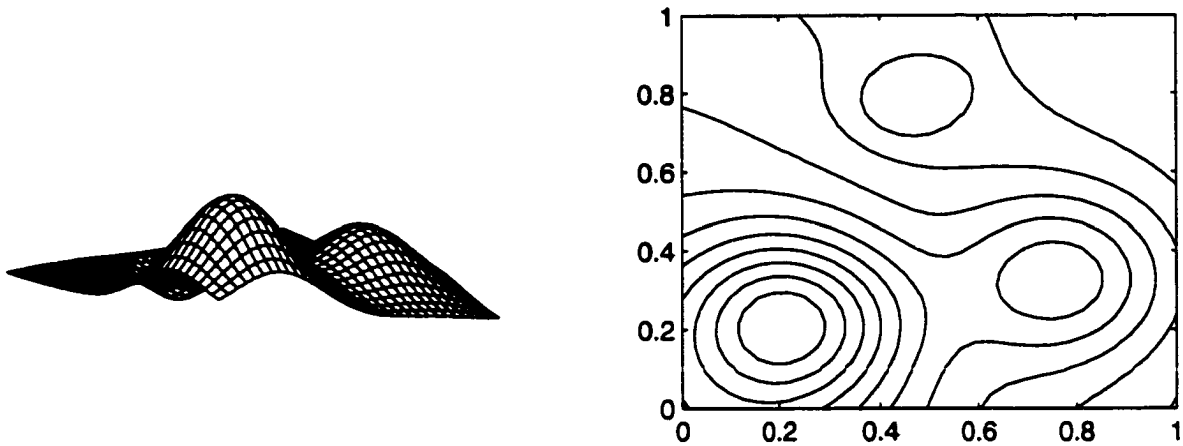


Figure D.1: Perspective view and level curves of $F^{(1)}$.

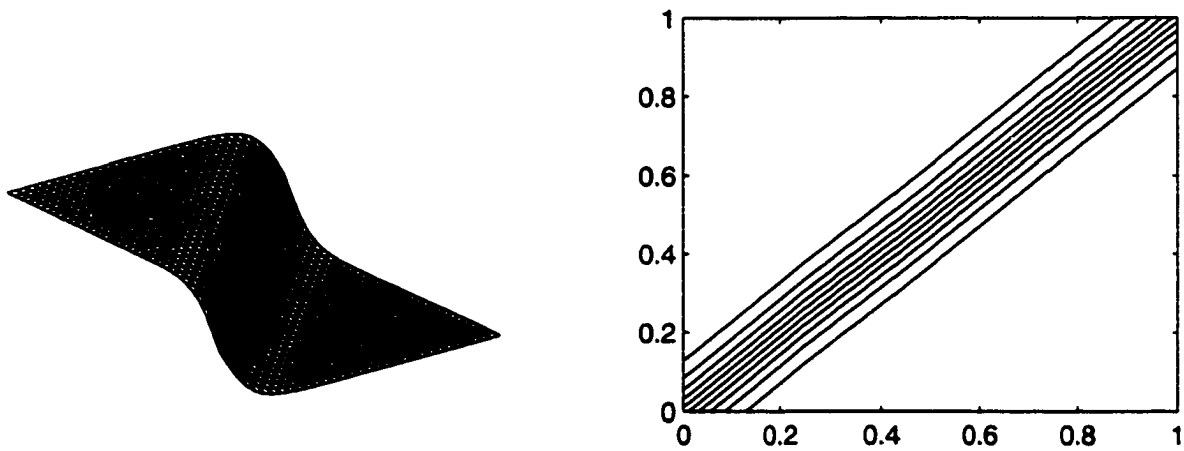


Figure D.2: Perspective view and level curves of $F^{(2)}$.

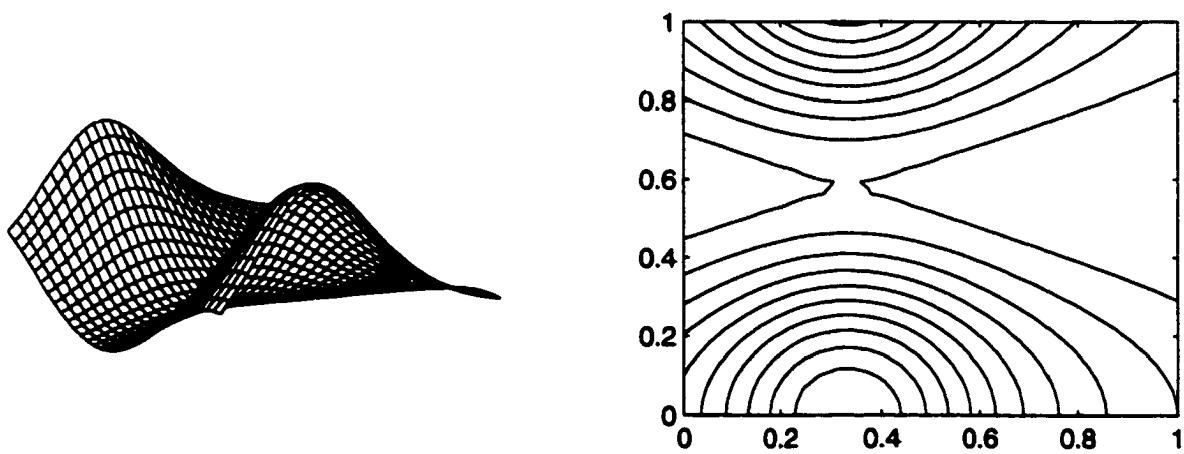


Figure D.3: Perspective view and level curves of $F^{(3)}$.

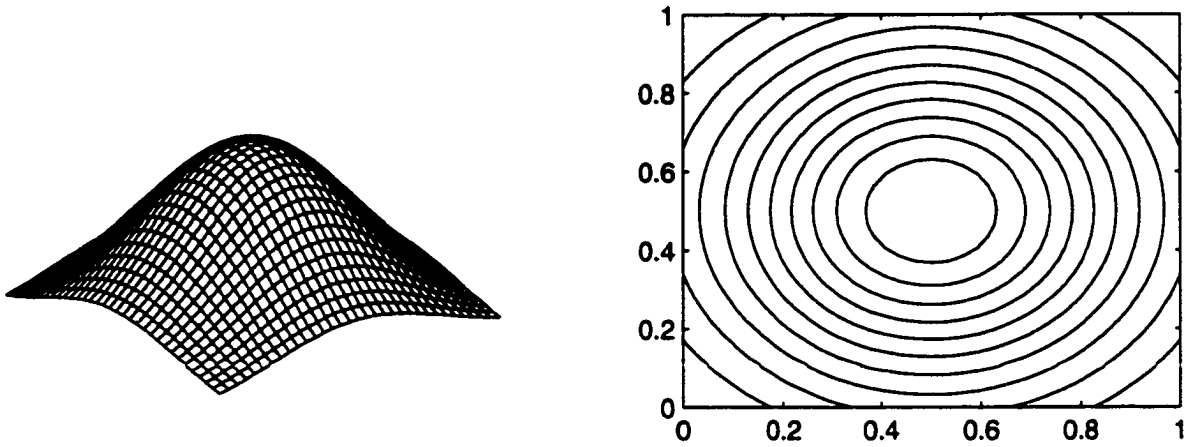


Figure D.4: Perspective view and level curves of $F^{(4)}$.

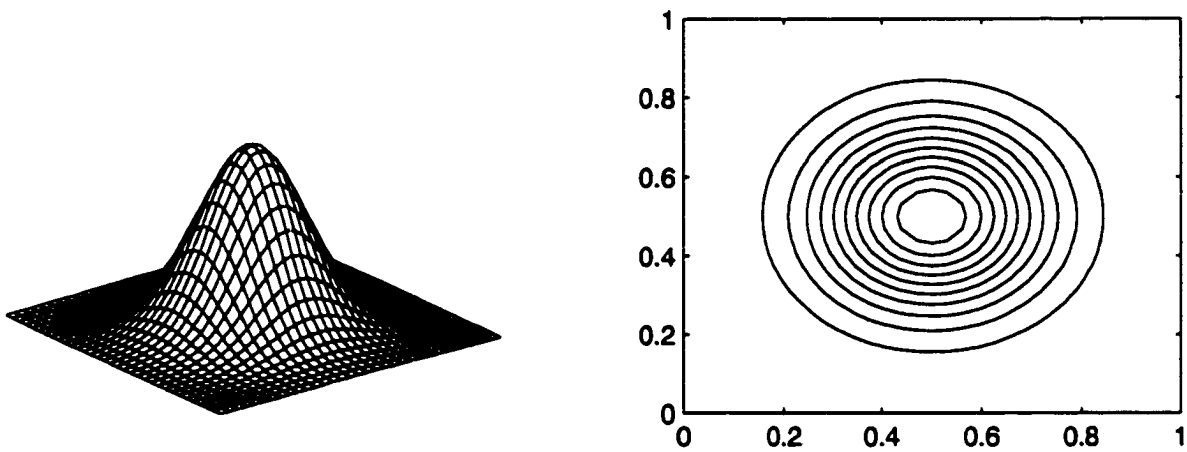


Figure D.5: Perspective view and level curves of $F^{(5)}$.

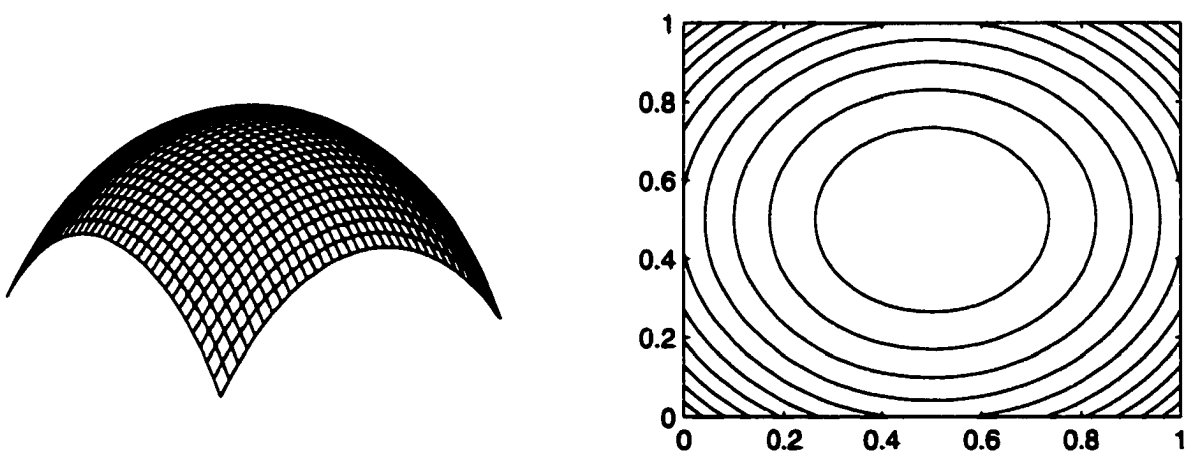


Figure D.6: Perspective view and level curves of $F^{(6)}$.

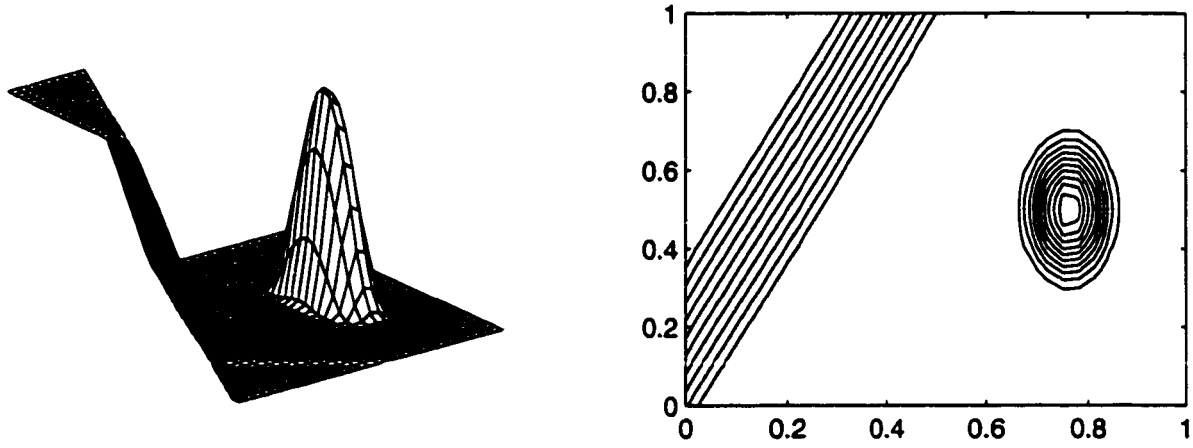


Figure D.7: Perspective view and level curves of $F^{(7)}$.

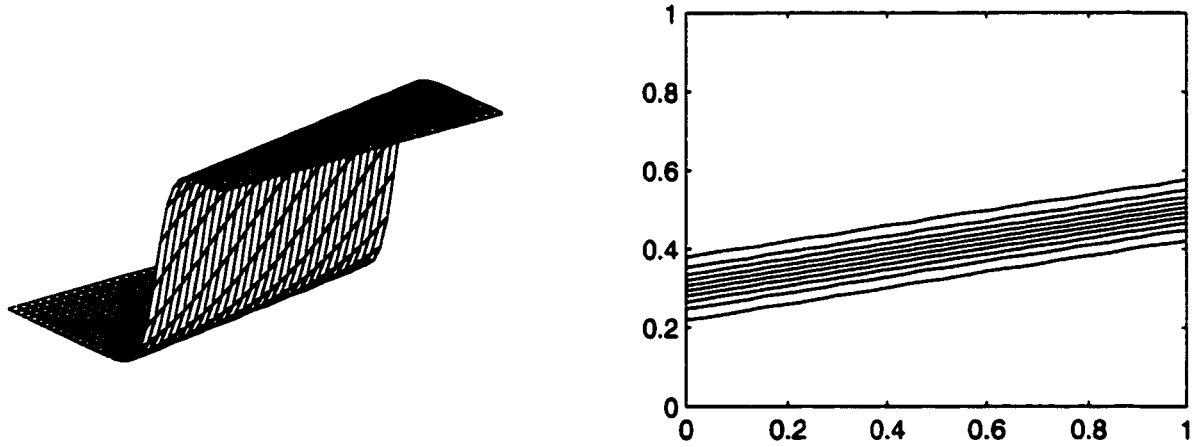


Figure D.8: Perspective view and level curves of $F^{(8)}$.

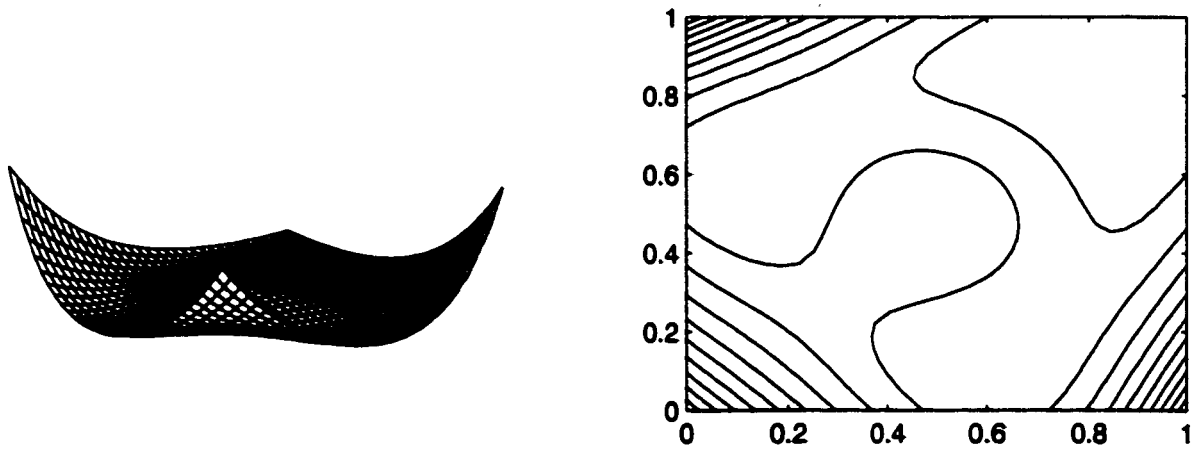


Figure D.9: Perspective view and level curves of $F^{(9)}$.

Bibliography

[1] Ahlberg, J.H., E.N. Nilson, and J.L. Walsh. External Orthogonality and Convergence Properties of Multidimensional Splines. *J. Math. Anal. Appl. 11* (1965), 27–48.

[2] Ahlberg, J.H., E.N. Nilson, and J.L. Walsh. *The Theory of Splines and their Applications.* Academic Press, New York, 1967.

[3] Akima, H. A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedure. *J. Assoc. Comput. Mach. 17* (1970), 589–602.

[4] Arge, Erlend, Morten Dæhlen, and Aslak Tveito. Approximation of Scattered Data Using Smooth Grid Functions. *J. of Computational and Applied Mathematics 59* (1995), 191–205.

[5] Asaturyan, S., and K. Unsworth. A C^1 Monotonicity Preserving Surface Interpolation Scheme. In *Mathematics Of Surfaces III*, D. C. Handscome, Ed. Oxford University Press, London, 1989, pp. 243–266.

[6] Barnhill, R.E. Representation and Approximation of Surfaces. In *Mathematical Software III*, J. R. Rice, Ed. Academic Press, New York, 1977, pp. 68–119.

[7] Beatson, R.K., and H. Wolkowicz. Post-Processing Piecewise Cubics For Monotonicity. *SIAM J. Numerical Analysis 26*, 2 (1989), 480–502.

[8] Beatson, R.K., and Z. Ziegler. Monotonicity Preserving Surface Interpolation. *SIAM J. Numerical Analysis 22*, 2 (1985), 401–411.

[9] Birkhoff, G., and Carl de Boor. Piecewise Polynomial Interpolation and Approximation. In *Approximation Of Functions*, H. Garabedian, Ed. Elsevier, Amsterdam, 1965, pp. 164–190.

[10] Brandt, Achi. Multi-Level Adaptive Solutions to Boundary-Value Problems. *Mathematics of Computation 31*, 138 (1977), 333–390.

[11] Briggs, William L. *A Multigrid Tutorial*. SIAM, Lancaster Press, Lancaster, PA, 1987.

[12] Burt, Peter J. Moment Images, Polynomial Fit Filters, and the Problem of Surface Interpolation. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (1988), 144–152.

[13] Carlson, R.E., and Fred N. Fritsch. Monotone Piecewise Bicubic Interpolation. *SIAM J. Numerical Analysis 22*, 2 (1985), 386–400.

[14] Carlson, R.E., and Fred N. Fritsch. An Algorithm for Monotone Piecewise Bicubic Interpolation. *SIAM J. Numerical Analysis 26*, 1 (1989), 230–238.

[15] Chebyshev, P.L. Sur les fonctions qui diffèrent le moins possible de zèro. *J. de Math. Pures et Appl. 19* (1876), 319–346.

[16] Chui, C., H. Chui, and T. X. He. Shape Preserving Interpolation by Bivariate C^1 Quadratic Splines. In *Workshop On Computational Geometry*, F. F. A. Conte, V. Demichelis and I. Galligani, Eds. World Scientific, Singapore, 1993, pp. 1–75.

[17] Clough, R.W., and J.L. Tocher. Finite Element Stiffness Matrices for Analysis of Plates in Bending. *Proc. Conf. on Matrix Methods in Structural Mechanics* (1965), 515–545.

[18] Costantini, Paolo. On Monotone and Convex Spline Interpolation. *Math. Comp. 46*, 173 (1986), 203–214.

[19] Costantini, Paolo. Co-Monotone Interpolating Splines of Arbitrary Degree: A Local Approach. *SIAM J. Sci. Stat. Comp 8*, 6 (1987), 1026–1034.

[20] Costantini, Paolo. An Algorithm for Computing Shape-Preserving Interpolating Splines of Arbitrary Degree. *J. Comp. Appl. Math 22* (1988), 89–136.

[21] Costantini, Paolo, and Ferruccio Fontanella. Shape Preserving Bivariate Interpolation. *SIAM J. Numerical Analysis 27* (1990), 488–506.

[22] Costantini, Paolo, and Rossana Morandi. An Algorithm for Computing Shape-Preserving Cubic Spline Interpolation to Data. *Calcolo 21* (1984), 295–305.

[23] Costantini, Paolo, and Rossana Morandi. Monotone and Convex Cubic Spline Interpolation. *Calcolo 21* (1984), 281–294.

[24] Crain, I., and K. Bhattacharyya. Treatment of Nonequispaced Two-dimensional Data with a Digital Computer. *Geoexploration 5* (1967), 173–194.

[25] Cressman, G.P. An Operational Objective Analysis System. *Mon. Wea. Rev. 87* (1959), 367–374.

[26] de Boor, Carl. Bicubic Spline Interpolation. *J. Math. and Phys. 41* (1962), 212–218.

[27] de Boor, Carl. *A Practical Guide to Splines*. Springer-Verlag, New York, 1978.

[28] Delbourgo, R., and J. Gregory. Shape Preserving Piecewise Rational Interpolation. *SIAM J. Sci. and Stat. Comp. 6* (1985), 967–976.

[29] Duchon, J. Spline Minimizing Rotation Invariant Semi-Norm in Sobolev Spaces. *Lecture Notes in Mathematics 571* (1977), 85–100.

[30] Dyn, Nira. Interpolation and Approximation by Radial and Related Function. In *Approximation Theory VI*, C. Chui, L. Schumaker, and J. Ward, Eds. Academic Press, San Diego, 1989, pp. 211–234.

[31] Dyn, Nira, David Levin, and Samuel Ruppa. Numerical Procedures for Global Surface Fitting of Scattered Data by Radial Functions. *SIAM J. Sci. Statis. Comput. 7* (1986), 639–659.

[32] Eisenstat, S.C., K.R. Jackson, and J.W. Lewis. The Order of Monotone Piecewise Cubic Interpolation. *SIAM J. Numerical Analysis 22*, 6 (1985), 1220–1237.

[33] Fasshauer, Gregory E., and Larry L. Schumaker. Minimal Energy Surfaces using Parametric Splines. *Computer Aided Geometric Design 13* (1996), 45–79.

[34] Foley, James D., Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*, 2nd ed. Addison-Wesley, Reading, MA, 1990.

[35] Foley, Thomas A., and Gregory M. Nielson. Multivariate Interpolation to Scattered Data Using Delta Iteration. In *Approximation Theory III*, E. Cheney, Ed. Academic Press, New York, 1980, pp. 419–424.

[36] Fontanella, Ferruccio. Shape Preserving Interpolation. In *Topics In Multivariate Approximation*, L. S. C. Chui and F. Utreras, Eds. Academic Press, New York, 1987, pp. 63–78.

[37] Fontanella, Ferruccio. Shape Preserving Interpolation. In *Computation of Curves and Surfaces*, M. G. W. Dehman and C. Micchelli, Eds. Kluwer Academic Publishers, Norwell, MA, 1990, pp. 187–214.

[38] Forsey, David R., and Richard H. Bartels. Hierarchical B-spline Refinement. *Computer Graphics (Proc. SIGGRAPH '88) 22*, 4 (1988), 205–212.

[39] Forsey, David R., and Richard H. Bartels. Tensor Products and Hierarchical Fitting. *Curves and Surfaces in Computer Vision and Graphics II (Proc. SPIE 1610)* (1991), 88–96.

[40] Forsey, David R., and Richard H. Bartels. Surface Fitting with Hierarchical Splines. *ACM Trans. on Graphics 14*, 2 (1995), 134–161.

[41] Franke, Richard. A Critical Comparison of Some Methods for Interpolation of Scattered Data. Tech. Rep. NPS-53-79-003, Naval Postgraduate School, 1979.

[42] Franke, Richard, and Gregory M. Nielson. Scattered Data Interpolation and Applications: A Tutorial and Survey. In *Geometric Modelling: Methods and Their Application* (Berlin, 1991), H. Hagen and D. Roller, Eds., Springer-Verlag, pp. 131–160.

[43] Fritsch, Fred N. Energy Comparison of Wilson Fowler Splines with Other Interpolating Splines. In *Geometric Modeling: Algorithms and New Trends*, G. E. Farin, Ed. SIAM, 1987.

[44] Fritsch, Fred N., and J. Butland. A Method for Constructing Local Monotone Piecewise Cubic Interpolants. *SIAM J. Sci. Stst. Comput. 5*, 2 (1984).

[45] Fritsch, Fred N., and R.E. Carlson. Monotone Piecewise Cubic Interpolation. *SIAM J. Numerical Analysis 17*, 2 (1980), 238–246.

[46] Fritsch, Fred N., and R.E. Carlson. Monotonicity Preserving Bicubic Interpolation: a Progress Report. *Computer Aided Geometric Design 2* (1985), 117–121.

[47] Glassner, Andrew. *Principles of Digital Image Synthesis*. Morgan Kaufmann, San Francisco, CA, 1995.

[48] Gonzalez, Rafael C., and Richard E. Woods. *Digital Image Processing*. Addison-Wesley, Reading, MA, 1992.

[49] Gortler, Steven J., and Michael F. Cohen. Hierarchical and Variational Geometric Modeling with Wavelets. In *1995 Symposium on Interactive 3D Graphics* (1995), pp. 35–42.

[50] Goshtasby, Ardeshir. Piecewise Cubic Mapping Functions for Image Registration. *Pattern Recognition 20*, 5 (1987), 525–533.

[51] Gregory, J., and R. Delbourgo. Piecewise Rational Quadratic Interpolation to Monotonic Data. *IMA J. Numerical Analysis 2* (1982), 123–130.

[52] Han, Lu, and Larry L. Schumaker. Fitting Monotone Surfaces to Scattered Data Using C^1 Piecewise Cubics. *SIAM J. Numerical Analysis 34*, 2 (1997), 569–585.

[53] Hardy, R.L. Multiquadratic Equations of Topography and Other Irregular Surfaces. *J. Geophysical Research 76*, 8 (1971), 1905–1915.

[54] Hoschek, Josef, and Dieter Lasser. *Computer Aided Geometric Design*. AK Peters, Ltd, Wellesley, MA, 1993.

[55] Hsu, William M., John F. Hughes, and Henry Kaufman. Direct Manipulation of Free-Form Deformations. *Computer Graphics (Proc. SIGGRAPH '92) 26*, 2 (1992), 177–184.

[56] Huynh, Hung T. Accurate Monotone Cubic Interpolation. *SIAM J. Numerical Analysis 30*, 1 (1993), 57–100.

[57] Lawson, C.L. Software for C^1 Surface Interpolation. In *Mathematical Software III*, J. R. Rice, Ed. Academic Press, New York, 1977, pp. 161–194.

[58] Lee, Seungyong, Kyung-Yong Chwa, James Hahn, and Sung Yong Shin. Image Morphing Using Deformable Surfaces. *Proc. Computer Animation '94* (1994), 31–39. IEEE Computer Society Press.

[59] Lee, Seungyong, George Wolberg, Kyung-Yong Chwa, and Sung Yong Shin. Image Metamorphosis With Scattered Feature Constraints. *IEEE Trans. Visualization and Computer Graphics 2*, 4 (1996), 337–354.

[60] Lee, Seungyong, George Wolberg, and Sung Yong Shin. Scattered Data Interpolation Using Multilevel B-Splines. *IEEE Trans. Visualization and Computer Graphics 3*, 3 (1997), 228–244.

[61] Litwinowicz, Peter, and Lance Williams. Animating Images with Drawings. *Computer Graphics (Proc. SIGGRAPH '94)* (1994), 409–412.

[62] Lustig, Irving, Roy Marsten, and David Shanno. Interior Point Methods for Linear Programming: Computational State of the Art. *ORSA Journal on Computing 6*, 1 (1994), 1–14.

[63] Lyche, T., and K. Morken. Knot Removal for Parametric B-spline Curves and Surfaces. *Computer Aided Geometric Design 4* (1987), 217–230.

[64] Malcolm, Michael M. On the Computation of Nonlinear Spline Functions. *SIAM J. Numerical Analysis 14*, 2 (1977), 254–282.

[65] McAllister, D., E. Passow, and J. Roulier. Algorithms for Computing Shape Preserving Splines Interpolations to Data. *Math. Comp. 31* (1977), 717–725.

[66] McLain, D.H. Two Dimensional Interpolation from Random Data. *Comp. Journal 19* (1976), 178–181.

[67] Mitchell, Don P. Generating Antialiased Images at Low Sampling Densities. *Computer Graphics (Proc. SIGGRAPH '87) 21*, 4 (1987), 65–72.

[68] Nielson, Gregory M. Some Piecewise Polynomial Alternatives to Splines Under Tension. In *Computer Aided Geometric Design*, R. Barnhill and R. Riesenfeld, Eds. Academic Press, New York, 1974, pp. 209–235.

[69] Pruess, S. Properties of Splines in Tension. *J. of Approximation Theory 17* (1976), 86–96.

[70] Pruess, S. Alternatives to the Exponential Spline in Tension. *Math. Comp. 33* (1979), 1273–1281.

[71] Pruess, S. Shape Preserving C^2 Cubic Spline Interpolation. *IMA J. Numerical Analysis 13* (1993), 493–507.

[72] Rao, Singiresu S. *Engineering Optimization*, 3 ed. John Wiley and Sons, New York, 1996.

[73] Rogers, David, and J. Alan Adams. *Mathematical Elements for Computer Graphics*. McGraw-Hill, New York, 1990.

[74] Schröder, Peter, and Wim Sweldens. Wavelets in Computer Graphics. *SIGGRAPH 96 Course Notes* (1996).

[75] Schumaker, Larry L. Fitting Surfaces to Scattered Data. In *Approximation Theory II*, C. Chui, L. Schumaker, and G. Lorentz, Eds. Wiley, New York, 1976, pp. 203–268.

[76] Schumaker, Larry L. On Shape Oreserving Quadratic Spline Interpolation. *SIAM J. Numerical Analysis 20* (1983), 854–864.

[77] Schweikert, D. An Interpolation Curve Using Splines in Tension. *J. Math and Phys. 45* (1966), 312–317.

[78] Shampine, L.F., Jr. R.C. Allen, and S. Pruess. *Fundamentals of Numerical Computing.* John Wiley and Sons, New York, 1997.

[79] Shepard, D. A Two Dimensional Interpolation Function for Irregularly Spaced Data. *Proc. 23rd Nat. Conf. ACM* (1968), 517–524.

[80] Späth, H. Exponential Spline Interpolation. *Computing 4* (1969), 225–233.

[81] Szeliski, Richard. Fast Surface Interpolation Using Hierarchical Basis Functions. *IEEE Trans. Pattern Analysis and Machine Intelligence 12*, 6 (1990), 513–528.

[82] Terzopoulos, Demetri. Multilevel Computational Processes for Visual Surface Reconstruction. *Computer Vision, Graphics, and Image Processing 24* (1983), 52–96.

[83] Timoshenko, S., and S. Woinowsky-Krieger. *Theory of Plates and Shells.* McGraw-Hill, New York, 1959.

[84] Welch, William, and Andrew Witkin. Variational Surface Modeling. *Computer Graphics (Proc. SIGGRAPH '86) 26*, 2 (1992), 157–166.

[85] Wolberg, George. *Digital Image Warping.* IEEE Computer Society Press, Los Alamitos, CA, 1990.

[86] Wolberg, George. Nonuniform Image Reconstruction Using Multilevel surface. *Proc. IEEE Intl. Conference on Image Processing* (1997).

[87] Wolberg, George, and Itzik Alfy. Monotonic Cubic Spline Interpolation. *Proc. of Computer Graphics International* (1999), 188–195.

[88] Wolfe, P. The simplex method for quadratic programming. *Econometrica 27* (1959), 382–398.