THE UNIVERSITY OF NEWCASTLE UPON TYNE
DEPARTMENT OF COMPUTING SCIENCE

Shape-Preserving Algorithms for Curve and Surface Design

by

Rafhat Iqbal

PhD Thesis

August 1998

Abstract

This thesis investigates, develops and implements algorithms for shape-preserving curve and surface design that aim to reflect the shape characteristics of the underlying geometry by achieving a visually pleasing interpolant to a set of data points in one or two dimensions. All considered algorithms are local and useful in computer graphics applications. The thesis begins with an introduction to existing methods which attempt to solve the shape-preserving curve interpolation problem using C^1 cubic and quadratic splines. Next, a new generalized slope estimation method involving a parameter t, which is used to control the size of the estimated slope and, in turn, produces a more visually pleasing shape of the resulting curve, is proposed. Based on this slope generation formula, new automatic and interactive algorithms for constructing shape-preserving curves from C^1 quadratic and cubic splines are developed and demonstrated on a number of data sets. The results of these numerical experiments are also presented. Finally, a method suggested by Roulier which generates C^1 surfaces interpolating arbitrary sets of convex data on rectangular grids is considered in detail and modified to achieve more visually pleasing surfaces. Some numerical examples are given to demonstrate the performance of the method.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr John Lloyd, for his guidance, cooperation and enormous patience in the progress of this research. He has been exceptionally kind to me and, through him, several opportunities have been made available that would have otherwise been impossible.

I wish to thank Dr Chris Phillips and Dr Kenneth Wright for their useful suggestions and serving as members of the thesis committee. Thanks also go to Richard Achmatowicz, Liaqat Hussain (Stoke-on-Trent), Dr Darren Hodge, Prof Zahid H Kazmi, Dr Maciej Koutny, Dr Murtaza Malik, Dr Amer Saeed, Hatim M Tahir, Shirley Craig (Dept. Librarian) and all the staff in the Department of Computing Science who helped me in one way or another during the course of this work.

I am deeply grateful to the Ministry of Education, Government of Pakistan for awarding me the Quaid-e-Azam scholarship which enabled me to accomplish this research.

The acknowledgements would be incomplete without recognizing the love and patience of the author's parents, sisters and brothers who, through the years, have incessantly boosted the morale and the ambition of the author, without which little could have been achieved. My loving gratitude and heartfelt thanks go to all of them for their everlasting support throughout the course of my education.

Dedication

This work is dedicated to the memory of my beloved grandmothers who passed away during my stay in the UK.

Contents

Abstract ii

Acknowledgements iii

Dedication iv

Contents v

List of Figures viii

List of Tables xii

Chapter 1 1

Introduction and Preliminaries

1.1 The Spline Function 4

1.1.1 C^k Continuity for Spline Curves 4

1.2 Shape-Preserving Interpolation 7

1.2.1 Polynomial Splines 9

1.2.2 Splines under Tension 10

1.2.3 Rational Splines 12

1.3 Organisation and Layout of the Thesis 15

1.3.1 Publications 16

1.4 Some Definitions 16

1.5 Definition of Test Problems 20

1.5.1 Univariate Data Sets 20

1.5.1.1 Monotone Data 20

1.5.1.2 Monotone and Convex Data 21

1.5.1.3 Convex Data 21

1.5.2 Bivariate Data Sets 21

1.5.2.1 Convex Data 21

Chapter 2 27

Shape-Preserving Algorithms Using Cubic Splines

2.1 Piecewise Cubic Hermite Interpolation 28

2.2 Fritsch and Carlson Method 30

2.3 Slope Estimation Methods 33

2.3.1 Butland Method 33

2.3.2 Brodlie Method 33

2.3.3 Fritsch and Butland Method 34

2.3.4 Costantini Method 34

2.3.5 Huynh Method 38

2.4 Yan Method 40

2.5 Gasparo and Morandi Method 43

2.6 Beatson and Wolkowicz Method 45

2.7 Pruess Method 47

2.8 Numerical Examples and Conclusions 52

Chapter 3 73

Shape-Preserving Algorithms Using Quadratic Splines

3.1 Slope Calculation Methods 74

3.2 Schumaker Algorithm 78

3.3 McAllister and Roulier Algorithm 83

3.4 Similarity Between McAllister-Roulier and Schumaker Algorithms 86

3.5 Iterative Improvement to the Slopes 89

3.6 Numerical Examples and Conclusions 92

Chapter 4 101

An Algorithm for Shape-Preserving Curve Drawing

4.1 New Generalised Slope Estimation Method 101

4.2 An Automatic Algorithm using Cubic Splines 102

4.3 Numerical Examples and Conclusions 112

Chapter 5 119

Interactive Algorithms for Shape-Preserving Curve Drawing

5.1 An Interactive Algorithm using Quadratic Splines 120

5.2 An Interactive Algorithm using Cubic Splines 129

5.3 Numerical Examples and Conclusions 132

Chapter 6 148

Algorithms for Shape-Preserving Surface Drawing

6.1 Literature Review 149

6.2 Roulier Algorithm 152

6.3 Modified Algorithm 155

6.4 Numerical Examples and Conclusions 157

Chapter 7 165

Conclusions and Future Work

7.1 Conclusions 165

7.2 Future Work 169

References 172

List of Figures

1.1	The Lagrange polynomial fit to the data given in Table 1.1	3
1.2	Standard cubic spline interpolant showing unwanted inflections to the data given in Table 1.1	7
1.3	Shape-preserving interpolant to the data given in Table 1.1	9
1.4	Standard bicubic spline interpolating the data given in Table 1.2	14
1.5	Shape-preserving interpolant to the data presented in Table 1.2	14
1.6	Graphic representation of Data 1	22
1.7	Graphic representation of Data 2	22
1.8	Graphic representation of Data 3	23
1.9	Graphic representation of Data 4	23
1.10	Graphic representation of Data 5	24
1.11	Graphic representation of Data 6	24
1.12	Graphic representation of Data 7 at the grid points	25
1.13	Graphic representation of Data 8 at the grid points	26
2.1	Monotonicity region S for cubic splines	31
2.2	Costantini method with different values of q and k	35
2.3	Costantini method with different values of k and fixed q	37
2.4	Costantini method with different values of q and fixed k	37
2.5	Beatson-Wolkowicz method which interpolates Data 1	57
2.6	Brodlie method which interpolates Data 1	57
2.7	Butland method which interpolates Data 1	58
2.8	Fritsch-Butland method which interpolates Data 1	58
2.9	Gasparo-Morandi method which interpolates Data 1	59
2.10	Huynh method which interpolates Data 1	59

2.11	Pruess method which interpolates Data 1	60
2.12	Yan method which interpolates Data 1	60
2.13	Beatson-Wolkowicz method which interpolates Data 2	61
2.14	Brodlie method which interpolates Data 2	61
2.15	Butland method which interpolates Data 2	62
2.16	Fritsch-Butland method which interpolates Data 2	62
2.17	Gasparo-Morandi method which interpolates Data 2	63
2.18	Huynh method which interpolates Data 2	63
2.19	Pruess method which interpolates Data 2	64
2.20	Yan method which interpolates Data 2	64
2.21	Beatson-Wolkowicz method which interpolates Data 3	65
2.22	Brodlie method which interpolates Data 3	65
2.23	Butland method which interpolates Data 3	66
2.24	Fritsch-Butland method which interpolates Data 3	66
2.25	Gasparo-Morandi method which interpolates Data 3	67
2.26	Huynh method which interpolates Data 3	67
2.27	Pruess method which interpolates Data 3	68
2.28	Yan method which interpolates Data 3	68
2.29	Beatson-Wolkowicz method which interpolates Data 4	69
2.30	Brodlie method which interpolates Data 4	69
2.31	Butland method which interpolates Data 4	70
2.32	Fritsch-Butland method which interpolates Data 4	70
2.33	Gasparo-Morandi method which interpolates Data 4	71
2.34	Huynh method which interpolates Data 4	71
2.35	Pruess method which interpolates Data 4	72
2.36	Yan method which interpolates Data 4	72
3.1	Data configuration for McAllister and Roulier's slope method when $\lvert\delta_{i-1}\rvert > \lvert\delta_i\rvert > 0$	76

3.2	Data configuration for McAllister and Roulier's slope method when $0 < \|\delta_{i-1}\| \leq \|\delta_i\|$	77
3.3	Case 1 configuration for McAllister-Roulier algorithm	84
3.4	Case 2 configuration for McAllister-Roulier algorithm	85
3.5	Schumaker algorithm with intial slopes estimates	94
3.6	Schumaker algorithm with improved slopes estimates	94
3.7	Shape-preserving quadratic spline interpolant to Data 1	95
3.8	Shape-preserving quadratic spline interpolant to Data 2	96
3.9	Shape-preserving quadratic spline interpolant to Data 3	97
3.10	Shape-preserving quadratic spline interpolant to Data 4	98
3.11	Shape-preserving quadratic spline interpolant to Data 6	99
3.12	Shape-preserving quadratic spline interpolant to Data 5	100
4.1	Visualization of slope formula with different values of t	104
4.2	Monotonicity region S (shaded with dots) and convexity region (hatched horizontally)	107
4.3	New automatic algorithm interpolating Data 1	114
4.4	New automatic algorithm interpolating Data 2	115
4.5	New automatic algorithm interpolating Data 3	116
4.6	New automatic algorithm interpolating Data 4	117
4.7	New automatic algorithm interpolating Data 6	118
5.1	Monotonicity region M for quadratic splines	122
5.2	Algorithm QCURVE with varying w_2; $w_1 = 1.0$ and $t_i = 1.0$	137
5.3	Algorithm CCURVE with varying w_2; $w_1 = 1.0$ and $t_i = 1.0$	137
5.4	Default curve: $t_i = t_i^*$ for each i, $2 \leq i \leq n-1$	138
5.5	Modified curve: $t_9 = 0.211, 0.6, 1.5$ and 10.0; $t_i = t_i^*$ otherwise	139
5.6	Modified curve: $t_{10} = 1.0, 2.0$ and 10.0; $t_i = t_i^*$ otherwise	139
5.7	Modified curve with large parameter values: $t_9 = 50$ and $t_{10} = 50$	140
5.8	Default curve: $t_i = t_i^*$ for $i = 2, ..., n-1$	141

5.9	Modified curve: $t_2 = 0.2, 0.8, 1.5$ and 3.0; $t_i = t_i^*$ otherwise	142
5.10	Default curve: $t_i = t_i^*$ for each i, $2 \leq i \leq n-1$	143
5.11	Modified curve: $t_9 = 0.283, 0.8, 1.5$ and 10.0; $t_i = t_i^*$ otherwise	144
5.12	Modified curve: $t_{10} = 0.794, 2.0$ and 10.0; $t_i = t_i^*$ otherwise	144
5.13	Modified curve with large parameter values: $t_9 = 40$ and $t_{10} = 40$	145
5.14	Default curve: $t_i = t_i^*$ for $i = 2, ..., n-1$	146
5.15	Modified curve: $t_2 = 0.515, 0.8, 1.3$ and 2.5; $t_i = t_i^*$ otherwise	147
6.1	Modified algorithm with Butland slopes	159
6.2	Modified algorithm with Frey slopes	159
6.3	Modified algorithm with Butland slopes	160
6.4	Modified algorithm with Frey slopes	160
6.5	Combined plot by Modified algorithm with Butland and Frey slopes	161
6.6	Combined plot by Modified algorithm with Butland and Frey slopes	161
6.7	Modified algorithm with Butland slopes	162
6.8	Modified algorithm with Frey slopes	162
6.9	Modified algorithm with Butland slopes	163
6.10	Modified algorithm with Frey slopes	163
6.11	Combined plot by Modified algorithm with Butland and Frey slopes	164
6.12	Combined plot by Modified algorithm with Butland and Frey slopes	164
7.1	Interactive rational spline interpolant to Data 4	170

List of Tables

1.1	Monotone and convex data	2
1.2	Bivariate monotone data	13
1.3	Monotone Data 1	22
1.4	Monotone Data 2	22
1.5	Monotone Data 3	23
1.6	Monotone and convex Data 4	23
1.7	Locally monotone and convex Data 5	24
1.8	Convex Data 6	24
1.9	Convex Data 7	25
1.10	Convex Data 8	26
2.1	The values associated with the curves in Figure 2.2	36
2.2	Jumps in the second derivative at the interior knots	36
3.1	Jumps in the second derivative at the interior knots for Data 4	90
3.2	Jumps in the second derivative at the interior knots for Data 6	90
3.3	The values associated with Figure 3.7	95
3.4	The values associated with Figure 3.8	96
3.5	The values associated with Figure 3.9	97
3.6	The values associated with Figure 3.10	98
3.7	The values associated with Figure 3.11	99
3.8	The values associated with Figure 3.12	100
4.1	Jumps in the second derivative at the interior knots	104
4.2	The values used in the construction of Figure 4.3	114
4.3	The values used in the construction of Figure 4.4	115
4.4	The values used in the construction of Figure 4.5	116

4.5	The values used in the construction of Figure 4.6	117
4.6	The values used in the construction of Figure 4.7	118
5.1	The values used in the construction of Figure 5.4	138
5.2	The values used in the construction of Figure 5.8	141
5.3	The values used in the construction of Figure 5.10	143
5.4	The values used in the construction of Figure 5.14	146
6.1	Timing information for the Figure 6.1 through Figure 6.10	158

Chapter 1

Introduction and Preliminaries

This thesis is concerned with the process of representing a given set of data by a smooth curve or surface which exactly reproduces the given values and preserves the shape characteristics of the data. This is a problem which arises frequently in a wide range of applications in science, engineering and computer graphics. The classical interpolation methods developed by Lagrange and Hermite are used in data interpolation even today. Unfortunately, these methods often fail to reproduce important qualitative aspects of the data such as monotonicity and/or convexity and may not meet the expectations of the designer. They typically generate unexpected bumps, oscillations, waves or spurious wrinkles that ruin the shape of curves or surfaces. Much work has been done subsequently, but surprisingly the curve and surface design problems have not yet been dealt with adequately. For example, before the advent of computers, the method used by a draftsman or engineer was to draw the curve manually through the data points. This was a skilled job but very tedious and the opinion of different people might vary as to what constituted the "best" curve. It is not completely understood why a draftsman's curve usually looks better than one generated by some automatic curve interpolation algorithms. The draftsman somehow senses the shape information contained in the data and draws curves that are correct, smooth and pleasing to the eye. However, with the advent of the computer and its ability to perform many calculations in a relatively short time, the question arises - why not

let the computer draw the curve? To do this, of course, we must provide a mathematical representation of the curve. Fortunately, there are several mathematical methods of representing a curve, given a set of ordinates and abscissas, and these have been well documented in the literature.

Perhaps the best known method is fitting by polynomials. In spite of its theoretical appeal, polynomial interpolation suffers from several serious drawbacks; for example, any local modification to the data has a global effect on the interpolant. Thus, they are unsuitable for applications where the data may be frequently changed, or where one wishes to compute the interpolant as the data arrives. Polynomials seem to do all right for small numbers of data points, but when we go to higher degree, that is, as the number of interpolation points increases, so does the complexity of the interpolant and severe oscillations often appear. This is a major disadvantage of polynomial interpolation and is especially undesirable in many design problems when practical considerations indicate that the data has some additional properties such as monotonicity and/or convexity. An example illustrates the problem: when Lagrange polynomial interpolation is applied to a set of monotone and convex data given in the following Table 1.1. The results obtained are displayed

x_i	-2.0	-1.0	-0.5	-0.25	-0.1
y_i	0.25	1.0	4.0	16.0	100.0

Table 1.1 Monotone and convex data.

in Figure 1.1. While this procedure can often yield quite satisfactory results, Figure 1.1 shows a very different behaviour from what one would normally expect. Clearly Lagrangian interpolation has introduced bumps and unwanted

inflection points in the first three subintervals that are not supported by the data. In most practical applications, these oscillations are unacceptable.

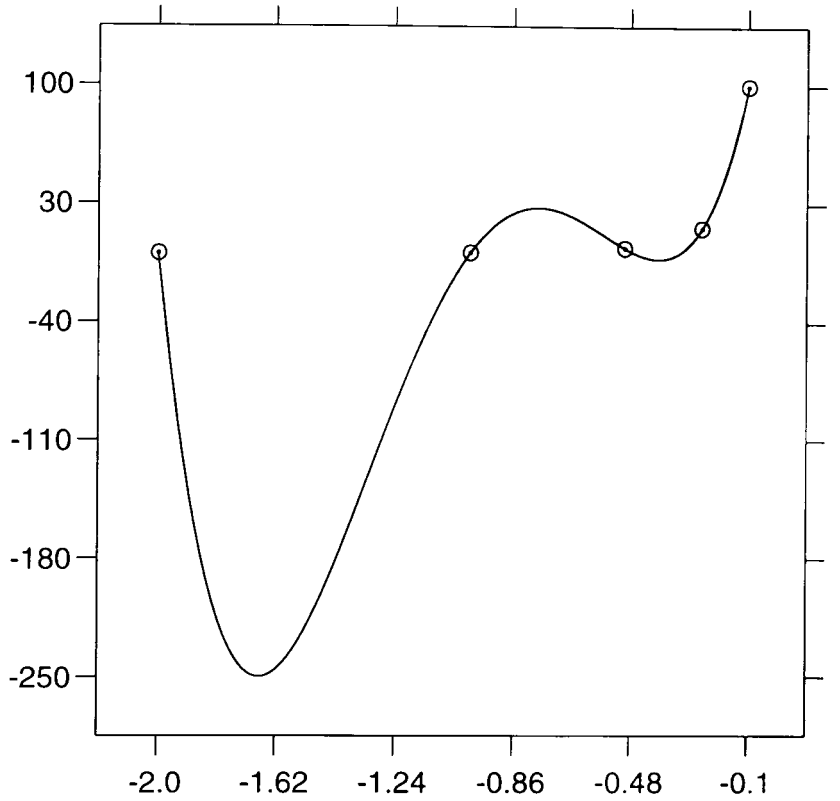

Figure 1.1 The Lagrange polynomial fit to the data given in Table 1.1.

Several schemes have been suggested for combating the difficulties encountered with polynomial interpolation. One of the most successful approaches is to use piecewise polynomial interpolation which leads to the division of the interval into small subintervals: instead of trying to interpolate the data over the entire interval by one polynomial of high degree, one interpolates the data by a piecewise polynomial function, where the degree of the polynomial pieces associated with each subinterval is small and where the pieces satisfy certain continuity conditions. This avoids the problems associated with high degree polynomials and retains the inherent conceptual simplicity of polyno-

mial interpolation. The use of piecewise polynomials for curve and surface interpolation was stimulated in the early 1960's by the introduction of spline functions. There are various possible methods of introducing the theory of spline functions and a considerable volume of literature has been written on these functions since their development by Schoenberg [60].

1.1 The Spline Function.

Let $(x_i, y_i), i = 1, ..., n$, be a given set of data points such that $x_1 < x_2, ..., < x_n$. Spline functions are piecewise polynomials with derivatives constrained for the purpose of making the resulting function smooth at the knot points x_i. An interpolating spline function of degree k with interpolating points $x_i, i = 1, ..., n$, is a piecewise polynomial $p(x)$ satisfying the following properties:

1. $p(x)$ is $k-1$ times differentiable at each point x_i, $i = 1, ..., n$.
2. On each subinterval $[x_i, x_{i+1}]$, $p(x)$ is a polynomial of degree not exceeding k.
3. $p(x_i) = y_i, i = 1, ..., n$.

In other words the property 1 ensures that the interpolating spline function $p(x)$ is of class C^{k-1} as $p(x)$ is continuous and has continuous derivatives for all orders less than or equal to $k-1$.

1.1.1 C^k Continuity for Spline Curves.

An interpolating spline function $p(x)$ on $[x_1, x_n]$ is called C^k continuous (k times continuously differentiable) at the knot x_i if the following conditions hold:

$$p_i(x_i) = p_{i+1}(x_i), \quad i = 1, ..., n-2$$
$$p_i'(x_i) = p_{i+1}'(x_i), \quad i = 1, ..., n-2$$
$$p_i''(x_i) = p_{i+1}''(x_i), \quad i = 1, ..., n-2$$
$$\cdots\cdots\cdots\cdots$$
$$\cdots\cdots\cdots\cdots$$
$$p_i^{(k)}(x_i) = p_{i+1}^{(k)}(x_i), \quad i = 1, ..., n-2$$

The spline function can be represented in various ways: the simplest, and one of the most common representations, being in terms of B-splines as basis functions. This basis is fairly well conditioned and has many other nice properties which usually lead to stable and simple algorithms; see de Boor [7] and Schumaker [62]. Interpolation schemes using a B-spline representation have been developed by Cox [19] and de Boor [8].

A spline function is the mathematical equivalent of the draftsman's physical spline - a long narrow strip of wood or plastic used to fit curves through specified data points, for example, in shipbuilding, automotive and aircraft design. The splines are shaped by lead weights called ducks. By varying the number and positions of the lead weights, the spline is made to pass through the specified data such that the resulting curve appears smooth, or fair and pleasing to the eye. The mechanical spline has the properties of continuity in slope and curvature while minimizing the "strain energy" of a thin elastic beam subject to the interpolation constraints. Cubic splines have the same properties of continuity, but only approximate to the minimization of energy, by minimizing $\int y''^2 \, dx$. The most attractive property of the cubic spline is that, at each point, in the range the function is represented by a cubic with C^2 continuity at the data points, while flexibility is retained due to the

discontinuities in the third derivatives at the "joins" of the individual cubics. Some of the earlier work on the use of spline functions in curve interpolation is due to Ahlberg, Nilson and Walsh [1]. Gordon [38] extended these techniques to surface fitting and bivariate interpolation through curve networks.

Standard cubic spline interpolation is a global method in the sense that estimation of the derivatives at the data points is made on the basis of all the data points by solving a system of linear equations. Consequently, all the data must be available before computation of the interpolant can begin. Also, if a small portion of the data is changed, the entire linear system of equations must be recomputed. Hence, this results in a change in the whole curve.

Cubic splines are among the most widely used interpolants because they offer attractive smoothness properties and computational convenience and efficiency. The practical utility of cubic splines is quite evident from their widespread use as finite element basis functions, in collocation approximations to differential equations and data fitting applications. On the other hand, they are not without their shortcomings and have a tendency to produce more inflection points and overshoots than a draftsman would ordinarily include when drawing a curve through the data. This occurs when the interpolant cannot bend sharply enough at certain points and, as a consequence, breaks out into spurious oscillations. Figure 1.2 displays the standard cubic spline interpolant that interpolates to the same data used for Figure 1.1 and we immediately notice some erratic behaviour in the resulting curve. Although it is neither monotone nor convex, it oscillates much less widely than the Lagrange polynomial interpolant in Figure 1.1. In fact, while one might desire a monotone and convex interpolant in this example, the cubic spline is neither.

Thus, if these shape-preserving constraints are needed, one certainly must look beyond the standard cubic spline.

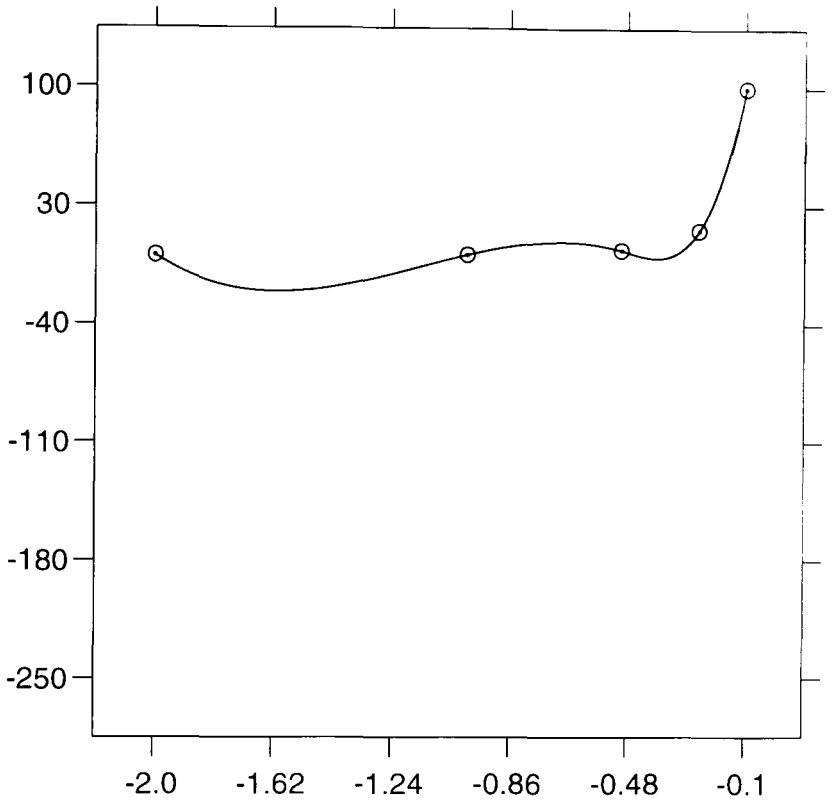

Figure 1.2 Standard cubic spline interpolant showing unwanted inflections to the data given in Table 1.1.

1.2 Shape-Preserving Interpolation.

Often data in an interpolation problem represents a physical quantity having a certain behaviour (monotone decreasing, concave etc) as a function of other quantities, but the previous examples show that polynomial and standard spline interpolation methods are not guaranteed to satisfy these shape characteristics of the data. Thus, one would like to describe an interpolation scheme preserving these shape properties. Such "shape-preserving interpolation" provides a means of avoiding the extraneous oscillations often

seen in standard cubic spline interpolation and similar algorithms. In addition to the shape-preserving requirements, it is often desirable that the method should be local. A local method is such that if a small portion of the data is changed, the effect will only be apparent in the neighbourhood of the change and one would then like the rest of it to remain unaffected. Here, the estimation of the slope at a data point is based only on information at the point itself and neighbouring points. In contrast with global methods, local methods often produce C^1 interpolants and the discontinuity in the second derivative at the data points does not constitute a serious problem as they usually provide curves which are visually pleasing. Their simplicity makes them popular in computer graphics, and because they are local methods, the interpolant can be generated as the data points are collected. A number of local methods have been developed which attempt to maintain shape characteristics of the data, amongst which are the so called shape-preserving methods. These methods are shape-preserving in the sense that, in those intervals where the data is monotone increasing or decreasing, the interpolant has the same property; similarly, in those intervals where the data is convex or concave, the same is also true of the interpolant. Figure 1.3 illustrates the result of interpolating to the same data used for Figure 1.1 by a shape-preserving method, and the resulting graph is both convex and monotonic increasing as well as pleasing to the eye, as compared with the curves drawn in Figure 1.1 and Figure 1.2 which are neither monotone nor convex.

Some progress has been made in the last decade in the field of shape-preserving interpolation and three different techniques which provide some control over the shape of the interpolating curve have been considered so far; namely,

a) Polynomial splines.

b) Spline under tension.

c) Rational splines.

These are discussed in the following sections.

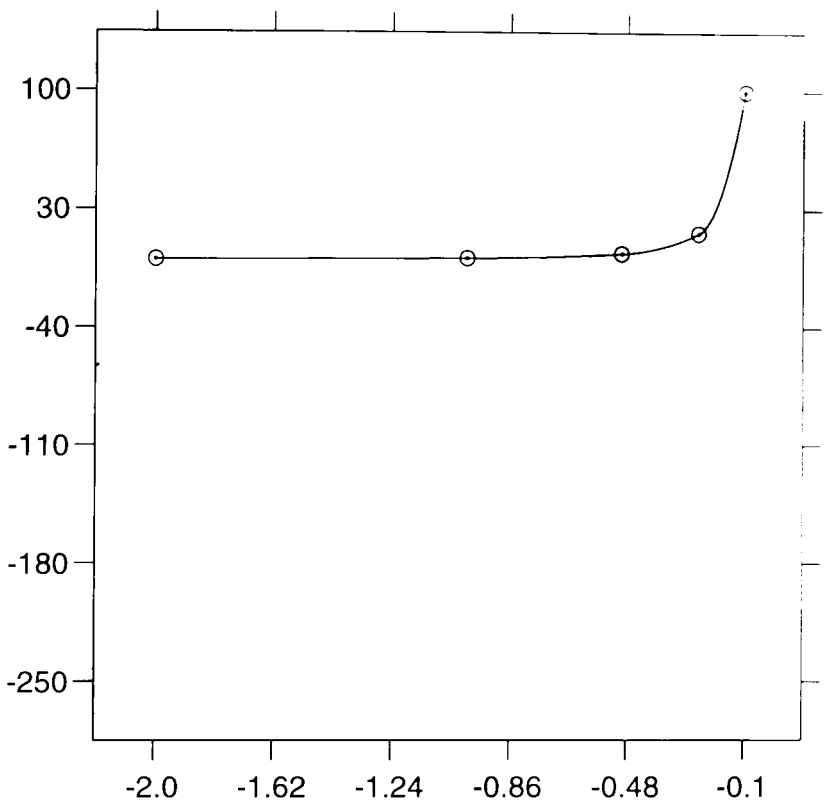

Figure 1.3 Shape-preserving interpolant to the data given in Table 1.1.

1.2.1 Polynomial Splines.

The specific problem of shape-preserving interpolation using polynomial splines, especially quadratic and cubic, has been considered by a number of authors. de Boor [7] proposed a taut spline algorithm which makes use of C^1 piecewise cubic polynomials and preserves the convexity of the data by inserting at most one additional knot between each pair of data points but does

not guarantee monotonicity. Fritsch and Carlson [33] describe an algorithm in which an initial piecewise cubic interpolant is modified by changing the derivative values of the interpolant (where necessary) to produce a monotone piecewise cubic interpolant. Costantini [16] has developed the Fritsch and Carlson method further to include interpolants of arbitrary degree and has proposed a new technique for slope estimation. Beatson and Wolkowicz [5] and Yan [67] describe methods which preserve the monotonicity of the data and provide an alternative to the Fritsch and Carlson [33] method. Here, derivative values are not modified; instead, when derivative values do not ensure monotonicity, rather than changing the slopes, extra knots are inserted in that subinterval. Dougherty, Edelman and Hyman [25] have introduced C^1 cubic and C^2 quintic spline polynomials for shape-preserving interpolation. Fiorot and Tabka [28] have described shape-preserving global C^2 cubic spline interpolants, where the existence of such an interpolant depends upon the existence of solutions of a system of linear inequalities representing the first derivative values at the data points.

McAllister, Passow and Roulier [49] consider the problem of interpolating monotone and convex data. They make use of piecewise polynomial Bernstein representations and introduce additional knots into their schemes. Such a scheme for quadratic spline interpolation is described by McAllister and Roulier [50] and this technique is developed further by Schumaker [61].

1.2.2 Splines under Tension.

The idea of an interpolating spline under tension was first introduced by Schweikert [63] in order to eliminate extraneous inflection points in curves fitted by cubic splines. Schweikert used uniform tension and obtained on each

subinterval an interpolant that was a linear combination of 1, x, e^{tx} and e^{-tx}, with t as "tension parameter". These interpolants, with a sufficiently large tension, do remove extraneous points of inflections, but are more expensive to construct and evaluate than cubic splines. His idea was further investigated and implemented by Cline [14] and generalized by Spath [64]. A detailed derivation of the generalized form based on a variational principle is given by Barsky [4]. These generalizations allow local choices of the tension parameter. A common feature of all of these notions is that of a tension parameter which may depend on the knot interval. As the tension parameters increase, the graph of the spline tends to pull closer to the shape of the polygonal segments connecting the data. Thus, for sufficiently high tension parameters, the spline will be shape-preserving. Typically, one knows from experience how to choose the tension parameters so that shape-preservation is achieved in two or three attempts. Each time a change is made in the tension parameters, the complete spline must be recalculated by solving a system of linear equations.

The main advantage of the spline under tension is that sufficient tension yields a shape-preserving interpolant; that is, an interpolant which matches the monotonicity and convexity properties of the data (see Fletcher and McAllister [29], Kaklis and Pandelis [44], McCartin [51], Pruess [53], and Rentrop [57]). In this connection, Renka [56] has suggested heuristic ways of choosing a local tension parameter associated with each subinterval which yield shape-preserving interpolants. In general, splines under tension are satisfactory for many purposes, but they have the added disadvantage that exponential functions must be computed for each evaluation.

1.2.3 Rational Splines.

Besides polynomials, other classical functions have also been used extensively for interpolation. The most popular include rational functions. Rational splines are also of use for producing smooth curves through the given data points. The computational effort involved in their calculation and their subsequent use for interpolation is not significantly greater than that required for polynomial splines, but is significantly less than that for splines under tension. In recent years, several shape-preserving methods have been developed which make use of piecewise rational quadratic or cubic interpolants. A rational spline solution to the problem of shape-preserving interpolation is provided by Delbourgo and Gregory [23] and Gregory [35]. The use of a piecewise rational quadratic function for constructing a C^1 monotonic interpolant which interpolates monotone data has been described by Gregory and Delbourgo [37]. These authors [22] use a similar representation to obtain a global C^2 interpolant which requires solving a set of non-linear equations in the knot derivatives, derived by the imposition of the C^2 continuity constraints at all interior knots. Rational spline approaches to achieve convexity-preserving interpolants to the convex data have been proposed by Delbourgo [20, 21], and Ramirez and Lorente [55].

The ideas of shape-preserving curve interpolation can be extended to surfaces if the defining data points lie on grid lines. The problem of generating shape-preserving surfaces is very important in computer aided geometric design applications and it is often desirable to interpolate three-dimensional surface data defined by two independent variables (x and y) and one dependent variable. A number of techniques have been developed for surface interpolation including Coons and Bezier patches and tensor products of Bezier curves, cubic splines, and B-splines. A difficulty arises with these methods, especially the

spline methods: abrupt changes in the dependent variable of the data may induce artificial or exaggerated hills and valleys in the interpolating surface. Figure 1.4 illustrates the interpolating bicubic spline surface for the EOS data given in the following Table 1.2. This data comes from Carlson and Fritsch [12] and represents an equation of state (EOS) surface for aluminum with pressure

	T	-2.30	-1.61	-0.92	-0.51	-0.22	0.00
ρ							
-0.07		-34.54	-13.82	-10.10	-7.26	-5.66	-4.53
0.33		-34.54	-13.82	-10.10	-7.26	-5.66	-4.13
0.55		-34.54	-13.82	-10.10	-7.26	-4.88	-3.35
0.69		-34.54	-13.82	-10.10	-4.82	-3.34	-2.73
0.84		-34.54	-13.82	-2.52	-2.22	-1.98	-1.78
0.93		-34.54	-2.68	-1.88	-1.56	-1.41	-1.28
0.98		-3.06	-2.28	-1.63	-1.32	-1.15	-1.05
1.02		-2.86	-1.92	-1.39	-1.10	-0.92	-0.81
1.08		-2.37	-1.60	-1.17	-0.90	-0.72	-0.60
1.13		-1.89	-1.30	-0.95	-0.71	-0.54	-0.41

Table 1.2 Bivariate monotone data.

as a function of density (ρ) and temperature (T). For univariate data having a drastic change in slope, the cubic spline typically deviates extensively from the desired trend between data points (see Figure 1.2). Figure 1.4 shows that the same phenomenon occurs with the bicubic spline interpolation. The interpolating bicubic spline has produced unwanted ripples or overshoots that are clearly unacceptable. A small number of methods have been proposed which eliminate these unwanted hills and valleys in an interpolating spline surface not indicated by the data. These methods are shape-preserving in the sense that if the data exhibits a given monotonicity and/or convexity along all grid lines parallel to the axes, then the resulting interpolant also exhibits

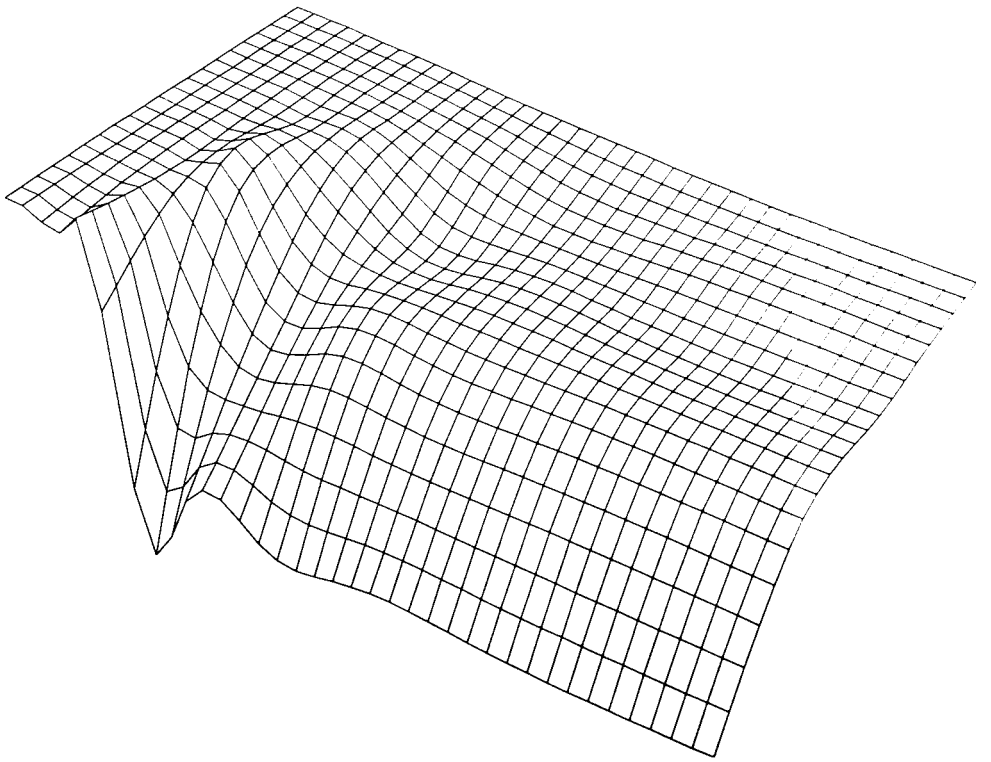

Figure 1.4 Standard bicubic spline interpolating the data given in Table 1.2.

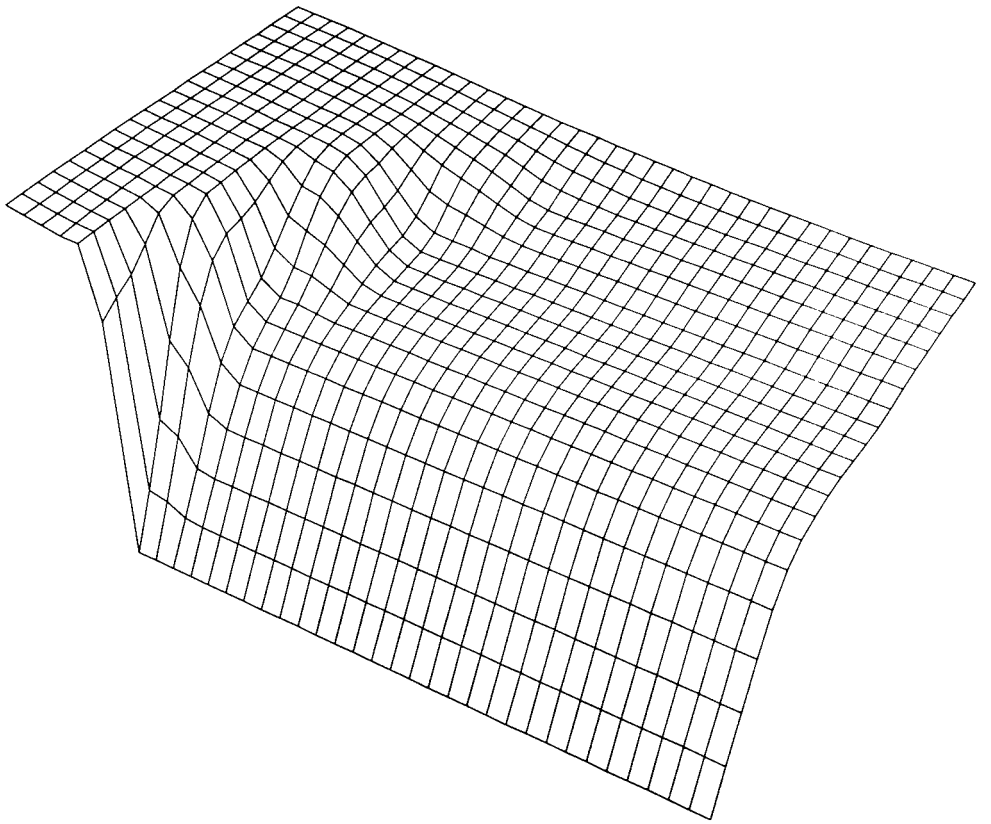

Figure 1.5 Shape-preserving interpolant to the data presented in Table 1.2.

the same monotonicity and/or convexity along lines parallel to these grid lines as well. Figure 1.5 shows the result of applying a shape-preserving method to the same data used for Figure 1.4, where all of the ripples or overshoots have been eliminated. The resulting surface is monotonic along the grid lines and visually pleasing.

1.3 Organisation and Layout of the Thesis.

The aim of this thesis is to investigate, develop and implement local algorithms for shape-preserving curve and surface design using C^1 quadratic and cubic interpolating splines. These algorithms are used to fit a number of data sets of practical significance and the results are compared to test their performance. The rest of the thesis is organized in the following manner. We conclude this chapter with some formal definitions to be used in the discussion to aid the understanding of subsequent chapters and a description of data sets for both curve and surface interpolation problems which will be used to illustrate the behaviour of the various algorithms to be considered. Chapter 2 is devoted to previous relevant research for the construction of shape-preserving curve interpolants using cubic splines. Here, several slope estimation methods which satisfy conditions of monotonicity for the cubic splines are discussed and compared. In Chapter 3, we study the methods of interpolating shape-preserving curves proposed by McAllister and Roulier [50] and Schumaker [61] which use C^1 quadratic splines, and prove that these are identical if the slopes required by the Schumaker algorithm are estimated by the formula proposed by Butland [11]. We also, in the case of convex data, improve slopes further by using an iterative technique to guarantee more visually pleasing shapes for the resulting curves. In Chapter 4, a new generalized slope estimation method is introduced and a new local automatic shape-preserving interpolation algorithm

based on this formula is constructed using C^1 cubic splines. The new algorithm is found to compare favourably with the best of the existing methods. In Chapter 5, the interactive curve building algorithms are developed with specific application to the shape-preserving problem. These algorithms are local and depend upon the slope estimation techniques involving a parameter which is used to control the size of the estimated slope. This, in turn, yields a more flexible tool for curve generation in computer graphics. The main work in this thesis is concerned with the study of shape-preserving algorithms for curve design and some of the results obtained for curves are then extended to the problem of surface design. In Chapter 6, we review some of the existing shape-preserving surface interpolation methods and describe some improvements. Finally, Chapter 7 presents the conclusions as well as some suggested future research.

1.3.1 Publications.

Parts of the work described in Chapter 3 and Chapter 6 of this thesis have appeared in Iqbal [41, 42]. Other parts will be submitted for publication in the near future.

1.4 Some Definitions.

The purpose of this section is to introduce some of the concepts that will be pertinent to this thesis. In these definitions we consider first the univariate case; that is, we assume that $(x_i, y_i), i = 1, ..., n$ is a given set of data points in a Cartesian co-ordinate system, where the x-values are monotonic increasing, that is, $x_1 < x_2, ..., < x_n$. It is also convenient to define the first-order divided differences.

$$\delta_i = \frac{y_{i+1} - y_i}{h_i}, \quad h_i = x_{i+1} - x_i, \quad i = 1, ..., n-1. \tag{1.1}$$

and the second-order divided differences

$$\Delta_i = \frac{\delta_{i+1} - \delta_i}{h_{i+1} + h_i}, \quad i = 1, ..., n-2. \tag{1.2}$$

The sign of δ_i is defined as

$$\text{sign}(\delta_i) = \begin{cases} \dfrac{\delta_i}{|\delta_i|}, & \text{if } \delta_i \neq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{1.3}$$

Definition 1.4.1

The univariate data set is said to be monotone increasing if the following condition holds:

$$\delta_i \geq 0, \ i = 1, ..., n-1 \tag{1.4}$$

and to be monotonic decreasing if

$$\delta_i \leq 0, \ i = 1, ..., n-1 \tag{1.5}$$

We say that the data set is strictly monotone if there is no equality in equations (1.4) and (1.5).

Definition 1.4.2

The univariate data set is said to be convex if we have

$$\delta_{i+1} \geq \delta_i, \ i = 1, ..., n-2 \tag{1.6}$$

and said to be concave if

$$\delta_{i+1} \leq \delta_i, \ i = 1, ..., n-2 \tag{1.7}$$

We say that the data set is strictly convex/concave when the inequalies (1.6) and (1.7) are strict. Note that under these definitions, straight line segments are considered to be both convex and concave.

Definition 1.4.3

An inflection point in the interval (x_i, x_{i+1}) is said to be extraneous if $\Delta_i \Delta_{i+1} > 0$. Perhaps most important of the several desirable properties of an interpolation curve is being able to control the extraneous points. Dealing with the problem of these inflection points is equivalent to preserving the convexity of the data. For example, Figure 1.2 shows the standard cubic spline interpolant that interpolates the data given in Table 1.1 for which all second-order differences are positive. The interpolant does not have a positive second derivative everywhere and instead introduces extraneous inflection points in the first three subintervals, $[-2.0, -1.0]$, $[-1.0, -0.5]$ and $[-0.5, -0.25]$.

The problem that we intend to address in the first part of this thesis is that of shape-preserving curve interpolation using spline interpolation. A piecewise interpolating function $p(x) \in C^1[x_1, x_n]$ is defined such that

$$p(x_i) = y_i, \quad p'(x_i) = d_i, \quad i = 1, ..., n. \tag{1.8}$$

where d_i are the derivative values at the endpoints of the subinterval. The interpolating function $p(x)$ is specified on the interval in terms of the data y_i, and the derivative values d_i at the endpoints of the subinterval. In order that the function $p(x)$ be monotonicity preserving on $[x_i, x_{i+1}]$, the sign of $p'(x)$ must agree with the sign of δ_i on $[x_i, x_{i+1}]$. We say that $p(x)$ is convexity preserving if $p'(x)$ is monotonic increasing in intervals in which the data are convex and $p'(x)$ is monotonic decreasing in intervals in which the data are concave. The interpolating function $p(x)$ is said to be shape-preserving if it is

both monotonicity and convexity preserving.

We end this section by introducing some important definitions for bivariate data sets. Let a given set of m by n data points in three dimensions be represented by $(x_i, y_j, f_{i,j})$, where $i = 1, ..., n$ and $j = 1, ..., m$. It is assumed that the independent variables are ordered ($x_1 < x_2 <, ... < x_n$ and $y_1 < y_2 <, ..., < y_m$) and form a rectangular grid, but are not necessarily equally spaced. Now we introduce some notation and definitions as follows .

$$\delta x_{i,j} = \frac{f_{i+1,j} - f_{i,j}}{x_{i+1} - x_i}, \quad i = 1, ..., n-1;\ j = 1, ..., m. \tag{1.9}$$

$$\delta y_{i,j} = \frac{f_{i,j+1} - f_{i,j}}{y_{j+1} - y_j}, \quad i = 1, ..., n;\ j = 1, ..., m-1. \tag{1.10}$$

Definition 1.4.4

The bivariate data is said to be monotone increasing (monotone decreasing) along the grid line $x = x_i$ if

$$\delta y_{i,j} \geq 0 \quad (\delta y_{i,j} \leq 0),\ j = 1, ..., m-1. \tag{1.11}$$

Definition 1.4.5

The bivariate data is said to be convex (concave) along the grid line $x = x_i$ if

$$\delta y_{i,j+1} \geq \delta y_{i,j} \quad (\delta y_{i,j+1} \leq \delta y_{i,j}),\ j = 1, ..., m-2. \tag{1.12}$$

Similarly, we can define the terms monotone increasing/decreasing, and convex/concave along the grid line $y = y_j$.

The problem we intend to address in the second part of this thesis is that of determining a surface which interpolates the given data and which preserves convexity and monotonicity along grid lines. Mathematically, the problem is that of finding a smooth bivariate function $P(x, y)$ with continuous first partial

derivatives such that

$$P(x_i, y_j) = f_{i,j}, \quad i = 1, ..., n; \; j = 1, ..., m. \tag{1.13}$$

subject to the shape-preserving constraints. The interpolant $P(x, y)$ is called shape-preserving if it is both monotonicity and convexity preserving on each grid line.

1.5 Definition of Test Problems.

Several sets of test data have appeared in the literature. In this thesis, we consider six data sets for univariate and two data sets for bivariate interpolation and all of them are either monotone and/or convex. These appear in Tables 1.3 through 1.10 and the corresponding data plots with data points marked with a circle (o) for the univariate cases are presented in Figures 1.6 through 1.11. The skeleton graphs of bivariate data sets are shown in Figure 1.12 and Figure 1.13, where the data points are connected by straight lines over a rectangular grid for display purposes. It is worthwhile to point out that there is no essential (theoretical) difference between monotone increasing and monotone decreasing data or between convex and concave data. Now, we characterize the data sets into following catagories.

1.5.1 Univariate Data Sets.

1.5.1.1 Monotone Data.

Data 1 (Table 1.3, Figure 1.6) is taken from Akima [2]. It has been used by several authors for comparison purposes.

Data 2 (Table 1.4, Figure 1.7) is used in Fritsch and Carlson [33] and represents data from Livermore radio-chemical calculations.

Data 3 (Table 1.5, Figure 1.8) is the third example reported in Pruess [53]. It represents data from a potentiometric titration calculation.

1.5.1.2 Monotone and Convex Data.

Data 4 (Table 1.6, Figure 1.9) is from McAllister, Passow and Roulier [49] and represents the convex function $\frac{1}{x^2}$ at $x = -2, -1, -0.3, -0.2$.

Data 5 (Table 1.7, Figure 1.10) comes from deBoor [7] and relates to a property of titanium as a function of temperature. This data is locally monotone and convex.

1.5.1.3 Convex Data.

Data 6 (Table 1.8, Figure 1.11) is used in Irvine, Marin and Smith [43] and comes from the convex function

$$\frac{1}{(0.05+x)(1.05-x)} \text{ at } x = 0.0, 0.1, 0.4, 0.7, 0.8, 1.0.$$

1.5.2 Bivariate Data Sets.

1.5.2.1 Convex Data.

Data 7 (Table 1.9, Figure 1.12) is obtained from Roulier [58] and is from the equation

$$f(x,y) = (x-3)^2 + (y-4)^2.$$

Data 8 (Table 1.10, Figure 1.13) is also provided by Roulier [58] and is from the equation

$$f(x,y) = 10 + 5\left(\frac{(y-4)^2}{16} - \frac{(x-4)^2}{9}\right).$$

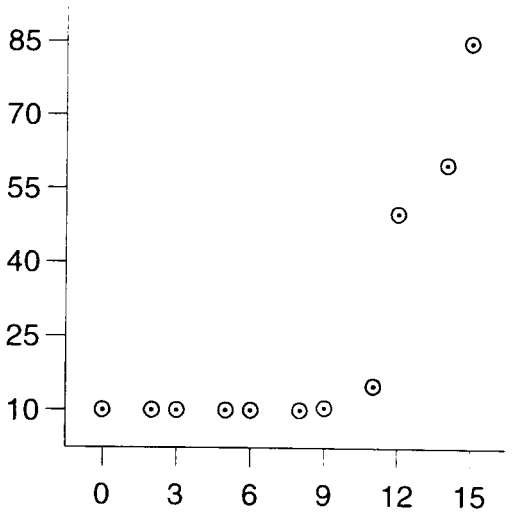

Figure 1.6 Graphic representation of Data 1.

x_i	0.0	2.0	3.0	5.0	6.0	8.0	9.0	11.0	12.0	14.0	15.0
y_i	10.0	10.0	10.0	10.0	10.0	10.0	10.5	15.0	50.0	60.0	85.0

Table 1.3 Monotone Data 1.

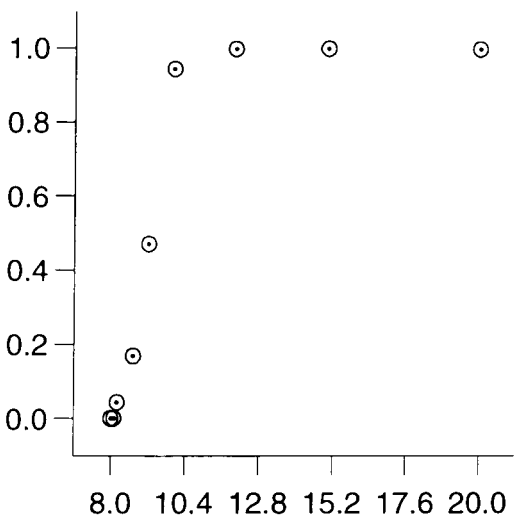

Figure 1.7 Graphic representation of Data 2.

x_i	7.99	8.09	8.19	8.70	9.20	10.0
y_i	0.0	2.76429E-05	4.37498E-02	0.169183	0.469428	0.943740
x_i	12.0	15.0	20.0			
y_i	0.998636	0.999919	0.999994			

Table 1.4 Monotone Data 2.

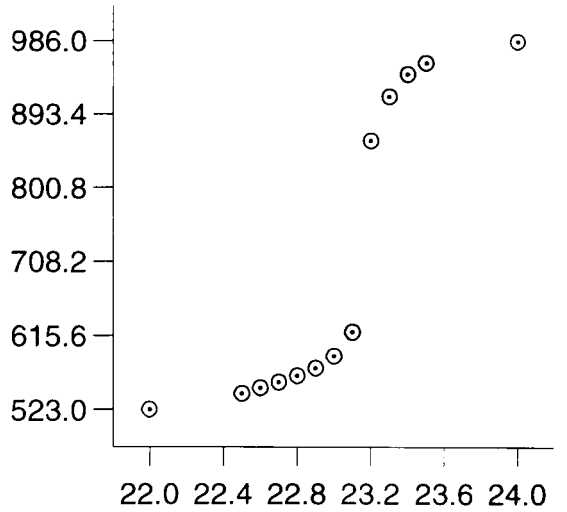

Figure 1.8 Graphic representation of Data 3.

x_i	22.0	22.5	22.6	22.7	22.8	22.9	23.0	23.1	23.2
y_i	523.0	543.0	550.0	557.0	565.0	575.0	590.0	620.0	860.0
x_i	23.3	23.4	23.5	24.0					
y_i	915.0	944.0	958.0	986.0					

Table 1.5 Monotone Data 3.

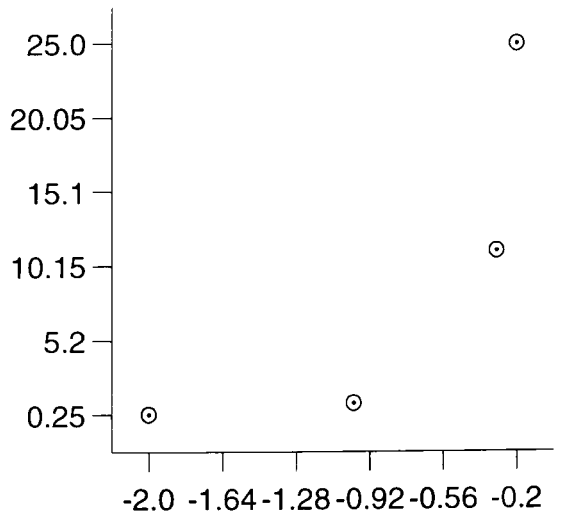

Figure 1.9 Graphic representation of Data 4.

x_i	-2.0	-1.0	-0.3	-0.2
y_i	0.2500	1.0000	11.1111	25.0000

Table 1.6 Monotone and convex Data 4.

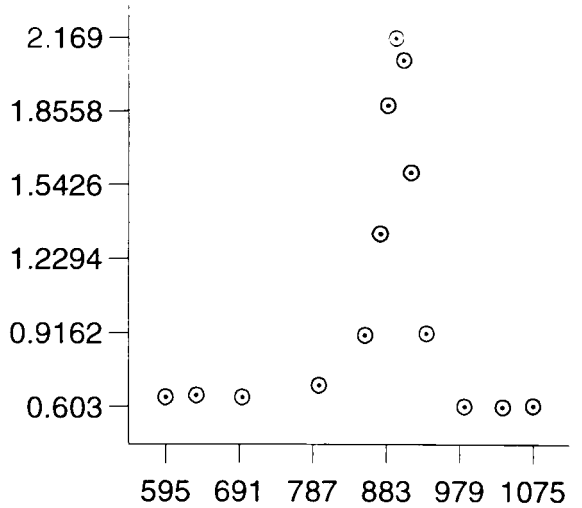

Figure 1.10 Graphic representation of Data 5.

x_i	595.0	635.0	695.0	795.0	855.0	875.0	885.0	895.0	905.0
y_i	0.644	0.652	0.644	0.694	0.907	1.336	1.881	2.169	2.075
x_i	915.0	935.0	985.0	1035.0	1075.0				
y_i	1.598	0.916	0.607	0.603	0.608				

Table 1.7 Locally monotone and convex Data 5.

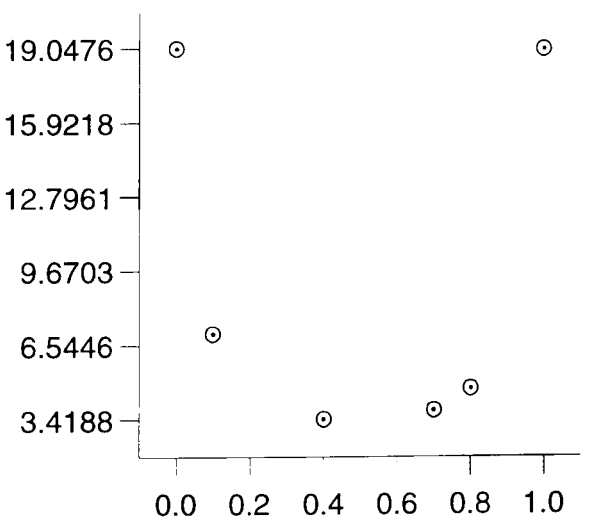

Figure 1.11 Graphic representation of Data 6.

x_i	0.0	0.1	0.4	0.7	0.8	1.0
y_i	19.0476	7.0175	3.4188	3.8095	4.7059	19.0476

Table 1.8 Convex Data 6.

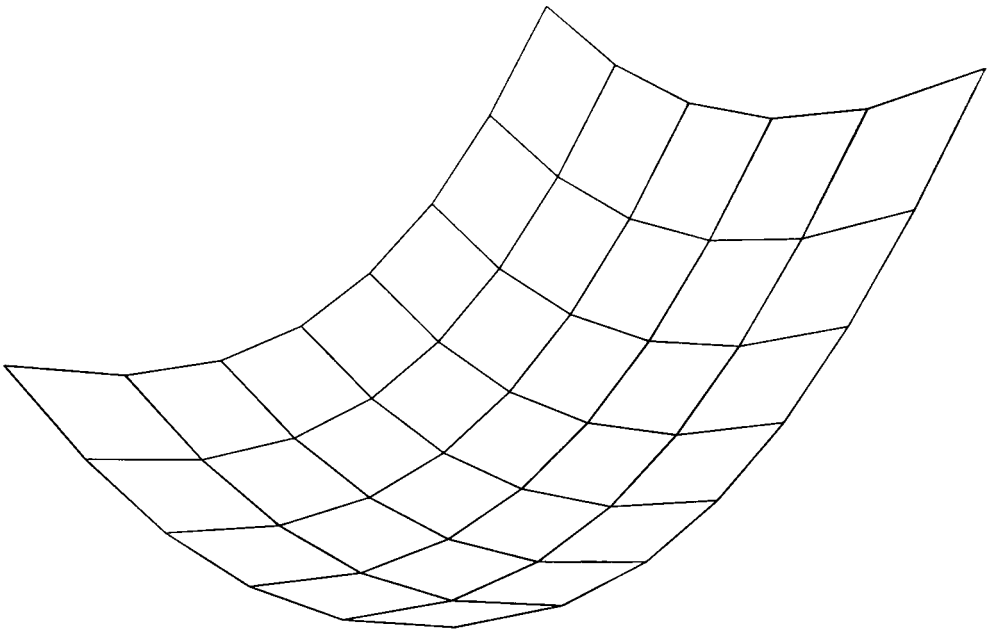

Figure 1.12 Graphic representation of Data 7 at the grid points.

X / Y	1.000	2.000	3.000	4.000	5.000	6.000
1.000	13.000	10.000	9.000	10.000	13.000	18.000
2.000	8.000	5.000	4.000	5.000	8.000	13.000
3.000	5.000	2.000	1.000	2.000	5.000	10.000
4.000	4.000	1.000	0.000	1.000	4.000	9.000
5.000	5.000	2.000	1.000	2.000	5.000	10.000
6.000	8.000	5.000	4.000	5.000	8.000	13.000
7.000	13.000	10.000	9.000	10.000	13.000	18.000
8.000	20.000	17.000	16.000	17.000	20.000	25.000

Table 1.9 Convex Data 7.

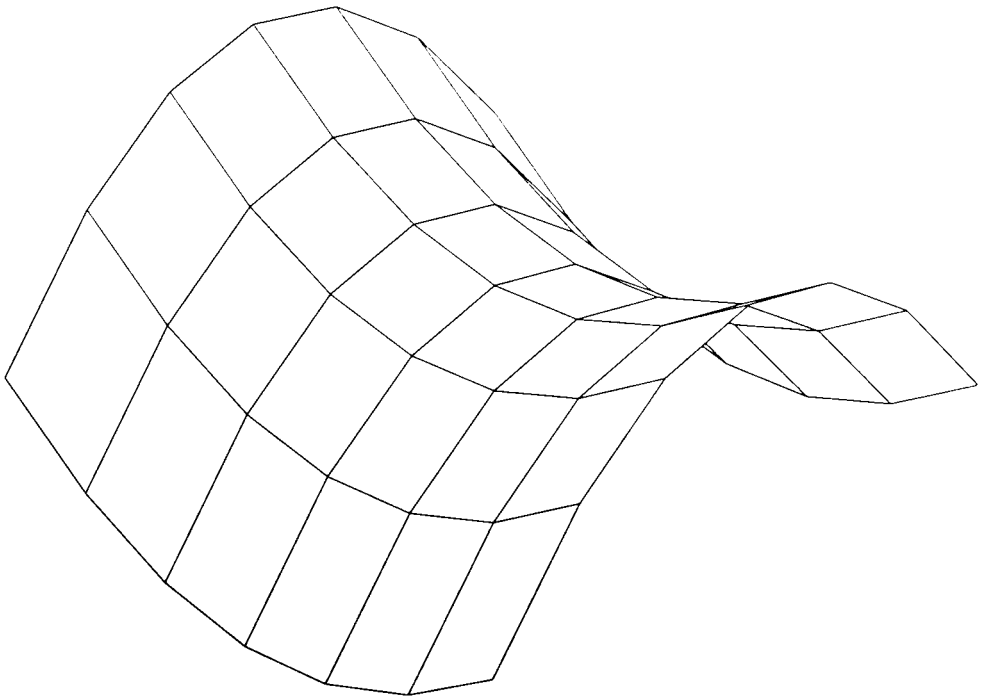

Figure 1.13 Graphic representation of Data 8 at the grid points.

X / Y	1.000	2.000	3.000	4.000	5.000	6.000	7.000
1.000	7.813	10.590	12.257	12.813	12.257	10.590	7.813
2.000	6.250	9.028	10.694	11.250	10.694	9.028	6.250
3.000	5.313	8.090	9.757	10.313	9.757	9.090	5.313
4.000	5.000	7.778	9.444	10.000	9.444	7.778	5.000
5.000	5.313	8.090	9.757	10.313	9.757	9.090	5.313
6.000	6.250	9.028	10.694	11.250	10.694	9.028	6.250
7.000	7.813	10.590	12.257	12.813	12.257	10.590	7.813

Table 1.10 Convex Data 8.

Chapter 2

Shape-Preserving Algorithms Using Cubic Splines

In this chapter we discuss the development of many effective local methods which use C^1 piecewise cubic splines to provide an additional opportunity to control the shape of an interpolatory curve. The detailed mathematical background of these methods may be found in the original articles. Here, we are mostly interested in the final form of the algorithm. The explicit form of the piecewise cubic Hermite interpolation, which is used to interpolate a set of data (x_i, y_i), $i = 1, ..., n$, is given in Section 2.1. The necessary and sufficient conditions for a C^1 cubic spline to preserve monotonicity are described in Section 2.2. The derivative values at the interpolation points may not be available as part of the data and one would be required to estimate these values before constructing an interpolant to a set of data. These values are usually specified in some manner in order to satisfy the shape related criteria. A review of some standard formulae for calculating the derivative values at the interpolation points is provided in Section 2.3. Various techniques for constructing shape-preserving curves with a specific order of continuity have been developed by a number of authors using piecewise cubic Hermite interpolants. Methods of this type are described in Section 2.4 through Section 2.7. Finally, conclusions and a number of experiments performed by applying different algorithms to several data sets are presented in Section 2.8.

2.1 Piecewise Cubic Hermite Interpolation.

The spline fit by its very nature is a global scheme as it requires the solution of a tridiagonal system. On the other hand, osculatory or Hermite interpolation provides a local means of interpolation. For this reason, Hermite interpolation is often preferred over spline interpolation. The local nature of this scheme is obtained at the expense of smoothness. For example, if we specify the derivatives, a cubic Hermite interpolant which is only C^1 as opposed to the C^2 smoothness provided by the standard cubic splines is achieved. Moreover, the required derivatives are typically not available and must themselves be estimated. With these provisos duly noted, we now proceed to discuss Hermite interpolation by cubic splines.

Let$(x_i, y_i), i = 1, ..., n$ be the given data set. A C^1 piecewise function $p(x)$ is constructed on $[x_1, x_n]$ with the following characteristics:

$$p(x_i) = y_i \quad \text{and} \quad p'(x_i) = d_i, \quad i = 1, ..., n \tag{2.1}$$

where for each $i = 1, ..., n-1$,

$$p(x) = y_i + d_i(x - x_i) + \frac{-2d_i - d_{i+1} + 3\delta_i}{x_{i+1} - x_i}(x - x_i)^2 + \frac{d_i + d_{i+1} - 2\delta_i}{(x_{i+1} - x_i)^2}(x - x_i)^3 \tag{2.2}$$

is a cubic polynomial interpolant defined on the subinterval $[x_i, x_{i+1}]$ and the d_i's are the approximations to the derivatives of y at x_i to be determined. The construction of $p(x)$ is then essentially based on the calculation of the derivative values $d_i, i = 1, ..., n$; and the process is known as piecewise cubic Hermite interpolation and the $p(x)$ in (2.2) is called the cubic Hermite interpolant in the subinterval $[x_i, x_{i+1}]$. We observe that $p(x)$ is determined

on each subinterval by four parameters. Two of these parameters are given by the interpolation constraint. The other two are specified in terms of the derivatives at the interpolation points.

In general, the slopes d_i are unknown and our problem can be seen as one of finding a formula to estimate these slopes so that the resulting approximation preserves the shape characteristics inherent in the data. Many slope calculation techniques are now available, amongst the earliest used is the Akima [2] method. In this method, the slope of the curve is determined locally at each given data point by a geometrical condition using coordinates of five data points. The data point in question is taken as a centre point with two neighbouring data points on each side. The slope at x_i is calculated as follows:

$$d_i = \begin{cases} \dfrac{\delta_{i-1} + \delta_i}{2}, & \text{if } \delta_{i-2} = \delta_{i-1} \quad \text{and} \quad \delta_i = \delta_{i+1} \\ \dfrac{|\delta_{i+1} - \delta_i|\delta_{i-1} + |\delta_{i-2} - \delta_{i-1}|\delta_i}{|\delta_{i+1} - \delta_i| + |\delta_{i-2} - \delta_{i-1}|}, & \text{otherwise.} \end{cases} \tag{2.3}$$

The slope d_i is the weighted average of the secant slopes δ_{i-1} and δ_i about x_i. When interpolation is made near the end points of the curve, two extra points are generated using an extrapolation technique. Akima does this by fitting a quadratic through the three end points (either x_1, x_2, x_3 or x_{n-2}, x_{n-1}, x_n) and reading off two extra points. These extra points can then be regarded as an extension of the data set for the purposes of applying the slope formula, enabling every slope to be calculated in the same way.

It is worthwhile to point out that the Akima [2] method usually does not preserve any shape characteristics present in the data such as monotonicity

and/or convexity, as it is not designed for such purposes. In order that the interpolants (2.2) should maintain shape-preserving properties the derivative values, d_i, must satisfy certain conditions. These conditions can be written in terms of restrictions on the derivative values d_i and d_{i+1} at the end points of the subinterval $[x_i, x_{i+1}]$ as a function of δ_i's. Thus, the problem is to determine a local method for finding the derivative values d_i which causes the conditions always to be satisfied. The conditions appropriate to particular methods are described below.

2.2 Fritsch and Carlson Method.

Several attempts have been made at finding an efficient and automatic method for constructing shape-preserving interpolants using quadratic splines. In the case of cubic splines, the major breakthrough was in the publication of the paper by Fritsch and Carlson [33]. In this paper, they consider monotone data and have shown necessary and sufficient conditions for a Hermite cubic interpolant to maintain monotonicity. In the case $\delta_i = 0$, the requirement that $p(x)$ be monotonicity preserving implies $d_i = d_{i+1} = 0$, and $p(x) = y_i$ in $[x_i, x_{i+1}]$. For $\delta_i \neq 0$, their conditions are defined in terms of $\alpha_i = \frac{d_i}{\delta_i}$ and $\beta_i = \frac{d_{i+1}}{\delta_i}$ where d_i are the derivatives to be determined. These conditions for monotone data sets can be summarized as follows:

Theorem 2.2.1.

If $\text{sign}(d_i) = \text{sign}(d_{i+1}) = \text{sign}(\delta_i)$, then $p(x)$ is monotone on $[x_i, x_{i+1}]$ if and only if either

(i) $\alpha_i + \beta_i - 2 \leq 0$ or

(ii) $\alpha_i + \beta_i - 2 > 0$ and

either

(ii.i) $2\alpha_i + \beta_i - 3 \leq 0$ or

(ii.ii) $\alpha_i + 2\beta_i - 3 \leq 0$ or

(ii.iii) $\alpha_i - \dfrac{(2\alpha_i + \beta_i - 3)^2}{3(\alpha_i + \beta_i - 2)} \geq 0.$

Fritsch and Carlson proved this theorem by considering the extremum of $p'(x)$. Conditions on α_i, β_i were derived so as to ensure that this extremum is forced outside of the interval $[x_i, x_{i+1}]$, or by considering necessary bounds if it does fall in the interval. The conditions α_i and β_i must satisfy in order to guarantee monotonicity are depicted in Figure 2.1, by the area shaded with dots. The region in which the conditions are satisfied is denoted by S and is in fact the finite region of the first quadrant of the $\alpha\beta$-plane bounded by the ellipse $\alpha_i^2 + \beta_i^2 + \alpha_i\beta_i - 6\alpha_i - 6\beta_i + 9 = 0$. In order that the interpolant $p(x)$

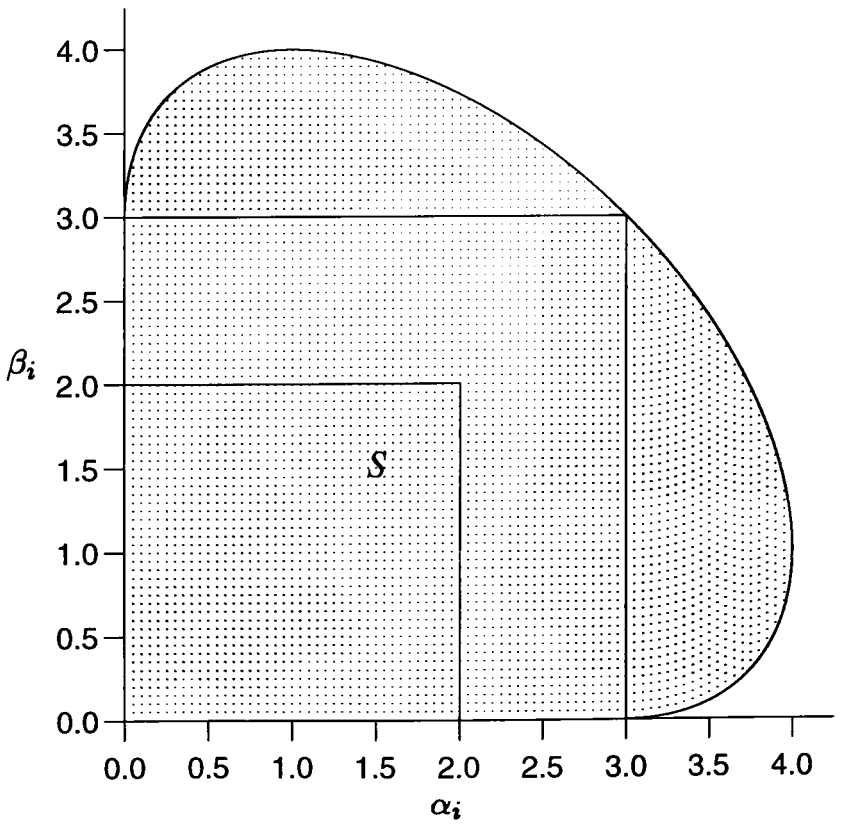

Figure 2.1 Monotonicity region S for cubic splines.

be monotone on $[x_i, x_{i+1}]$, the derivative values d_i and d_{i-1} must be chosen so that (α_i, β_i) fall within region S. If $(\alpha_i, \beta_i) \notin S$, then d_i and d_{i+1} are modified to satisfy appropriate conditions such that all $(\alpha_i, \beta_i) \in S$. Fritsch and Carlson suggest working with different subregions of S and amongst them, the following is an easier condition to satisfy:

$$0 \leq \alpha_i \leq 3, \quad 0 \leq \beta_i \leq 3 \tag{2.4}$$

The authors initially estimate the slopes d_i using a 3-point difference formula i.e., for $i = 2, ..., n-1$,

$$d_i = \begin{cases} \dfrac{h_i \delta_{i-1} + h_{i-1} \delta_i}{h_{i-1} + h_i}, & \text{if } \delta_{i-1} \neq 0 \text{ and } \delta_i \neq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

The derivatives at the end points x_1 and x_n are estimated using a non-centered 3-point difference formula. With these values of d_i, α_i and β_i are calculated for each segment and tested to check that they lie within the allowable monotonicity region S. If not, then the d_i are modified such that all $(\alpha_i, \beta_i) \in S$. Their process of modification requires two passes over the data and is essentially a non-local approach. The modification step is complicated and is dependent on the direction in which the data is scanned.

The success of the Fritsch-Carlson method has inspired a series of papers, all of which have abandoned the old one-pass techniques for the two-pass approach. Eisenstat, Jackson and Lewis [27] have subsequently analyzed some of these algorithms, which they term "fit and modify", and show that the Fritsch-Carlson method is only third-order accurate to an underlying C^3 monotone function. They also produce a fourth-order algorithm of their own.

2.3 Slope Estimation Methods.

In this section, several methods for estimating the derivative values d_i at the interpolation points are presented which directly satisfy condition (2.4). All the methods are local and calculate the derivative at a point using a mean of first-order divided differences about the point in question. In the following, δ_i and h_i are defined as in (1.1), and $i = 2, ..., n-1$.

2.3.1 Butland Method.

This is the method introduced by Butland [11]. The formula is the Harmonic mean between δ_{i-1} and δ_i:

$$d_i = \begin{cases} \dfrac{2\delta_{i-1}\delta_i}{\delta_{i-1}+\delta_i}, & \text{if } \delta_{i-1}\delta_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

This formula restricts the d_i's to points of (α_i, β_i) in Figure 2.1 within a square $[0,2]\times[0,2]$ and may not produce a visually pleasing curve because it fails to consider the relative spacing of the data points as noted in Fritsch and Butland [32].

2.3.2 Brodlie Method.

This procedure developed by Brodlie [9] is a modification of (2.6) and calculates the derivatives by taking account of the relative spacing of the data points:

$$d_i = \begin{cases} \dfrac{\delta_{i-1}\delta_i}{\lambda\delta_i + (1-\lambda)\delta_{i-1}}, & \text{if } \delta_{i-1}\delta_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

where $\lambda = \dfrac{1}{3}(1 + \dfrac{h_i}{h_{i-1}+h_i})$. Note that $\lambda = \dfrac{1}{2}$ gives the Butland formula (2.6).

2.3.3 Fritsch and Butland Method.

This is the method proposed by Fritsch and Butland [32]. The formula is a weighted Harmonic mean as follows:

$$d_i = \begin{cases} 0, & \text{if } \delta_{i-1}\delta_i \leq 0 \\ \dfrac{3\delta_{i-1}\delta_i}{\delta_i + 2\delta_{i-1}}, & \text{if } |\delta_{i-1}| \leq |\delta_i| \\ \dfrac{3\delta_{i-1}\delta_i}{\delta_{i-1} + 2\delta_i}, & \text{if } |\delta_i| \leq |\delta_{i-1}| \end{cases} \tag{2.8}$$

Both formulae given in (2.7) and (2.8) fulfill condition (2.4). For this choice of d_i, the values of α_i and β_i lie in the square $[0,3]\times[0,3]$, a larger subset of S than in the case for the Butland formula (2.6), allowing more visually pleasing curves to be produced (see, for example, Figure 2.31 and Figure 2.32).

2.3.4 Costantini Method.

This technique presented by Costantini [16] also employs the weighted Harmonic mean in which weights can be varied, subject to certain conditions, and the slopes are computed using the following formula:

$$d_i = \begin{cases} 0, & \text{if } \delta_{i-1}\delta_i \leq 0 \\ \dfrac{\rho(q,k)\delta_{i-1}\delta_i}{\delta_i + (\rho(q,k)-1)\delta_{i-1}}, & \text{if } |\delta_{i-1}| \leq |\delta_i| \\ \dfrac{\rho(q,k)\delta_{i-1}\delta_i}{\delta_{i-1} + (\rho(q,k)-1)\delta_i}, & \text{if } |\delta_i| \leq |\delta_{i-1}| \end{cases} \tag{2.9}$$

where

$$\rho(q,k) = \frac{\dfrac{q}{q-2k}\displaystyle\sum_{j=k}^{q-k-1}\binom{q-1}{j}}{\dfrac{2k}{q-2k}\displaystyle\sum_{j=k}^{q-k-1}\binom{q-1}{j} - 2\sum_{j=0}^{k-1}\binom{q-1}{j}}$$

and q, k are integers such that $0 < k < q - k$.

This formula satisfies the monotonicity condition (2.4) provided that $\rho(q,k) \leq 3$. In particular when $q = 3$ or $q = 4$ and $k = 1$, this corresponds to the Fritsch-Butland formula (2.8). When $q > 5$ and $\frac{q}{k} \leq 3$, then the values of (α_i, β_i) are always restricted to the smaller square $[0,2]\times[0,2]$ in Figure 2.1 and consequently tighter curves are produced. As an illustration, the curves drawn in Figure 2.2 are produced by using values of $q = 3, k = 1$; $q = 5, k = 2$; $q = 7, k = 3$ and $q = 11, k = 5$ in the formula (2.9) for Data 4. The corresponding values of α_i and β_i, as well as the jumps in the second derivative at the interior knots, denoted by J_i, are listed in Table 2.1 and Table 2.2 respectively. In Figure 2.2, clearly increasing the values of q and k give the effect of tightening the curve toward the straight line between the points

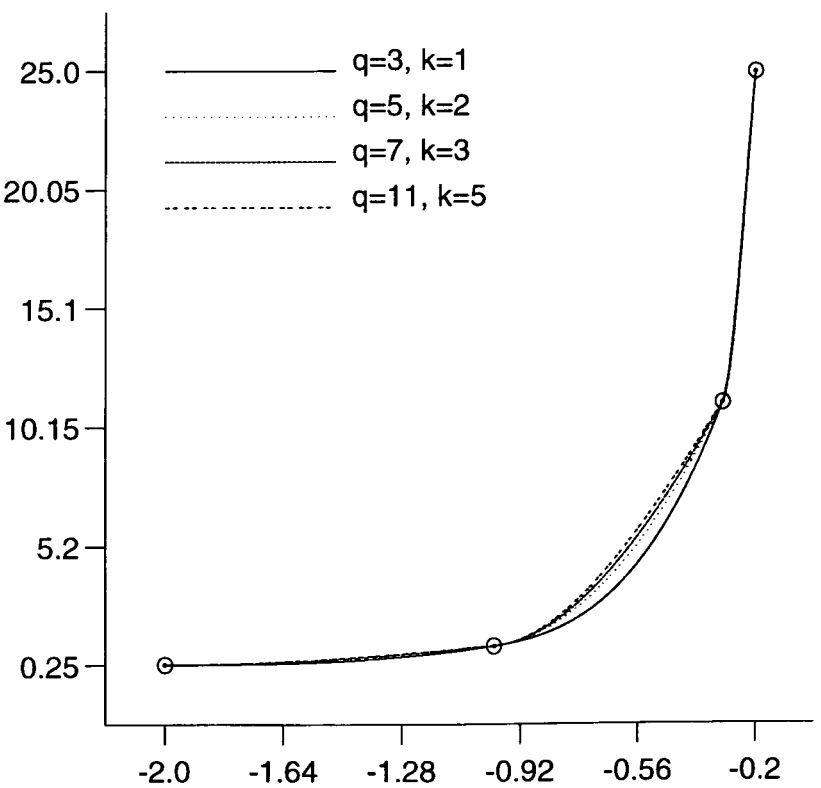

Figure 2.2 Costantini Method with different values of q and k.

	Data Points		$q=3,k=1$		$q=5,k=2$		$q=7,k=3$		$q=11,k=5$	
i	x_i	y_i	α_i	β_i	α_i	β_i	α_i	β_i	α_i	β_i
1	-2.0	0.25	0.00	2.72	0.00	2.02	0.00	1.76	0.00	1.54
2	-1.0	1.0	0.14	2.48	0.11	1.92	0.09	1.69	0.08	1.49
3	-0.3	11.1111	0.26	1.11	0.20	1.11	0.18	1.11	0.16	1.11
4	-0.2	25.0								

Table 2.1 The values associated with the curves in Figure 2.2.

	Data Points		$q=3,k=1$	$q=5,k=2$	$q=7,k=3$	$q=11,k=5$
i	x_i	y_i	J_i	J_i	J_i	J_i
1	-2.0	0.25				
2	-1.0	1.0	6.02	34.53	45.55	55.41
3	-0.3	11.1111	3722.57	4099.27	4246.05	4377.95
4	-0.2	25.0				

Table 2.2 Jumps in the second derivative at the interior knots.

(x_i, y_i) and (x_{i+1}, y_{i+1}), and move the values of (α_i, β_i) presented in Table 2.1 towards the origin of Figure 2.1. Note that for $q > 5$ and $\frac{q}{k} \leq 3$, these values fall within the smaller square $[0,2]\times[0,2]$ of the monotonicity region S, as observed by Fritsch and Butland [32], and the jump in the second derivative at the knots increases with the rise in the values of q and k as shown in Table 2.2. Larger values of α_i and β_i, within the square $[0,3]\times[0,3]$ of Figure 2.1, generate smaller discontinuities in the second derivative and produce the more visually pleasing shape of the curves. It is worthwhile to mention here that varying either q or k, while keeping the other fixed such that $\rho(q,k) \leq 3$, will alter the shape of the curve in a manner similar to that noticed in the case of the simultaneous variation of these parameters in Figure 2.2. The effects due to these operations on the shape of the curves are demonstrated in Figure 2.3 and Figure 2.4.

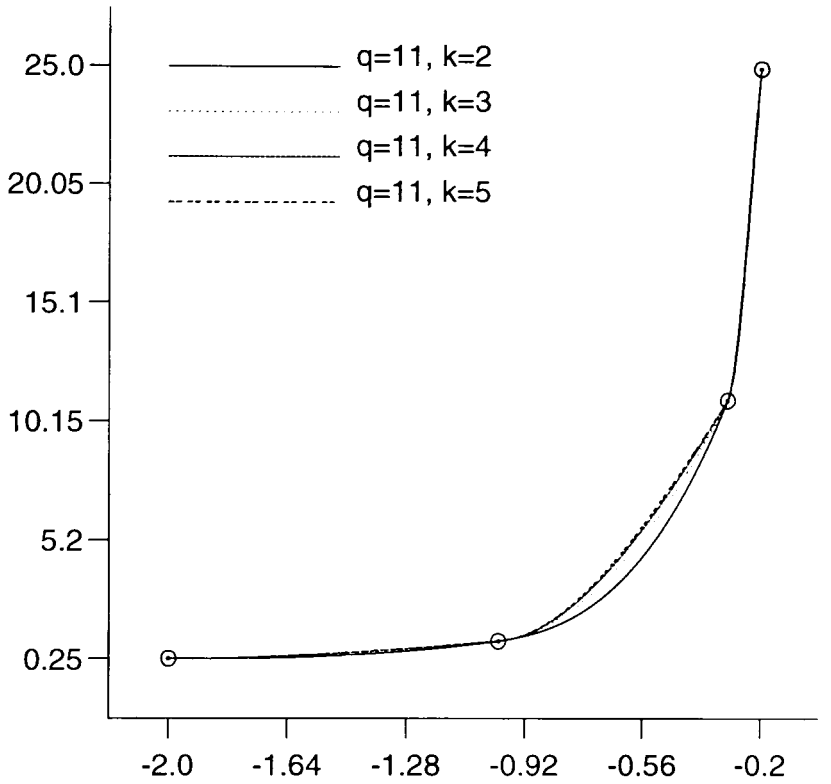

Figure 2.3 Costantini method with different values of k and fixed q.

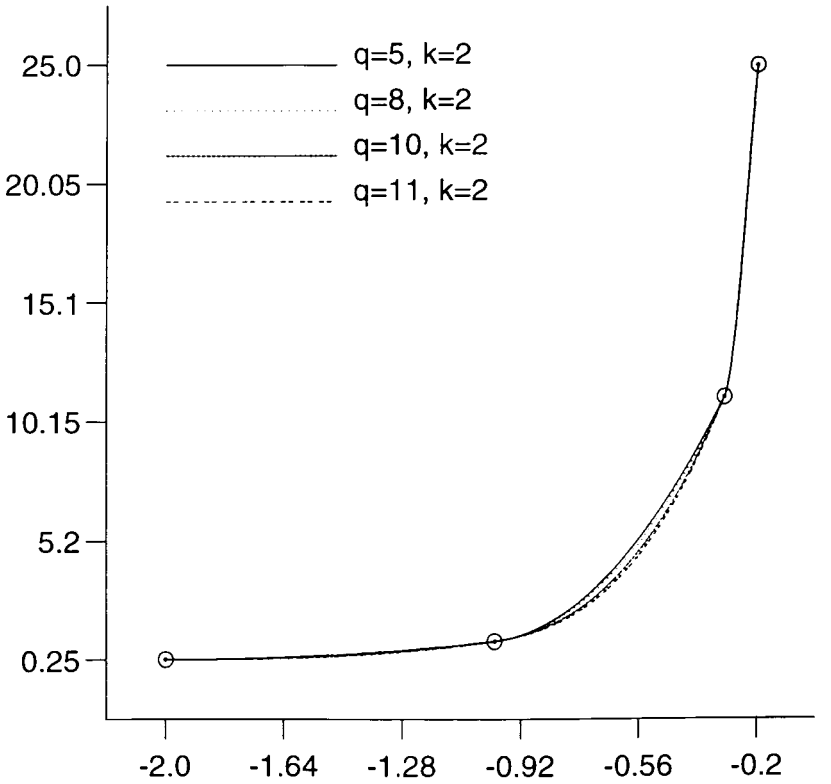

Figure 2.4 Costantini method with different values of q and fixed k.

2.3.5 Huynh Method.

Huynh [40] has described several effective slope calculation formulae by naming them as limiter functions and some of these yield the values of d_i such that condition (2.4) is directly satisfied. These formulae are described below:

$$d_i = \begin{cases} \min\Big(\max(\delta_{i-1}, \delta_i), 3\min(\delta_{i-1}, \delta_i)\Big), & \text{if } \delta_{i-1}\delta_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

$$d_i = \begin{cases} \min\Big(\dfrac{\delta_{i-1} + \delta_i}{2}, 3\min(\delta_{i-1}, \delta_i)\Big), & \text{if } \delta_{i-1}\delta_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

$$d_i = \begin{cases} \dfrac{3\delta_{i-1}\delta_i(\delta_{i-1} + \delta_i)}{\delta_{i-1}^2 + 4\delta_{i-1}\delta_i + \delta_i^2}, & \text{if } \delta_{i-1}\delta_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.12}$$

Formulae (2.10), (2.11) and (2.12) are called the Superbee, average and rational limiter respectively. It is worthwhile to mention here that, in practice, the formula (2.12) produces more visually pleasing curves than that of (2.10) and (2.11). In this chapter, unless otherwise stated, the Huynh method means formula (2.12) and its performance is compared with other existing methods in Section 2.8.

We see that all the formulae given in (2.6) through (2.12) satisfy the following condition

$$\min(\delta_{i-1}, \delta_i) \leq d_i \leq \max(\delta_{i-1}, \delta_i) \tag{2.13}$$

This means that the slope of the curves lies between the slopes of the adjacent data segments and, as a consequence, when the data is linear (i.e. where

$\delta_{i-1} = \delta_i$), then $d_i = \delta_{i-1}$. While this condition is not necessary for monotonicity, it seems intuitively reasonable as noted by Fritsch and Butland [32]. The standard cubic spline often exhibits unwanted oscillations due to the emergence of overshoots and/or extraneous inflection points. The above slope estimation methods ensure us that the C^1 cubic splines remedy this situation, for appropriate values of d_i chosen through them.

The slope estimation formulae described in (2.6) through (2.12) only provide derivative values d_i at interior data points. There are a variety of ways to determine the derivative values d_1 and d_n at the end points. We have used the non-centered 3-point difference formula, setting the result to zero if its sign does not agree with that of δ_1 (or δ_{n-1}). This procedure yields values of d_1 and d_n that fulfill the condition (2.4). Unless otherwise stated, these end conditions given by

$$d_1 = \begin{cases} 0, & \text{if } \delta_1 = 0 \text{ or } \operatorname{sign}(d_1) \neq \operatorname{sign}(\delta_1) \\ \delta_1 + \dfrac{h_1(\delta_1 - \delta_2)}{h_1 + h_2}, & \text{otherwise.} \end{cases} \tag{2.14}$$

$$d_n = \begin{cases} 0, & \text{if } \delta_{n-1} = 0 \text{ or } \operatorname{sign}(d_n) \neq \operatorname{sign}(\delta_{n-1}) \\ \delta_{n-1} + \dfrac{h_{n-1}(\delta_{n-1} - \delta_{n-2})}{h_{n-1} + h_{n-2}}, & \text{otherwise.} \end{cases} \tag{2.15}$$

are used for all curves shown in Section 2.8.

For completeness, we show here that, if $\delta_1 \neq 0$, sign(δ_1)=sign(δ_2), and d_1 is computed as described above, then $\alpha_1 = \dfrac{d_1}{\delta_1} < 2$, so that this method of

determining end derivative values is compatible with monotonicity preserving conditions at the extreme points. After simplification, from (2.14), we have:

$$\alpha_1 = \left(1 + \frac{h_1}{h_1 + h_2}\right) - \frac{h_1}{h_1 + h_2}\left(\frac{\delta_2}{\delta_1}\right)$$

If δ_2 is too large compared to δ_1, this can be negative, which is why the adjustment to zero may be needed. However, we are interested only in an upper bound and assuming δ_1 and δ_2 have the same sign, we have:

$$\alpha_1 \leq 1 + \frac{h_1}{h_1 + h_2}$$

Since h_1 and h_2 are assumed to be positive, $h_1 < (h_1 + h_2)$, so that $\alpha_1 < 2$, as claimed.

2.4 Yan Method.

The method developed by Yan [67] considers the interpolation problem $p(x_i) = y_i, i = 1, ..., n$ by constructing the interpolant as a C^1 cubic spline with knots at the data points $x_1, ..., x_n$ and with two extra knots inserted in those subintervals at which the points $(\alpha_i, \beta_i) \notin S$, in order to preserve monotonicity of the data. It shows when it is necessary to add knots to a subinterval and where they should be placed.

In contrast to the Fritsch and Carlson [33] method, Yan's method does not modify the chosen values of d_i. Instead, when (α_i, β_i) does not lie in S, rather than changing the slopes, two extra knots $\xi_{i,1}$ and $\xi_{i,2}$ such that $x_i < \xi_{i,1} \leq x^* \leq \xi_{i,2} < x_{i+1}$ are inserted in the subinterval $[x_i, x_{i+1}]$, where

$$x^* = x_i + h_i \frac{(2\alpha_i + \beta_i - 3)}{3(\alpha_i + \beta_i - 2)}$$

is the unique extremum of the quadratic which is obtained by differentiating the cubic (2.2) in the subinterval $[x_i, x_{i+1}]$.

Let $\delta_i \neq 0$ and define

$$\mu = x^* - x_i, \quad \eta = x_{i+1} - x^*, \quad \varpi = p'(x^*) \tag{2.16}$$

It is clear that $p(x)$ is not monotone on $[x_i, x_{i+1}]$ if and only if

$$0 < \mu,\ \eta < h_i \quad \text{and} \quad \delta_i \varpi < 0 \tag{2.17}$$

In this case, the new interpolant on $[x_i, x_{i+1}]$ is chosen to have a derivative $\tilde{p}'(x)$ of the following form:

$$\tilde{p}'(x) = \begin{cases} a_1(x - \xi_{i,1})^2 + b, & x \in [x_i, \xi_{i,1}] \\ b, & x \in [\xi_{i,1}, \xi_{i,2}] \\ a_2(x - \xi_{i,2})^2 + b, & x \in [\xi_{i,2}, x_{i+1}] \end{cases} \tag{2.18}$$

where a_1, a_2 and b are the constants in which b is chosen as zero (see Section 2.5 for the case where $b \neq 0$) and are determined in such a way that $\tilde{p}(x)$ satisfies all interpolation and monotonicity requirements.

By imposing the continuity and interpolation conditions on (2.18), and after simple calculations, the expressions for the additional knots $\xi_{i,1}$ and $\xi_{i,2}$ are given by

$$\xi_{i,1} = x_i + 3\mu \frac{(y_{i+1} - y_i)}{d_i\mu + d_{i+1}\eta} \quad \text{and} \quad \xi_{i,2} = x_{i+1} - 3\eta \frac{(y_{i+1} - y_i)}{d_i\mu + d_{i+1}\eta}$$

The interpolant $p(x)$ is modified so that it becomes a C^1 cubic with respect to the expanded set of knots and preserves the monotonicity of the data. The

shape of the interpolant becomes flat between the additional inserted knots, that is, on the interval $[\xi_{i,1}, \xi_{i,2}]$, preserving the monotonicity of the data. An algorithm is described briefly as follows:

Algorithm.

Step 1

Calculate the slopes d_i at x_i using 4-point difference formula as

For i = 2 to n-2 do

$$d_i = \delta_{i-1} + h_i \cdot \frac{\delta_i - \delta_{i-1}}{h_i + h_{i-1}} + h_i h_{i-1} \cdot \frac{\dfrac{\delta_i - \delta_{i-1}}{h_i + h_{i-1}} - \dfrac{\delta_{i+1} - \delta_i}{h_{i+1} + h_i}}{h_{i+1} + h_i + h_{i-1}}$$

At the end points, for $i = 1, n$, use non-centered 4-point difference formula to compute d_i, which uses the value of y_i at the four points nearest to x_i.

Step 2

Guarantee that no two of d_i, d_{i+1}, δ_i are of opposite signs: if $d_i \delta_i \leq 0$, then set $d_i = 0$ and if $d_{i+1} \delta_i \leq 0$, then set $d_{i+1} = 0$.

Step 3

For each subinterval $[x_i, x_{i+1}]$ in which $(\alpha_i, \beta_i) \in S$, represent $p(x)$ as cubic Hermite polynomial (2.2) with slopes d_i, d_{i+1} and which is monotone on $[x_i, x_{i+1}]$.

Step 4

If $(\alpha_i, \beta_i) \notin S$, then choose extra knots $\xi_{i,1}$ and $\xi_{i,2}$ as proposed above and defined $p(x)$ as

$$p(x) = \begin{cases} \dfrac{a_1(x-\xi_{i,1})^3}{3} + c, & x \in [x_i, \xi_{i,1}] \\ c, & x \in [\xi_{i,1}, \xi_{i,2}] \\ \dfrac{a_2(x-\xi_{i,2})^3}{3} + c, & x \in [\xi_{i,2}, x_{i+1}] \end{cases}$$

where $a_1 = \dfrac{d_i}{(x_i - \xi_{i,1})^2}$, $\quad a_2 = \dfrac{d_{i+1}}{(x_{i+1} - \xi_{i,2})^2}$ and

$c = y_i - \dfrac{d_i(x_i - \xi_{i,1})}{3} = y_{i+1} - \dfrac{d_{i+1}(x_{i+1} - \xi_{i,2})}{3}$.

Then $p(x)$ is also monotone on $[x_i, x_{i+1}]$.

2.5 Gasparo and Morandi Method.

The method proposed by Gasparo and Morandi [34] is very similar to the Yan [67] method except that the former draws a straight line between the additional inserted knots by imposing $b \neq 0$ in (2.18), as opposed to the latter which always selects b as zero and yields a constant value throughout the interval $[\xi_{i,1}, \xi_{i,2}]$. The process of finding the extra knots is achieved by integrating $\tilde{p}'(x)$ on $[x_i, x_{i+1}]$ and imposing the interpolation conditions. Then $\xi_{i,1}$ and $\xi_{i,2}$ are calculated as

$$\xi_{i,1} = x_i + \tilde{\mu} \quad \text{and} \quad \xi_{i,2} = x_{i+1} - \tilde{\eta}$$

where

$$\tilde{\mu} = \rho(b)\mu, \quad \tilde{\eta} = \rho(b)\eta \quad \rho(b) = \frac{3h_i(\delta_i - b)}{d_i\mu + d_{i+1}\eta - bh_i}.$$

In order to determine b, the authors suggest the formula

$$b = \varphi.\text{sign}(\delta_i)\min(|\varpi|, |\delta_i|)$$

where $\varphi \in [0, 1)$ is a given real number and the slope b in the interval $[x_i, x_{i-1}]$ can be varied by choosing different values of φ, which in turn, may lead to better results. An algorithm is given as follows:

Algorithm.

Here Steps 1, 2 and 3 are the same as given in the Yan method, except Step 4 which is outlined as.

Step 4

If $(\alpha_i, \beta_i) \notin S$, then add extra knots $\xi_{i,1}$ and $\xi_{i,2}$ as mentioned above and define

$$p(x) = \begin{cases} \dfrac{a_1(x - \xi_{i,1})^3}{3} + bx + c, & x \in [x_i, \xi_{i,1}] \\ bx + c, & x \in [\xi_{i,1}, \xi_{i,2}] \\ \dfrac{a_2(x - \xi_{i,2})^3}{3} + bx + c, & x \in [\xi_{i,2}, x_{i+1}] \end{cases}$$

where $a_1 = \dfrac{d_i - b}{(x_i - \xi_{i,1})^2}$, $\quad a_2 = \dfrac{d_{i+1} - b}{(x_{i+1} - \xi_{i,2})^2}$ and

$$c = y_i - bx_i - \frac{(d_i - b)(x_i - \xi_{i,1})}{3} = y_{i+1} - bx_{i+1} - \frac{(d_{i+1} - b)(x_{i+1} - \xi_{i,2})}{3}.$$

Then $p(x)$ is also monotone on $[x_i, x_{i+1}]$.

The above algorithms described by Yan and Gasparo-Morandi give us a fourth-order approximation to monotone functions. We should point out that when the slopes for the data change abruptly from large to small values, the

curves produced by both algorithms change quickly due to the high order of convergence and are not as pleasing as those produced by other algorithms (see Section 2.8).

2.6 Beatson and Wolkowicz Method.

Beatson and Wolkowicz [5] have described methods which preserve monotonicity of the data by extending the monotonicity region S into a superset Ψ which consists of union of monotonicity region S with the squares $[0,1]\times[3,4]$ and $[3,4]\times[0,1]$. Here, when $(\alpha_i, \beta_i) \notin S$, then α_i, β_i are projected onto the superset of the monotonicity region for each subinterval (x_i, x_{i+1}) and then an extra knot is added where necessary to allow monotonicity without further modifying the derivative values d_i at the data points. The authors have presented two algorithms but here we only consider the second algorithm which provides more visually pleasing curves than the first. The basis of the method rests on the result of a lemma and a relaxation function $g(x)$ such that $g(x) \leq x$ for all $x \in [0,1]$ and $(1-g(x))/(1-x)$ is bounded on $[0,1)$. Suppose we are given a set of monotone data and a number χ such that $1 \leq \chi \leq 2$. An algorithm implementing the method is outlined briefly as follows:

Algorithm

Step 1

Calculate the derivative values d_i using standard C^2 cubic spline interpolation with 'not-a-knot' end conditions.

Step 2

For i = 1 to n do

Correct the sign of derivative, that is, if $d_i\delta_i < 0$, then set $d_i = -d_i$

Step 3

For each subinterval $[x_i, x_{i+1}]$ in which $(\alpha_i, \beta_i) \in S$, represent $p(x)$ as cubic Hermite polynomial (2.2) with slopes d_i, d_{i+1} and which is monotone on $[x_i, x_{i+1}]$.

Step 4

For i = 1 to n-1 do

If $(\alpha_i, \beta_i) \notin S$, then perform a relaxed projection of (α_i, β_i) onto Ψ by computing $\nu > 0$ such that $(1,1) + \nu(\alpha_i - 1, \beta_i - 1)$ lie on the boundary of S and define a relaxation function $g(\nu)$ as

$$g(\nu) = \begin{cases} \nu/2, & \nu < \frac{2}{3} \\ 2\nu - 1, & \nu \geq \frac{2}{3}. \end{cases}$$

If $\alpha_i \geq 1$ and $\beta_i \geq 1$ Then

$$\alpha_i = 1 + g(\nu)(\alpha_i - 1)$$

$$\beta_i = 1 + g(\nu)(\beta_i - 1)$$

Else

If $\alpha_i < 1$ Then

$$\beta_i = 1 + g(\nu)(\beta_i - 1)$$

Else

$$\alpha_i = 1 + g(\nu)(\alpha_i - 1)$$

Step 5

For i = 1 to n-1 do

$$\Theta = \frac{2\alpha_i + \beta_i - 3}{3(\alpha_i + \beta_i - 2)}$$

If $(\alpha_i, \beta_i) \notin S$, then choose an additional knot $\xi_{i,1}$ as

If $\alpha_i < 1$ then

$$\xi_{i1} = x_i + 2h_i\Theta$$

Calculate $p(\xi_{i,1})$ and $p'(\xi_{i,1})$ using (2.2) and define

$$p(\xi_{i,1}) = p(\xi_{i,1}) + \tfrac{4}{3}h_i\delta_i\chi\Theta\Big(\Theta(2\alpha_i + \beta_i - 3) - \alpha_i\Big)$$

$p'(\xi_{i,1})=p'(\xi_{i,1})$

Else

$\xi_{i1} = x_{i+1} - 2h_i(1-\Theta)$

Compute $p(\xi_{i,1})$ and $p'(\xi_{i,1})$ using (2.2) and set

$p(\xi_{i,1}) = p(\xi_{i,1}) - \frac{4}{3}h_i\delta_i\chi(1-\Theta)\Big(\Theta(2\alpha_i+\beta_i-3)-\alpha_i\Big)$

$p'(\xi_{i,1})=p'(\xi_{i,1})$

Step 6

If $(\alpha_i,\beta_i) \notin S$, then choose an extra knot ξ_{i1} as proposed above and modify the original piecewise single cubic spline on $[x_i, x_{i+1}]$ into two cubic splines on $[x_i, \xi_{i,1}]$ and $[\xi_{i,1}, x_{i+1}]$.

This algorithm becomes completely local if the derivative values d_i in Step 1 are replaced with the 4-point difference formula as given in Step 1 of the Yan method.

2.7 Pruess Method.

In contrast to the above methods, Pruess [54] presents a local C^2 cubic spline method which preserves monotonicity and convexity of the data by dividing each data subinterval $[x_i, x_{i+1}]$ into three pieces using two additional knots $\xi_{i,1}$ and $\xi_{i,2}$ such that $x_i < \xi_{i,1} < \xi_{i,2} < x_{i+1}$. The additional knots are constructed as follows:

$$\xi_{i,1} = x_i + \sigma_i h_i, \quad \sigma_i \in (0,1/2) \quad \text{and} \quad \xi_{i,2} = x_{i+1} - \tau_i h_i, \quad \tau_i \in (0,1/2)$$

A piecewise interpolating function $p(x) \in C^2[x_1, x_n]$ is then defined such that

$$p(x_i) = y_i, \quad p'(x_i) = d_i \quad \text{and} \quad p''(x_i) = d_i', \quad i = 1, ..., n.$$

where d_i and $d_i^{'}$ are the first and second derivative values at the endpoints of the subinterval $[x_i, x_{i+1}]$ respectively. Now $p(x)$ is defined on each $[x_i, x_{i-1}]$ as

$$p(x) = \begin{cases} y_i + d_i(x - x_i) + d_i^{'}\dfrac{(x - x_i)^2}{2} \\ \qquad\qquad + p^{'''}(x_i^+)\dfrac{(x - x_i)^3}{6}, & x \in [x_i, \xi_{i,1}], \\ p(\xi_{i,1}) + p^{'}(\xi_{i,1})(x - \xi_{i,1}) + p^{''}(\xi_{i,1})\dfrac{(x - \xi_{i,1})^2}{2}, \\ \qquad\qquad + p^{'''}(\xi_{i,1}^+)\dfrac{(x - \xi_{i,1})^3}{6}, & x \in [\xi_{i,1}, \xi_{i,2}], \\ p(\xi_{i,2}) + p^{'}(\xi_{i,2})(x - \xi_{i,2}) + p^{''}(\xi_{i,2})\dfrac{(x - \xi_{i,2})^2}{2} \\ \qquad\qquad + p^{'''}(\xi_{i,2}^+)\dfrac{(x - \xi_{i,2})^3}{6}, & x \in [\xi_{i,2}, x_{i+1}] \end{cases} \tag{2.19}$$

where

$$p(\xi_{i,1}) = y_i + \sigma_i h_i d_i + \left[2d_i^{'} + p^{''}(\xi_{i,1})\right]\frac{(\sigma_i h_i)^2}{6} \tag{2.20}$$

$$p(\xi_{i,2}) = y_{i+1} - \tau_i h_i d_{i+1} + \left[2d_{i+1}^{'} + p^{''}(\xi_{i,2})\right]\frac{(\tau_i h_i)^2}{6} \tag{2.21}$$

$$\begin{aligned} p^{''}(\xi_{i,1}) = \big[&6\delta_i - 2(2 + \sigma_i - \tau_i)d_i - 2(1 - \sigma_i + \tau_i)d_{i+1} \\ &- \sigma_i h_i(2 - \tau_i)d_i^{'} + \tau_i h_i(1 - \sigma_i)d_{i+1}^{'}\big]\frac{1}{h_i(1 - \tau_i)} \end{aligned} \tag{2.22}$$

$$\begin{aligned} p^{''}(\xi_{i,2}) = \big[&-6\delta_i + 2(1 + \sigma_i - \tau_i)d_i + 2(2 - \sigma_i + \tau_i)d_{i+1} \\ &+ \sigma_i h_i(1 - \tau_i)d_i^{'} - \tau_i h_i(2 - \sigma_i)d_{i+1}^{'}\big]\frac{1}{h_i(1 - \sigma_i)} \end{aligned} \tag{2.23}$$

$$p^{'}(\xi_{i,1}) = d_i + \left[d_i^{'} + p^{''}(\xi_{i,1})\right]\frac{\sigma_i h_i}{2} \tag{2.24}$$

$$p^{'}(\xi_{i,2}) = d_{i+1} - \left[d_{i+1}^{'} + p^{''}(\xi_{i,2})\right]\frac{\tau_i h_i}{2} \tag{2.25}$$

$$p^{'''}(x_i^+) = \frac{p^{''}(\xi_{i,1}) - d_i^{'}}{\sigma_i h_i}$$

$$p^{'''}(\xi_{i,1}^+) = \frac{p^{''}(\xi_{i,2}) - p^{''}(\xi_{i,1})}{(1 - \sigma_i - \tau_i)h_i}$$

$$p'''(\xi_{i,2}^{+}) = \frac{d_{i+1}^{'} - p''(\xi_{i,2})}{\tau_i h_i}$$

$p(x)$ is then in $C^2[x_1, x_n]$ for any choice of d_i and $d_i^{'}$, $i = 1, ..., n$ and for any choice of σ_i and τ_i, $i = 1, ..., n-1$.

In order to derive the sufficient conditions attached to preserving local monotonicity and convexity, the author has, for simplicity, assumed $\sigma_i = \tau_i$ for each interval. It is convenient to define

$$\gamma_i = \text{sign}(\delta_i), \quad i = 1, ..., n-1.$$

$$\psi_{i,1} = \text{sign}(d_i), \quad \psi_{i,2} = \text{sign}(d_i^{'}), \quad i = 1, ..., n.$$

These signs are defined as in definition 1.3 of Chapter 1 and determined computationally as

$$\psi_{i,1} = \text{sign}\Big(\frac{h_{i-1}\delta_i + h_i\delta_{i-1}}{h_i + h_{i-1}}\Big) \quad \psi_{i,2} = \text{sign}(\delta_i - \delta_{i-1}), \quad i = 2, ..., n-1. \tag{2.26}$$

and at the end points $\psi_{1,1} = \psi_{2,1}$, $\psi_{n,1} = \psi_{n-1,1}$, $\psi_{1,2} = \psi_{2,2}$ and $\psi_{n,2} = \psi_{n-1,2}$.

When $\sigma_i = \tau_i$, $i = 1, ..., n-1$, the sufficient conditions for monotonicity are obtained as follows:

$$\gamma_i d_i \geq 0, \quad \gamma_i d_{i+1} \geq 0, \tag{2.27}$$

$$\Big[-\tau_i(2 - 4\tau_i + \tau_i^2)h_i d_i^{'} - \tau_i^2(1 - \tau_i)h_i d_{i+1}^{'}\Big]\gamma_i \leq \Big[6\tau_i\delta_i + 2(2 - 4\tau_i)d_i - 2\tau_i d_{i+1}\Big]\gamma_i \tag{2.28}$$

$$\Big[-\tau_i(1-3\tau_i+\tau_i^2)h_i d_i^{'} - \tau_i^2(1-\tau_i)h_i d_{i+1}^{'}\Big]\gamma_i \leq \Big[6\tau_i\delta_i + 2(1-3\tau_i)d_i - 2\tau_i d_{i+1}\Big]\gamma_i \tag{2.29}$$

$$-\tau_i h_i d_i^{'}\gamma_i \leq 4d_i\gamma_i \tag{2.30}$$

$$\Big[\tau_i(1-\tau_i)h_i d_i^{'} - \tau_i^2 h_i d_{i+1}^{'}\Big]\gamma_i \leq \Big[6\delta_i - 2d_i - 4\tau_i d_{i+1}\Big]\gamma_i \tag{2.31}$$

$$\Big[\tau_i^2 h_i d_i^{'} - \tau_i(1-\tau_i)h_i d_{i+1}^{'}\Big]\gamma_i \leq \Big[6\delta_i - 4\tau_i d_i - 2d_{i+1}\Big]\gamma_i \tag{2.32}$$

$$\tau_i h_i d_{i+1}^{'}\gamma_i \leq 4d_{i+1}\gamma_i \tag{2.33}$$

$$\Big[\tau_i^2(1-\tau_i)h_i d_i^{'} + \tau_i(1-3\tau_i+\tau_i^2)h_i d_{i+1}^{'}\Big]\gamma_i \leq \Big[6\tau_i\delta_i - 2\tau_i d_i + 2(1-3\tau_i)d_{i+1}\Big]\gamma_i \tag{2.34}$$

$$\Big[\tau_i^2(1-\tau_i)h_i d_i^{'} + \tau_i(2-4\tau_i+\tau_i^2)h_i d_{i+1}^{'}\Big]\gamma_i \leq \Big[6\tau_i\delta_i - 2\tau_i d_i + 2(2-4\tau_i)d_{i+1}\Big]\gamma_i \tag{2.35}$$

If conditions (2.27) through (2.35) hold true, then $\text{sign}\Big(p^{'}(x)\Big) = \text{sign}(\delta_i)$ for each $x \in [x_i, x_{i+1}]$.

Similarly, the sufficient conditions given by Pruess for convexity preserving are:

$$\text{sign}\Big(p^{''}(\xi_{i-1,2})\Big) = \text{sign}\Big(p^{''}(x_i)\Big) = \text{sign}\Big(p^{''}(\xi_{i,1})\Big), \quad i = 2, ..., n-1. \tag{2.36}$$

or, equivalently, from (2.22) with $\sigma_i = \tau_i$, $p^{''}(\xi_{i,1})$ has the correct sign if and only if

$$\Big[4d_i + 2d_{i+1} + h_i d_i^{'}\tau_i(2-\tau_i) - h_i d_{i+1}^{'}\tau_i(1-\tau_i)\Big]\psi_{i,2} \leq 6\delta_i\psi_{i,2}. \tag{2.37}$$

Similarly, from (2.23) with $\sigma_i = \tau_i$, $p^{''}(\xi_{i,2})$ has the correct sign if and only if

$$\Big[2d_i + 4d_{i+1} + h_i d_i^{'}\tau_i(1-\tau_i) - h_i d_{i+1}^{'}\tau_i(2-\tau_i)\Big]\psi_{i+1,2} \geq 6\delta_i\psi_{i+1,2}. \tag{2.38}$$

To preserve monotonicity, some feasible techniques for choosing the free parameters d_i and d_i' are needed and for this purpose, if we choose the slope d_i satisfying the monotonicity condition (2.4), then the inequalities (2.27) through (2.35) are satisfied for any choice of d_i' as long as τ_i is sufficiently small. Pruess suggests to use $\tau_i \leq 1/3$ and the formula for d_i', $i = 1, ..., n$, as follows:

$$d_i' = \begin{cases} \psi_{i,2} \min\left(\frac{\psi_{i,2} R_{2i-1}}{h_i}, \frac{\psi_{i,2} R_{2i-2}}{h_{i-1}}\right), & \text{if } R_{2i-1} \text{ and } R_{2i-2} > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.39}$$

where

$$R_{2i-1} = 6\delta_i - 4d_i - 2d_{i+1}, \quad R_{2i} = 2d_i + 4d_{i+1} - 6\delta_i, \quad i = 1, ..., n-1.$$

Now, we consider the convexity preserving case which is more complicated than the monotonicity preserving case. We first consider a reduced case by taking $\tau_i = 0$. The inequalities (2.37) and (2.38) in terms of R's can then be written as:

$$\psi_{i,2} R_{2i-1} \geq 0 \tag{2.40}$$

$$\psi_{i,2} R_{2i} \geq 0 \tag{2.41}$$

The quadratic optimization method of Burmeister et al [10] can now be employed to get a C^1 convex cubic spline interpolant satisfying conditions (2.40) and (2.41). Here again, $\tau_i \leq 1/3$ is chosen and the formula (2.39) is used for the d_i' to get C^2 convexity preserving interpolants. An algorithm concerning the above theory has been implemented and is compared with other methods in the next section. It is worthwhile to note that Schmidt and Heb [59] have also described a C^2 convexity preserving method using cubic splines which is very similar to the above method.

2.8 Numerical Examples and Conclusions.

In this section, the different methods introduced earlier in this chapter are compared. For this purpose, a series of experiments have been conducted using the data sets introduced in Chapter 1 to examine the performance of each method. The results of these tests are now described.

Data 1 is monotone increasing, so it is reasonable to expect a monotone increasing interpolant. However, the data contains both flat and steep regions which must be accommodated, making monotonicity preservation more difficult to achieve. The methods described in this chapter were applied to this test data and the resulting curves are shown in Figures 2.5 through 2.12. The Gasparo-Morandi and Yan methods introduce a pair of extra knots in the interval $[9.0, 11.0]$ at $(9.097, 10.781)$ and $(9.184, 10.584)$ respectively. In Figure 2.9 and Figure 2.12, although shapes of the curves are monotone increasing, there exists a non-pleasing behaviour in the above interval, where shapes are linear and flat between the additional inserted knots. For this data, it is clear that the Gasparo-Morandi and Yan methods do not give visually pleasing curves, but the other methods cope fairly well in general and produce visually pleasing shapes.

Data 2 contains a very steep region between points 8.7 and 10.0, a very flat region between points 10.0 and 20.0 and a difficult step feature between points 7.99 and 8.7. The data is also convex from the point 8.7 onwards. The methods were applied to this data set and the resulting curves are shown in Figures 2.13 through 2.20. The most striking feature of the curves in Figures 2.13 to 2.20 is the sharp corner at point 8.09 in each case. This occurs because each method preserves monotonicity in the data and the point 8.09 is the

junction of a very flat and a very steep segment. In Figure 2.17 and Figure 2.20, the results for this data show once again that the Gasparo-Morandi and Yan methods produce sharper changes in slope of the curve at the additional knots, and as a result the curve segments are not visually pleasing between the additional inserted knots at $(10.227, 11.894)$ and $(10.430, 11.798)$ in the interval $[10.0, 12.0]$ respectively. As before, the other methods produce curves which are visually pleasing; in particular, the shape of curve in the region of the interval $[10.0, 12.0]$.

Data 3 contains a flattish region at each end and a very steep section in the middle. Note also that this data is convex in the interval $[22.0, 23.1]$ in which the points 22.5, 22.6 and 22.7 are collinear, and concave in the interval $[23.2, 24.0]$. This shows that the interpolating curve will have a single point of inflection in $[23.1, 23.2]$. Figures 2.21 through 2.28 show the results of interpolating this data set by various the methods. Figure 2.21 is the graph of the interpolating curve generated by Beatson-Wolkowicz method. It can be seen that some erratic and non-pleasing behaviour appears in the shape of the curve segments, specifically in the intervals $[22.8, 23.0]$ and $[23.2, 24.0]$. From the plots of the interpolating curves shown in Figure 2.25 and Figure 2.28, we note, in particular that shapes of the curves in the intervals $[23.0, 23.1]$ and $[23.5, 24.0]$ show unpleasant behaviour due to the linear and flat segments drawn over the extra knots inserted at $(23.009, 23.074)$ and $(23.659, 23.733)$ by the Gasparo-Morandi method and at $(23.016, 23.055)$ and $(23.669, 23.717)$ by the Yan method, respectively. The Beatson-Wolkowicz, Gasparo-Morandi and Yan methods produce totally unpleasant curves for this difficult data and in fact they are trying to preserve monotonicity instead of convexity in the data. The Brodlie, Fritsch-Butland, Huynh and Pruess methods give more natural

looking and visually pleasing curves, but the remaining methods produce sub-standard interpolants to the data, where a noticeable change in the shape of the curve at the intervals $[22.9, 23.1]$ and $[23.2, 23.4]$ can be seen.

Data 4 is convex and monotone increasing and Figures 2.29 through 2.36 are the graphs of the interpolation curves drawn by the methods in question. In Figure 2.33, the pair of additional knots inserted by the Gasparo-Morandi method are placed at $(-1.991, -1.004)$ and $(-0.945, -0.441)$, while in Figure 2.36, the extra knots introduced by the Yan method are located at $(-1.983, -1.008)$ and $(-0.900, -0.558)$, in the intervals $[-2.0, -1.0]$ and $[-1.0, -0.3]$ respectively. Once again, the Gasparo-Morandi and Yan methods fail to maintain the shape of the data because they are trying to preserve monotonicity rather than convexity and the effects of linear and flat curve segments drawn due to the insertion of extra knots can be seen in the region of the first, second and third interpolation points. Figure 2.29 shows a curve generated by the Beatson-Wolkowicz method. The shape of this curve, particularly in the region of the first and second interpolation points, is slightly unpleasant. Figure 2.31 is the graph of the interpolating curve by the Butland method. It can be seen that the curve produces a shape with a rapid turn at the interior data points. The reason for these may be due to the rapid change in the slopes at those points. As a comparison, the interpolating curves discussed above are compared to the rest of the curves shown in Figures 2.30, 2.32, 2.34 and 2.35: we focus, in particular, on the shape of the curve in the region of the second and third interpoaltion points. It can be seen that these curves produced by the Brodlie, Fritsch-Butland, Huynh and Pruess methods are more satisfactory and visually pleasing than those obtained by the other methods.

The Gasparo-Morandi and Yan methods generate a C^1 piecewise cubic spline interpolant and only add two extra knots between the existing data points at which $(\alpha_i, \beta_i) \notin S$, in order to preserve monotonicity of the data. The disadvantage of adding extra knots is that the resulting curve generally yields an undesirable feature on the shape of the interpolant. This is due to the fact that former method always contructs a straight line with non-zero slope and the latter always draws a horizontal line, between the additional inserted knots. Obviously, the linear or flat curve segments drawn by these methods can not produce a visually pleasing graph, as is evident from the aforementioned examples. Furthermore, these methods require more storage and increased search time during evaluation.

The C^2 shape-preserving interpolation method proposed by Pruess has an advantage in terms of accuracy and additional smoothness, which may be necessary in some applications, but is less efficient in both the preprocessing and evaluation phase. It always adds two additional knots in each interval and thus, using three cubics per interval, is less cost effective in terms of evaluation. In the preprocessing phase, particularly in the case of convex data, it requires solving an optimization problem in order to satisfy the convexity conditions. Also, this process must be repeated if an additional data point is added to the data set, whereas other methods depend only on a few nearby data points, which allow for efficient updating of the data set. Thus, the Pruess method is not competitive in terms of computational cost with the methods described in this chapter.

All other methods produce shape-preserving interpolating curves using C^1 cubic splines and are simple, fast, very easy to implement, and cheap in terms of computational cost in that they do not require the insertion of

additional knots. Also, they demand a lesser number of operations per single piece of interpolant compared to the Beatson-Wolkowicz, Gasparo-Morandi, Yan and Pruess methods. The main disadvantages for the methods which add extra knots are that the amount of extra storage required for the data, and the amount of search time needed to evaluate the spline interpolant are both increased.

Furthermore, on the basis of appearance of the shapes of the curves, we conclude from the above results that the Brodlie, Fritsch-Butland, Huynh and Pruess methods are, in general, able to produce both smooth and visually pleasing shapes of the interpolating curves and are the best among the existing local methods.

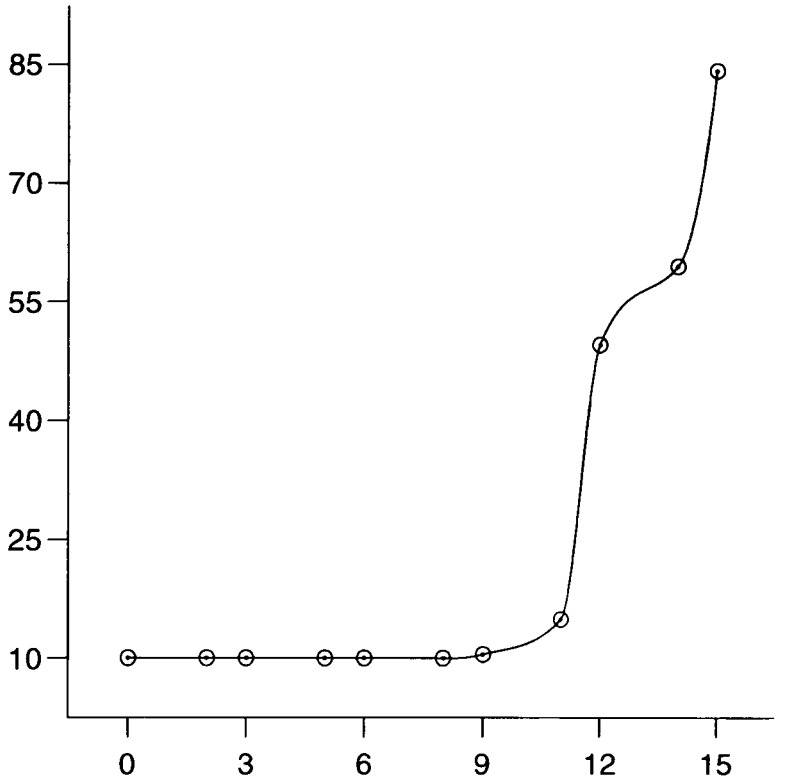

Figure 2.5 Beatson-Wolkowicz method which interpolates Data 1.

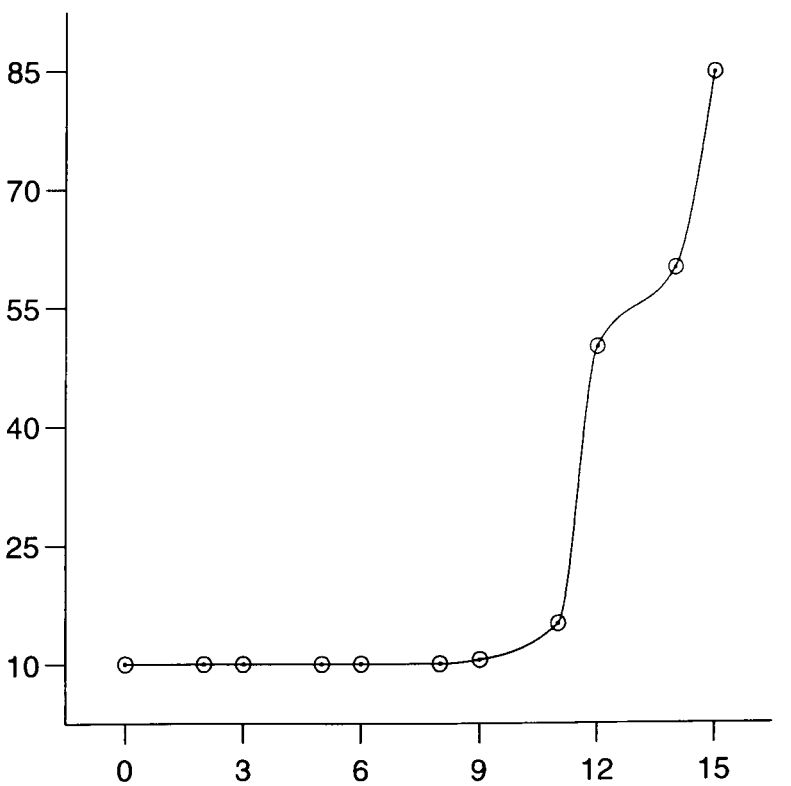

Figure 2.6 Brodlie method which interpolates Data 1.

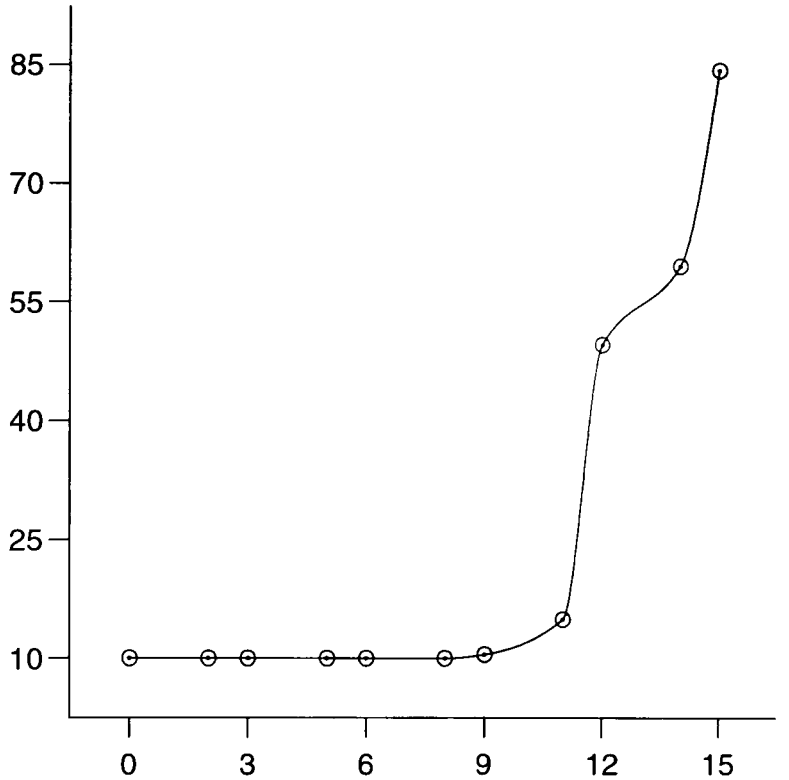

Figure 2.7 Butland method which interpolates Data 1.

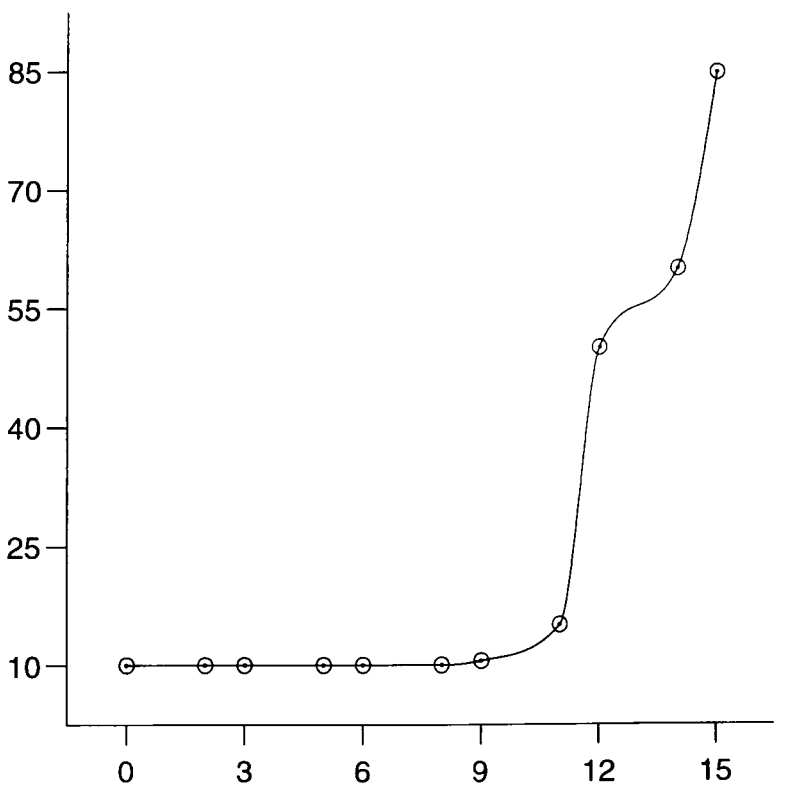

Figure 2.8 Fritsch-Butland method which interpolates Data 1.

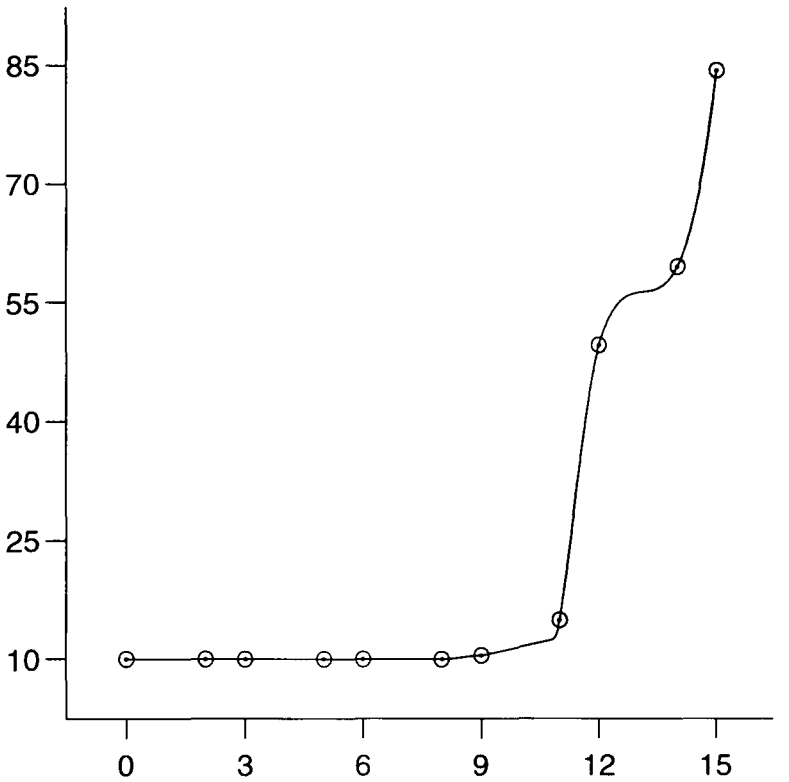

Figure 2.9 Gasparo-Morandi method which interpolates Data 1.

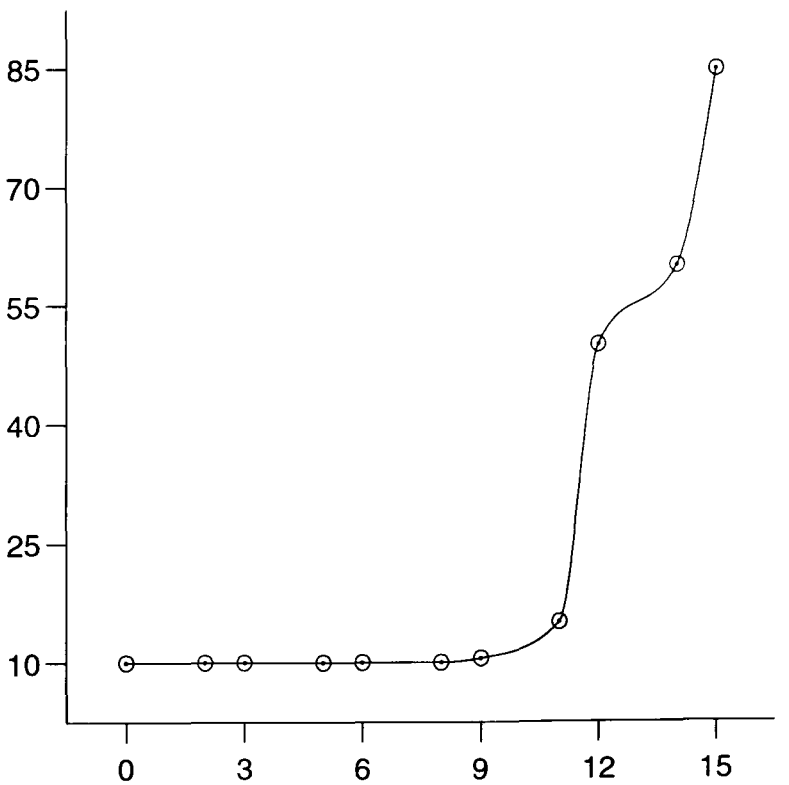

Figure 2.10 Huynh method which interpolates Data 1.

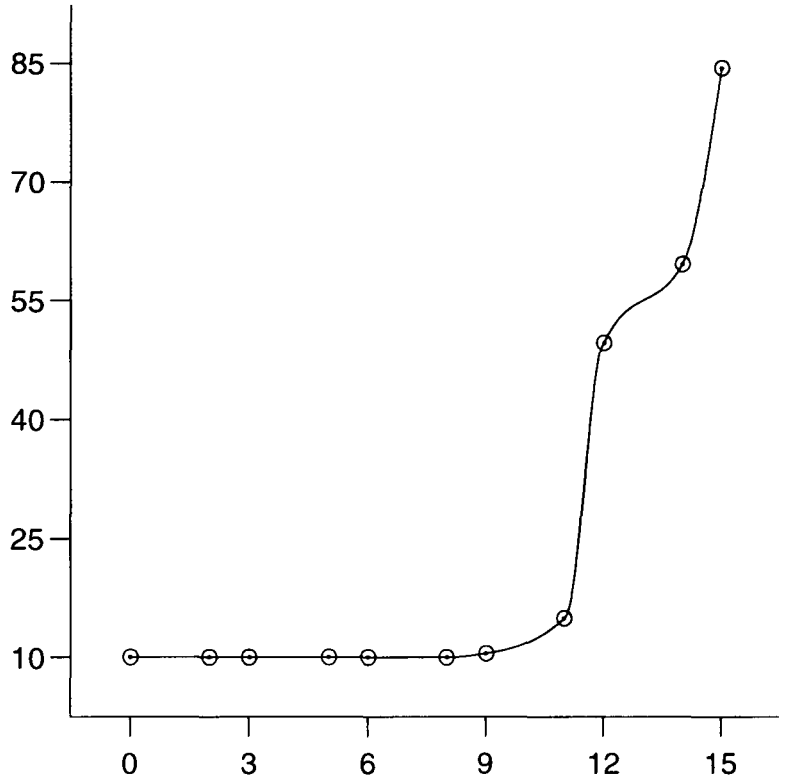

Figure 2.11 Pruess method which interpolates Data 1.

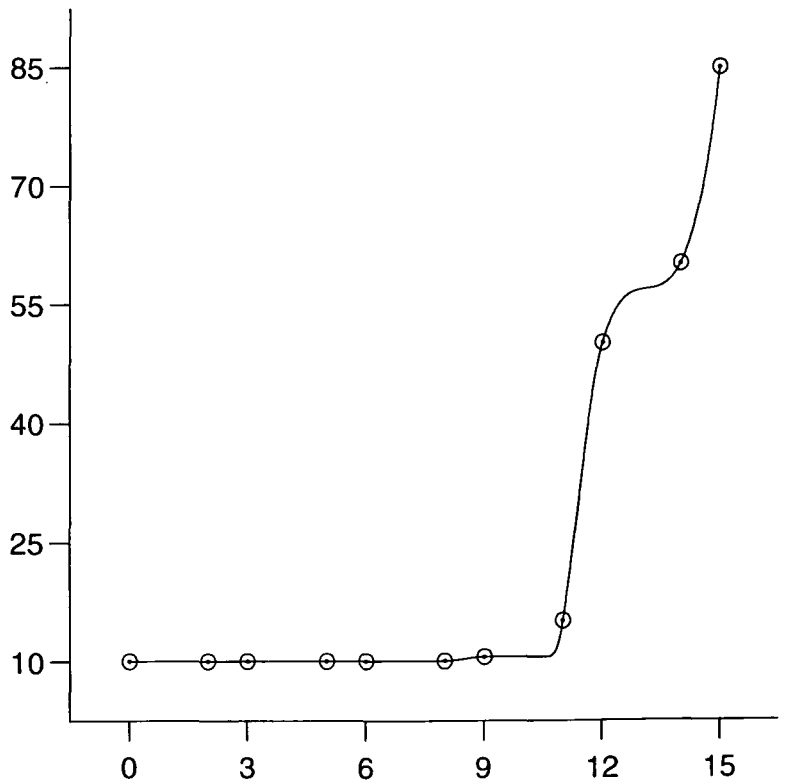

Figure 2.12 Yan method which interpolates Data 1.

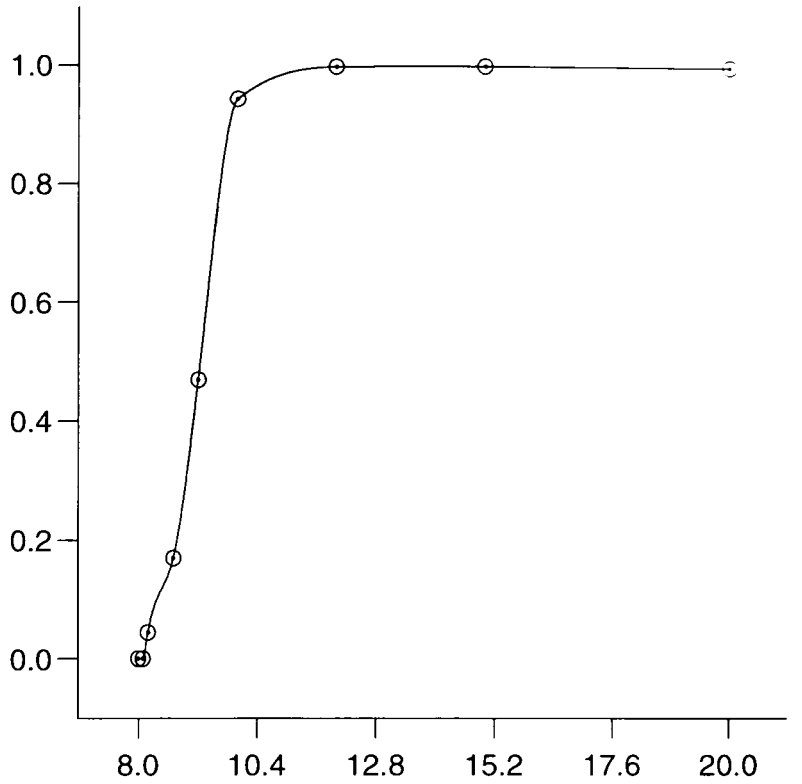

Figure 2.13 Beatson-Wolkowicz method which interpolates Data 2.

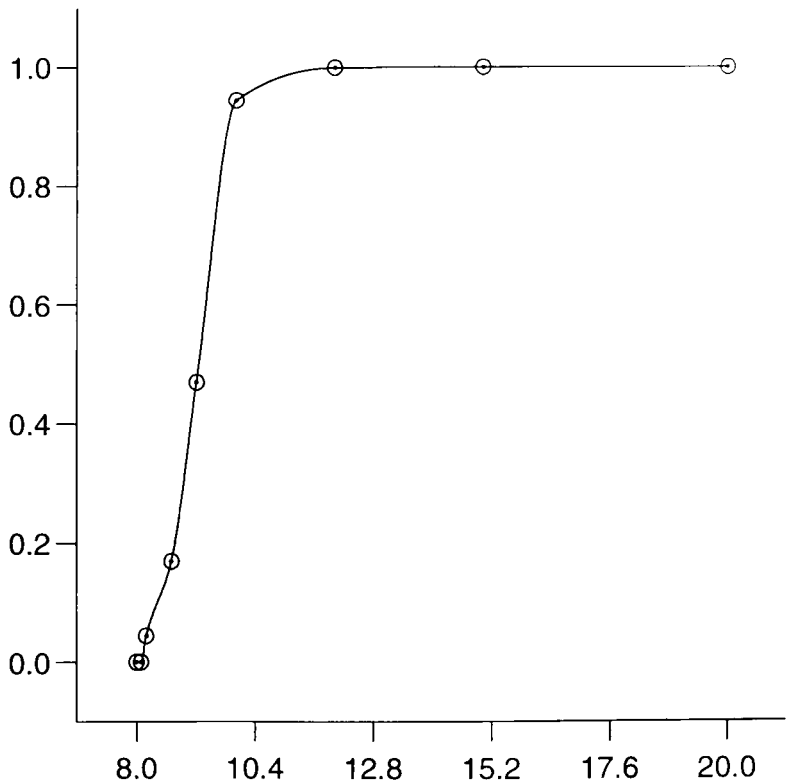

Figure 2.14 Brodlie method which interpolates Data 2.

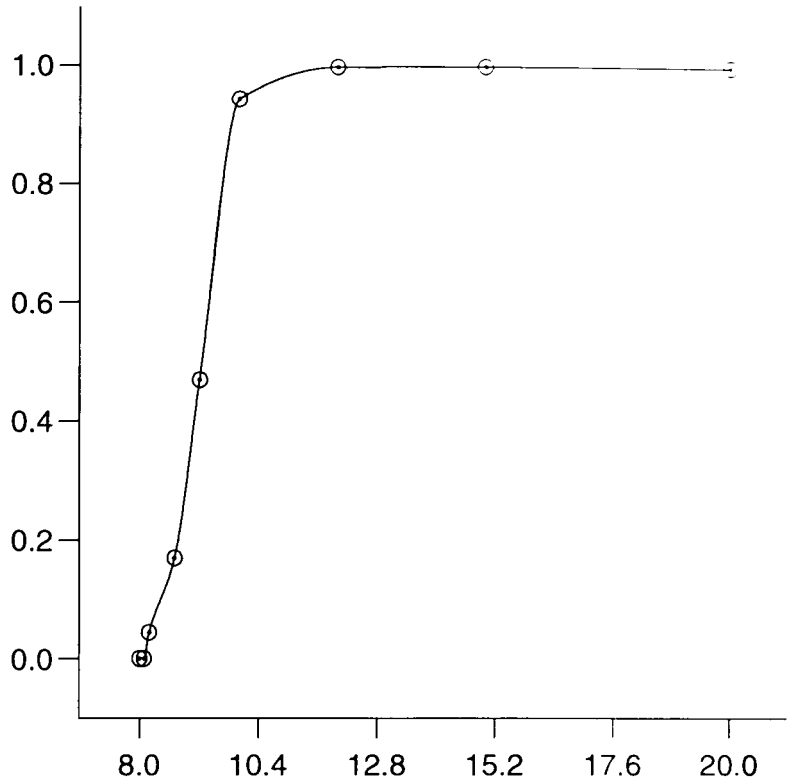

Figure 2.15 Butland method which interpolates Data 2.

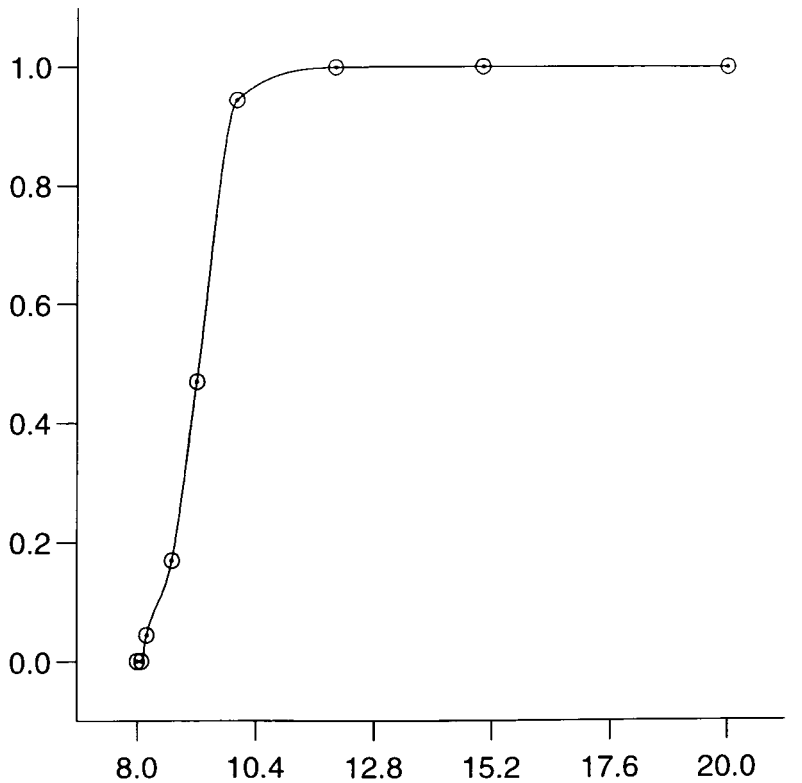

Figure 2.16 Fritsch-Butland method which interpolates Data 2.

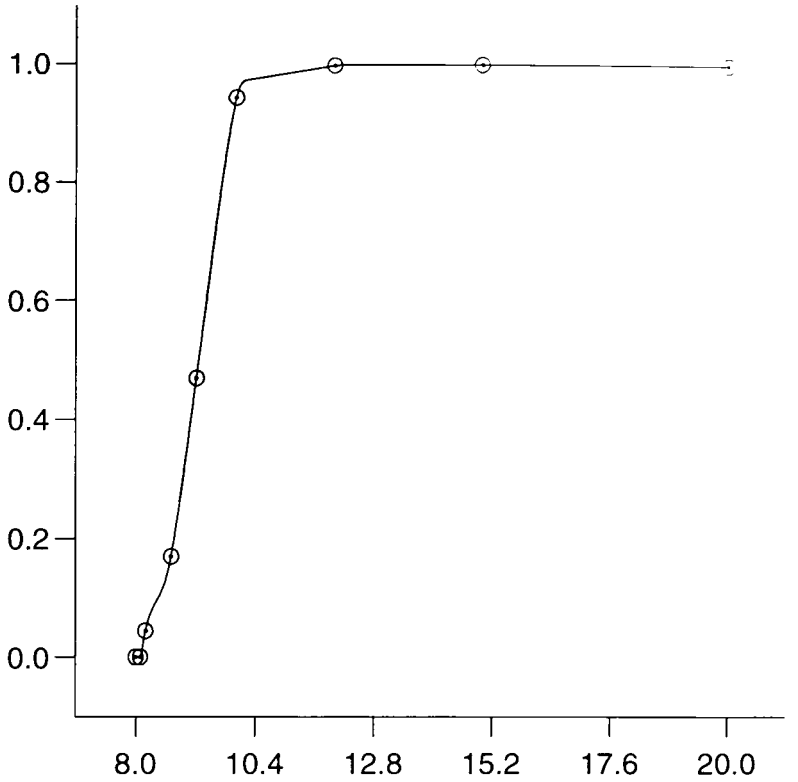

Figure 2.17 Gasparo-Morandi method which interpolates Data 2.

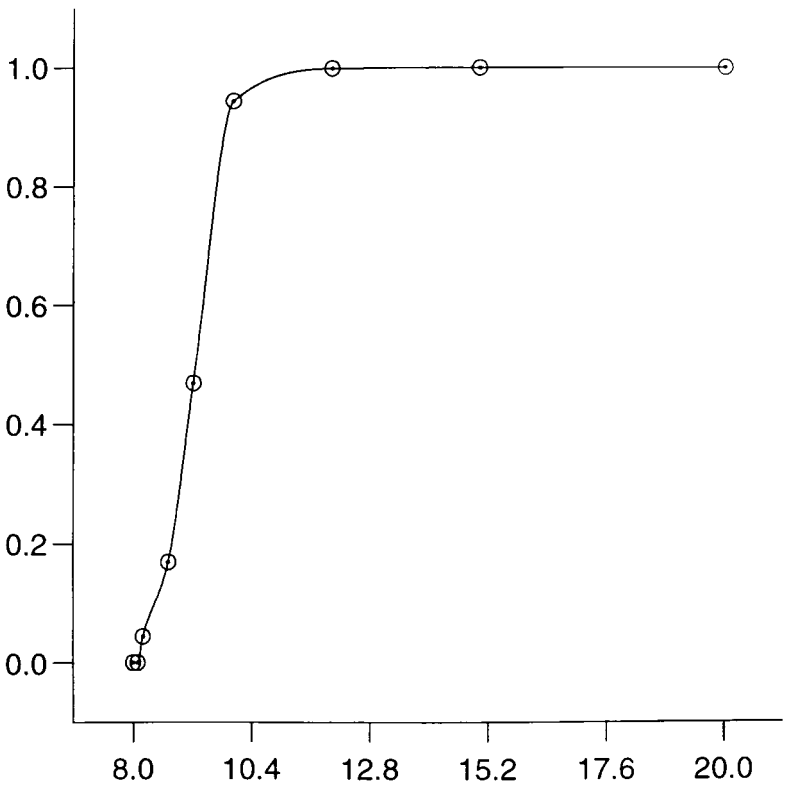

Figure 2.18 Huynh method which interpolates Data 2.

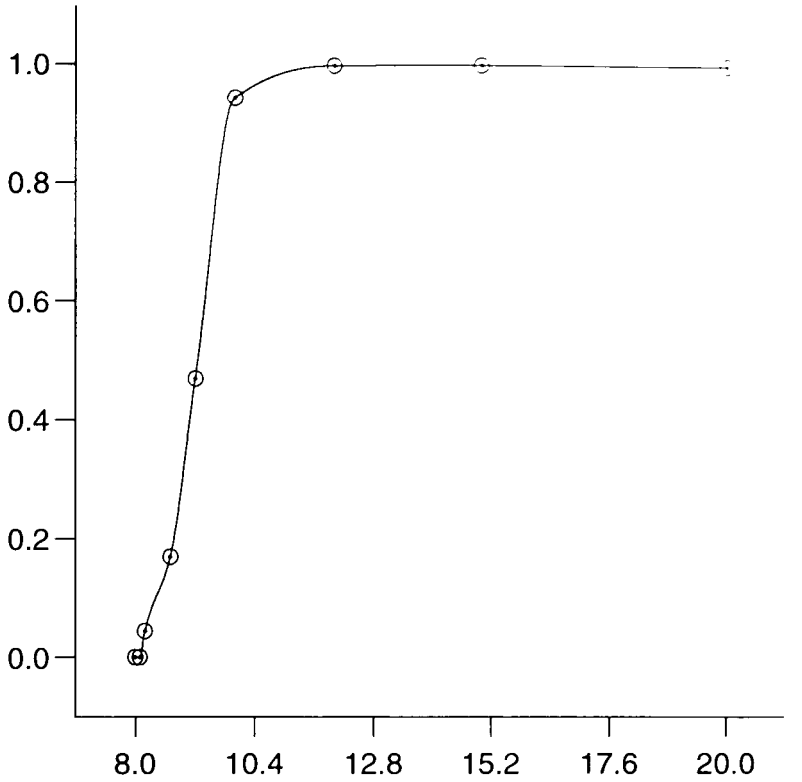

Figure 2.19 Pruess method which interpolates Data 2.

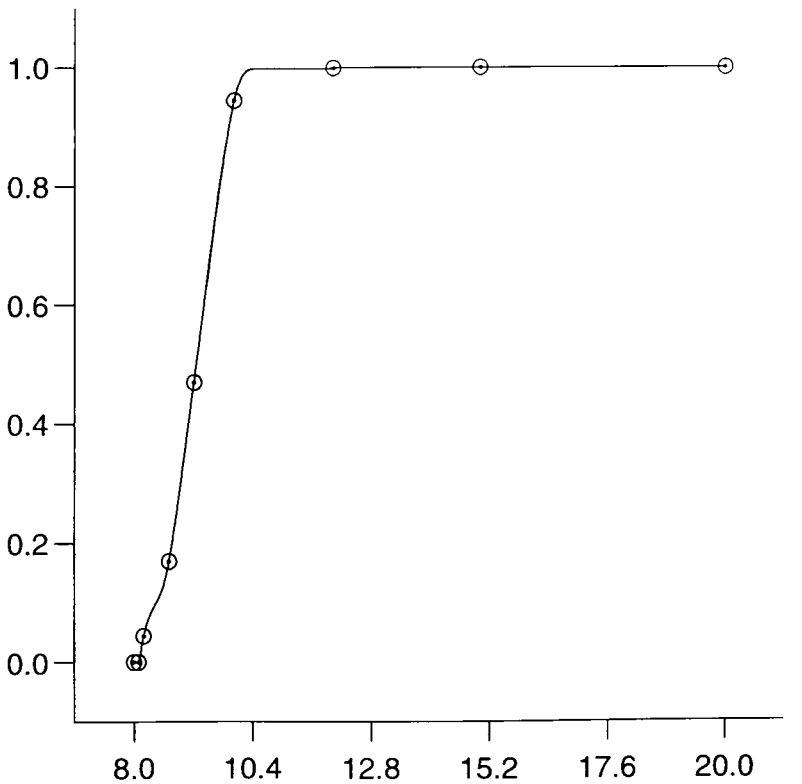

Figure 2.20 Yan method which interpolates Data 2.

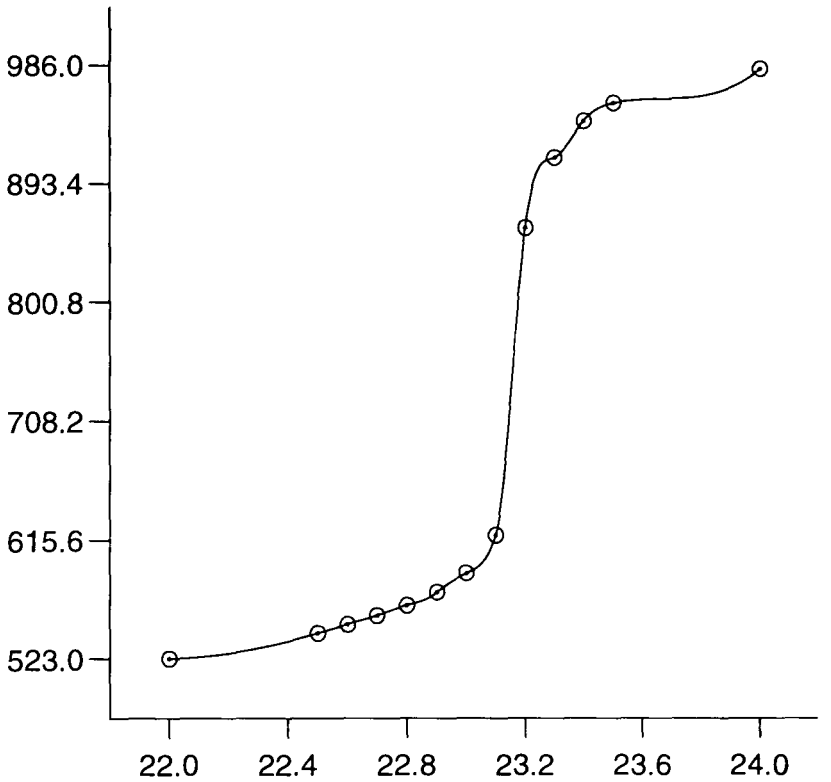

Figure 2.21 Beatson-Wolkowicz method which interpolates Data 3.

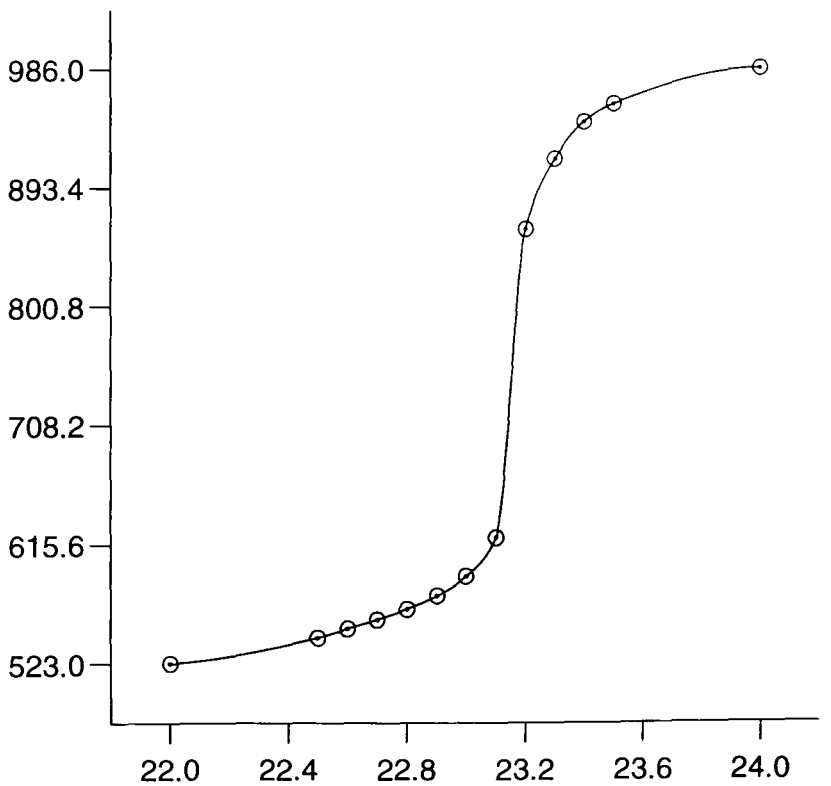

Figure 2.22 Brodlie method which interpolates Data 3.

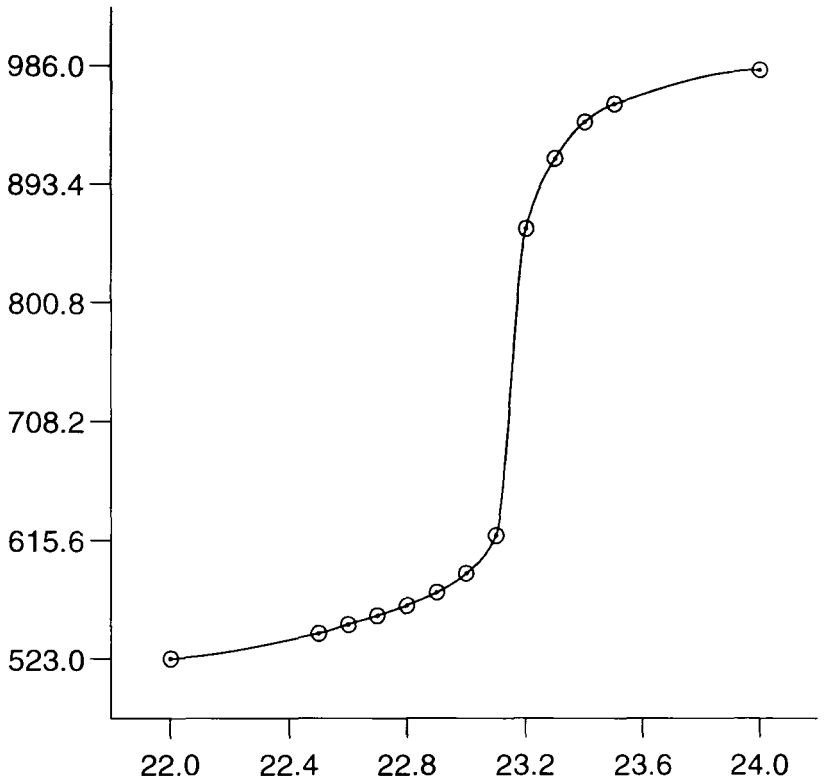

Figure 2.23 Butland method which interpolates Data 3.

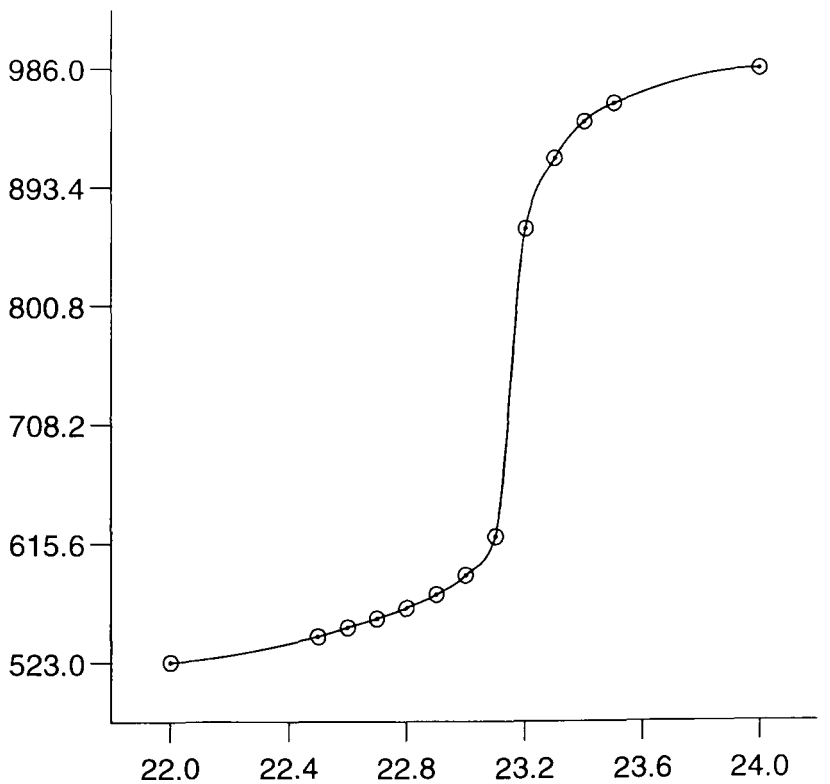

Figure 2.24 Fritsch-Butland method which interpolates Data 3.

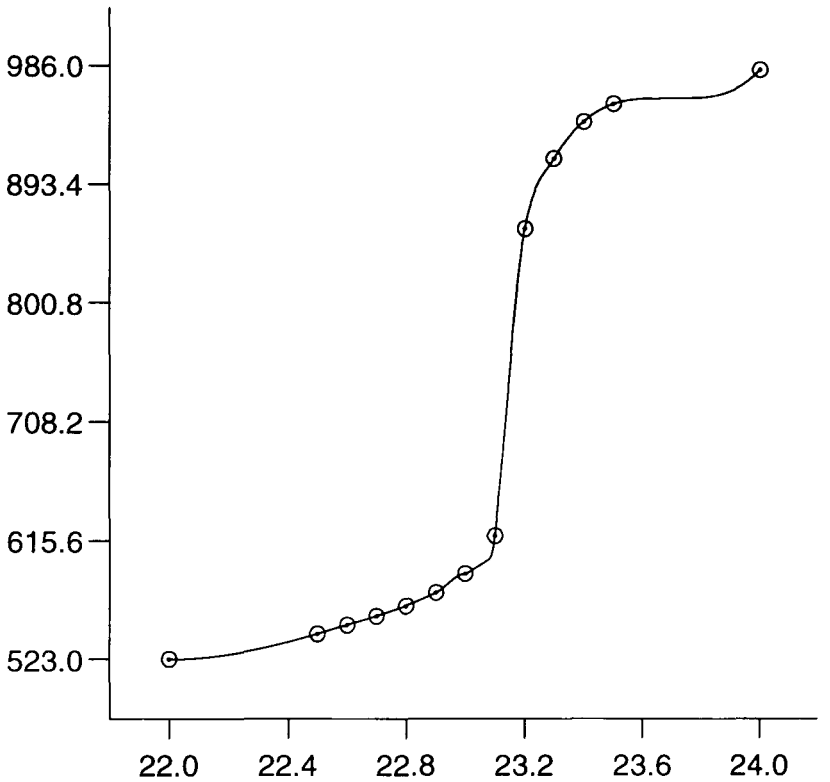

Figure 2.25 Gasparo-Morandi method which interpolates Data 3.

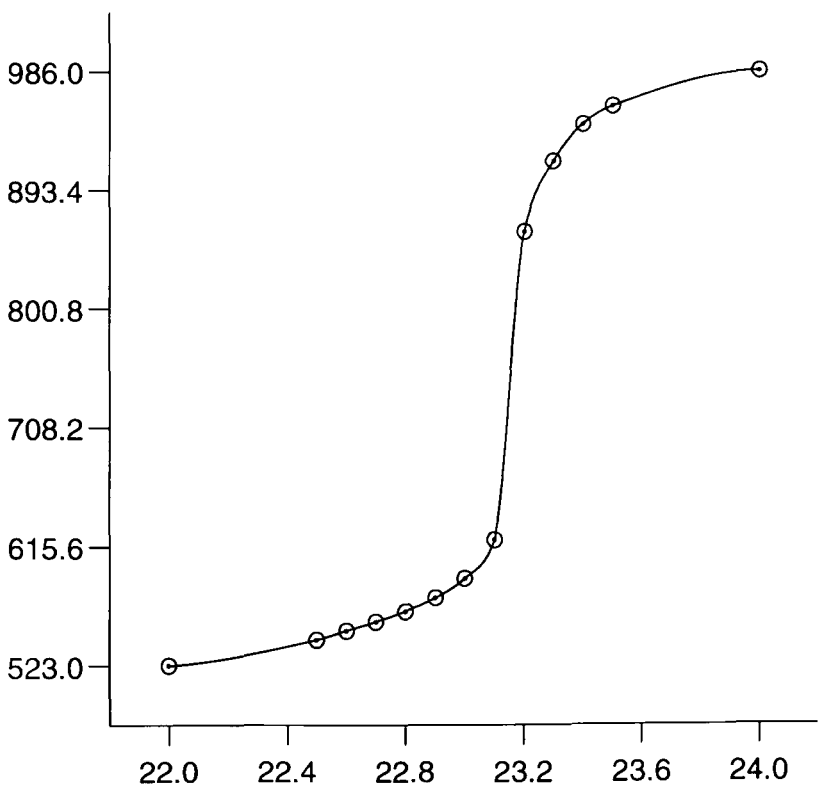

Figure 2.26 Huynh method which interpolates Data 3.

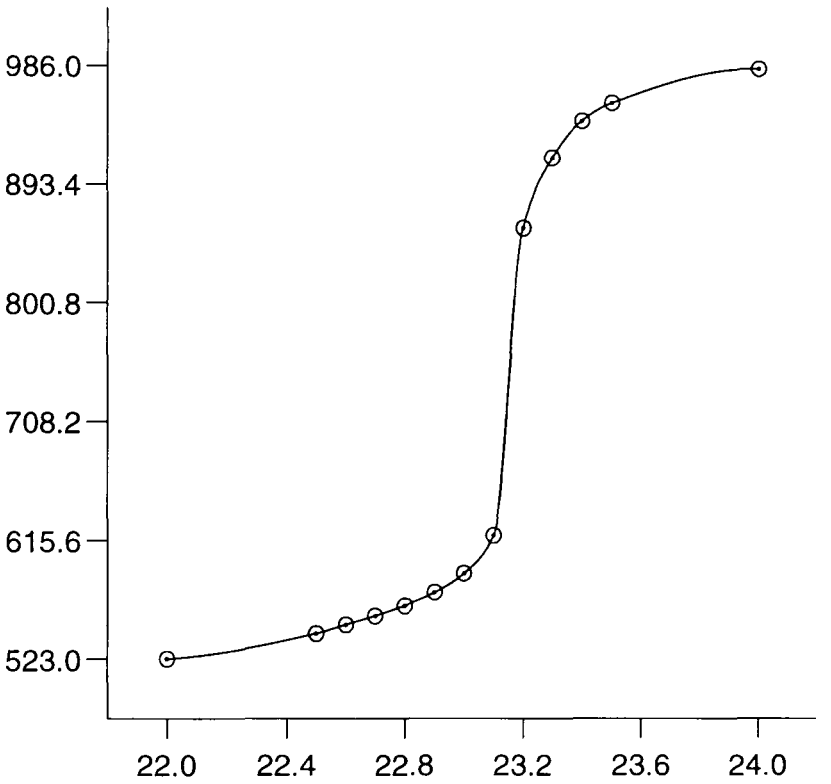

Figure 2.27 Pruess method which interpolates Data 3.

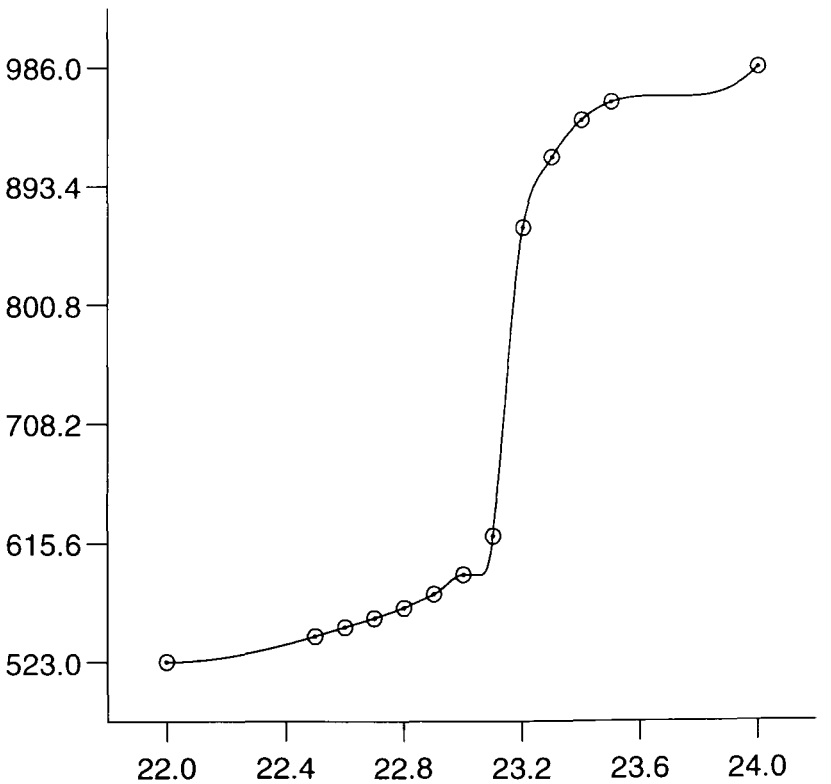

Figure 2.28 Yan method which interpolates Data 3.

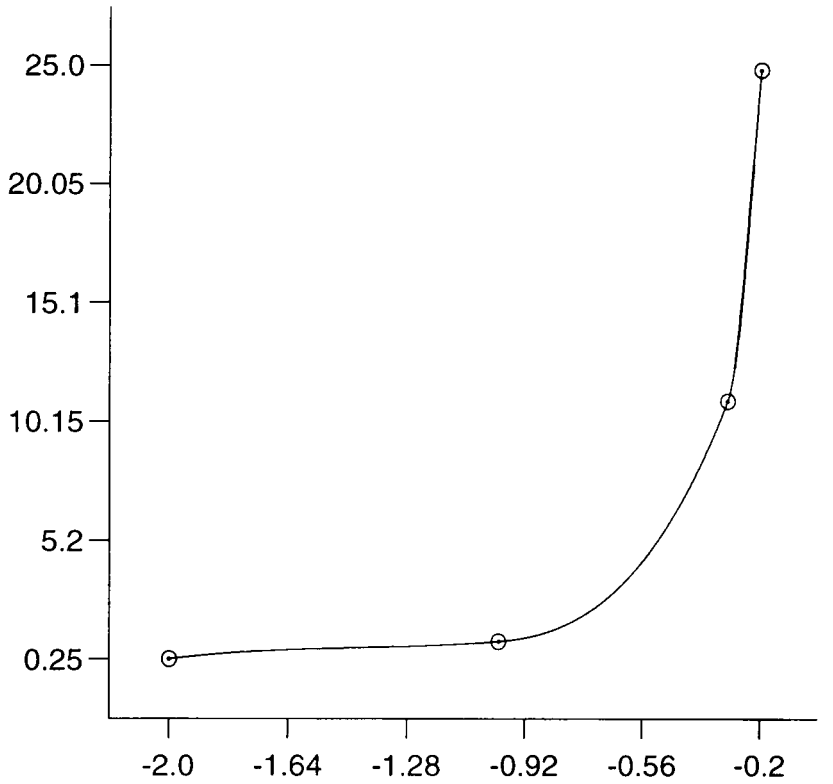

Figure 2.29 Beatson-Wolkowicz method which interpolates Data 4.

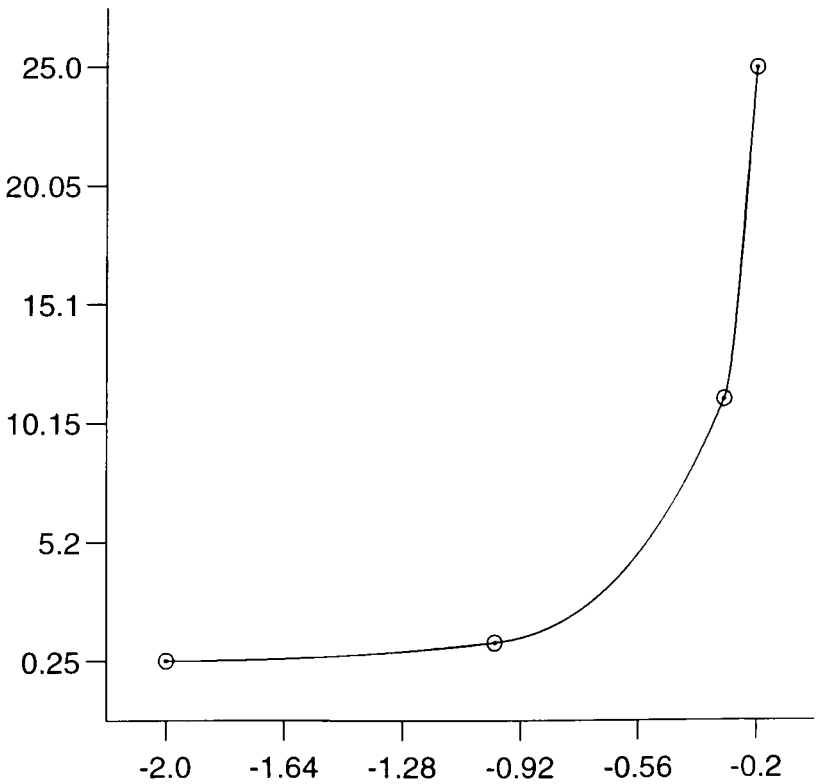

Figure 2.30 Brodlie method which interpolates Data 4.

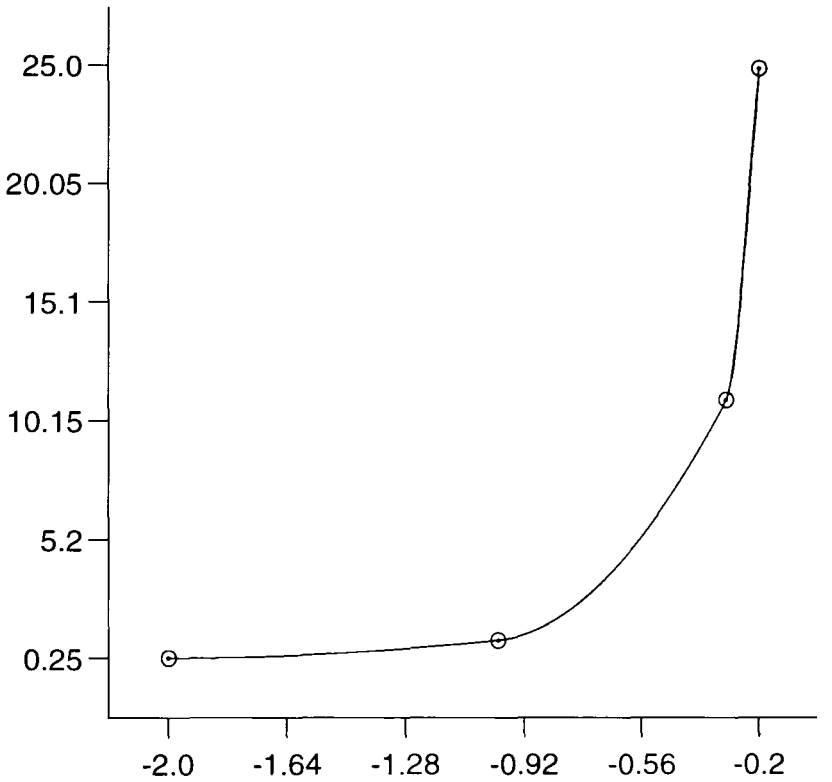

Figure 2.31 Butland method which interpolates Data 4.

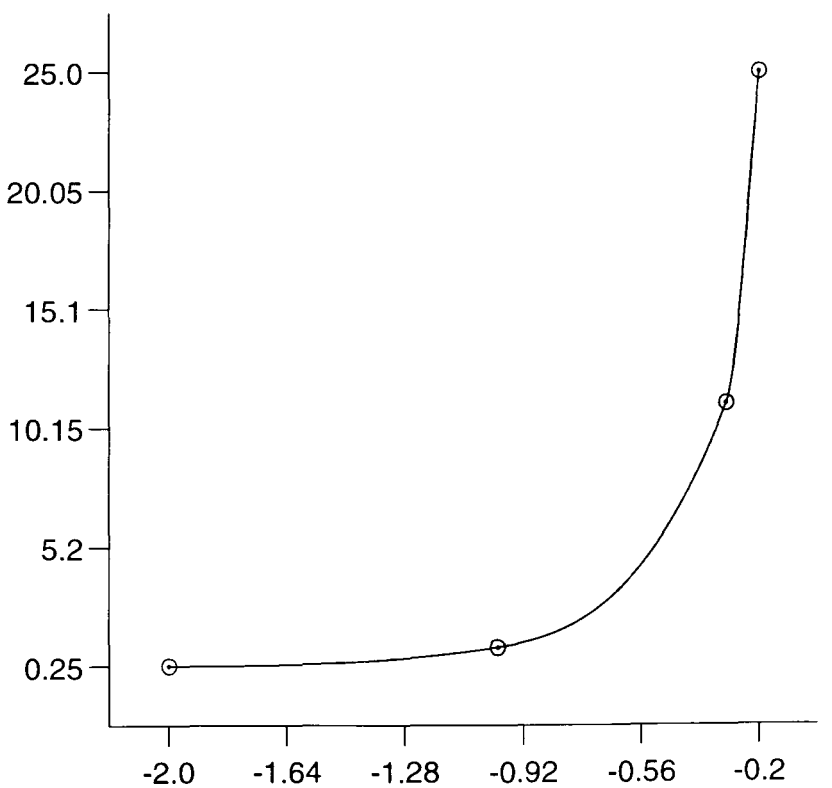

Figure 2.32 Fritsch-Butland method which interpolates Data 4.

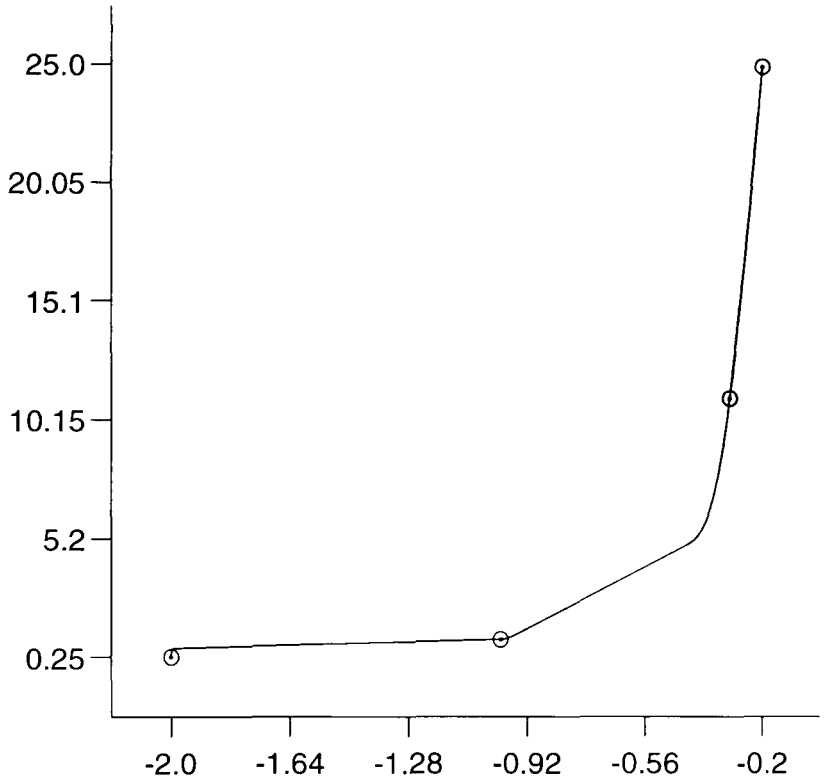

Figure 2.33 Gasparo-Morandi method which interpolates Data 4.

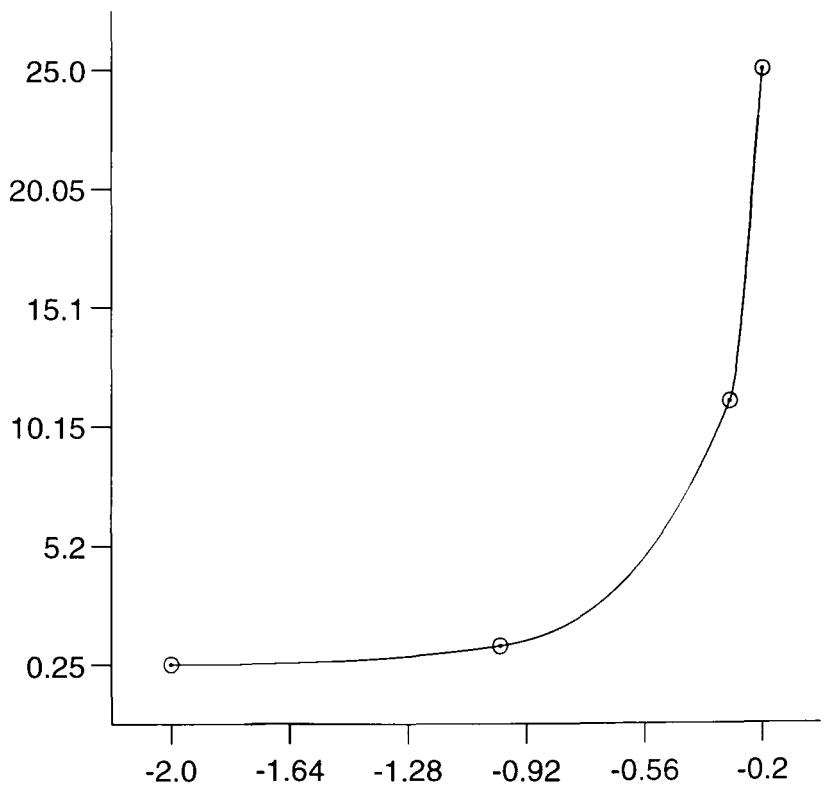

Figure 2.34 Huynh method which interpolates Data 4.

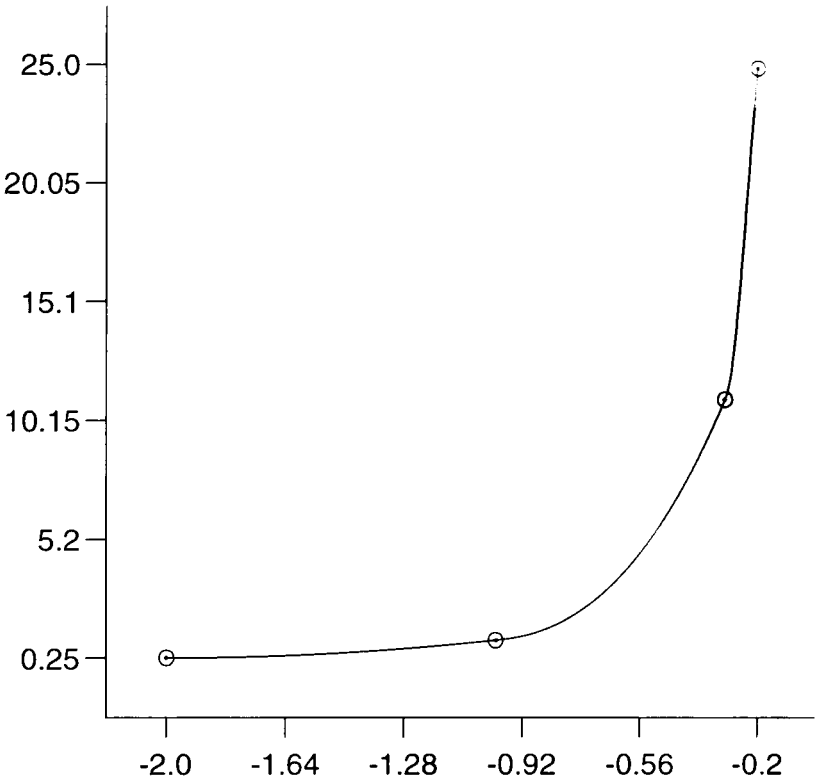

Figure 2.35 Pruess method which interpolates Data 4.

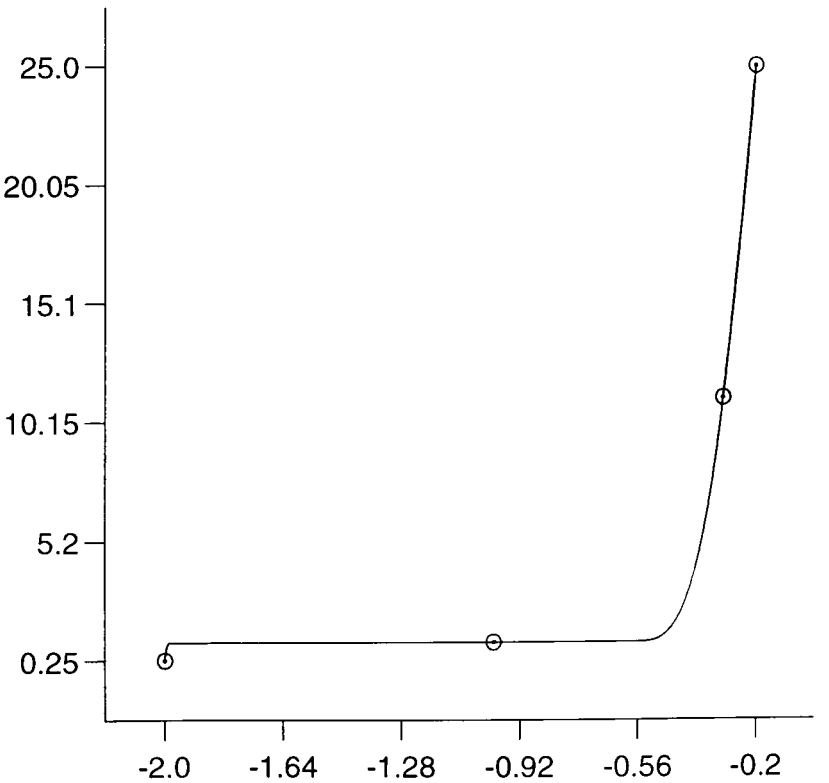

Figure 2.36 Yan method which interpolates Data 4.

Chapter 3

Shape-Preserving Algorithms Using Quadratic Splines

As we have discussed in Chapter 2, by relaxing the C^2 continuity of cubic splines, we get enough degrees of freedom to satisfy the monotonicity and/or convexity conditions. Nevertheless, one of the very important reasons to prefer cubics over quadratics is their C^2 continuity. If we do not have this, then an important advantage is lost. An attractive alternative over cubic splines is then to use the C^1 piecewise quadratic splines with additional knots chosen so as to preserve the shape characteristics of the data. Methods of this type are described below.

McAllister and Roulier [50] developed an algorithm in which the slope and knot assignments are made automatically to preserve monotonicity and/or convexity, based on a geometrical argument. The resulting quadratic piecewise polynomial is constructed from Bernstein polynomials. A similar algorithm using quadratic piecewise polynomials is presented by Schumaker [61]. This preserves both monotonicity and/or convexity by the addition of at most one knot in each data subinterval. This algorithm leaves judgement about control of inflection points, monotonicity, etc. up to an interactive computer user who modifies the interpolant either by changing the slopes or by altering the knot locations. Lahtinen [45, 46] has described the implementation and application of a minor modification of Schumaker's algorithm.

This chapter has been published in Iqbal [41] and expores the relationship between the ideas of the Schumaker and the McAllister and Roulier algorithms by presenting them in a common notation. In Section 3.1, we show that at the internal points the slope calculation method used by McAllister and Roulier [50], which is derived geometrically, generates slopes which are identical to those proposed by Butland [11]. In Section 3.2, we then prove that the slopes obtained from Butland's method satisfy all of the monotonicity and/or convexity conditions described in the algorithm of Schumaker [61] and that, using these slopes, the algorithm generates a shape-preserving interpolant in one pass, rather than requiring the adjustment of any slope or knot location. In Section 3.3, we determine the location of the knot introduced by the algorithm of McAllister and Roulier, and in Section 3.4 it is shown that the interpolating quadratic splines generated by this method and by Schumaker's algorithm are identical if the slopes required by Schumaker's algorithm are estimated by Butland's slope method. In Section 3.5, the Butland slopes are further improved iteratively for the convex data using a technique described in Frey [31]. Finally, conclusions and numerical examples are presented in Section 3.6.

3.1 Slope Calculation Methods.

The local slope estimation method of Butland [11], referred to above, was proposed for use with cubic interpolants. Fritsch and Butland [32] show that the slopes lie within the monotonicity region for piecewise cubic interpolation as defined by Fritsch and Carlson [33]. Here, we consider the use of the Butland slope formula with the piecewise quadratic interpolation methods of Schumaker [61] and McAllister and Roulier [50]. At the interior data points $(x_i, y_i), i = 2, ..., n-1$, the slopes are estimated as

$$d_i = \begin{cases} \dfrac{2\delta_{i-1}\delta_i}{\delta_{i-1}+\delta_i}, & \text{if } \delta_{i-1}\delta_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

At the extreme points (x_1, y_1) and (x_n, y_n), the slopes are calculated by using a forward and a backward 3-point difference formulae. Now from (3.1), it is obvious that

$$0 \leq \frac{d_i}{\delta_{i-1}} < 2 \qquad \text{and} \qquad 0 \leq \frac{d_i}{\delta_i} < 2 \tag{3.2}$$

Now, we prove that the Butland and the McAllister and Roulier slope calculation methods are identical. In the McAllister and Roulier slope estimation method, for $i = 2, ..., n-1, d_i = 0$ if $\delta_{i-1}\delta_i \leq 0$. Otherwise, if $|\delta_{i-1}| > |\delta_i| > 0$, then the line through (x_i, y_i) with slope δ_{i-1} is extended until it meets the horizontal line through (x_{i+1}, y_{i+1}) at the point $(\bar{x}, y_{i+1})$ as shown in Figure 3.1. The equation of the line passing through (x_i, y_i) with slope δ_{i-1} is given by

$$y - y_i = \delta_{i-1}(x - x_i) \tag{3.3}$$

McAllister and Roulier set $d_i = \dfrac{y_{i+1} - y_i}{\hat{x} - x_i}$, where $\hat{x} = \dfrac{\bar{x} + x_{i+1}}{2}$. From (3.3),

$$\hat{x} - x_i = \frac{(\bar{x} - x_i) + (x_{i+1} - x_i)}{2} = \frac{y_{i+1} - y_i}{2\delta_{i-1}} + \frac{x_{i+1} - x_i}{2}$$

Thus,

$$d_i = \frac{y_{i+1} - y_i}{\hat{x} - x_i} = \frac{2}{\dfrac{1}{\delta_{i-1}} + \dfrac{x_{i+1} - x_i}{y_{i+1} - y_i}} = \frac{2}{\dfrac{1}{\delta_{i-1}} + \dfrac{1}{\delta_i}}$$

$$\Longrightarrow \qquad d_i = \frac{2\delta_{i-1}\delta_i}{\delta_{i-1} + \delta_i} \tag{3.4}$$

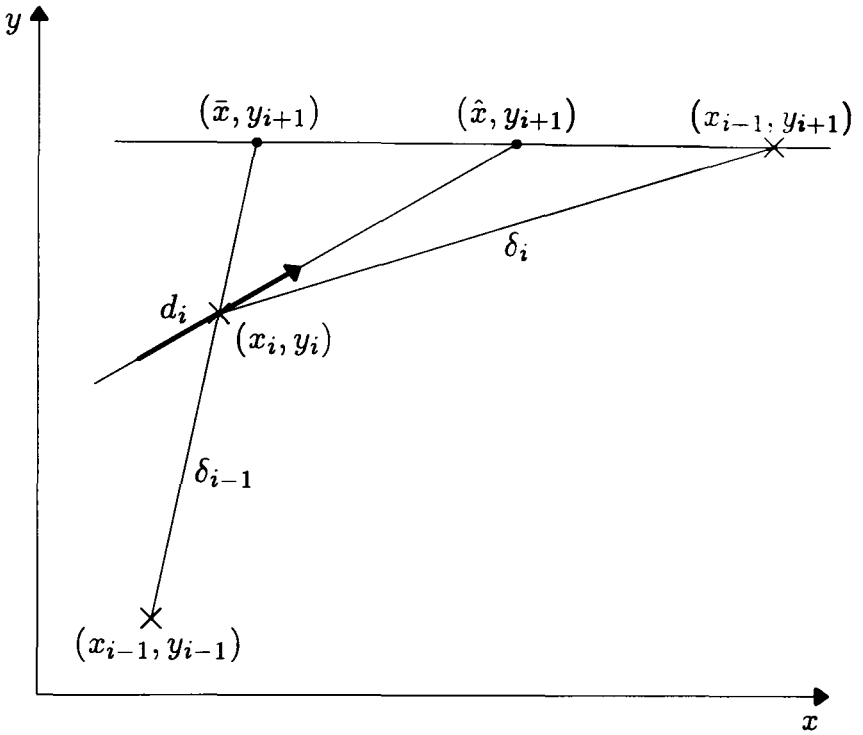

Figure 3.1 Data configuration for McAllister and Roulier's slope method when $|\delta_{i-1}| > |\delta_i| > 0$.

On the other hand, if $0 < |\delta_{i-1}| \leq |\delta_i|$ as shown in Figure 3.2, then the line through (x_i, y_i) with slope δ_i is extended until it meets the horizontal line through (x_{i-1}, y_{i-1}) at the point $(\bar{x}, y_{i-1})$. By applying a similar argument as above leads to

$$d_i = \frac{2\delta_{i-1}\delta_i}{\delta_{i-1} + \delta_i} \tag{3.5}$$

From (3.4) and (3.5), we can write, for $i = 2, ..., n-1$

$$d_i = \begin{cases} \dfrac{2\delta_{i-1}\delta_i}{\delta_{i-1} + \delta_i}, & \text{if } \delta_{i-1}\delta_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

At the end points, for $i = 1, n$, the slopes are estimated as follows:

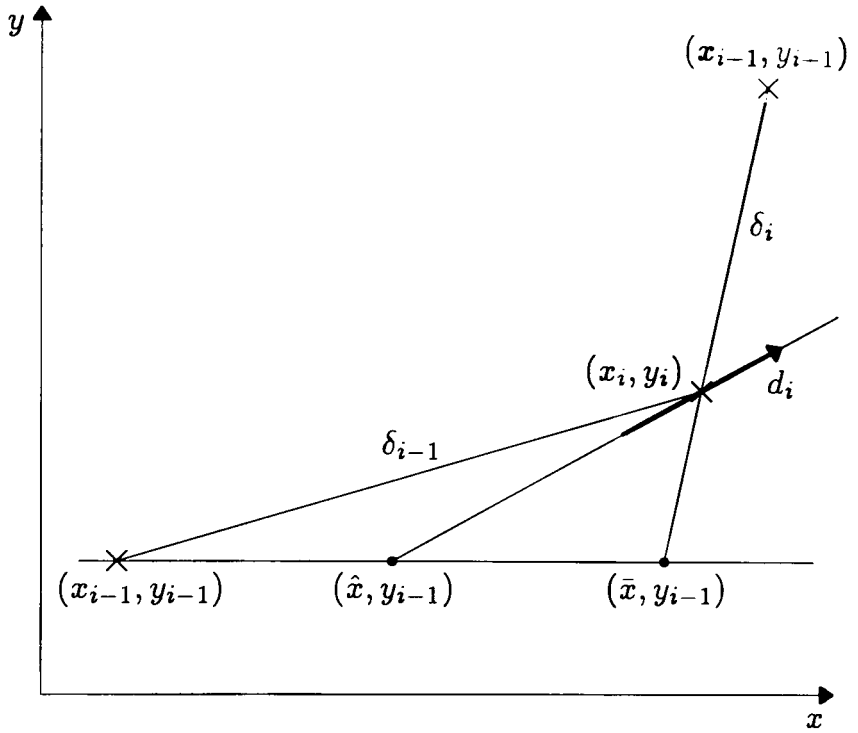

Figure 3.2 Data configuration for McAllister and Roulier's slope method when $0 < |\delta_{i-1}| \leq |\delta_i|$.

$$d_1 = \begin{cases} 2\delta_1 - d_2, & \text{if } \delta_1(2\delta_1 - d_2) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

and

$$d_n = \begin{cases} 2\delta_{n-1} - d_{n-1}, & \text{if } \delta_{n-1}(2\delta_{n-1} - d_{n-1}) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

From (3.1) and (3.6), we deduce that the internal slopes derived from the Butland formula and from the construction of McAllister and Roulier are identical. The term "Butland formula" is used hereafter for the slopes that are estimated by the Butland method at the interior points and at the end points by the McAllister and Roulier method.

3.2 Schumaker Algorithm.

Schumaker [61] describes a fairly straightforward shape-preserving method which constructs the interpolant as a C^1 quadratic spline with knots at the data points $x_1, ..., x_n$ and with at most one additional knot in each subinterval $(x_i, x_{i+1}), i = 1, ..., n-1$. The additional knot is inserted under certain conditions and, when it is inserted, there is some freedom of choice in exactly where it can be placed.

The method is based on a number of lemmas which characterise the solution of the interpolation problem (1.8). These state that if $d_i + d_{i+1} = 2\delta_i$, then a single quadratic polynomial

$$p(x) = y_i + d_i(x - x_i) + \frac{(d_{i+1} - d_i)(x - x_i)^2}{2(x_{i+1} - x_i)}, \qquad x \in [x_i, x_{i+1}] \tag{3.9}$$

interpolates the data in $[x_i, x_{i+1}]$, and otherwise a quadratic spline with a single additional knot at an arbitrarily chosen point ξ_i is required. This spline is given by

$$p(x) = \begin{cases} A_1 + B_1(x - x_i) + C_1(x - x_i)^2, & x \in [x_i, \xi_i] \\ A_2 + B_2(x - \xi_i) + C_2(x - \xi_i)^2, & x \in [\xi_i, x_{i+1}] \end{cases} \tag{3.10}$$

with

$$A_1 = y_i, \quad B_1 = d_i, \quad C_1 = \frac{(\bar{d}_i - d_i)}{2(\xi_i - x_i)},$$

$$A_2 = y_i + d_i(\xi_i - x_i) + \frac{1}{2}(\bar{d}_i - d_i)(\xi_i - x_i), \quad B_2 = \bar{d}_i, \quad C_2 = \frac{d_{i+1} - \bar{d}_i}{2(x_{i+1} - \xi_i)},$$

where

$$\bar{d}_i = p'(\xi_i) = 2\delta_i - d_{i+1} + \frac{(d_{i+1} - d_i)(\xi_i - x_i)}{x_{i+1} - x_i} \tag{3.11}$$

Schumaker has presented the following results about the shape characteristics of the interpolant.

Proposition 3.2.1.

If $d_i + d_{i+1} = 2\delta_i$, then the single quadratic polynomial $p(x)$ in the interval $[x_i, x_{i+1}]$ will be monotone if and only if d_i and d_{i+1} are of same sign and convex (concave) when $d_{i+1} > d_i$ $(d_{i+1} < d_i)$.

Proposition 3.2.2.

If $d_i + d_{i+1} \neq 2\delta_i$, and d_i, d_{i+1} and δ_i have the same sign, then the quadratic spline $p(x)$ on $[x_i, x_{i+1}]$:

(I) is monotone if and only if

$$\omega d_i + (1-\omega) d_{i+1} \leq 2\delta_i \qquad \text{and} \qquad d_i d_{i+1} \geq 0 \tag{3.12}$$

where $\omega = \dfrac{\xi_i - x_i}{x_{i+1} - x_i}$.

(II) is convex when

$$d_i \leq \bar{d}_i \leq d_{i+1} \qquad \text{if} \quad d_i < d_{i+1} \tag{3.13}$$

and concave when

$$d_{i+1} \leq \bar{d}_i \leq d_i \qquad \text{if} \quad d_i > d_{i+1} \tag{3.14}$$

where $\bar{d}_i$ is the slope at ξ_i.

(III) If $(d_{i+1} - \delta_i)(d_i - \delta_i) < 0$, then p(x) is convex (concave) if $d_{i+1} > d_i$ $(d_{i+1} < d_i)$ assuming that the additional knot ξ_i is chosen so that

$$\begin{cases} x_i < \xi_i \leq x_i + \dfrac{2(x_{i+1}-x_i)(d_{i+1}-\delta_i)}{d_{i+1}-d_i}, & \text{if } |d_{i+1}-\delta_i| < |d_i-\delta_i| \\ x_{i+1} + \dfrac{2(x_{i+1}-x_i)(d_i-\delta_i)}{d_{i+1}-d_i} \leq \xi_i < x_{i+1}, & \text{if } |d_{i+1}-\delta_i| > |d_i-\delta_i| \end{cases} \tag{3.15}$$

Moreover, if $d_i d_{i+1} \geq 0$, then $p(x)$ is also monotone on $[x_i, x_{i+1}]$.

(IV) If $(d_{i+1}-\delta_i)(d_i-\delta_i) \geq 0$, then p(x) has a point of inflection on $[x_i, x_{i+1}]$.

The requirements on $\bar{d}_i$ restrict the choice of ξ_i for given d_i, d_{i+1} and it is not possible to satisfy conditions (3.13) and (3.14) for arbitrary knot locations. So ξ_i must satisfy conditions (3.15) that show which knot locations lead to convex or concave splines. Schumaker proposes the choice as follows:

Case 1.

If $(d_{i+1}-\delta_i)(d_i-\delta_i) < 0$, then

$$\xi_i = x_{i+1} + \frac{(d_i-\delta_i)(x_{i+1}-x_i)}{d_{i+1}-d_i} \tag{3.16}$$

Case 2.

if $(d_{i+1}-\delta_i)(d_i-\delta_i) \geq 0$, then

$$\xi_i = \frac{x_i + x_{i+1}}{2} \tag{3.17}$$

which are the centre points of the region of admissibility (note that, for 3.16 at least, this implies $\bar{d}_i = \delta_i$). When the data is convex or concave, only Case 1 will be selected.

In order to ensure that the interpolants preserve the monotonicity and/or convexity of the data, it is necessary to specify how the slopes d_i are to be chosen from the given data. Schumaker proposes a formula for generating

initial slope estimates which does not automatically ensure this (see, for example, Figures 3.5 and 3.6) and proposes instead to allow the user to modify the slopes and/or the knot locations to obtain shape-preserving interpolants.

We now show that the Butland formula yields values of d_i and d_{i+1} which directly satisfy all the conditions specified in Propositions 3.2.1 and 3.2.2 and that the resulting interpolant becomes monotone and/or convex automatically, without adjusting the slopes or the inserted knot. It is obvious that the Butland formula satisfies Proposition 3.2.1. Now we prove the other results in Proposition 3.2.2.

1. If $\xi_i = x_{i+1} + \dfrac{(d_i - \delta_i)(x_{i+1} - x_i)}{d_{i+1} - d_i}$, then

$$\omega = \frac{\xi_i - x_i}{x_{i+1} - x_i} = \frac{d_{i+1} - \delta_i}{d_{i+1} - d_i} \tag{3.18}$$

and, from (3.11), we have

$$\begin{aligned} \bar{d}_i &= 2\delta_i - d_{i+1} + \frac{(d_{i+1} - d_i)(\xi_i - x_i)}{x_{i+1} - x_i} \\ &= 2\delta_i - d_{i+1} + (d_{i+1} - d_i)\omega \\ &= 2\delta_i - d_{i+1} + \frac{(d_{i+1} - d_i)(d_{i+1} - \delta_i)}{d_{i+1} - d_i} \\ \bar{d}_i &= \delta_i \end{aligned} \tag{3.19}$$

To ensure monotonicity in this case, we must show that (3.12) is satisfied for the Butland slopes. Now if $\delta_{i-1}, \delta_i, \delta_{i+1}$ are of the same sign, then the Butland slopes d_i, d_{i+1} are also of the same sign and hence $d_i d_{i+1} \geq 0$. So, from (3.18), we have

$$\omega d_i + (1-\omega)d_{i+1} = \omega(d_i - d_{i+1}) + d_{i-1}$$

$$= \frac{(d_{i+1} - \delta_i)}{d_{i+1} - d_i}(d_i - d_{i+1}) + d_{i+1}$$

$$= \delta_i < 2\delta_i \tag{3.20}$$

2. If $\xi_i = \dfrac{x_i + x_{i+1}}{2}$, then

$$\omega = \frac{\xi_i - x_i}{x_{i+1} - x_i} = \frac{1}{2}$$

and (3.12) becomes

$$\frac{d_i + d_{i+1}}{2} \leq 2\delta_i$$

or

$$\frac{d_i}{\delta_i} + \frac{d_{i+1}}{\delta_i} \leq 4, \tag{3.21}$$

which follows immediately from (3.2). Hence, we conclude from (3.20) and (3.21) that the Butland formula satisfies the monotonicity condition (3.12).

Let us now consider the convexity condition (3.13) i.e.,

$$d_i \leq \bar{d}_i \leq d_{i+1} \ ,$$

which in view of (3.19) may be written as

$$d_i \leq \delta_i \leq d_{i+1} \tag{3.22}$$

We know that the Butland formula satisfies the inequalities

$$\delta_{i-1} \leq d_i \leq \delta_i \qquad \text{and} \qquad \delta_i \leq d_{i+1} \leq \delta_{i+1}$$

Now (3.22) follows from these inequalities automatically and so we see that inequality (3.13) always holds. Similarly, it can be proved that inequality (3.14) always holds with Butland formula when the data is concave.

Now we are in a position to say that by substituting the slopes estimated by the Butland formula into the Schumaker method, it becomes a one-pass algorithm for shape-preserving interpolation.

3.3 McAllister and Roulier Algorithm.

The algorithm proposed by McAllister and Roulier [50] produces a local, C^1 quadratic spline interpolant which preserves monotonicity and/or convexity of the data by inserting at most two additional knots per data interval. The selection of slopes and knots is based on the geometric argument described below and the polynomial pieces are constructed using Bernstein polynomials.

Let $S = (x_i, y_i)$ and $T = (x_{i+1}, y_{i+1})$ be two non-decreasing data points with $x_i < x_{i+1}$ having slopes d_i and d_{i+1} respectively. Let L_1 and L_2 be the two straight lines through points S and T with slopes d_i and d_{i+1} respectively. Let R be the set of points,

$$R = \{(x, y) : x_i \leq x \leq x_{i+1}, \quad \text{and} \quad y_i \leq y \leq y_{i+1}\} - \{S, T\}$$

that is, R is the boundary and interior of the rectangle defined by the points (x_i, y_i), (x_i, y_{i+1}), (x_{i+1}, y_{i+1}) and (x_{i+1}, y_i) minus the points S and T. Suppose M is the midpoint line segment through the points $F = \left(\frac{x_i + x_{i+1}}{2}, y_i\right)$ and $G = \left(\frac{x_i + x_{i+1}}{2}, y_{i+1}\right)$. Let $Z = (z_1, z_2)$ be a point of intersection of L_1 and L_2. We now show how to construct the desired quadratic spline $p(x)$ on $[x_i, x_{i+1}]$.

McAllister and Roulier consider four cases which can arise in general, but describe a local method for assigning slopes to the data points (as shown in

Section 3.1) which yields only the first two cases. For our comparison purpose, only the these two cases are relevent here.

Case 1.

Here L_1 and L_2 intersect each other at the point $Z = (z_1, z_2)$ in R, as shown in Figure 3.3, where

$$z_1 = \xi_i = \frac{y_i - y_{i+1} + d_{i+1}x_{i+1} - d_i x_i}{d_{i+1} - d_i} = x_{i+1} + \frac{(d_i - \delta_i)(x_{i+1} - x_i)}{d_{i+i} - d_i} \tag{3.23}$$

The algorithm inserts an additional knot at $x = \xi_i$.

Now suppose that

$$V = (v_1, v_2) = \left(\frac{x_i + \xi_i}{2}, L_1(\frac{x_i + \xi_i}{2})\right) \tag{3.24}$$

$$W = (w_1, w_2) = \left(\frac{x_{i+1} + \xi_i}{2}, L_2(\frac{x_{i+1} + \xi_i}{2})\right) \tag{3.25}$$

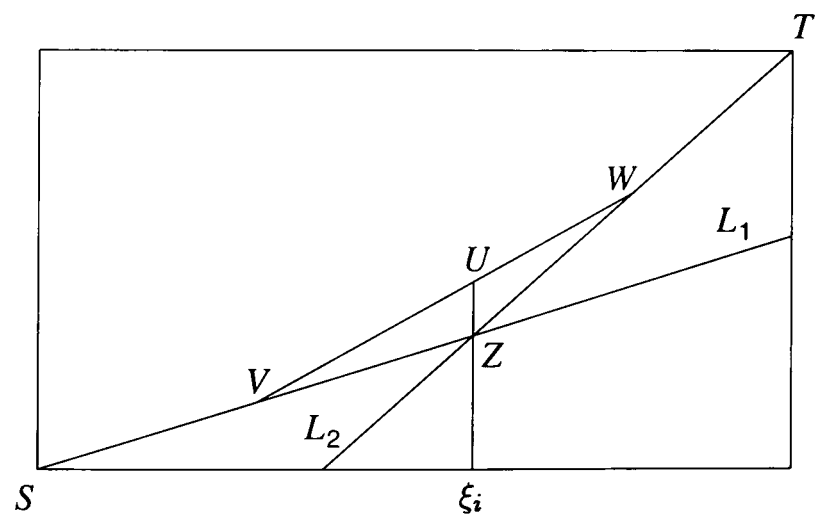

Figure 3.3 Case 1 configuration for McAllister-Roulier algorithm.

Let us define $\eta_i = L(\xi_i)$, where L is the line passing through the points V and W. Then define $p(x)$ on $[x_i, x_{i+1}]$ with a join point $U = (\xi_i, \eta_i)$ as follows:

$$p(x) = \begin{cases} \dfrac{1}{(\xi_i - x_i)^2}\Big[y_i(\xi_i - x)^2 + 2v_2(x - x_i)(\xi_i - x) \\ \qquad\qquad +\eta_i(x - x_i)^2\Big], \quad x \in [x_i, \xi_i], \\ \dfrac{1}{(x_{i+1} - \xi_i)^2}\Big[\eta_i(x_{i+1} - x)^2 + 2w_2(x - \xi_i)(x_{i+1} - x) \\ \qquad\qquad +y_{i+1}(x - \xi_i)^2\Big], \quad x \in [\xi_i, x_{i+1}] \end{cases} \tag{3.26}$$

If the first degree spline defined by the join points S, V, U, W and T is convex (concave) and/or monotone, then $p(x)$ is also convex(concave) and/or monotone.

Case 2.

In this case, L_1 and L_2 do not intersect in R. Instead both intersect the line segment M (see Figure 3.4), and the method introduces one additional knot

$$\xi_i = \frac{x_i + x_{i+1}}{2} \tag{3.27}$$

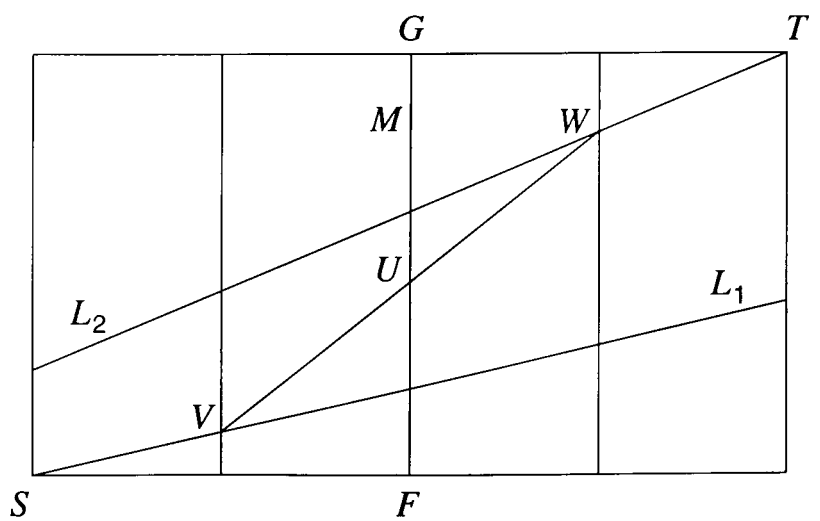

Figure 3.4 Case 2 configuration for McAllister-Roulier algorithm.

in (x_i, x_{i+1}). Then define V, W, U and the spline $p(x)$ with a common point $U = (\xi_i, \eta_i)$ on $[x_i, x_{i+1}]$ as in Case 1. Then $p(x)$ will have a continuous first derivative and preserve the shape of the data on $[x_i, x_{i+1}]$. The details of other two cases are omitted here, as they are rather complex and cannot be succinctly described with a few equations or figures. The reader should refer to the descriptions found in McAllister and Roulier [50].

3.4 Similarity Between McAllister-Roulier and Schumaker Algorithms.

In the following section, we show that the McAllister-Roulier and Schumaker algorithms produce identical interpolants if the slopes used are calculated by the Butland formula. In Section 3.3, we observed that in Case 1, from (3.23), the McAllister-Roulier algorithm selects the additional knot at

$$\xi_i = \frac{y_i - y_{i+1} + d_{i+1}x_{i+1} - d_i x_i}{d_{i+1} - d_i}$$

and in Case 2, from (3.27), at

$$\xi_i = \frac{x_i + x_{i+1}}{2},$$

which are the same as the Schumaker algorithm does in equations (3.16) and (3.17) if $d_{i+1} + d_i \neq 2\delta_i$, though the latter does not introduce any knot if $d_{i+1} + d_i = 2\delta_i$. We will now show that the ordinate η_i at ξ_i is the same for both methods when slopes are calculated by the Butland formula.

We first consider the Schumaker algorithm. In the case where $d_{i+1} + d_i = 2\delta_i$, the Schumaker algorithm does not insert a knot. If ξ_i is the knot inserted

by the McAllister-Roulier algorithm, then from (3.9), the value at ξ_i of the single polynomial used by Schumaker is

$$\eta_i = p(\xi_i) = y_i + d_i(\xi_i - x_i) + \frac{(d_{i+1} - d_i)(\xi_i - x_i)^2}{2(x_{i+1} - x_i)}$$

$$= y_i + \frac{1}{2}(d_i + d_i)(\xi_i - x_i) + \frac{(d_{i+1} - d_i)(\xi_i - x_i)^2}{2(x_{i+1} - x_i)}$$

and, since $d_i = 2\delta_i - d_{i+1}$, we have

$$\eta_i = y_i + \frac{1}{2}(2\delta_i + d_i - d_{i+1})(\xi_i - x_i) + \frac{(d_{i+1} - d_i)(\xi_i - x_i)^2}{2(x_{i+1} - x_i)} \tag{3.28}$$

Now when $d_{i+1} + d_i \neq 2\delta_i$, from (3.10), we obtain

$$\eta_i = p(\xi_i) = A_1 + B_1(\xi_i - x_i) + C_1(\xi_i - x_i)^2$$

By substituting the values of A_1, B_1 and C_1, we have

$$\eta_i = y_i + d_i(\xi_i - x_i) + \frac{1}{2}(\bar{d}_i - d_i)(\xi_i - x_i),$$

and on simplification, we get

$$\eta_i = y_i + \frac{1}{2}(\xi_i - x_i)(d_i + \bar{d}_i) \tag{3.29}$$

Now by substituting the value of $\bar{d}_i$ from(3.11) in (3.29), we obtain

$$\eta_i = y_i + \frac{(\xi_i - x_i)}{2}\left(d_i + 2\delta_i - d_{i+1} + \frac{(d_{i+1} - d_i)(\xi_i - x_i)}{x_{i+1} - x_i}\right)$$

$$= y_i + \frac{1}{2}(2\delta_i + d_i - d_{i+1})(\xi_i - x_i) + \frac{(d_{i+1} - d_i)(\xi_i - x_i)^2}{2(x_{i+1} - x_i)} \tag{3.30}$$

From (3.28) and (3.30), we conclude that Schumaker's algorithm gives

$$\eta_i = y_i + \frac{1}{2}(2\delta_i + d_i - d_{i+1})(\xi_i - x_i) + \frac{(d_{i+1} - d_i)(\xi_i - x_i)^2}{2(x_{i+1} - x_i)} \tag{3.31}$$

We now look at the McAllister-Roulier algorithm where η_i can be calculated as

$$\eta_i = L(\xi_i) = v_2 + (w_2 - v_2)\frac{\xi_i - v_1}{w_1 - v_1} \tag{3.32}$$

where

$$v_1 = \frac{x_i + \xi_i}{2}, \quad v_2 = y_i + d_i(v_1 - x_i) = y_i + \frac{1}{2}d_i(\xi_i - x_i)$$

$$w_1 = \frac{\xi_i + x_{i+1}}{2}, \quad w_2 = y_{i+1} + d_{i+1}(w_1 - x_{i+1}) = y_{i+1} + \frac{1}{2}d_{i+1}(\xi_i - x_{i+1})$$

By substituting the values of v_1, v_2, w_1 and w_2 in (3.32), we obtain

$$\eta_i = y_i + \frac{1}{2}d_i(\xi_i - x_i) + (\xi_i - x_i)\frac{w_2 - v_2}{x_{i+1} - x_i}$$

$$= y_i + (\xi_i - x_i)(\frac{d_i}{2} + \frac{w_2 - v_2}{x_{i+1} - x_i}) \tag{3.33}$$

Now consider

$$w_2 - v_2 = y_{i+1} - y_i + \frac{1}{2}d_{i+1}(\xi_i - x_{i+1}) - \frac{1}{2}d_i(\xi_i - x_i)$$

$$= y_{i+1} - y_i - \frac{1}{2}d_{i+1}(x_{i+1} - x_i) + \frac{1}{2}(d_{i+1} - d_i)(\xi_i - x_i)$$

$$\frac{w_2 - v_2}{x_{i+1} - x_i} = \delta_i - \frac{d_{i+1}}{2} + \frac{(d_{i+1} - d_i)(\xi_i - x_i)}{2(x_{i+1} - x_i)}$$

Substituting this expression into (3.33), we get

$$\eta_i = y_i + \frac{1}{2}(2\delta_i + d_i - d_{i+1})(\xi_i - x_i) + \frac{(d_{i+1} - d_i)(\xi_i - x_i)^2}{2(x_{i+1} - x_i)} \tag{3.34}$$

From (3.31) and (3.34), we see that the η_i are identical in both methods. This means that both methods select the same ordinate value for the additional inserted knot. We have then proved that both methods make use of same slopes at the original data points, select the same additional knots in the same circumstances, and use the same slope and ordinate values at the extra inserted knots. The uniqueness of the quadratic spline follows immediately from a well-known theorem on splines (see Schumaker [62], theorem 4.53, page 160).

We conclude that the McAllister-Roulier and the Schumaker algorithms are identical if the slopes in Schumaker algorithm are estimated by the Butland formula. Furthermore, when $d_{i+1} + d_i = 2\delta_i$, then there is no need for an extra knot and so, in this case, the additional knot introduced by the McAllister-Roulier algorithm is redundant.

3.5 Iterative Improvement to the Slopes.

In Section 3.2, it is shown that the Schumaker algorithm yields shape-preserving interpolants for all data types, provided the slopes required at the data points are estimated by the Butland formula (3.1). However, in some convex data cases, especially where curves with sharply varying curvature implicit in the data are to be produced, the Schumaker interpolant with Butland slopes sometimes displays excessively large local curvatures. For the purpose of illustration, the curves drawn in Figure 3.10 and Figure 3.11 are produced using Butland's slopes (full line) and Frey's improved slopes (broken line) and the corresponding jumps in the second derivative at the interior knots, denoted by J_i, are presented in Table 3.1 and Table 3.2 respectively.

	Data Points		Butland Method	Frey Method
i	x_i	y_i	J_i	J_i
1	-2.0	0.25		
2	-1.0	1.0	37.90	11.89
3	-0.3	11.1111	2222.60	803.10
4	-0.2	25.0		

Table 3.1 Jumps in the second derivative at the interior knots for Data 4.

	Data Points		Butland Method	Frey Method
i	x_i	y_i	J_i	J_i
1	0.0	19.0476		
2	0.1	7.0175	1910.17	838.43
3	0.4	3.4188	78.67	46.24
4	0.7	3.8095	125.43	25.55
5	0.8	4.7059	415.35	200.59
6	1.0	19.0476		

Table 3.2 Jumps in the second derivative at the interior knots for Data 6.

The graph shown by a full line exhibits a shape with a rapid turn and tightening effect in the region of the third interpolation point in Figure 3.10, while a similar behaviour is observed in the region of the second interpolation point in Figure 3.11 where curve is also turning sharply. The reason for these may be due to the rapid change in the slopes at those points. Also, this is in agreement with the discussion on page 12 of Lancaster and Salkauskas [47], which states that if the curvature value is large at x_i, then the slope of the curve is turning 'rapidly' with x as x increases through x_i. In that sense, the shape of the curve is tightened at x_i. In the case of the curvature value being small at x_i, then the slope of the curve is turning 'slowly' with x as x increases through x_i. As a result, 'flat spots' are produced on the curve

corresponding to almost zero curvature values. The curves drawn by a broken line display a visually pleasing shape of the interpolants, particularly in the region of the second and third interpolation points in Figure 3.10 and Figure 3.11 where improvement in the shape of curve segments can be clearly seen, than those obtained by a full line. The reason may be due to the fact that the Frey method of improving the slopes keeps the jumps in the second derivative at the knots small in comparision with the Butland method, as shown in Table 3.1 and Table 3.2, and hence produces more visually pleasing shapes of the curves.

In this section, the Butland slope calculation method is further developed with an iterative technique to overcome the above problem. For this purpose, we adopt an iterative improvement method for slopes, due to Frey [31], where Butland slopes can be employed as an initial guess. Like the Butland slope method, the Frey technique is also local, and the slope of the curve at a particular data point is improved iteratively by influencing the slopes of two other data points in its vicinity (the slope at data point in question is taken as a central point with one data point on each side).

Frey proposes a method of iterative improvement of the slopes applicable to successive initial guesses of slopes at given data points. He assumes a convex data set consisting of three consecutive data points $(x_{i-1}, y_{i-1}), (x_i, y_i)$ and (x_{i+1}, y_{i+1}). The paper describes his method in great detail, but here we simply list the iterative process as follows. First compute

$$A_1 = \delta_{i-1}, \qquad A_2 = \delta_i, \qquad A_3 = \frac{h_i\delta_i + h_{i-1}\delta_{i-1}}{h_i + h_{i-1}}$$

$$B_1 = A_2 - d_{i+1}, \qquad B_2 = d_{i-1} - A_1, \qquad B_3 = A_1 + A_3$$

$$C_1 = \frac{h_{i-1}}{h_{i-1}+h_i} B_2(A_1 - A_3), \qquad C_2 = \frac{h_i}{h_{i-1}+h_i} B_1$$

$$E_1 = C_1 d_{i+1} + B_3 C_2 d_{i-1} - 2A_1 A_2 C_2, \qquad E_2 = C_1 + 2C_2 d_{i-1} - B_3 C_2$$

Then d_i is computed as

$$d_i = \frac{E_1}{E_2} \tag{3.35}$$

We use the Butland slope formula (3.1) for the initialization of slopes required in the above steps, and apply (3.35) iteratively to each slope in succession to produce improved estimates. Frey does not perform convergence analysis; however, in applications his scheme converges adequately within ten to fifteen iterations through the data points.

3.6 Numerical Examples and Conclusions.

The algorithm described in the preceding section has been tested on the six data sets which have been already discussed in Chapter 1. The corresponding plots of the interpolants, showing the data points marked by circles (o), are presented in Figures 3.7 through 3.12. Data points, slopes, additional inserted knots and case numbers are listed in Tables 3.3 through 3.8. The additional (unnecessary) knots introduced by the McAllister-Roulier algorithm and not by the Schumaker algorithm are marked with an asterisk (*). We observe that the algorithm generates visually pleasing interpolants automatically as shown in Figures 3.7 to 3.12. Comparing Figure 3.7 with the corresponding Figures 3.5 and 3.6 generated by the Schumaker [61], where three adjustments to the slopes are made to obtain a similar shape, we see that the present algorithm behaves very well.

In Figure 3.10, the curve obtained using Butland's slopes (solid line) and the curve achieved by the iterative scheme (dotted line) are displayed. The two curves lie almost on top of one another, apart from the interval [-1.0, -0.3] where the curve produced with Butland slopes is tighter than that generated by the iterative scheme. Similarly, in Figure 3.11, the two curves agree very closely with one another except in the interval [0.1, 0.4], where the curve displayed with Butland slopes is tighter than the curve produced by the iterative scheme.

We have shown that the Schumaker algorithm using the Butland slope formula generates a shape-preserving interpolant in one pass and is identical to the McAllister-Roulier algorithm if the same slopes are used in each. Schumaker's algorithm has advantages over the McAllister-Roulier algorithm, as it is much simpler to implement and requires less storage since it introduces extra knots only in those intervals where $d_i + d_{i+1} \neq 2\delta_i$, whereas the latter method always generates one additional knot per data interval.

The Schumaker algorithm can be used in conjunction with the Roulier [58] method for shape-preserving surface interpolation. This is presented in Chapter 6.

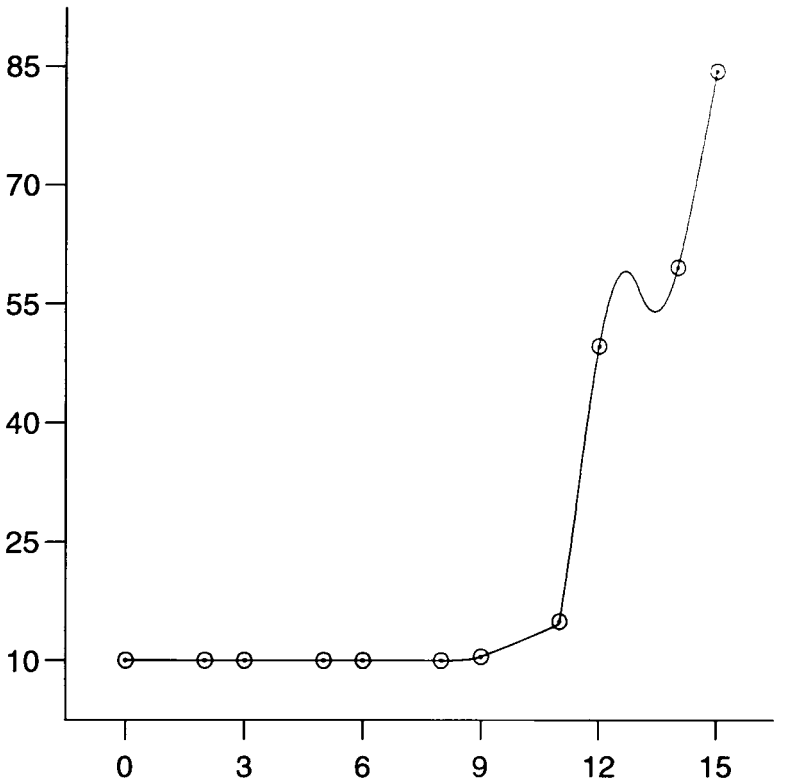

Figure 3.5 Schumaker algorithm with initial slopes estimates.

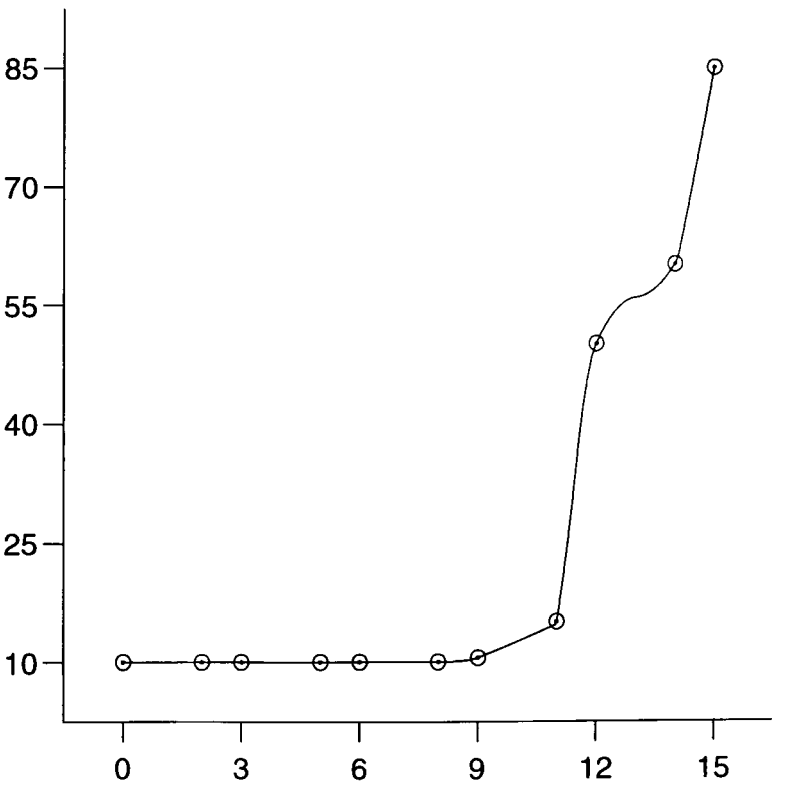

Figure 3.6 Schumaker algorithm with improved slopes estimates.

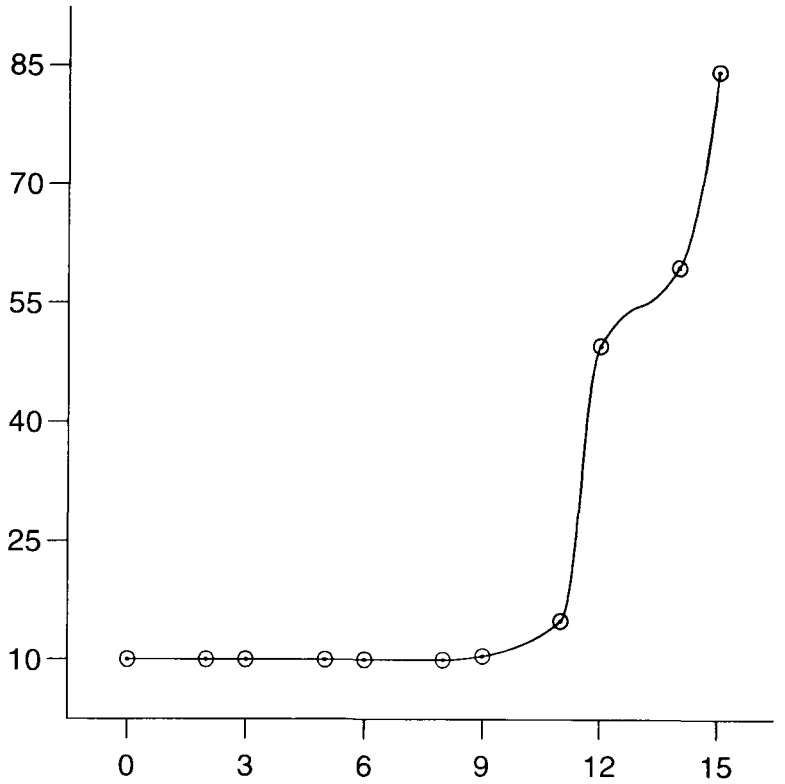

Figure 3.7 Shape-preserving quadratic spline interpolant to Data 1.

	Data Points		Slopes	Additional Knots	Case on
i	x_i	y_i	d_i	ξ_i	$[x_i, x_{i+1}]$
1	0.0	10.0	0.0	1.0*	2
2	2.0	10.0	0.0	2.5*	2
3	3.0	10.0	0.0	4.0*	2
4	5.0	10.0	0.0	5.5*	2
5	6.0	10.0	0.0	7.0*	2
6	8.0	10.0	0.0	8.389	1
7	9.0	10.5	0.8182E+00	10.160	1
8	11.0	15.0	0.4228E+01	11.5	2
9	12.0	50.0	0.8750E+01	13.0	2
10	14.0	60.0	0.8333E+01	14.5*	1
11	15.0	85.0	0.4167E+02		

Table 3.3 The values associated with Figure 3.7.

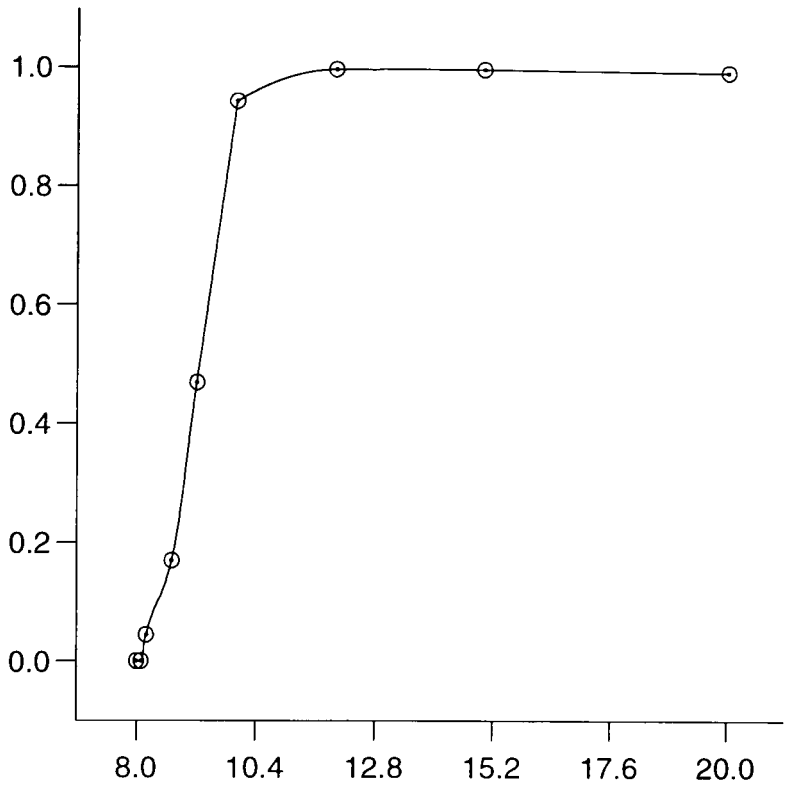

Figure 3.8 Shape-preserving quadratic spline interpolant to Data 2.

	Data Points		Slopes	Additional Knots	Case on
i	x_i	y_i	d_i	ξ_i	$[x_i, x_{i+1}]$
1	7.99	0.0	0.3750E-06	8.040*	1
2	8.09	2.76429E-05	0.5725E-03	8.140	2
3	8.19	4.37498E-02	0.3148E+00	8.445	2
4	8.70	0.169183	0.3490E+00	8.950	2
5	9.20	0.469428	0.5967E+00	9.994	1
6	10.0	0.943740	0.5247E-01	11.031	1
7	12.0	0.998636	0.8422E-03	13.471	1
8	15.0	0.999919	0.2898E-04	17.499*	1
9	20.0	0.999994	0.1016E-05		

Table 3.4 The values associated with Figure 3.8.

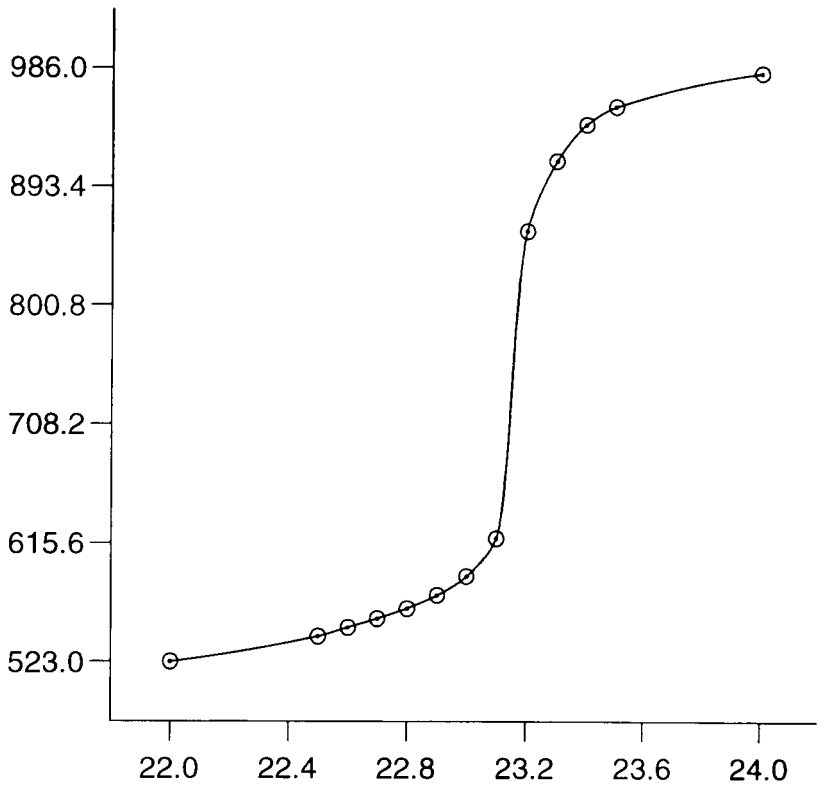

Figure 3.9 Shape-preserving quadratic spline interpolant to Data 3.

	Data Points		Slopes	Additional Knots	Case on
i	x_i	y_i	d_i	ξ_i	$[x_i, x_{i+1}]$
1	22.0	523.0	0.2909E+02	22.250*	1
2	22.5	543.0	0.5091E+02	22.550	2
3	22.6	550.0	0.7000E+02	22.650	2
4	22.7	557.0	0.7467E+02	22.763	1
5	22.8	565.0	0.8889E+02	22.864	1
6	22.9	575.0	0.1200E+03	22.963	1
7	23.0	590.0	0.2000E+03	23.070	1
8	23.1	620.0	0.5333E+03	23.150	2
9	23.2	860.0	0.8949E+03	23.233	1
10	23.3	915.0	0.3798E+03	23.353	1
11	23.4	944.0	0.1888E+03	23.455	1
12	23.5	958.0	0.8000E+02	23.750*	1
13	24.0	986.0	0.3200E+02		

Table 3.5 The values associated with Figure 3.9.

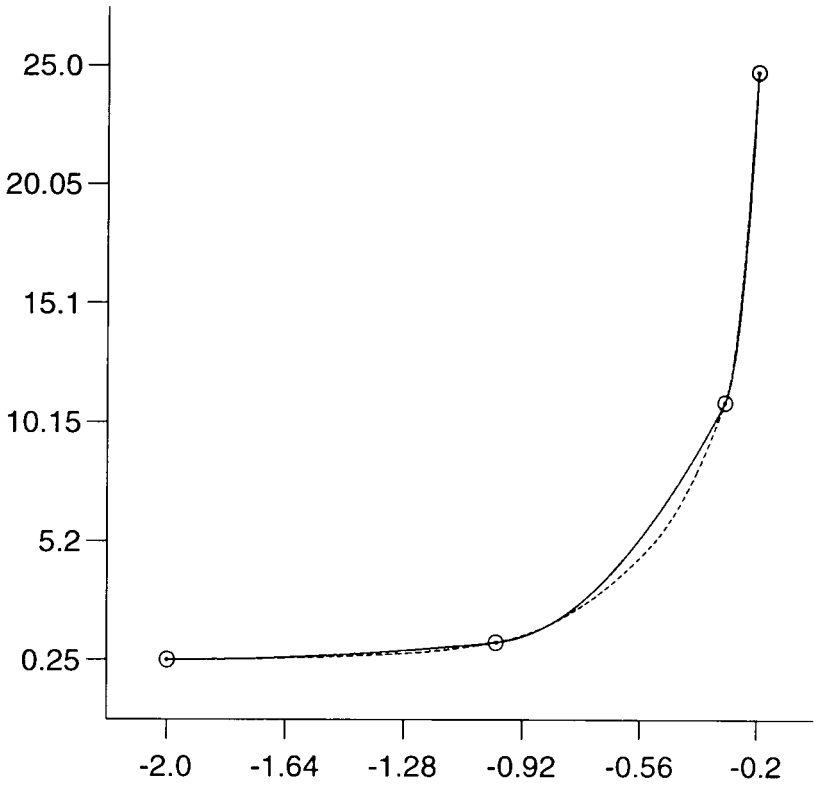

Figure 3.10 Shape-preserving quadratic spline interpolant to Data 4.

	Data Points		Slopes	Additional Knots	Case on
i	x_i	y_i	d_i	ξ_i	$[x_i, x_{i+1}]$
1	-2.0	0.2500	0.7404E-01	-1.500^*	1
2	-1.0	1.0000	0.1426E+01	-0.668	1
3	-0.3	11.1111	0.2617E+02	-0.250^*	1
4	-0.2	25.0000	0.2516E+03		

Table 3.6 The values associated with Figure 3.10.

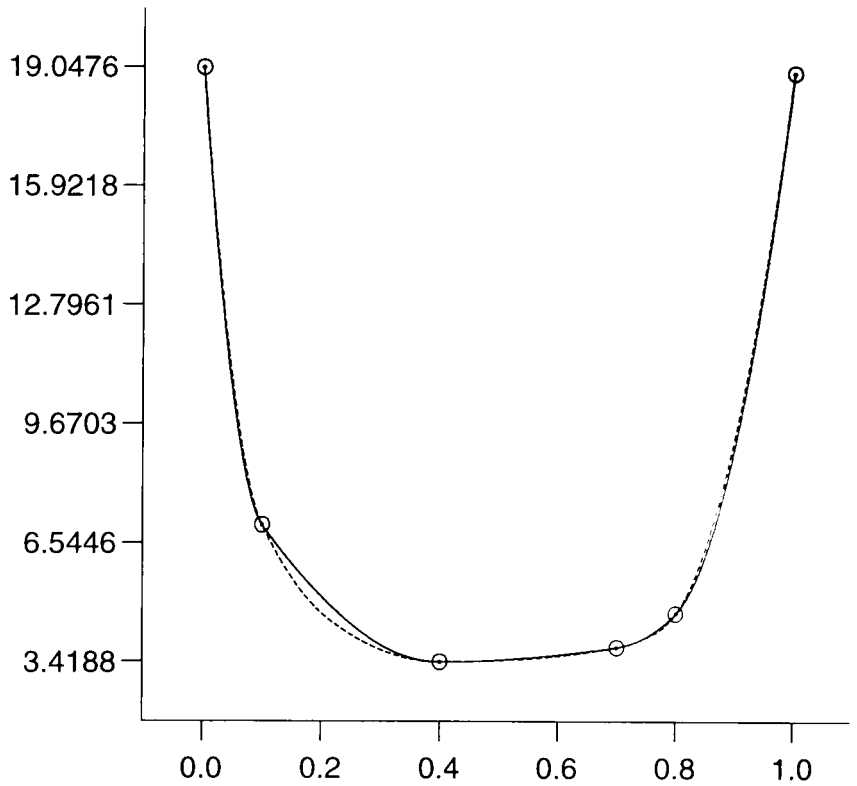

Figure 3.11 Shape-preserving quadratic spline interpolant to Data 6.

	Data Points		Slopes	Additional Knots	Case on
i	x_i	y_i	d_i	ξ_i	$[x_i, x_{i+1}]$
1	0.0	19.0476	-0.2188E+03	0.050*	1
2	0.1	7.0175	-0.2182E+02	0.265	1
3	0.4	3.4188	0.0	0.528	1
4	0.7	3.8095	0.2274E+01	0.751	1
5	0.8	4.7059	0.1594E+02	0.900*	1
6	1.0	19.0476	0.1275E+03		

Table 3.7 The values associated with Figure 3.11.

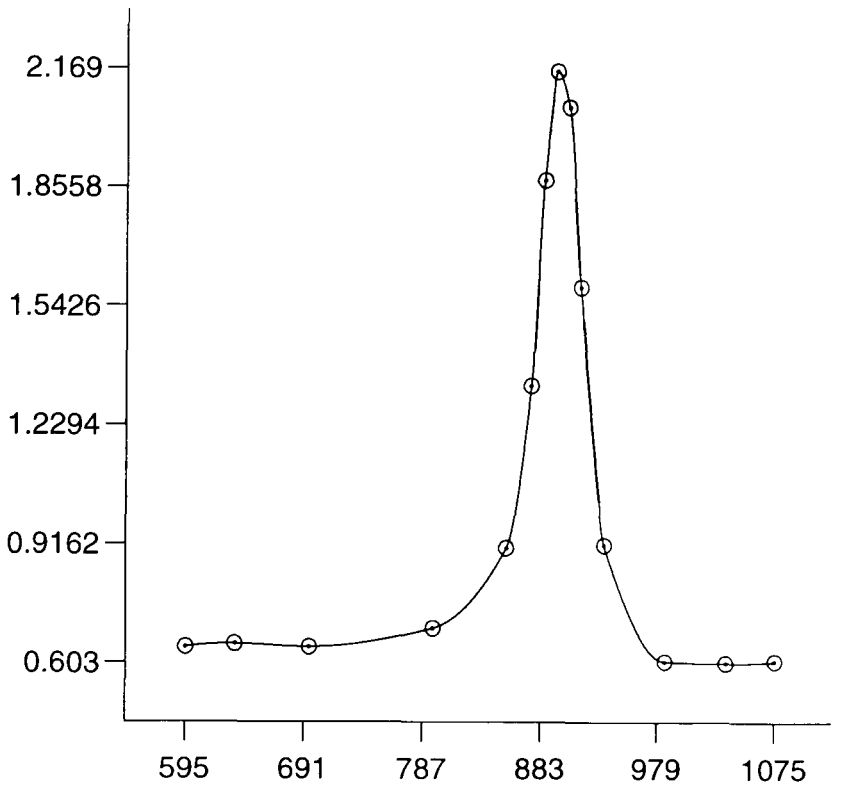

Figure 3.12 Shape-preserving quadratic spline interpolant to Data 5.

	Data Points		Slopes	Additional Knots	Case on
i	x_i	y_i	d_i	ξ_i	$[x_i, x_{i+1}]$
1	595.0	0.644	0.4000E-03	615.000*	1
2	635.0	0.652	0.0	665.000	2
3	695.0	0.644	0.0	737.958	1
4	795.0	0.694	0.8765E-03	824.243	1
5	855.0	0.907	0.6092E-02	862.560	1
6	875.0	1.336	0.3078E-01	880.000	2
7	885.0	1.881	0.3769E-01	892.642	1
8	895.0	2.169	0.0	899.015	1
9	905.0	2.075	-0.1570E-01	910.000	2
10	915.0	1.598	-0.3977E-01	931.131	1
11	935.0	0.916	-0.1046E-01	964.217	1
12	985.0	0.607	-0.1580E-03	1010.324	1
13	1035.0	0.603	0.0	1055.000*	1
14	1075.0	0.608	0.2500E-03		

Table 3.8 The values associated with Figure 3.12.

Chapter 4

An Algorithm for Shape-Preserving Curve Drawing

In this chapter we develop a new automatic algorithm for interpolating the data using local procedures. The algorithm is arranged to produce monotone and/or convex interpolants using C^1 cubic splines when the data have the corresponding properties by means of a robust slope algorithm and is designed to deal with the data points (x_i, y_i), $i = 1, ..., n$, and so is comparable with the methods presented in Chapter 2. We also give an analysis of the slope parameter t that provides a quantitative means by which the derivative values d_i that control the shape of the curve are reliably generated without experimentation or interactive user direction. It is assumed that the data is monotonic in x but the y values are arbitrary. In Section 4.1, we introduce a new slope estimation method which provides the basis of the new algorithm and discuss the derivation and analysis of the algorithm in Section 4.2. Finally, conclusions and several numerical examples illustrating the inherent superiority of the new algorithm to the existing methods are described in Section 4.3.

4.1 New Generalized Slope Estimation Method.

Here, we describe a new slope estimation method which generalizes the harmonic mean, satisfies condition (2.4) and produces more visually pleasing

curves. This method uses the minimum number of data values for assigning the slope at each data point, that is, the coordinates of the point itself and one point on either side. The idea is motivated by the introduction of slope estimation formulae given in (2.6) through (2.12) and is derived from Hardy, Littlewood and Polya [39], where they define the general Harmonic mean as

$$\left(\frac{\sum_{i=1}^{m} w_i}{\sum_{i=1}^{m} w_i \cdot \frac{1}{\delta_i^t}} \right)^{\frac{1}{t}}$$

By considering the case $m = 2$, our method calculates the derivative values by using:

$$d_i = \begin{cases} \dfrac{(w_1 + w_2)^{\frac{1}{t}} \delta_{i-1} \delta_i}{(w_1 \delta_{i-1}^t + w_2 \delta_i^t)^{\frac{1}{t}}}, & \text{if } \delta_{i-1}\delta_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

where t is a non-negative real number and, w_1 and w_2 are positive weighting factors. Here, when $w_1 = 1, w_2 = 1$ and $t = 1$, we have the Butland original formula (2.6). Also note that with the restrictions of $w_1 = 1, w_2 = 2$ and $t = 1$, we get the Fritsch-Butland formula (2.8). Similarly, by choosing the appropriate values for w_1, w_2 and taking $t = 1$, the formulas (2.7) and (2.9) for the Brodlie and Costantini's methods can be easily deduced.

4.2 An Automatic Algorithm Using Cubic Splines.

Here, we present an algorithm which preserves both monotonicity and convexity of the data using C^1 cubic splines. Our algorithm is based on a new slope estimation formula which is deduced from (4.1) and computes the

derivative by using:

$$d_i = \begin{cases} \dfrac{2^{\frac{1}{t}}\delta_{i-1}\delta_i}{(\delta_{i-1}^t + \delta_i^t)^{\frac{1}{t}}}, & \text{if } \delta_{i-1}\delta_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

where $t > 0$ is chosen to be as small as possible, while still ensuring that d_i maintains monotonicity and convexity, whenever the data is monotone and convex. We have plotted the characteristics of this formula given various slopes and some values of t in Figure 4.1. Consider

$$d_i = \frac{2^{\frac{1}{t}}u}{(1+r^t)^{\frac{1}{t}}} \tag{4.3}$$

where $u = \min(\delta_{i-1}, \delta_i), L = \max(\delta_{i-1}, \delta_i)$, and $r = u/L$. Then d_i/L will be normalized between 0 and 1. Note that as t decreases, it will tend to favour the larger δ_i. That is, d_i will lie closer to L than to u. This, in turn, creates larger values of (α_i, β_i) and spreads these throughout the entire region S of Figure 2.1 more completely. The advantage of this method appears to be that it produces smaller second derivative discontinuities than the corresponding methods described in Section 2.3 of Chapter 2. As an illustration, Data 4 is used for comparison purposes. The jumps in the second derivative at the interior data points, denoted by J_i, generated by the new method with $t = 0.3$ and other methods are shown in Table 4.1. Comparing the discontinuities given in Table 2.2 of Chapter 2 and Table 4.1 clearly shows that the new method reduces these as compared to all other methods and hence will produce more visually pleasing curves. Similar comparisons also apply to the other data sets.

Conceptually, a "visually pleasing curve" is a curve that looks good to the human eye and it can not be described in precise mathematical terms. This definition is, of course, purely subjective as it is impossible to construct a curve

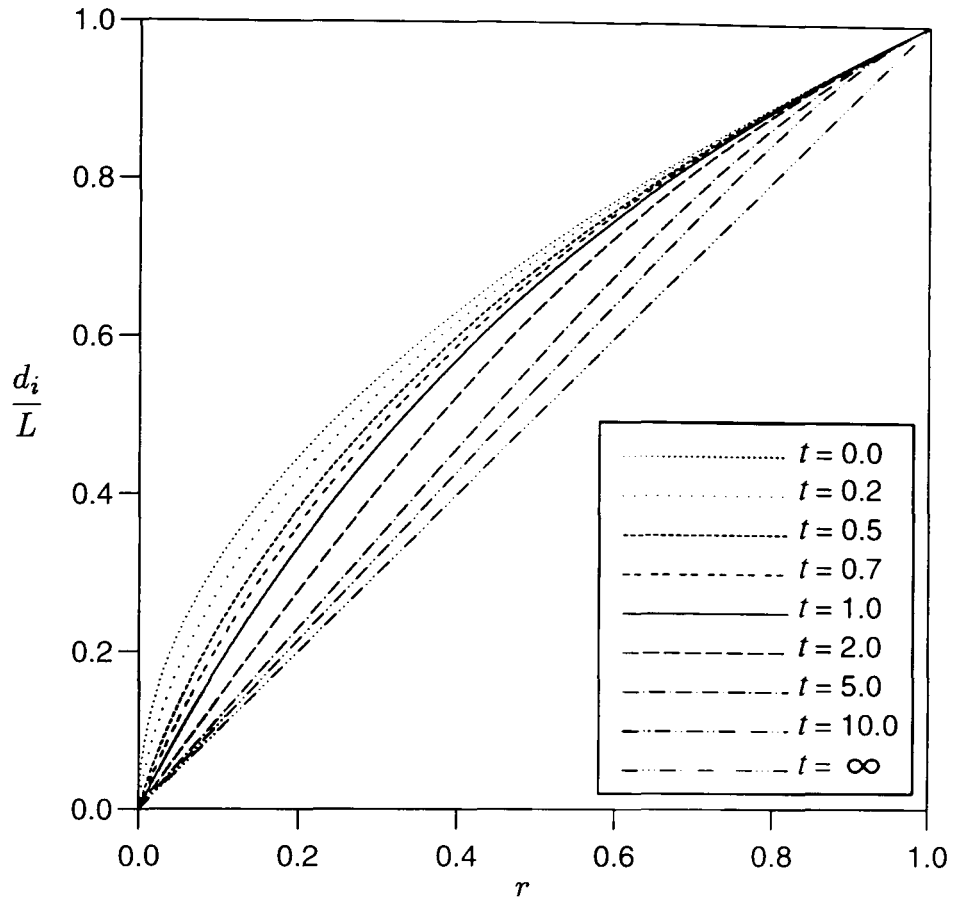

Figure 4.1 Visualization of slope formula with different values of t.

i	Data Points x_i	y_i	Butland Method J_i	Brodlie Method J_i	Huynh Method J_i	Fritsch-Butland method J_i	New Method with $t = 0.3$ J_i
1	-2.0	0.25					
2	-1.0	1.0	39.69	19.89	13.52	6.02	0.94
3	-0.3	11.1111	4167.96	3863.23	3829.91	3722.57	3665.67
4	-0.2	25.0					

Table 4.1 Jumps in the second derivative at the interior knots.

to everyone's liking. However, this property may be very important to designers, as they would like to get visually pleasing curves for their work.

Now we prove that formula (4.2) satisfies inequality (2.13). Suppose $0 \leq \delta_{i-1} \leq \delta_i$. Then, for $t > 0$

$$\delta_{i-1} = \frac{2^{\frac{1}{t}}\delta_{i-1}\delta_i}{2^{\frac{1}{t}}\delta_i} = \frac{2^{\frac{1}{t}}\delta_{i-1}\delta_i}{(2\delta_i^t)^{\frac{1}{t}}} \leq \frac{2^{\frac{1}{t}}\delta_{i-1}\delta_i}{(\delta_{i-1}^t + \delta_i^t)^{\frac{1}{t}}}$$

$$\Rightarrow \qquad \delta_{i-1} \leq d_i \tag{4.4}$$

Now

$$\delta_i = \frac{2^{\frac{1}{t}}\delta_{i-1}\delta_i}{2^{\frac{1}{t}}\delta_{i-1}} = \frac{2^{\frac{1}{t}}\delta_{i-1}\delta_i}{(2\delta_{i-1}^t)^{\frac{1}{t}}} \geq \frac{2^{\frac{1}{t}}\delta_{i-1}\delta_i}{(\delta_{i-1}^t + \delta_i^t)^{\frac{1}{t}}}$$

$$\Rightarrow \qquad d_i \leq \delta_i \tag{4.5}$$

Combining (4.4) and (4.5), we obtain

$$\delta_{i-1} \leq d_i \leq \delta_i \tag{4.6}$$

A similar result holds for $0 \leq \delta_i \leq \delta_{i-1}$. That is

$$\delta_i \leq d_i \leq \delta_{i-1} \tag{4.7}$$

Hence from (4.6) and (4.7), we conclude that $\min(\delta_{i-1}, \delta_i) \leq d_i \leq \max(\delta_{i-1}, \delta_i)$.

Now we show some results that are useful for determining the limits on t in order to get shape-preserving interpolants using cubic splines. By considering $p'(x)$ of (2.2), we see that it has a unique extremum at

$$x^* = x_i + \frac{(x_{i+1} - x_i)(2d_i + d_{i+1} - 3\delta_i)}{3(d_i + d_{i+1} - 2\delta_i)} \tag{4.8}$$

Hence, if $x^* \notin (x_i, x_{i+1})$, convexity would be satisfied. That is, for $x^* < x_i$

$$\frac{(x_{i+1} - x_i)(2d_i + d_{i+1} - 3\delta_i)}{3(d_i + d_{i+1} - 2\delta_i)} \leq 0 \tag{4.9}$$

or $2d_i + d_{i+1} - 3\delta_i \leq 0$ when $d_i + d_{i+1} - 2\delta_i > 0$ and, for $x^* > x_{i+1}$, then

$$\frac{(x_{i+1} - x_i)(2d_i + d_{i+1} - 3\delta_i)}{3(d_i + d_{i+1} - 2\delta_i)} \geq (x_{i+1} - x_i) \tag{4.10}$$

or $d_i + 2d_{i+1} - 3\delta_i \leq 0$ when $d_i + d_{i+1} - 2\delta_i > 0$. If $d_i + d_{i+1} - 2\delta_i \leq 0$, a similar result holds. This can be summarized in terms of $\alpha_i = \frac{d_i}{\delta_i}$ and $\beta_i = \frac{d_{i+1}}{\delta_i}$ in the following lemmas:

Lemma 4.2.1

If $\alpha_i + \beta_i - 2 > 0$ then $p(x)$ is convex on $[x_i, x_{i+1}]$ if and only if either:

(i) $2\alpha_i + \beta_i - 3 \leq 0$ or

(ii) $\alpha_i + 2\beta_i - 3 \leq 0$

Lemma 4.2.2

If $\alpha_i + \beta_i - 2 \leq 0$ then $p(x)$ is convex on $[x_i, x_{i+1}]$ if and only if either:

(i) $2\alpha_i + \beta_i - 3 \geq 0$ or

(ii) $\alpha_i + 2\beta_i - 3 \geq 0$

Using these two lemmas and the condition that (α_i, β_i) lie in the region S of Figure 4.2, we now have the following theorem:

Theorem 4.2.1

If α_i, β_i are both positive and either Lemma 4.2.1 or Lemma 4.2.2 is satisfied, then $p(x)$ maintains both monotonicity and convexity on $[x_i, x_{i+1}]$.

The proof of this follows by observing that if $\text{sign}(\alpha_i) = \text{sign}(\beta_i) \geq 0$ then $\text{sign}(d_i) = \text{sign}(d_{i+1}) = \text{sign}(\delta_i)$. Also, Lemma 4.2.1 and Lemma 4.2.2 provide the sufficient conditions for Theorem 2.2.1, hence monotonicity is satisfied.

The above provides us with regions of convexity and monotonicity that are shown in Figure 4.2, where the horizontally hatched area shows the convexity region and the shaded area with dots shows the monotonicity region.

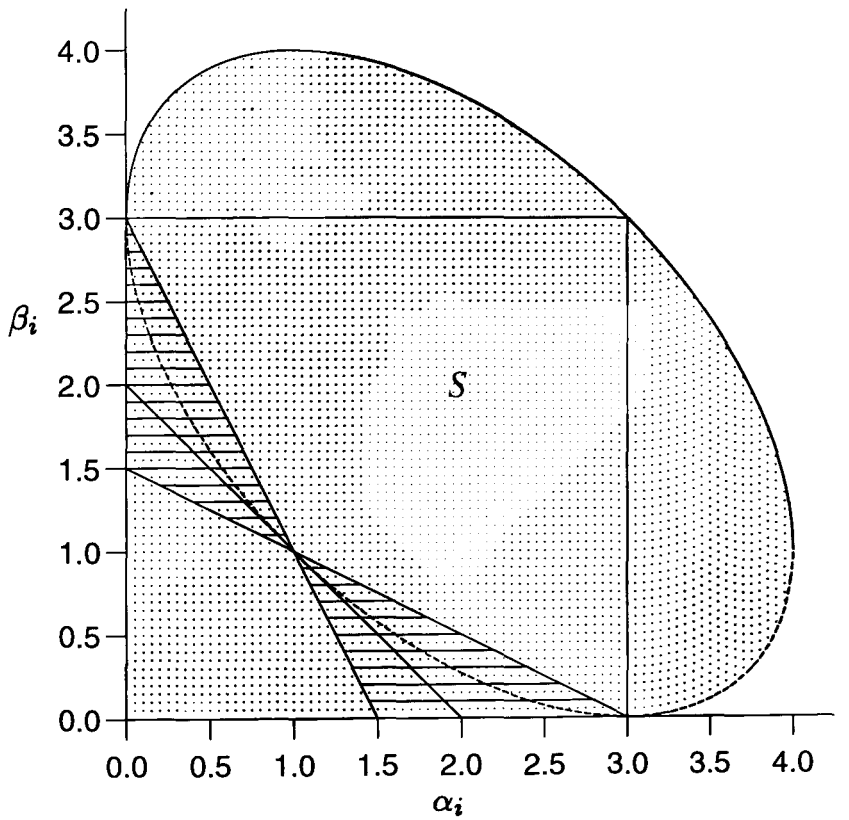

Figure 4.2 Monotonicity region S (shaded with dots) and convexity region (hatched horizontally).

By considering $d_i \leq \beta_i \delta_i$ $(= d_{i+1})$ for both $\alpha_i, \beta_i \geq 0$, then where given α_i, β_i is computed as follows:

(I) If $[y_{i-1}, y_{i+2}]$ forms a convex data set (i.e., $(\delta_{i-1} - \delta_i)(\delta_i - \delta_{i+1}) > 0$),

then by Theorem 4.2.1:

$$\begin{cases} \text{If} \quad \alpha_i \leq 1 \text{ then} \quad \frac{1}{2}(3-\alpha_i) \leq \beta_i \leq 3-2\alpha_i, \\ \text{If} \quad \alpha_i > 1 \text{ then} \quad 3-2\alpha_i \leq \beta_i \leq \frac{1}{2}(3-\alpha_i). \end{cases} \tag{4.11}$$

(II) If $[y_{i-1}, y_{i+2}]$ forms a non-convex monotone data set, then set

$$\beta_i = \frac{1}{2}\Big(6-\alpha_i+\sqrt{3\alpha_i(4-\alpha_i)}\Big) \tag{4.12}$$

This follows from Theorem 2.2.1 (ii.iii) of Chapter 2, which leads to a quadratic equation in terms of β_i and after some simplifications, is given in the form:

$$(\beta_i-1)^2+(\alpha_i-4)(\beta_i-1)+(\alpha_i^2-5\alpha_i+4)=0$$

In fact, β_i in (4.12) is the larger root of this equation and represents the upper half of the elliptical boundary drawn by a full line in Figure 4.2, covering the entire monotonicity region S.

Now, in order to establish a bound on the value of t in (4.3), let $r = \min(\delta_i, \delta_{i+1})/\max(\delta_i, \delta_{i+1})$ and

$$a = \begin{cases} 1 & \text{if } \delta_{i+1} \geq \delta_i \\ r & \text{if } \delta_{i+1} < \delta_i \end{cases} \tag{4.13}$$

and

$$\frac{d_i}{\delta_i} = \frac{2^{\frac{1}{t}}a}{(1+r^t)^{\frac{1}{t}}},$$

then $\dfrac{d_i}{\delta_i} \leq 2^{\frac{1}{t}}a \leq \beta_i$. That is,

$$t \geq \frac{\ln 2}{\ln\beta_i - \ln a} \tag{4.14}$$

As an example of maintaining monotonicity, consider the case of monotone data with $\max(\beta_i) = 4$ when $\alpha_i = 1$ and $t = 0.5$, hence from (4.3), we have

$$\lim_{\substack{r\to 1\\ t=0.5}} \frac{2^{\frac{1}{t}}u}{(1+r^t)^{\frac{1}{t}}} = \frac{4u}{4} = u$$

also

$$\lim_{\substack{r\to 0\\ t=0.5}} \frac{2^{\frac{1}{t}}u}{(1+r^t)^{\frac{1}{t}}} = \frac{4u}{1} = 4u$$

If $u = \delta_i$, then, since $\alpha_i = 1$, $\beta_i = d_i/\delta_i = 4u/u = 4$, which is within the monotonicity region. If $u = \delta_{i+1}$ then $\delta_i > \delta_{i+1}$ and $\beta_i = d_i/\delta_i = 4\delta_{i-1}/\delta_i < 4$, which again is within the necessary region. Similar results hold for convexity in the case of convex data by considering $\max(\beta_i) = 3$ when $t = \dfrac{\ln 2}{\ln 3}$.

In the above discussion, we have assumed that the data are monotone increasing and convex. The case of monotone decreasing and convex data can be treated in a similar manner. For these data, we only need to use $|\delta_i|$ rather than δ_i when applying the formula (4.14). Having computed the derivative value d_i in this way, only its sign is chosen to be negative for the monotone decreasing data ($\delta_i < 0$). This approach has been followed for Data 6, which is comprised of a mixture of monotone decreasing, monotone increasing and convex data segments. The resulting curve is displayed in Figure 4.7 in the next Section 4.3.

For practical purposes, it is worthwhile mentioning here that when t tends to ∞, then we have (see, [39]):

$$d_i = \lim_{t\to\infty} \frac{2^{\frac{1}{t}}u}{(1+r^t)^{\frac{1}{t}}} = u \tag{4.15}$$

The above proceeding can now be summarized by presenting an algorithm for estimating the derivative values d_i at each data point. We refer the algorithm as Algorithm SLOP.

Algorithm SLOP.

Given $\delta_{i-1}, \delta_i, \delta_{i+1}$ and d_1, then the derivative values d_i, $i = 2, ..., n-1$ are estimated as follows:

Step 1

Set $\alpha_i = \dfrac{d_{i-1}}{\delta_{i-1}}$

Step 2

If $(\delta_{i-1} - \delta_i)(\delta_i - \delta_{i+1}) > 0$ then

Compute β_i to satisfy (4.11) (convex).

Else

Compute β_i to satisfy (4.12) (monotone).

Step 3

Choose a as in (4.13) and then set

$t = \dfrac{\ln 2}{\ln \beta_i - \ln a}$ to satisfy (4.14).

Step 4

If $\delta_i \delta_{i+1} \leq 0$ then

Set $d_i = 0$

Else

If $t = \infty$ then

$d_i = u$

Else

Set $d_i = \dfrac{2^{\frac{1}{t}} u}{(1 + r^t)^{\frac{1}{t}}}$

The Algorithm SLOP described above only provides a means for setting interior derivative values d_i. For the end point derivatives, d_1 and d_n, we shall use the method proposed by McAllister-Roulier [50]. This has been discussed in Section 3.1 (see (3.7) and (3.8)). The McAllister-Roulier method is robust for calculating the derivative values at the end points, produces good results in all cases and automatically satisfies the required shape-preserving conditions at the extreme points. This is the scheme adopted for all curves drawn in the next Section 4.3.

Now, an algorithm for constructing the interpolating curve using the piecewise cubic Hermite interpolant (2.2) according to proposals discussed above is outlined.

Algorithm.

Step 1

Input number of data points, n.

Step 2

Input data $\{x_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$.

Step 3

Calculate the derivative values d_i by applying the
Algorithm SLOP mentioned above.

Step 4

For i=1 to n-1 do

Use equation (2.2) to generate the curve segment
on the subinterval $[x_i, x_{i+1}]$.

Display the resulting curve.

4.3 Numerical Examples and Conclusions.

In this section, our algorithm is applied to a selection of standard data sets introduced earlier in Chapter 1 and the results are compared with those obtained using the existing local methods described in Chapter 2. The basis of comparison is a collection of data sets drawn from existing literature on the comparison of shape-preserving interpolation methods. It follows that the comparison is unbiased to the methods since it utilises established data sets.

The algorithm introduced above has been applied to the data sets Data 1 through Data 4. The resulting plots of the interpolating curves are shown in Figures 4.3 through 4.6, and each figure is accompanied by a table detailing the data points, slopes and corresponding selected values for the parameter t. It can be observed in Table 4.4 that the new algorithm has selected $t = \infty$ at the data point $x_4 = 22.7$. This is due to the fact that points 22.5, 22.6 and 22.7 are collinear and are part of a convex region between points 22.0 and 23.2. As a consequence, the Algorithm SLOP computes $\alpha_4 = 1$ in Step 1, $\beta_4 = 1$ in Step 2 and $a = 1$ in Step 3 respectively and then $t = \infty$ is obtained from (4.14). Several other examples taken from various sources have also been worked out and in all of these cases, we have not encountered $t = \infty$ except for the above Data 3. However, the selection of $t = \infty$ to estimate the derivative value d_i at any data point does not present a problem as is evident from (4.15) which establishes $d_i = u = \min(\delta_{i-1}, \delta_i)$ and this always restricts the values of (α_i, β_i) to be within the region S of Figure 4.2.

As a comparison, the interpolating curves shown in Figure 4.3 to Figure 4.6 are compared to the corresponding curves drawn in Figure 2.2 through Figure 2.33 which were obtained by applying the different methods described

in Chapter 2. A study of these curves indicates that the new algorithm produces curves that are smooth, shape-preserving, more visually pleasing and offers substantial confirmation of the superiority of the new algorithm. The new algorithm also has an advantage that it does not insert additional knots between the original data points in order to preserve the shape characteristics of the given data.

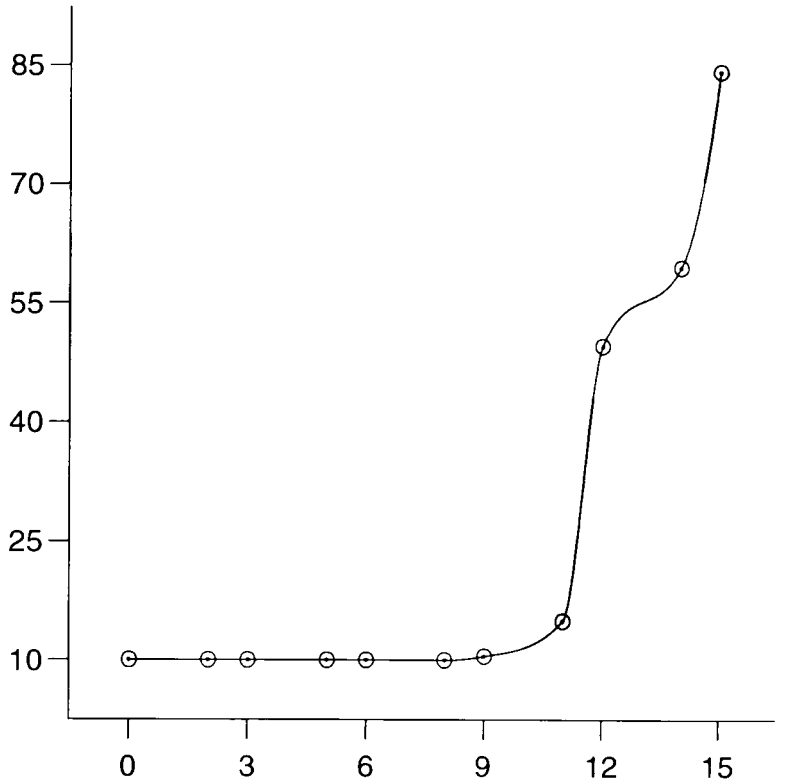

Figure 4.3 New automatic algorithm interpolating Data 1.

	Data Points		Slopes	Value of t on
i	x_i	y_i	d_i	$[x_i, x_{i+1}]$
1	0.0	10.0	0.0	
2	2.0	10.0	0.0	
3	3.0	10.0	0.0	
4	5.0	10.0	0.0	
5	6.0	10.0	0.0	
6	8.0	10.0	0.0	
7	9.0	10.5	0.8930E+00	0.631
8	11.0	15.0	0.5661E+01	0.515
9	12.0	50.0	0.1196E+02	0.215
10	14.0	60.0	0.9400E+01	0.553
11	15.0	85.0	0.4060E+02	

Table 4.2 The values used in the construction of Figure 4.3.

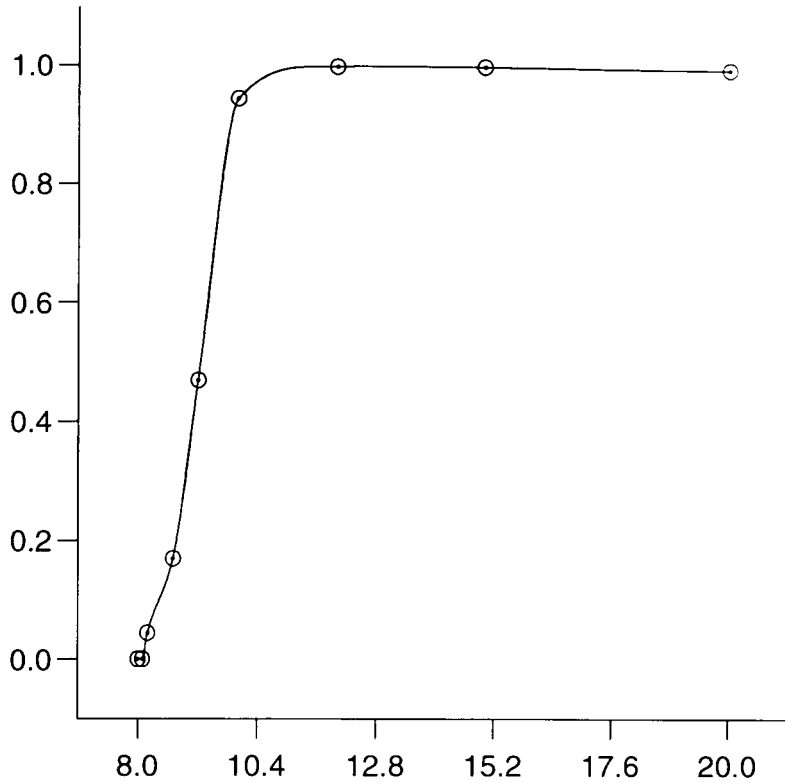

Figure 4.4 New automatic algorithm interpolating Data 2.

	Data Points		Slopes	Value of t on
i	x_i	y_i	d_i	$[x_i, x_{i+1}]$
1	7.99	0.0	0.0	
2	8.09	2.76429E-05	0.1089E-02	0.500
3	8.19	4.37498E-02	0.3225E+00	0.407
4	8.70	0.169183	0.3657E+00	0.503
5	9.20	0.469428	0.5967E+00	1.174
6	10.0	0.943740	0.9823E-01	0.226
7	12.0	0.998636	0.2546E-02	0.139
8	15.0	0.999919	0.6024E-04	0.207
9	20.0	0.999994	0.0	

Table 4.3 The values used in the construction of Figure 4.4.

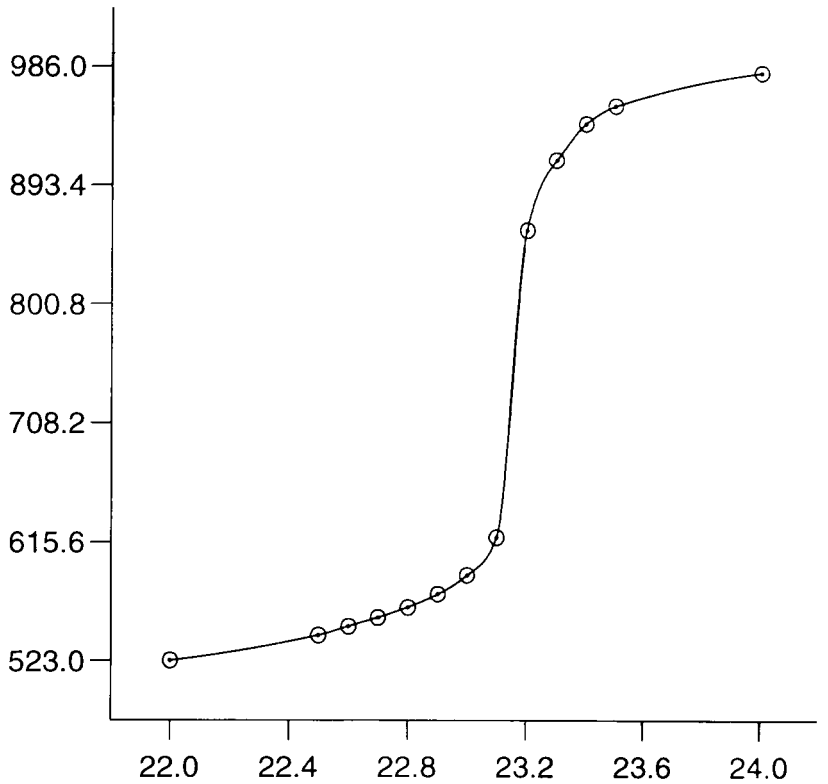

Figure 4.5 New automatic algorithm interpolating Data 3.

	Data Points		Slopes	Value of t on
i	x_i	y_i	d_i	$[x_i, x_{i+1}]$
1	22.0	523.0	0.2811E+02	
2	22.5	543.0	0.5189E+02	0.503
3	22.6	550.0	0.7000E+02	0.502
4	22.7	557.0	0.7000E+02	∞
5	22.8	565.0	0.8776E+02	3.106
6	22.9	575.0	0.1152E+03	3.116
7	23.0	590.0	0.1914E+03	1.819
8	23.1	620.0	0.6535E+03	0.505
9	23.2	860.0	0.1062E+04	0.292
10	23.3	915.0	0.2940E+03	50.247
11	23.4	944.0	0.1898E+03	0.961
12	23.5	958.0	0.8031E+02	0.959
13	24.0	986.0	0.3169E+02	

Table 4.4 The values used in the construction of Figure 4.5.

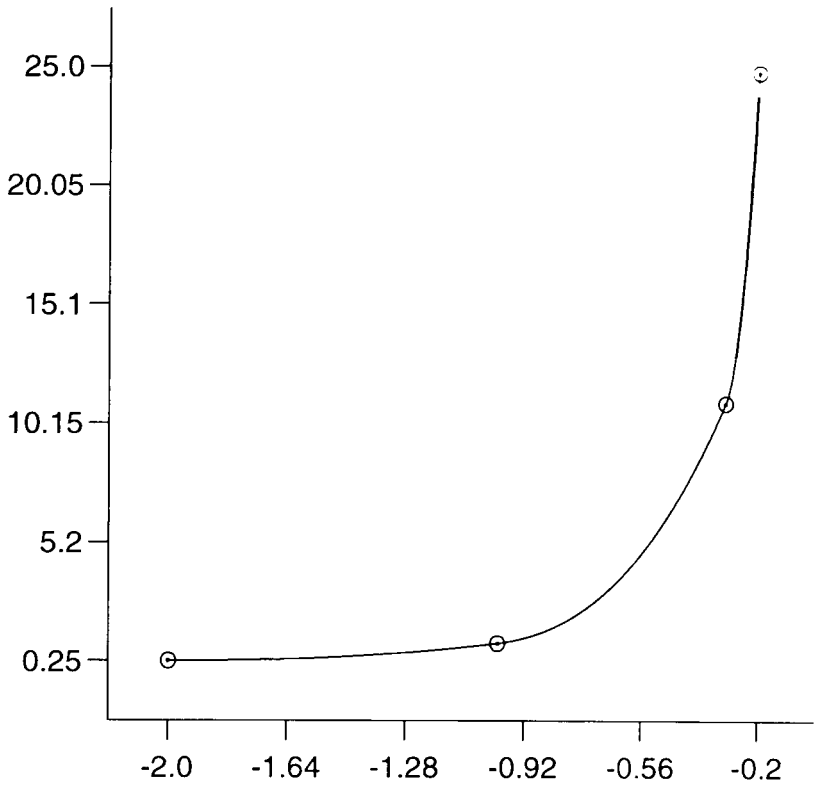

Figure 4.6 New automatic algorithm interpolating Data 4.

	Data Points		Slopes	Value of t on
i	x_i	y_i	d_i	$[x_i, x_{i+1}]$
1	-2.0	0.2500	0.0	
2	-1.0	1.0000	0.1737E+01	0.673
3	-0.3	11.1111	0.3216E+02	0.549
4	-0.2	25.0000	0.2456E+03	

Table 4.5 The values used in the construction of Figure 4.6.

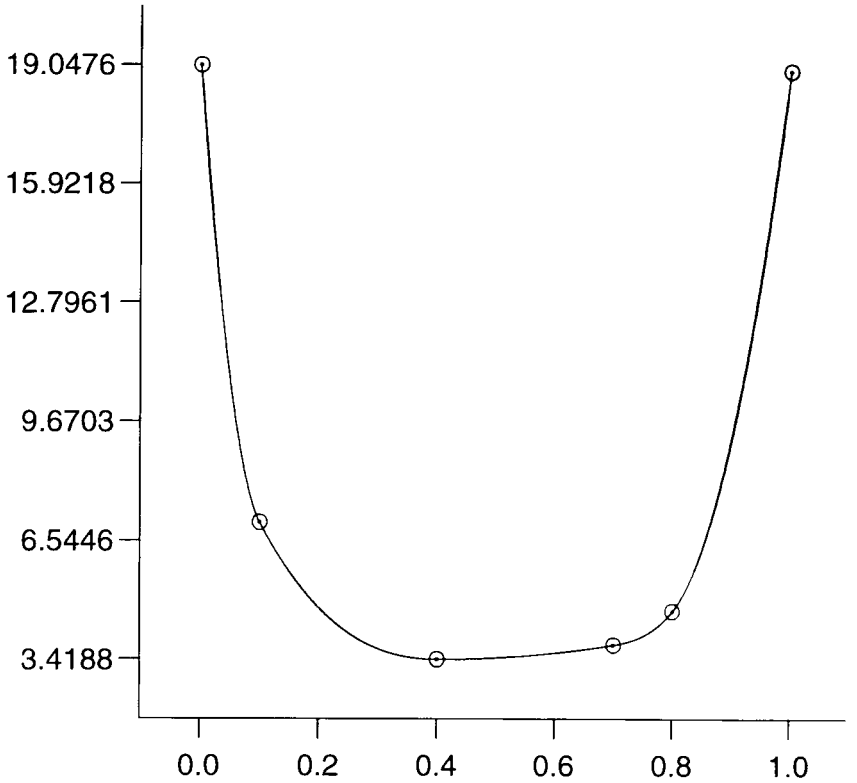

Figure 4.7 New automatic algorithm interpolating Data 6.

	Data Points		Slopes	Value of t on
i	x_i	y_i	d_i	$[x_i, x_{i+1}]$
1	0.0	19.0476	-0.2110E+03	
2	0.1	7.0175	-0.2957E+02	0.390
3	0.4	3.4188	0.0	
4	0.7	3.8095	0.2590E+01	0.631
5	0.8	4.7059	0.1935E+02	0.524
6	1.0	19.0476	0.1241E+03	

Table 4.6 The values used in the construction of Figure 4.7.

Chapter 5

Interactive Algorithms for Shape-Preserving Curve Drawing

In the previous chapters we have described several one-pass algorithms for constructing C^1 shape-preserving curves through the data points, using quadratic and cubic splines. This chapter deals with interactive shape-preserving interpolation which is very important for curve drawing in CAD and other scientific areas. An interactive system for curve drawing permits the user to model the form of the curve following his own needs. Here, unlike the slope estimation formulae (2.6) through (2.12), one seeks a means which allows, among possibilities, the choice of an interactive algorithm that always guarantees the construction of a shape-preserving curve. In this chapter, we create this flexibility by using the generalized slope estimation formula (4.1) which satisfies the shape-preserving conditions associated with the method in question. The formula uses the minimum number of data points for assigning the slope at each data point; that is, the coordinates of the point itself and one point on either side. It also involves a parameter t which is used to control the size of the estimated slope. As t increases, so the estimated slope decreases in magnitude and a tighter curve results. The extreme choices $t \to 0$ and $t \to \infty$ will generate the largest and the smallest values of the slope at the data points. In Sections 5.1 and 5.2, we discuss the slope estimation formula along with corresponding interactive algorithms using C^1 piecewise quadratic and cubic splines respectively. Several numerical examples and conclusions are presented in Section 5.3.

5.1 An Interactive Algorithm Using Quadratic Splines.

As we have shown in Chapter 3, a piecewise quadratic Hermite interpolant preserves monotonicity if the derivative values at the data points satisfy the condition (3.12), and maintains convexity/concavity of the data if the conditions (3.13) and (3.14) are fulfilled together with (3.15). In this section, we consider the derivation and analysis of a specific interactive shape-preserving interpolating curve algorithm based on the results presented in Chapter 3 and the new slope estimation formula (4.1) which provides the basis of the new algorithm. As a first step, let us assume that $\delta_i \neq 0$ and $\alpha_i = \dfrac{d_i}{\delta_i}$ and $\beta_i = \dfrac{d_{i+1}}{\delta_i}$ be the respective ratios of the end slopes to the chord slope. Now suppose that there is no additional knot inserted in the interval $[x_i, x_{i+1}]$. Then it is easy to see from Proposition 3.2.1 that the quadratic spline $p(x)$ defined in (3.9) is monotone on $[x_i, x_{i+1}]$ if sign(d_i)=sign(d_{i+1}) and $d_i + d_{i+1} = 2\delta_i$; that is,

$$\alpha_i + \beta_i = 2 \tag{5.1}$$

Moreover, $p(x)$ is also convex (concave) on $[x_i, x_{i+1}]$ if $d_i < d_{i+1}$ ($d_i > d_{i+1}$). In Figure 5.1 below, we display (5.1) which is exactly the line segment drawn by a broken line and this is in agreement with the results described in Edelman and Micchelli [26]. In order that the interpolant $p(x)$ be monotone and/or convex on $[x_i, x_{i+1}]$, the derivative values d_i and d_{i+1} must be chosen so that (α_i, β_i) lie on the broken line.

Next, we consider the situation when $d_i + d_{i+1} \neq 2\delta_i$ and there is an extra knot placed in the interval $[x_i, x_{i+1}]$. To ensure monotonicity, it is necessary to enforce a condition on the relationship between the size of the slopes and

the location of the additional knot ξ_i. In fact this has already been achieved in Chapter 3, where Case 2 (see page 80) is always selected in the case of a monotone data and the additional knot is inserted at $\xi_i = \frac{x_i + x_{i+1}}{2}$ (from (3.17)). It has also been observed that this choice always leads to a monotone interpolant if the slopes are calculated using the Butland formula (3.1). The following lemma provides the general condition to be satisfied by the derivative values d_i and d_{i+1} in order to preserve the monotonicity of the data when $\xi_i = \frac{x_i + x_{i+1}}{2}$.

Lemma 5.1.1

Let $\xi_i = \frac{x_i + x_{i+1}}{2}$ be an additional knot inserted in the interval $[x_i, x_{i+1}]$. Now if sign(d_i)=sign(d_{i+1})=sign(δ_i), then the quadratic spline $p(x)$ defined in (3.10) is monotone on $[x_i, x_{i+1}]$ if and only if

$$\alpha_i + \beta_i \leq 4 \tag{5.2}$$

The proof of this statement follows directly from Proposition 3.2.2 (I) together with (3.12), (3.17) and (3.21) of Chapter 3. As a consequence of Lemma 5.1.1, it is possible to construct a region M of allowable values for (α_i, β_i), which always guarantee to produce a monotone interpolant on $[x_i, x_{i+1}]$. This region is depicted with dots in Figure 5.1 and is in fact the triangular area with vertices (0,0), (4,0) and (0,4). It is useful here to point out that for the monotone data, Edelman and Micchelli [26] have shown that a C^1 piecewise quadratic spline with an arbitrary additional knot in $[x_i, x_{i+1}]$ is monotone if (α_i, β_i) lies in the region enclosed by the triangle with vertices (0,0), (2,0) and (0,2). This region is also contained within the monotonicity region M.

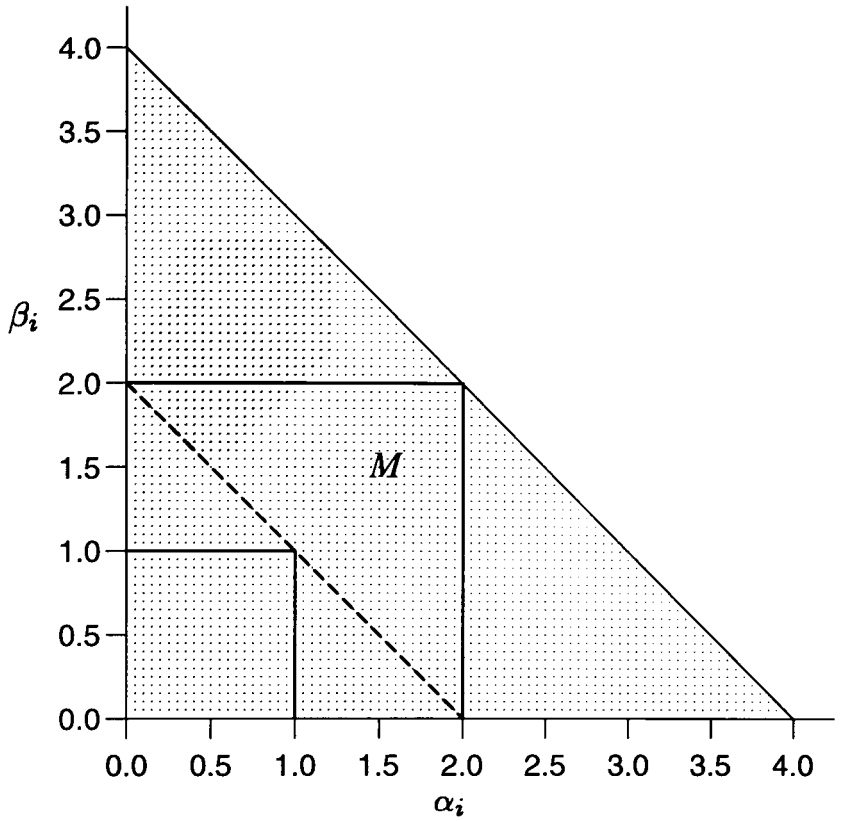

Figure 5.1 Monotonicity region M for quadratic splines.

Now, we turn our attention to the convexity/concavity and first assume that the data is convex; that is $0 < \delta_{i-1} \leq \delta_i$. In this case, we know that Case 1 (see page 80) is always chosen and the additional knot is inserted at $\xi_i = x_{i+1} + \dfrac{(d_i - \delta_i)(x_{i+1} - x_i)}{d_{i+1} - d_i}$ (from (3.16)). From Proposition 3.2.2 (II), the convexity condition to be satisfied by the derivative values d_i and d_{i+1} along with the additional knot ξ_i is given by

$$d_i \leq \bar{d}_i \leq d_{i+1} \qquad \text{if} \quad d_i < d_{i+1} \tag{5.3}$$

which in view of (3.16), (3.18) and (3.19) may be written as $d_i \leq \delta_i \leq d_{i+1}$ or equivalently:

$$\alpha_i \leq 1 \quad \text{and} \quad \beta_i \geq 1 \tag{5.4}$$

Similarly, in the case of concave data, that is, $\delta_{i-1} \geq \delta_i$, it can easily be proved from (3.14), (3.16), (3.18) and (3.19) that

$$\alpha_i \geq 1 \quad \text{and} \quad \beta_i \leq 1 \tag{5.5}$$

In addition, when $d_i d_{i+1} \geq 0$, then (5.4) and (5.5) also preserve the monotonicity of the data.

We claim that the modified form of the slope estimation formula (4.1), that is

$$d_i = \begin{cases} 0, & \text{if } \delta_{i-1}\delta_i \leq 0 \\ \dfrac{(w_1+w_2)^{\frac{1}{t}}\delta_{i-1}\delta_i}{(w_1\delta_{i-1}^t + w_2\delta_i^t)^{\frac{1}{t}}}, & \text{if } |\delta_i| \leq |\delta_{i-1}| \text{ and } \delta_{i-1}\delta_i > 0 \\ \dfrac{(w_1+w_2)^{\frac{1}{t}}\delta_{i-1}\delta_i}{(w_1\delta_i^t + w_2\delta_{i-1}^t)^{\frac{1}{t}}}, & \text{if } |\delta_{i-1}| \leq |\delta_i| \text{ and } \delta_{i-1}\delta_i > 0 \end{cases} \tag{5.6}$$

where $t > 0$ and, w_1 and w_2 are positive weighting factors such that $w_1 = 1$ and $1 \leq w_2 \leq 2$, produces the derivative values d_i and d_{i+1} which ensure that α_i and β_i always satisfy the monotonicity condition (5.2) and the convexity/concavity conditions (5.4) and (5.5). Note that the weights w_1 and w_2 in (5.6) can be scaled arbitrarily and assigning them high values will not have a significant effect, since a common factor in weights will simply cancel out. However, there are no specific criteria for choosing these but the work of Costantini [16] and Fritsch and Butland [32] show that $w_1 = 1$ and $1 \leq w_2 \leq 2$ are suitable for all data types and will produce visually pleasing curves. The effects of varying w_2 will be further discussed in Section 5.3. For practical

purposes, it is useful here to point out that when $t = 0$, the well-known generalized geometric mean (see [39]) is achieved; namely,

$$d_i = \lim_{t \to 0} \frac{(w_1 + w_2)^{\frac{1}{t}} \delta_{i-1} \delta_i}{(w_1 \delta_{i-1}^t + w_2 \delta_i^t)^{\frac{1}{t}}} = (\delta_{i-1}^{w_1} \delta_i^{w_2})^{\frac{1}{w_1 + w_2}} \tag{5.7}$$

and as $t \to \infty$ we obtain

$$d_i = \lim_{t \to \infty} \frac{(w_1 + w_2)^{\frac{1}{t}} \delta_{i-1} \delta_i}{(w_1 \delta_{i-1}^t + w_2 \delta_i^t)^{\frac{1}{t}}} = \min(\delta_{i-1}, \delta_i) \tag{5.8}$$

In fact, the formula (5.6) generates its maximum and minimum derivative values for $t = 0$ and $t = \infty$ respectively and t serves as a control parameter for the slope in the sense that as t tends to ∞ the curve tightens towards the chord segments connecting the data points.

The formula (5.6) normally calculates derivative values on interior data points only, so that a different approach must be used for the endpoints and one needs to supply the derivative values d_1 and d_n which affect the behaviour of the curve near its endpoints. The simplest way is to simply ask the user to supply them. The only danger here is that the user-supplied values may be incompatible with monotonicity and/or convexity in the end intervals. We recommend using the method described in Section 3.1 (see (3.7) and (3.8)) to evaluate the derivative values at the endpoints. This method automatically matches the required shape-preserving conditions in the extreme intervals. This approach has been followed for all curves shown in this chapter.

Now, we devise a method for determining the limits on t in order to get the derivative values d_i consistent with the monotonicity-preserving condition.

By considering $d_i \leq \beta_i \delta_i$ $(= d_{i+1})$ for both $\alpha_i, \beta_i \geq 0$, then, given α_i, β_i is computed as follows:

If the data is monotone, then from (5.2) set:

$$\beta_i = 4.0 - \alpha_i \tag{5.9}$$

Now, let $u = \min(\delta_i, \delta_{i+1})$, $r = \dfrac{\min(\delta_i, \delta_{i+1})}{\max(\delta_i, \delta_{i+1})}$ and

$$a = \begin{cases} 1 & \text{if } \delta_{i+1} \geq \delta_i \\ r & \text{if } \delta_{i+1} < \delta_i \end{cases} \tag{5.10}$$

Then (5.6) can be written as

$$d_i = \begin{cases} \dfrac{(w_1 + w_2)^{\frac{1}{t}} u}{(w_1 + w_2 r^t)^{\frac{1}{t}}}, & \text{if } \delta_{i-1}\delta_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{5.11}$$

and now from

$$\frac{d_i}{\delta_i} = \frac{(w_1 + w_2)^{\frac{1}{t}} a}{(w_1 + w_2 r^t)^{\frac{1}{t}}},$$

it is easy to see that $\dfrac{d_i}{\delta_i} \leq (w_1 + w_2)^{\frac{1}{t}} a \leq \beta_i$. That is,

$$t \geq \frac{\ln(w_1 + w_2)}{\ln \beta_i - \ln a} \tag{5.12}$$

Next, we prove that the slope estimation formula (5.6) always satisfies the convexity/concavity conditions (5.4) and (5.5) for every choice of parameter $t > 0$. Let us first consider the convex data case; that is, when $0 < \delta_{i-1} \leq \delta_i \leq \delta_{i+1}$. We then have:

$$\delta_{i-1} \leq \delta_i$$

$$\delta_{i-1}^t \leq \delta_i^t$$

$$(w_1 + w_2)\delta_{i-1}^t \leq (w_1\delta_{i-1}^t + w_2\delta_i^t)$$

$$\frac{(w_1 + w_2)^{\frac{1}{t}}\delta_{i-1}}{(w_1\delta_{i-1}^t + w_2\delta_i^t)^{\frac{1}{t}}} \leq 1$$

$$\alpha_i \leq 1 \tag{5.13}$$

Also, we have:

$$\delta_i \leq \delta_{i+1}$$

$$\delta_i^t \leq \delta_{i+1}^t$$

$$(w_1\delta_i^t + w_2\delta_{i+1}^t) \leq (w_1 + w_2)\delta_{i+1}^t$$

$$1 \leq \frac{(w_1 + w_2)^{\frac{1}{t}}\delta_{i+1}}{(w_1\delta_i^t + w_2\delta_{i+1}^t)^{\frac{1}{t}}}$$

$$\beta_i \geq 1 \tag{5.14}$$

The convexity condition (5.4) follows from (5.13) and (5.14). Similary, it can be proved that inequality (5.5) always holds with the slope estimation formula (5.6) when the data is concave; that is, when $\delta_{i-1} \geq \delta_i \geq \delta_{i+1}$. Hence, we infer the conclusion that formula (5.6) always provides the derivative values consistent with the convexity/concavity preserving conditions (5.4) and (5.5) for every choice of parameter $t > 0$. Thus, if the data is convex/concave, then t can be chosen arbitrarily to be any positive value and we set it equal to one in the algorithm QSLOPE given below, for the generation of the default curve.

The above discussion can now be concluded by presenting an explicit algorithm for obtaining the derivative values d_i at each interior data point. We refer the algorithm as Algorithm QSLOPE.

Algorithm QSLOPE.

Given δ_i's and d_1, then the derivative values d_i, $i = 2, ..., n-1$ are calculated as follows:

Step 1

If monotone data, then

Set $\alpha_i = \dfrac{d_{i-1}}{\delta_{i-1}}$

Compute β_i by applying the formula (5.9).

Choose a as in (5.10) and then set

$t_i = \dfrac{\ln(w_1 + w_2)}{\ln \beta_i - \ln a}$ to satisfy (5.12).

If convex/concave data, then

Set t_i=1.0

Step 2

If $\delta_i \delta_{i+1} \leq 0$ then

Set $d_i = 0$

Else

If $t_i = \infty$ then

$d_i = u$

Else

Set $\quad d_i = \dfrac{(w_1 + w_2)^{\frac{1}{t}} u}{(w_1 + w_2 r^t)^{\frac{1}{t}}}$

The above conditions (5.2), (5.4) and (5.5), along with the slope estimation formula (5.6), leave considerable freedom for choosing the derivative values d_i such that the resulting interpolant exhibits desirable shape properties present in the data. An interactive algorithm for generating shape-preserving interpolating curve is now described. We refer the algorithm as Algorithm QCURVE. A default curve is first generated by setting $w_1 = 1.0$, $w_2 = 1.0$ and initial values for t_i, denoted by t_i^* are determined using the Algorithm

QSLOPE for each $i = 2, ..., n-1$. The shape of any curve segments can then be altered interactively by varying the values of t_i.

Algorithm QCURVE.

Step 1

Input number of data points, n.

Step 2

Input data $\{x_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$.

Step 3

For i=1 to n-1 do

Use Algorithm QSLOPE to find t_i^* and the corresponding derivative values d_i.

Step 4

For i=1 to n-1 do

If $d_i + d_{i+1} = 2\delta_i$ then

Use equation (3.9) to generate the curve segment on $[x_i, x_{i+1}]$.

Else

If $(d_i - \delta_i)(d_{i+1} - \delta_i) \geq 0$ then

Choose Case 2 (monotone data).

Else

Select Case 1 (convex/concave data).

Use equation (3.10) to produce the curve segment on $[x_i, x_{i+1}]$.

Display the resulting curve.

Step 5

Modification of the shape of the curve.

If Case 2, then input a new value for $t_i > t_i^*$ (using (5.12)).

If Case 1, then input any new value for $t_i > 0$.

Use formula (5.11) to compute the new d_i.

Repeat Step 4 until the desired picture of the curve is achieved.

5.2 An Interactive Algorithm Using Cubic Splines.

In this section, we describe an interactive algorithm for generating shape-preserving interpolating curve using C^1 cubic splines. Suppose that the given data set is monotone and/or convex. The case of the monotone and/or concave data set can be tackled in a similar fashion. We claim that the slope estimation formula (5.6) for $t > 0$ and with positive weighting factors w_1 and w_2 such that $w_1 = 1$ and $1 \leq w_2 \leq 2$, always yields values of d_i and d_{i+1} which ensure that α_i and β_i lie directly inside the allowable region S of Figure 4.2. The overall idea for constructing an interactive algorithm which produces a C^1 interpolant is similar to that discussed in Section 4.2 of Chapter 4. Now, we proceed with determining the bounds on the magnitude of t so that the interpolant will preserve the monotonicity and/or convexity of the given data in each subinterval, as follows:

By assuming $d_i \leq \beta_i \delta_i \ (= d_{i+1})$ for both $\alpha_i, \beta_i \geq 0$, then, given α_i, β_i is computed as follows:

If the data set is non-convex monotone, then from Theorem 2.2.1 (ii.iii) of Chapter 2, we set:

$$\beta_i = \frac{1}{2}\Big(6 - \alpha_i + \sqrt{3\alpha_i(4 - \alpha_i)}\Big) \tag{5.15}$$

If the data set is convex (i.e., $(\delta_{i-1} - \delta_i)(\delta_i - \delta_{i+1}) > 0$), then using Lemma 4.2.1 and Lemma 4.2.2. of Chapter 4, we have:

$$\begin{cases} \text{If} \quad \alpha_i \leq 1 \text{ then} \quad \frac{1}{2}(3 - \alpha_i) \leq \beta_i \leq 3 - 2\alpha_i, \\ \text{If} \quad \alpha_i > 1 \text{ then} \quad 3 - 2\alpha_i \leq \beta_i \leq \frac{1}{2}(3 - \alpha_i). \end{cases} \tag{5.16}$$

Now, as in the preceeding section (see (5.10) through (5.12)), a range of

acceptable values for t is obtained:

$$t \geq \frac{\ln(w_1 + w_2)}{\ln \beta_i - \ln a} \tag{5.17}$$

From the above discussion, we propose the following algorithm for calculating the derivative values d_i at each interior data point. We refer the algorithm as Algorithm CSLOPE.

Algorithm CSLOPE.

Given δ_i's and d_1, then the derivative values d_i, $i = 2, ..., n-1$ are estimated as follows:

Step 1

Set $\alpha_i = \dfrac{d_{i-1}}{\delta_{i-1}}$

Step 2

(a) Case of monotone data: Compute β_i using the formula (5.15).

(b) Case of convex data: Compute β_i from the formula (5.16).

Step 3

Choose a as in (5.10) and then set

$t_i = \dfrac{\ln(w_1 + w_2)}{\ln \beta_i - \ln a}$ to satisfy (5.17).

Step 4

If $\delta_i \delta_{i+1} \leq 0$ then

Set $d_i = 0$

Else

If $t_i = \infty$ then

$d_i = u$

Else

Set $\quad d_i = \dfrac{(w_1 + w_2)^{\frac{1}{t}} u}{(w_1 + w_2 r^t)^{\frac{1}{t}}}$

An interactive algorithm for constructing the shape-preserving curve is now outlined. We refer the algorithm as Algorithm CCURVE. A default curve is first produced with $w_1 = 1.0$, $w_2 = 1.5$ and initial values for t_i, denoted by t_i^* for each $i = 2, ..., n-1$, are chosen applying the Algorithm CSLOPE.

Algorithm CCURVE.

Step 1

Input number of data points, n.

Step 2

Input data $\{x_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$.

Step 3

For i=1 to n-1 do

Apply Algorithm CSLOPE to find t_i^* and the corresponding derivative values d_i.

Step 4

For i=1 to n-1 do

Use equation (2.2) to produce the curve segment on $[x_i, x_{i+1}]$.

Display the resulting curve.

Step 5

Modification of the shape of the curve.

Set new value for $t_i > t_i^*$ (using (5.17)).

Apply formula (5.11) to estimate new d_i.

Repeat Step 4 until the desired shape of the curve is obtained.

In practical computation, rather than attempt to decide directly which parameter values to use in the algorithms of Section 5.1 and Section 5.2 above, the idea is to choose the initial values for t_i, denoted by t_i^*, using the alorithms QSLOPE and CSLOPE for each i, $i = 2, ..., n-1$, to generate an initial default

curve. Once this curve has been generated and displayed, then the shape of any of the curve segments can be modified interactively by assigning a new value to the parameter $t_i > t_i^*$, $2 \leq i \leq n-1$.

It should be noted that formula (5.6) also satisfies the condition (2.13) i.e.,

$$\min(\delta_{i-1}, \delta_i) \leq d_i \leq \max(\delta_{i-1}, \delta_i)$$

and varying t_i for a particular $2 \leq i \leq n-1$, will change the derivative value d_i. There are two curve segments which will be affected by a change in t_i: the curve segment preceding and succeeding the knot x_i. The need to use the derivatives as a tool to modify the shape of the curve generally arises in the case when the curve exhibits any sharp turns at a data point where the slopes of the adjacent data segments change rapidly, or at the turning point of a given data set. Hence, we can loosen or tighten the shape of the curve at that point by changing the derivative value.

5.3 Numerical Examples and Conclusions.

The proposed algorithms have been incorporated into an interactive package which allows the user to examine different shapes of the curve by varying the values of the parameter t_i. An important property of the schemes is that, for any $t_i \geq t_i^*$, they produce a shape-preserving curve. The local nature of the schemes and the control of the shape by the parameter t_i are also important features for curve design. The interactive algorithms discussed above combine the three ingredients: locality, interpolation and shape control. As is the case for any local interpolation algorithm, the above algorithms are very fast and convenient for computing all the points required for the graphical

display of a curve. The evaluation of a point on the curve at a prescribed parameter value is also quite fast, due to the local nature of the schemes. The following examples show that the algorithms described in the previous section are robust and provide a control parameter t_i for the derivative values associated with each data point, which can be used to flatten or tighten the curve locally. Hence these schemes are capable of producing shape-preserving curves which can be made visually more pleasing than the curves produced by existing one-pass methods.

Now we first investigate the effect of changing w_2 on the shape of the resulting curve such that $1 \leq w_2 \leq 2$. As an illustration, Data 4 is used to show the application of the algorithm QCURVE to the case when w_2 is varied, while keeping $w_1 = 1.0$ and $t_i = 1.0$ fixed. The resulting curves are displayed in Figure 5.2. Similarly, the effect of varying w_2 is demonstrated in Figure 5.3, where interpolating curves are generated applying the algorithm CCURVE. However, testing on several sets of data has shown that varying w_2 has little visual effect on the shape of the curve as compared to variation in t_i. It should also be noted that the simultaneous variation of w_2 and t_i does not give a clear indication of the scale of the impact on the shape of the curve. Hence in subsequent testing, we have decided to fix $w_2 = 1.0$ for generating the default curve applying the Algorithm QCURVE and $w_2 = 1.5$ for construction of the default curve using algorithm CCURVE. In the next step, varying t_i will change the shape of the curve in a predictable manner as shown in the following examples.

We now present some numerical output and discuss the results of applying the Algorithm QCURVE of Section 5.1 to two typical data sets. For each set of data, we first present the default curve accompanied by a table detailing the

data points, slopes, additional inserted knots, case numbers and corresponding selected initial values for the parameter t_i^*. Next, various effects of varying t_i will also be demonstrated. Figure 5.4 shows the default curve to Data 1 obtained by taking $w_1 = 1.0$ and $w_2 = 1.0$. In the following Figures 5.5 through 5.7, a magnified portion of the curve between the knots $x_7 = 9.0$ and $x_{11} = 15.0$ is shown to highlight the differences in the interpolant with the variation in the parameter t_i. The effect of varying the values of parameter $t_9 \geq t_9^*$, keeping $t_i = t_i^*$ (see Table 5.1) otherwise, is illustrated in Figure 5.5. Only two curve segments are affected, namely the curve segments between the intervals $[11.0, 12.0]$ and $[12.0, 14.0]$ respectively. Figure 5.6 exhibits the effect of varying an individual t_{10}, while all other are kept fixed with $t_i = t_i^*$. The curve in Figure 5.7 is generated by letting $t_9 = t_{10} = 50$ simultaneously. The effects due to these variations on the shape of the curve between the interval $[12.0, 14.0]$ are clearly seen, where the curve tends to a linear form. The default curve for the interpolation of Data 6 is shown in Figure 5.8 with $w_1 = 1.0$ and $w_2 = 1.0$. As the data is convex, this leaves us considerable freedom for choosing any value for $t_i > 0$ such that the resulting interpolant will exhibit desirable shape properties present in the data. Thus, Figure 5.9 illustrates the effect of successively increasing the value of parameter $t_2 = 0.2, 0.8, 1.5$ and 3.0, while other t_i are the same as t_i^* listed in Table 5.2. The effect of the high parameter value is clearly seen in that the resulting interpolant approaches the chord segment connecting the data points.

The result of applying the interactive cubic spline algorithm CCURVE of Section 5.2 with $w_1 = 1.0$ and $w_2 = 1.5$ to the same data sets used in the applications of the algorithm QCURVE above, is now presented. For each data set, the default curve is first displayed and then a table containing the data

points, derivative values and the initial computed values for the parameter t_i^* is provided. Next, the effects of varying t_i are also considered. Figure 5.10 is the graph of the default curve to Data 1. In the following Figures 5.11 to 5.13, an enlarged part of the curve in the interval $[9.0, 15.0]$ is given to show the effects on the resulting interpolant with the variation in the parameter t_i. The effects due to these variations on the shape of the curves are analogous to those shown in Figure 5.5 through Figure 5.7, where noticeable change in the magnitudes of the slope of the curves at the data points $x_9 = 12.0$ and $x_{10} = 14.0$ can be seen. Now the default curve is modified by changing the parameter t_9 such that $t_9 \geq t_9^*$, while all other are kept fixed with $t_i = t_i^*$ as given in Table 5.3. The resulting interpolants are shown in Figure 5.11 where the variation in t_9 only effects the shape of the curve in the neighbourhood of the data point $x_9 = 12.0$. Similarly, Figure 5.12 illustrates the effect of varying the parameter $t_{10} \geq t_{10}^*$, while letting all other $t_i = t_i^*$. Figure 5.13 is the graph of the interpolating curve where the curve segment in the interval $[12.0, 14.0]$ is modified by allowing $t_9 = t_{10} = 40$ simultaneously, so that the resulting curve segment approaches a linear segment. Figure 5.14 shows the default curve to Data 6 and the progressive increase in parameter $t_2 \geq t_2^*$, while keeping $t_i = t_i^*$ (see Table 5.4) otherwise, is demonstrated in the curves of Figure 5.15. This is accomplished for the values $t_2 = 0.618, 0.9, 1.5$ and 2.5 respectively. Clearly the effects of varying t_2 is that increasing it pulls the curve segment in the interval $[0.1, 0.4]$ towards the chord joining the two end points of this subinterval.

For design flexibility in manipulating the curves, the user should be given an interactive option to adjust the value of t_i in those intervals where the curve is to be loosened or tightened, rather than using a uniform value for the tension

parameter throughout all intervals. After the default curve has been produced, the algorithms QCURVE and CCURVE provide considerable freedom in choosing different values for t_i on each curve segment and produce visually pleasing shape-preserving C^1 interpolating curves. Figures 5.4 through 5.15 clearly demonstrate the increase in the tightness of the curve with the increase in t_i. Thus, the effect of varying t_i may be thought of as operations which "loosen" and "tighten" the corresponding curve segments.

In comparison with Montefusco's interactive shape-preserving method [52] which is based on C^1 cubic interpolating splines, the interactive algorithms proposed here are completely local and more efficient since they do not require solving optimization problems as in [52]. Also, they have an added advantage of always producing shape-preserving interpolants for every choice of parameter $t_i \geq t_i^*$, while the tensioning procedure in the Montefusco method requires more than one adjustment to generate acceptable shape-preserving curves.

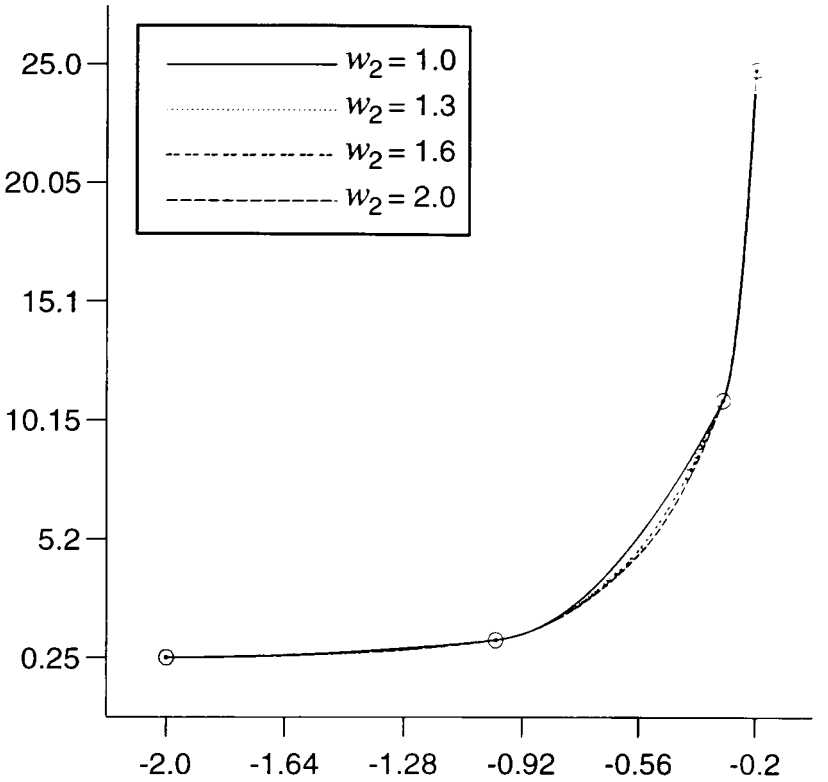

Figure 5.2 Algorithm QCURVE with varying w_2; $w_1 = 1.0$ and $t_i = 1.0$.

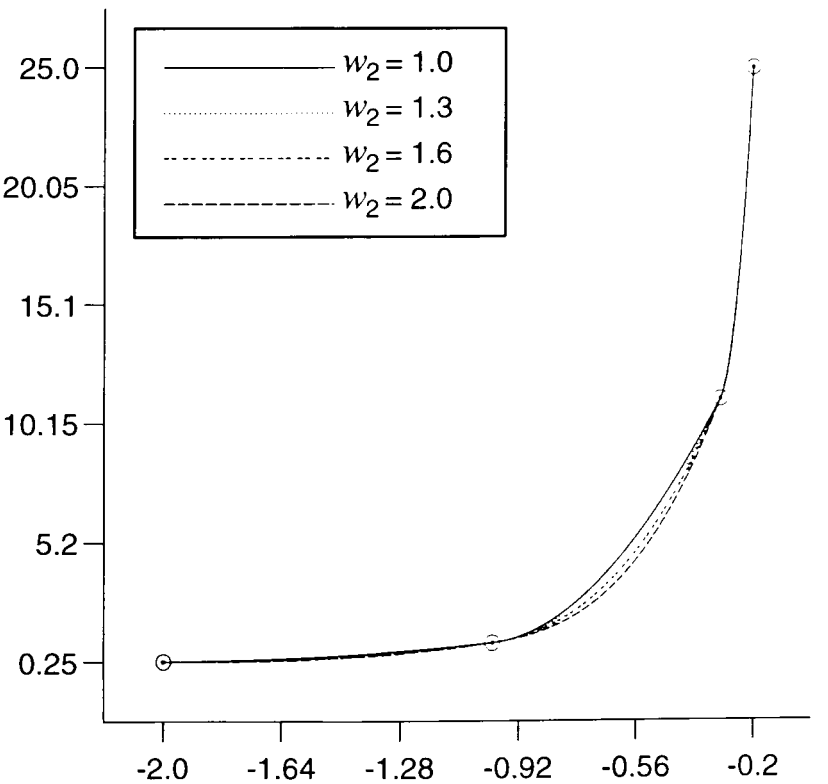

Figure 5.3 Algorithm CCURVE with varying w_2; $w_1 = 1.0$ and $t_i = 1.0$.

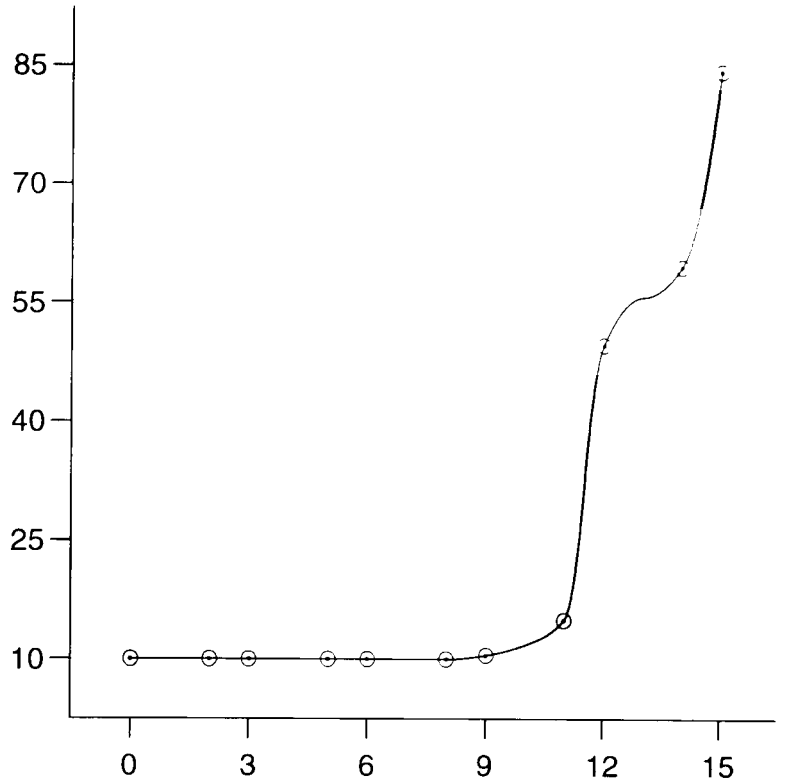

Figure 5.4 Default curve: $t_i = t_i^*$ for each i, $2 \leq i \leq n-1$.

	Data Points		Slopes	Additional Knots	Case on	Value of t_i^* at
i	x_i	y_i	d_i	ξ_i	$[x_i, x_{i+1}]$	d_i
1	0.0	10.0	0.0			
2	2.0	10.0	0.0			
3	3.0	10.0	0.0			
4	5.0	10.0	0.0			
5	6.0	10.0	0.0			
6	8.0	10.0	0.0	8.389	1	
7	9.0	10.5	0.8182E+00	10.398	1	1.000
8	11.0	15.0	0.5572E+01	11.5	2	0.537
9	12.0	50.0	0.1198E+02	13.0	2	0.211
10	14.0	60.0	0.8333E+01		1	1.000
11	15.0	85.0	0.4167E+02			

Table 5.1 The values associated with Figure 5.4.

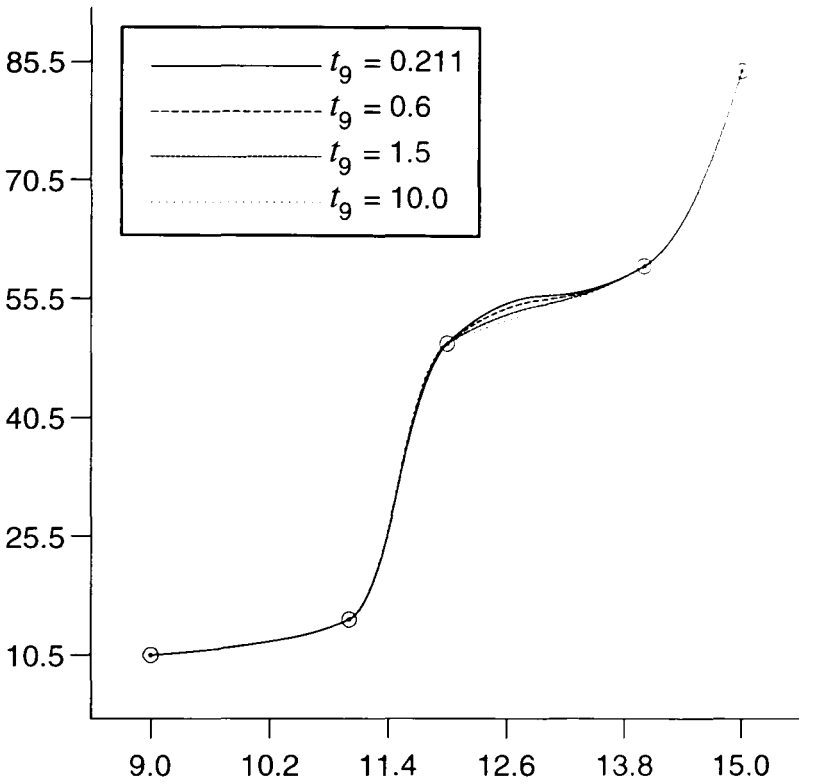

Figure 5.5 Modified curve: $t_9 = 0.211, 0.6, 1.5$ and 10.0; $t_i = t_i^*$ otherwise.

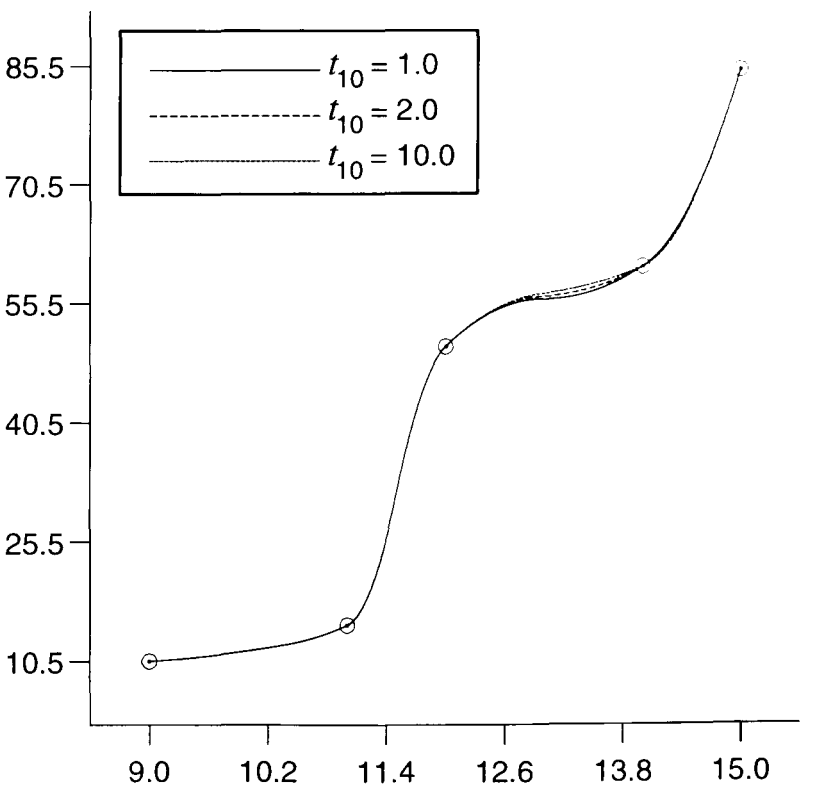

Figure 5.6 Modified curve: $t_{10} = 1.0, 2.0$ and 10.0; $t_i = t_i^*$ otherwise.

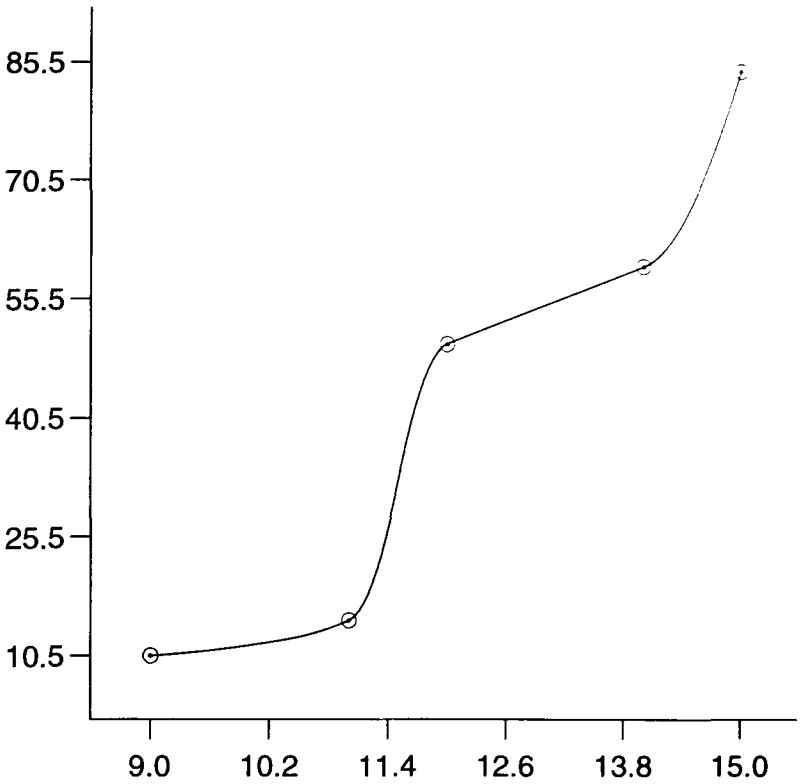

Figure 5.7 Modified curve with large parameter values: $t_9 = 50$ and $t_{10} = 50$.

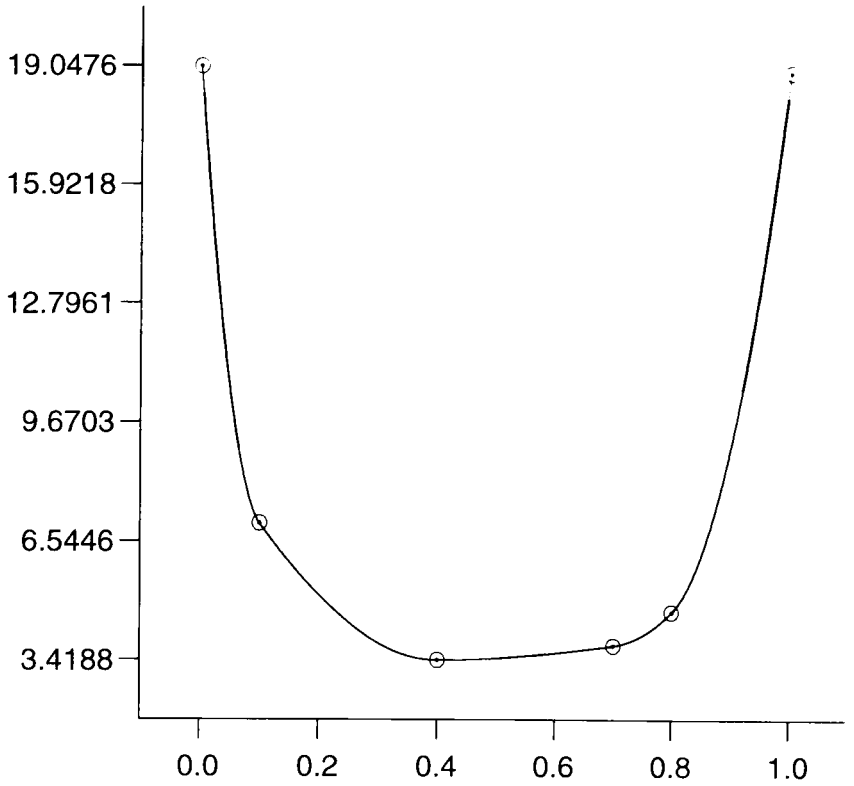

Figure 5.8 Default curve: $t_i = t_i^*$ for $i = 2, ..., n-1$.

	Data Points		Slopes	Additional Knots	Case on	Value of t_i^* at
i	x_i	y_i	d_i	ξ_i	$[x_i, x_{i+1}]$	d_i
1	0.0	19.0476	-0.2188E+03		1	1.0
2	0.1	7.0175	-0.2182E+02	0.265	1	1.0
3	0.4	3.4188	0.0	0.528	1	1.0
4	0.7	3.8095	0.2274E+01	0.751	1	1.0
5	0.8	4.7059	0.1594E+02		1	1.0
6	1.0	19.0476	0.1275E+03			

Table 5.2 The values associated with Figure 5.8.

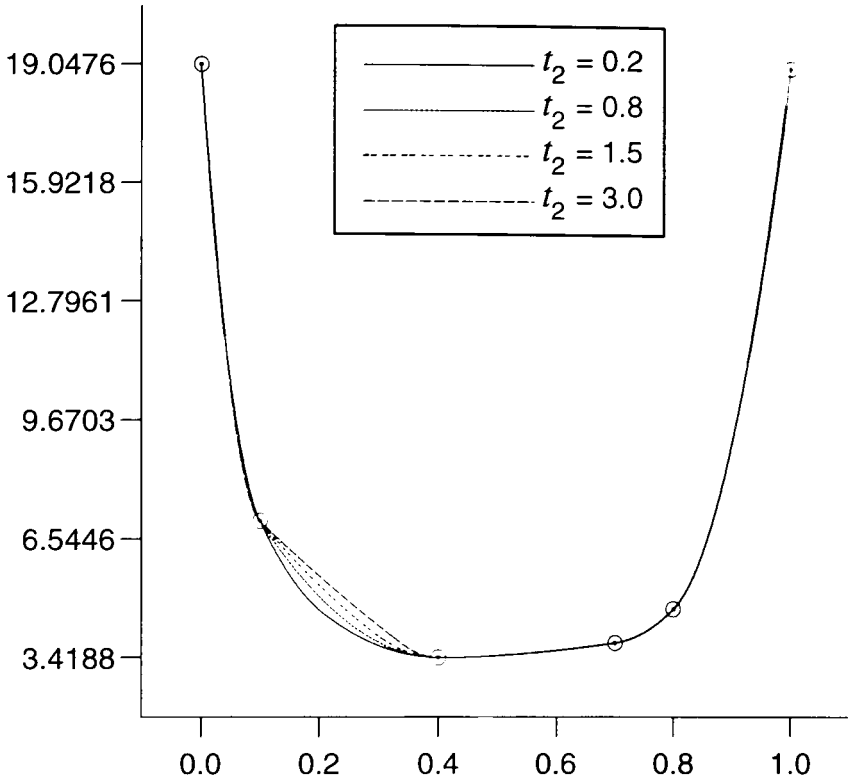

Figure 5.9 Modified curve: $t_2 = 0.2, 0.8, 1.5$ and 3.0; $t_i = t_i^*$ otherwise.

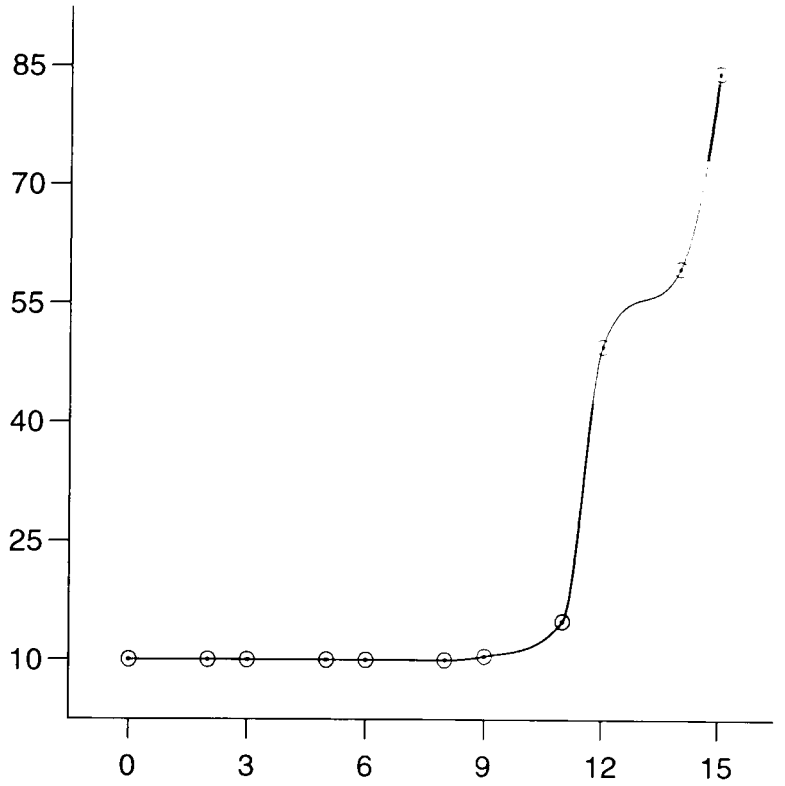

Figure 5.10 Default curve: $t_i = t_i^*$ for each i, $2 \leq i \leq n-1$.

	Data Points		Slopes	Value of t_i^* at
i	x_i	y_i	d_i	d_i
1	0.0	10.0	0.0	
2	2.0	10.0	0.0	
3	3.0	10.0	0.0	
4	5.0	10.0	0.0	
5	6.0	10.0	0.0	
6	8.0	10.0	0.0	
7	9.0	10.5	0.9787E+00	0.834
8	11.0	15.0	0.6378E+01	0.678
9	12.0	50.0	0.1409E+02	0.283
10	14.0	60.0	0.1021E+02	0.794
11	15.0	85.0	0.3979E+02	

Table 5.3 The values associated with Figure 5.10.

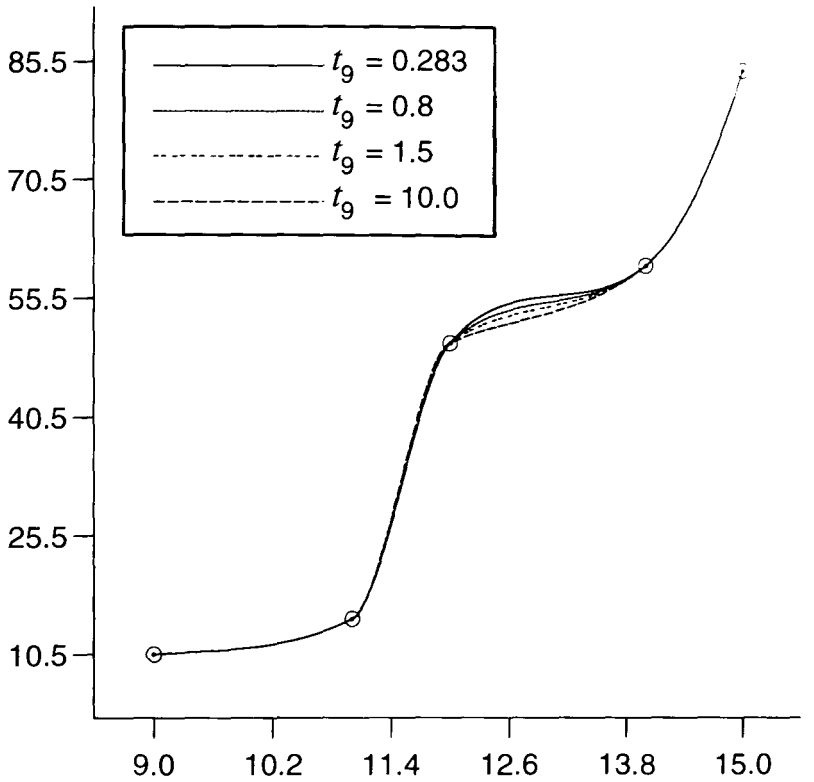

Figure 5.11 Modified curve: $t_9 = 0.283, 0.8, 1.5$ and 10.0; $t_i = t_i^*$ otherwise.

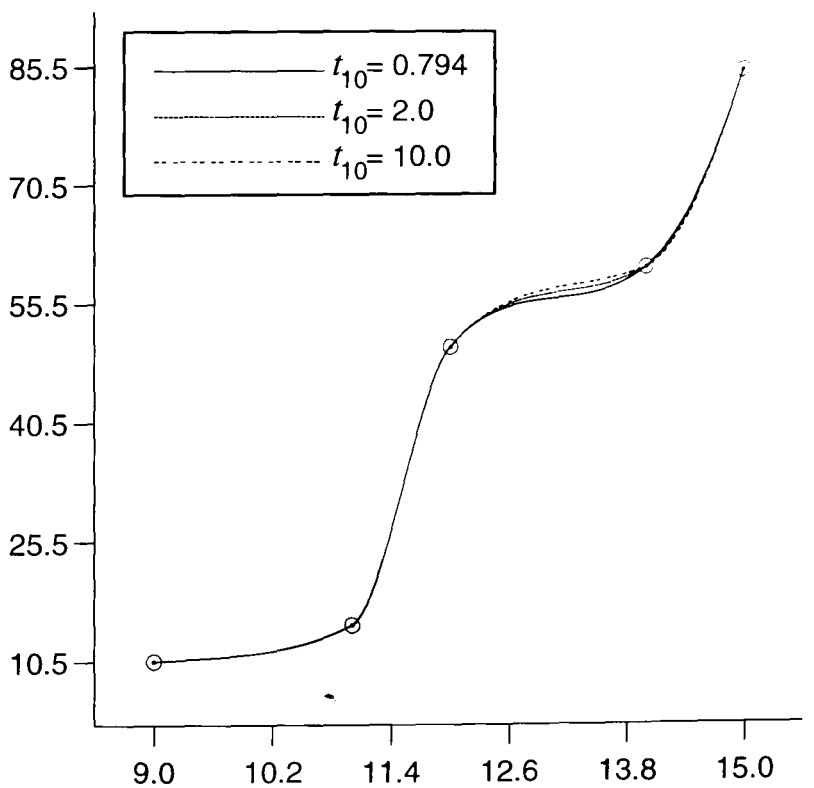

Figure 5.12 Modified curve: $t_{10} = 0.794, 2.0$ and 10.0; $t_i = t_i^*$ otherwise.

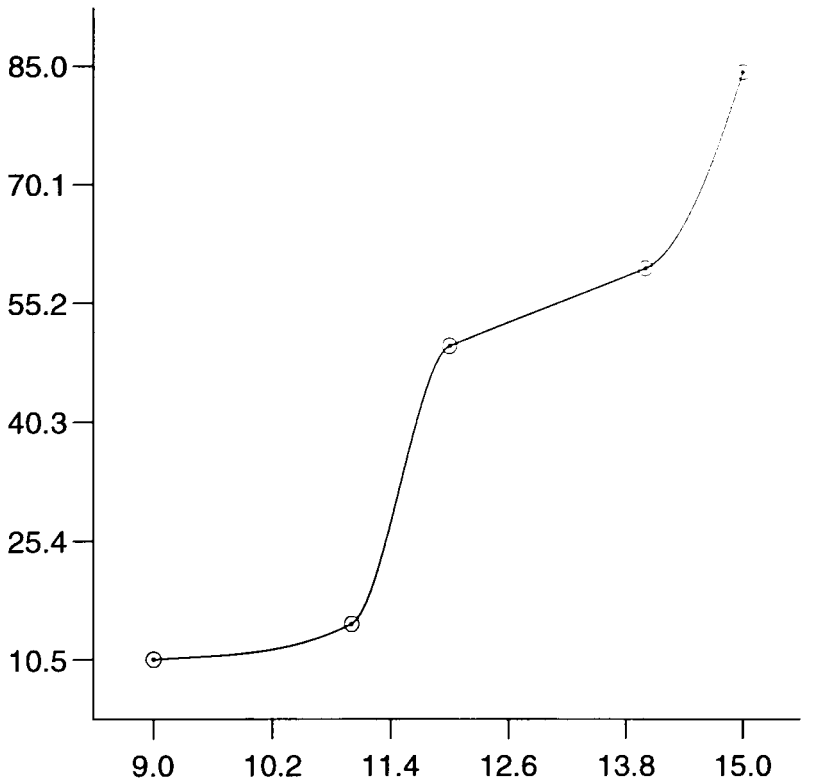

Figure 5.13 Modified curve with large parameter values: $t_9 = 40$ and $t_{10} = 40$.

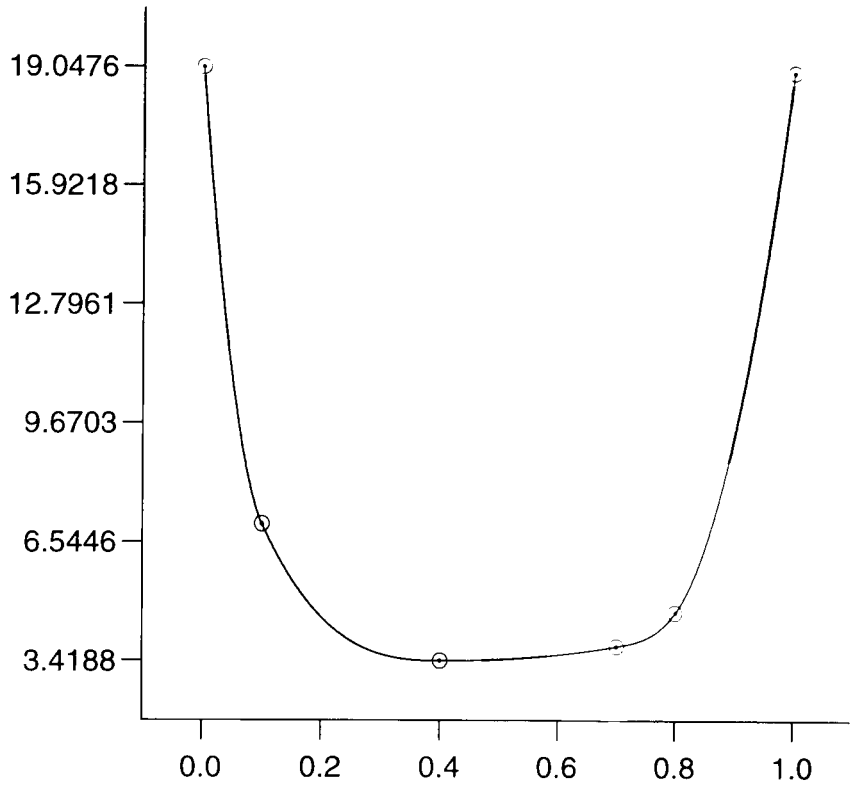

Figure 5.14 Default curve: $t_i = t_i^*$ for $i = 2, ..., n-1$.

	Data Points		Slopes	Value of t_i^* at
i	x_i	y_i	d_i	d_i
1	0.0	19.0476	-0.2064E+03	
2	0.1	7.0175	-0.3420E+02	0.515
3	0.4	3.4188	0.0	
4	0.7	3.8095	0.2852E+01	0.834
5	0.8	4.7059	0.2173E+02	0.689
6	1.0	19.0476	0.1217E+03	

Table 5.4 The values associated with Figure 5.14.

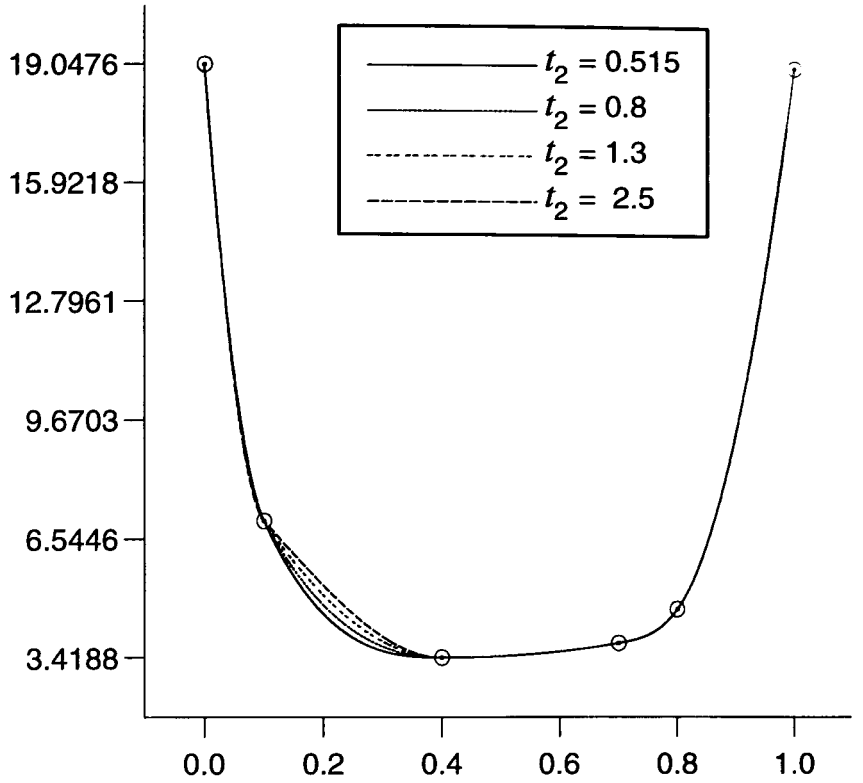

Figure 5.15 Modified curve: $t_2 = 0.515, 0.8, 1.3$ and 2.5; $t_i = t_i^*$ otherwise.

Chapter 6

Algorithms for Shape Preserving Surface Drawing

In this chapter we concern ourselves with the matters related to bivariate shape-preserving interpolation. For the case of curve generation, several methods which preserve properties such as monotonicity and/or convexity of the data have been presented in Chapters 2 through 5. However, shape-preserving interpolation techniques for the surface generation problem to a set of bivariate data defined over a rectangular grid have not yet been adequately dealt with, and only a few methods are available, such as in [3], [6], [12], [13], [18] and [24]. The methods used for generating C^1 interpolating surfaces are the blending function method and the tensor product method. Some of these methods are briefly describe in Section 6.1. This chapter has been published in Iqbal [42] and discusses the use of the Butland [11] slope estimation method in the method of Roulier [58]. In Section 6.2, we review the Roulier algorithm briefly and present a Modified algorithm in Section 6.3 which illustrates the ways in which Butland slopes and the one-dimensional shape-preserving method of Schumaker [61] can be used in the Roulier method. Finally, conclusions and numerical results showing the performance of our algorithm are presented in Section 6.4.

6.1 Literature Review.

The representation of a surface using a blending function technique is an approach taken by Dodd, McAllister and Roulier [24]. The method constructs surfaces which preserve the shape of the data along grid lines, such that if the data points form a convex shape along the grid lines, then the resulting curve passing through these points will also have a convex shape, and if the data points have a maximum or minimum point, then the interpolating curve should also have a maximum or minimum at that point. In this scheme, the one-dimensional shape-preserving method of McAllister and Roulier [50] is applied to estimate the functions along the grid lines, and then rectangular patches are constructed using blending functions as proposed by Gregory [36]. The specification of the first and mixed partial derivatives at each grid point are required. Their algorithm generally produces visually pleasing results, but may not preserve the shape of the data inside the rectangles in extreme cases. Also, the condition of imposing zero mixed partial derivatives, suggested and used by the authors, at the grid points causes undesirable flat spots in the resulting surface.

Beatson and Ziegler [6] give a method which preserves the monotonicity of the data by a C^1 piecewise quadratic function defined over triangular elements which are obtained by subdividing each grid rectangle into a grid of sixteen triangles. The quadratic polynomials are uniquely determined by the function value and first partial derivatives at the vertices of the grid rectangle. The method requires the specification of the values for the first partial derivatives at each of the grid points. The first partial x and y derivatives are initialised using a divided difference formula. A similar algorithm is presented by Asaturyan and Unsworth [3], where monotonicity is achieved using biquadratic

splines defined over rectangular elements formed by the partition of each initial rectangle into four subrectangles. Over each subrectangle, the surface is defined by biquadratic functions, requiring the specification of the first and mixed partial derivatives at each grid point. These derivative values are constrained to satisfy conditions which ensure the generation of shape-preserving surfaces.

Carlson and Fritsch [12, 13] develop schemes which produce a monotone interpolant to monotone data defined on a rectangular grid. The interpolating surfaces are obtained using the tensor product of C^1 cubic splines which are based upon their univariate piecewise cubic monotone interpolation schemes described in [32, 33]. These methods require the specification of values for the first partial derivatives and the first mixed partial derivatives at each grid point. A bicubic Hermite polynomial is defined over each grid rectangle and monotonicity constraints are imposed on the first partial derivatives and the first mixed partial derivatives at the grid points. These constraints are then satisfied by steadily reducing the magnitude of the first partial derivatives and the mixed partial derivatives from their initial values specified at the four corner of the grid rectangle. The schemes guarantee that the generated surface preserves the monotonicity of the surface along the grid lines and inside the surface patches.

Costantini and Fontanella [18] have approached the bivariate shape-preserving interpolation problem in a different way, by local adjustment of the degree of the interpolating functions (in either of the variables). No constraints are imposed on the partial derivatives as in above methods. The scheme is extension of the univariate shape-preserving interpolation scheme described in [15, 16, 17] to the bivariate case, and requires the specification of

values for the first partial derivatives and the first mixed partial derivatives at each grid point. Over each grid rectangle, the surface patch is defined by a tensor product spline using bivariate Bernstein polynomials and the resulting interpolant is of an arbitrary continuity class. The shape preserving constraints such as monotonicity and/or convexity are satisfied by adjusting the degree of the function interpolating the data. These constraints are applied and satisfied along the grid lines, which preserves the monotonicity and/or convexity of the data along the boundaries of the generated surface. The scheme is not however local since the degrees of an edge of the grid rectangle, say in the x direction, must be the same throughout the corresponding column in the y direction.

In contrast to above methods, Roulier [58] presents a refinement technique which is local and successively refines grid data which is convex along grid lines in such a way that the refined data exhibit the same convexity and monotonicity along the appropriate grid lines. This method uses the shape-preserving quadratic splines of McAllister and Roulier [50], and includes some geometric observations of the data. The method is first applied to the original data to generate refined grid data which exhibits the convexity and/or monotonicity of the original data along new grid lines. The method produces the refined data by inserting new data points between the original data points using McAllister and Roulier's algorithm [50]. These new points are within the convexity and monotonicity limits which are derived from the piecewise linear interpolant to the original data. The algorithm is applied repeatedly to the updated refined grid data and the final refined data can be used as points on a convex and monotone surafce, or can be used as bivariate data for an alternative surface interpolation scheme.

6.2 Roulier Algorithm.

Roulier [58] describes the following algorithm for convex bivariate grid data. For brevity and simplicity, we assume throughout the remainder of this chapter that the initial grid data is convex and monotone increasing in both x and y directions:

$$0 < \delta x_{i,j} < \delta x_{i+1,j} \qquad i = 1, ..., n-2, \quad j = 1, ..., m. \tag{6.1}$$

$$0 < \delta y_{i,j} < \delta y_{i,j+1} \qquad i = 1, ..., n, \quad j = 1, ..., m-2. \tag{6.2}$$

where $\delta x_{i,j}$ and $\delta y_{i,j}$ are defined in (1.9) and (1.10) of Chapter 1 and the extension to convex decreasing grid data is trivial. Here we only give a very brief review of the Roulier [58] method. This is necessary for us to describe our Modified algorithm.

In the first step of the method, one dimensional shape-preserving splines are found for the sets of data $\{(x_i, y_j, f_{i,j}), i = 1, ..., n\}$ for each j. These one-dimensional approximations are used to determine an approximate value for the function at the mid-points of each interval on grid lines in the x directions. At the next step the roles of x and y are reversed. The typical one-dimensional shape-preserving interpolation step on a grid line is characterised by data of the form $(t_k, f_k), k = 1, ..., n$, where t_k is either x_k or y_k, and where

$$0 < \delta_1 < \delta_2 < ... < \delta_{n-1} \tag{6.3}$$

and

$$\delta_k = \frac{f_{k+1} - f_k}{t_{k+1} - t_k}, \quad k = 1, ..., n-1. \tag{6.4}$$

The step computes estimates d_k of the slopes at t_k which satisfy

$$\delta_k < d_{k+1} < \delta_{k+1}, \quad k = 1, ..., n-2, \quad d_1 < \delta_1, \quad \delta_{n-1} < d_n \tag{6.5}$$

In terms of these slopes, we define constants A_k, B_k as follows:

$$A_1 = d_2(t_1 - t_2) + f_2$$

$$A_k = \max\Big(d_{k-1}(t_k - t_{k-1}) + f_{k-1}, \quad d_{k+1}(t_k - t_{k+1}) + f_{k+1}\Big), \quad k = 2, ..., n-1.$$

$$A_n = d_{n-1}(t_n - t_{n-1}) + f_{n-1}$$

and

$$B_k = \max\Big(d_k(\bar{t}_k - t_k) + f_k, \quad d_{k+1}(\bar{t}_k - t_{k+1}) + f_{k+1}\Big), \quad k = 1, ..., n-1.$$

where $\bar{t}_k = \dfrac{t_k + t_{k+1}}{2}$.

This one-dimensional scheme forms the basis of each step of the 2-dimensional interpolation algorithm described by the Pascal-like pseudocode given below.

Step 1

For $j := 1$ to m do

Begin

For $i := 1$ to n do

Use the slope calculation method as proposed by McAllister and Roulier [50] to produce estimates of slopes $dx_{i,j}$.

For $i := 1$ to n do

Use the shape-preserving quadratic interpolant of the McAllister and Roulier [50] to produce P_j such that $P_j(x_i) = f_{i,j}$ and $P_j'(x_i) = dx_{i,j}$.

End

Step 2

For $j := 1$ to m do

For $i := 1$ to $n - 1$ do

Begin

$$\bar{x}_i = \frac{x_i + x_{i+1}}{2}; \quad f^*_{i,j} = P_j(\bar{x}_i); \quad \bar{f}_{i,j} = \frac{f_{i,j} + f_{i-1,j}}{2}$$

End

For $j := 1$ to m do

For $i :=1$ to $n - 1$ do

Begin

Generate numbers $B_{i,j}$ (corresponding to B_k above) with $dx_{i,j}$ from Step 1 and they satisfy $B_{i,j} < f^*_{i,j} < \bar{f}_{i,j}$.

End

Step 3

For $i := 1$ to $n - 1$ do

For $j := 2$ to m do

Begin

$$\delta^*_{y_{i,j}} = \frac{f^*_{i,j} - f^*_{i,j-1}}{y_j - y_{j-1}}$$

End

Step 4

If for some i_o we have a j_o such that $\delta^*_{y_{i_o,j_o}} > \delta^*_{y_{i_o,j_{o+1}}}$, then use the data $(y_j, \bar{f}_{i_o,j})$ to generate $A_{i_o,j}$, $j = 1, ..., m$ (corresponding to A_k above).

Step 5

For this i_o if $f^*_{i_o,j} < A_{i_o,j}$ then set $f^*_{i_o,j} = A_{i_o,j}$. This guarantees that $(y_j, f^*_{i_o,j})$, $j = 1, ..., m$ are convex.

Step 6

Go to Step 1 and use new grid data consisting of $(x_i, y_j, f_{i,j})$ $i = 1, ..., n$, $j = 1, ..., m$. and $(\bar{x}_i, y_j, f^*_{i,j})$ $i = 1, ..., n-1$, $j = 1, ..., m$ but reverse the roles of x and y (i and j).

The points on a surface are generated and monotonicity and convexity parallel to the grid lines is maintained by the algorithm when applied to convex grid data.

6.3 Modified Algorithm.

Now we present a new modified algorithm based on an alternative one-dimensional shape-preserving algorithm due to Schumaker, using a slope estimation technique due to Butland. The Butland slope formula (3.1) satisfies (6.5), since from (6.3), we have

$$\delta_k < \delta_{k+1} \tag{6.6}$$

$$\delta_k^2 < \delta_k \delta_{k+1}$$

$$\delta_k^2 + \delta_k \delta_{k+1} < 2\delta_k \delta_{k+1}$$

$$\delta_k < \frac{2\delta_k \delta_{k+1}}{\delta_k + \delta_{k+1}}$$

$$\delta_k < d_{k+1} \tag{6.7}$$

and also, from (6.6),

$$\delta_k \delta_{k+1} < \delta_{k+1}^2$$

$$2\delta_k \delta_{k+1} < \delta_k \delta_{k+1} + \delta_{k+1}^2$$

$$\frac{2\delta_k \delta_{k+1}}{\delta_k + \delta_{k+1}} < \delta_{k+1}$$

$$d_{k+1} < \delta_{k+1} \tag{6.8}$$

Now by (6.7) and (6.8), we obtain (6.5)

In Chapter 3 we have shown that Schumaker's algorithm becomes a one-pass algorithm automatically if the Butland slopes are used and that, in this case, it produces an interpolant which is identical with the McAllister-Roulier interpolant, though by an easier computation. The bivariate interpolant produced by the Modified algorithm given below is therefore identical.

In the Modified algorithm Step 1 of the main algorithm is replaced using the Butland [11] slopes and the Schumaker [61] shape-preserving interpolant as described in Chapter 3 in such a manner that resulting interpolant preserves the convexity and monotonicity of the grid data. Step 1 is now performed as:

Step 1

For $j := 1$ to m do

Begin

For $i := 1$ to n do

Use the Butland slope formula to produce slopes $dx_{i,j}$.

For $i := 1$ to n do

Use the shape-preserving quadratic interpolation method of Schumaker to produce P_j such that $P_j(x_i) = f_{i,j}$ and $P_j'(x_i) = dx_{i,j}$.

End.

Step 2 through Step 6 are the same as given in the Roulier algorithm above. The Butland slopes can be further improved iteratively using Frey's iteration technique as described in Section 3.5 of Chapter 3. An advantage

of this improvement is that it leads more visually pleasing interpolants than those using just Butland slopes. This is demonstrated practically in the next section.

6.4 Numerical Examples and Conclusions.

In this section, we present the results of some numerical experiments using the schemes described in previous sections. The algorithms have been tested on several sets of grid data widely used in the literature, but here we consider only the two convex data sets used earlier by Roulier [58] and described in Chapter 1, to illustrate the performance of the method. In each of the following figures, the interpolant has been evaluated on a uniform 3×3 refinement of the original rectangular partition and the resulting points joined with straight lines for display purposes. The surfaces shown in Figures 6.1 and 6.2 are the result of applying the Modified algorithm using the Butland and Frey slope schemes respectively to Data 7 (Table 1.9). The plots for Data 8 (Table 1.10) produced by the Modified algorithm with the Butland and Frey slope methods are represented in Figures 6.7 and 6.8 respectively. Figures 6.5, 6.6, 6.11 and 6.12 show combined plots of the interpolating surfaces generated by the Modified algorithm with Butland and with Frey slopes, and which allow direct comparison of the different surfaces. Areas in which the resulting approximations show noticeable visual differences are highlighted and clearly observed on the diagrams. The algorithms have been implemented in Fortran-77 under the SunOS 4.1.3 operating system and run on a SPARCserver-670MP machine. For each algorithm, the computation time required for interpolating the surfaces is very important. In order to compare CPU time, we further interpolate the above surfaces by evaluating on a uniform 7×7 refinement of the

original rectangular grid shown in Figures 6.3, 6.4, 6.9 and 6.10 respectively. Table 6.1 summarizes the CPU seconds required to produce above Figures. It is clear from Table 6.1 that the Modified algorithm with Butland slopes is faster than the other two algorithms.

Figures	Modified Algorithm (with Butland Slopes)	Modified Algorithm (with Frey Slopes)	Roulier Algorithm (Original)
6.1, 6.2	0.081	0.356	0.262
6.3, 6.4	0.111	0.571	0.509
6.7, 6.8	0.091	0.375	0.326
6.9, 6.10	0.110	0.540	0.461

Table 6.1 Timing information for the Figure 6.1 through Figure 6.10.

These examples reveal that both algorithms with Butland slopes give identical interpolants and the resulting surfaces are shape-preserving. We conclude from these figures that the Modified algorithm with the Frey slope scheme produces more visually pleasing surfaces than that which uses Butland slopes.

We want to point out that our Modified algorithm provides an alternative and simpler method for constructing the interpolant given by Roulier's method. The Modified algorithm has the merit that it is very easy to implement, and that it is more efficient in terms of both CPU time and storage requirements.

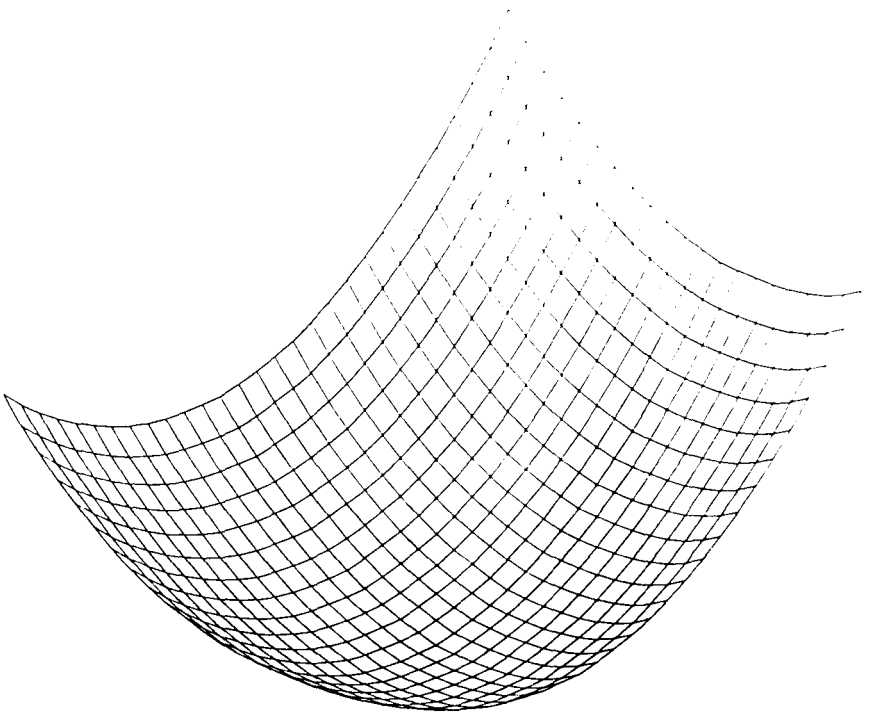

Figure 6.1 Modified algorithm with Butland slopes.

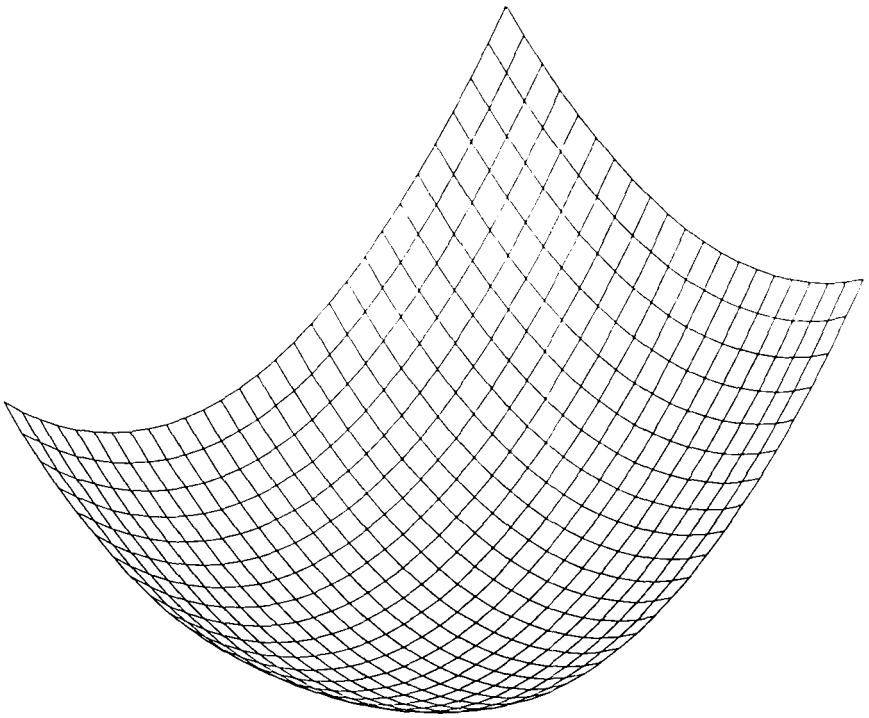

Figure 6.2 Modified algorithm with Frey slopes.

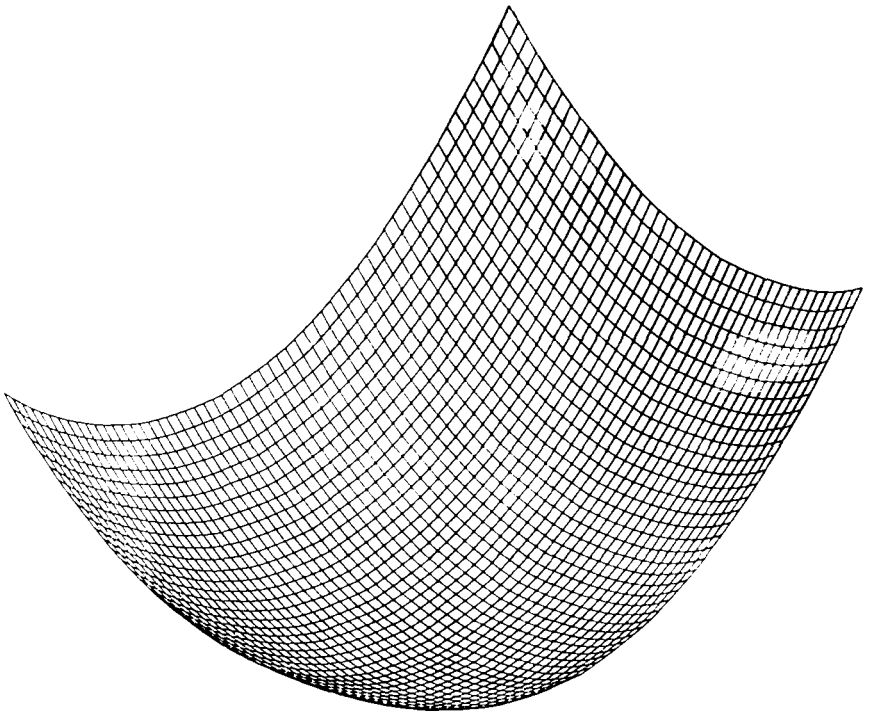

Figure 6.3 Modified algorithm with Butland slopes.

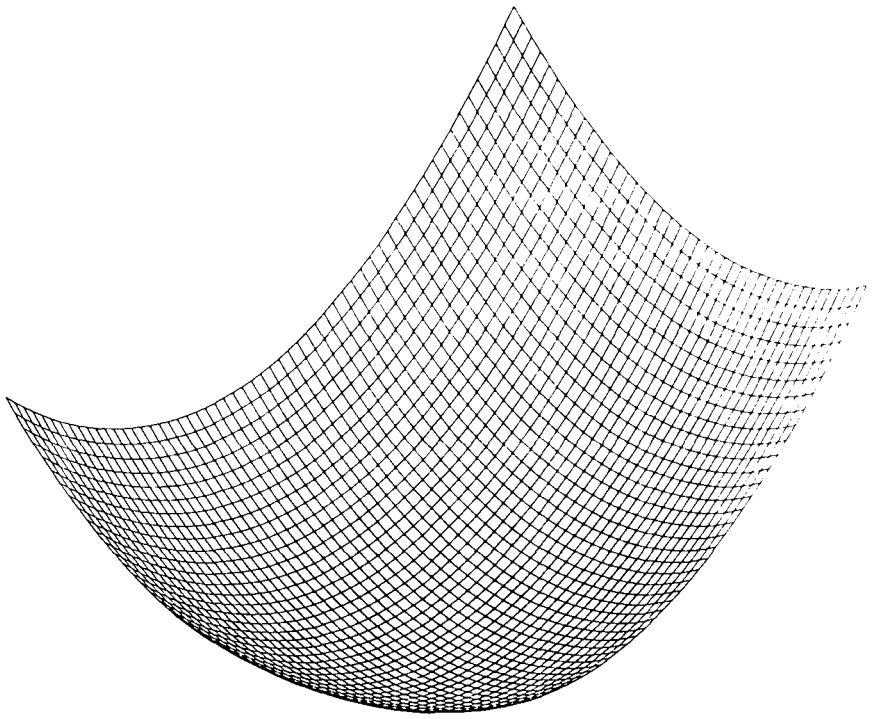

Figure 6.4 Modified algorithm with Frey slopes.

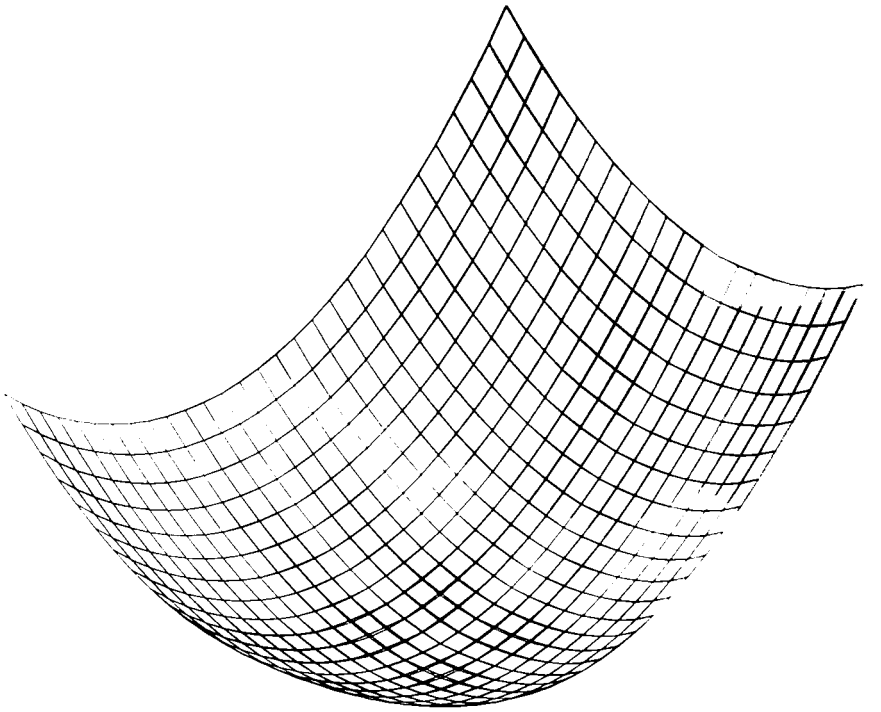

Figure 6.5 Combined plot by Modified algorithm with Butland and Frey Slopes.

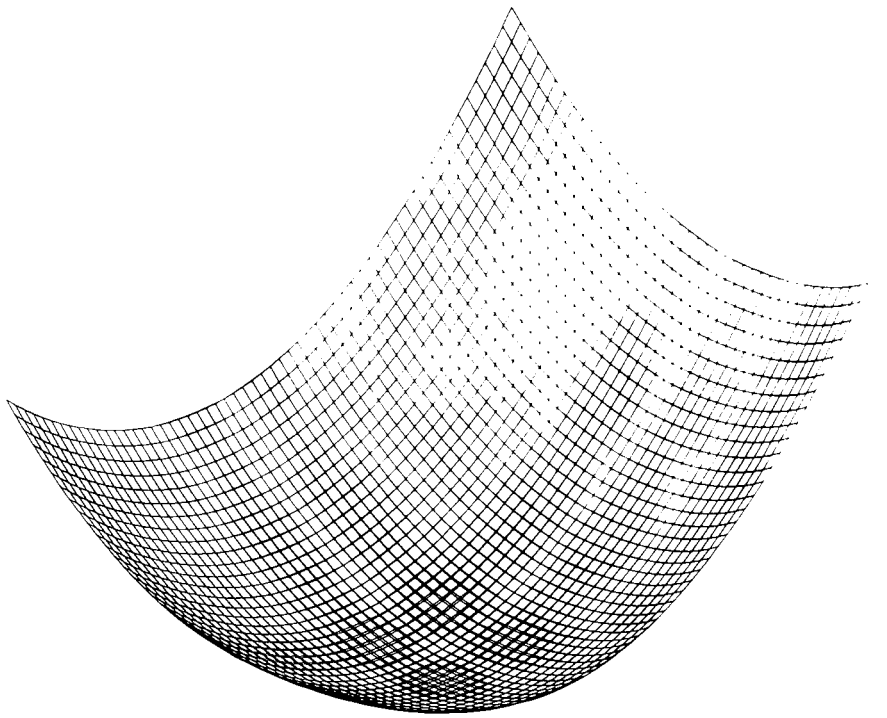

Figure 6.6 Combined plot by Modified algorithm with Butland and Frey Slopes.

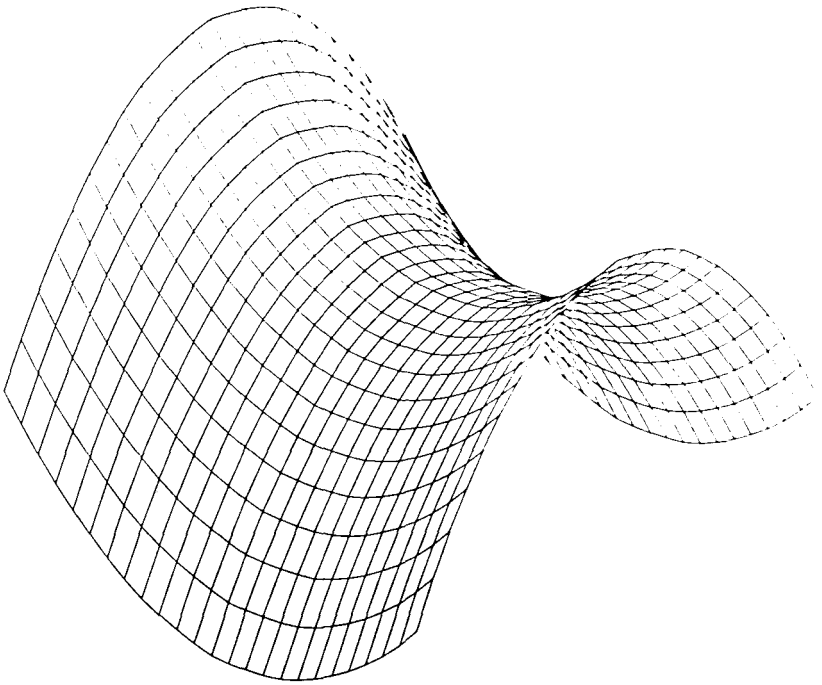

Figure 6.7 Modified algorithm with Butland slopes.

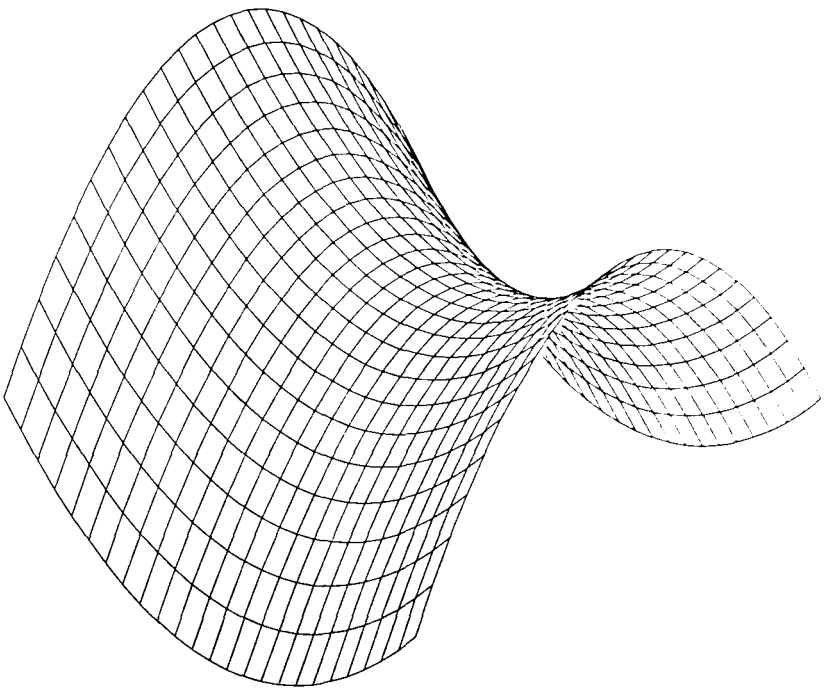

Figure 6.8 Modified algorithm with Frey slopes.

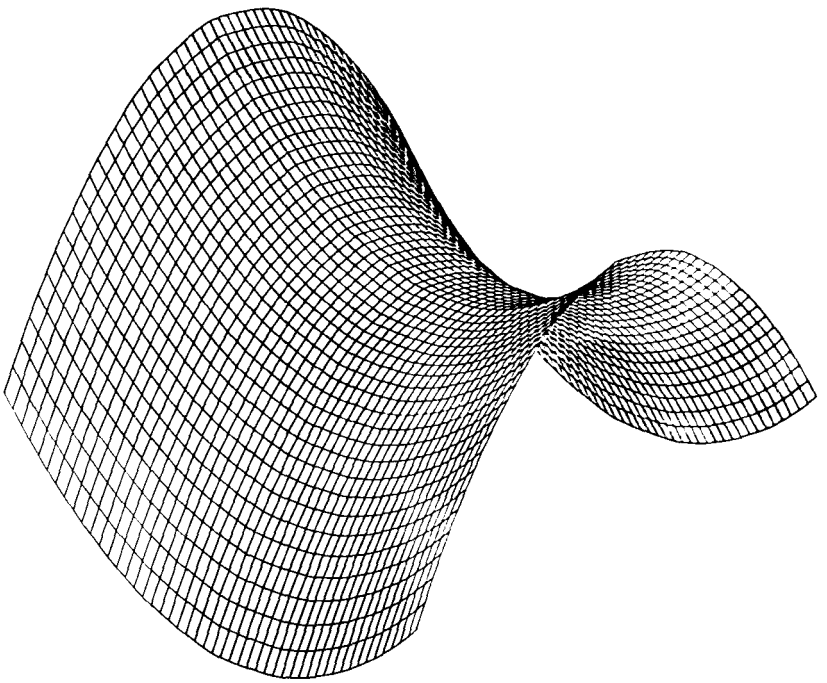

Figure 6.9 Modified algorithm with Butland slopes.

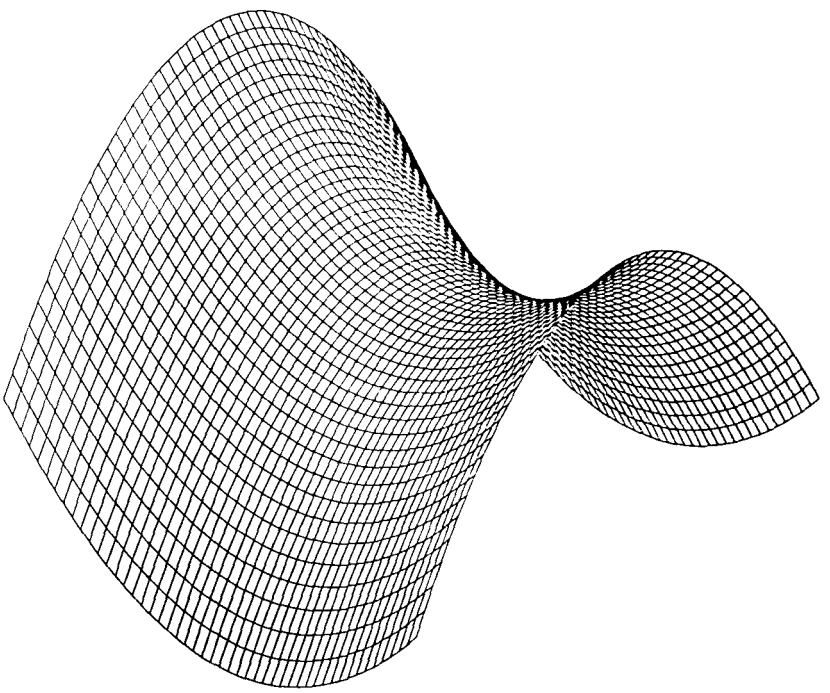

Figure 6.10 Modified algorithm with Frey slopes.

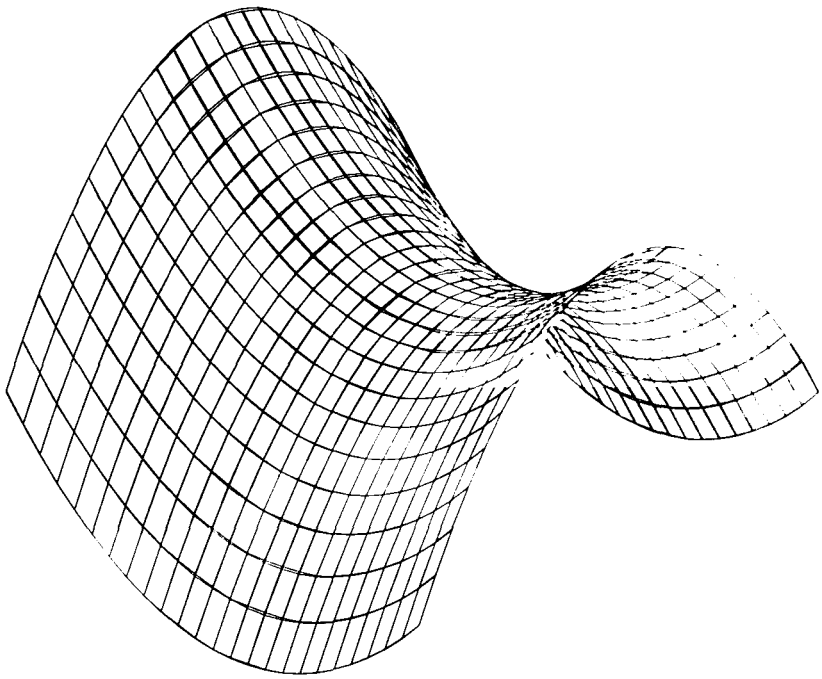

Figure 6.11 Combined plot by Modified algorithm with Butland and Frey Slopes.

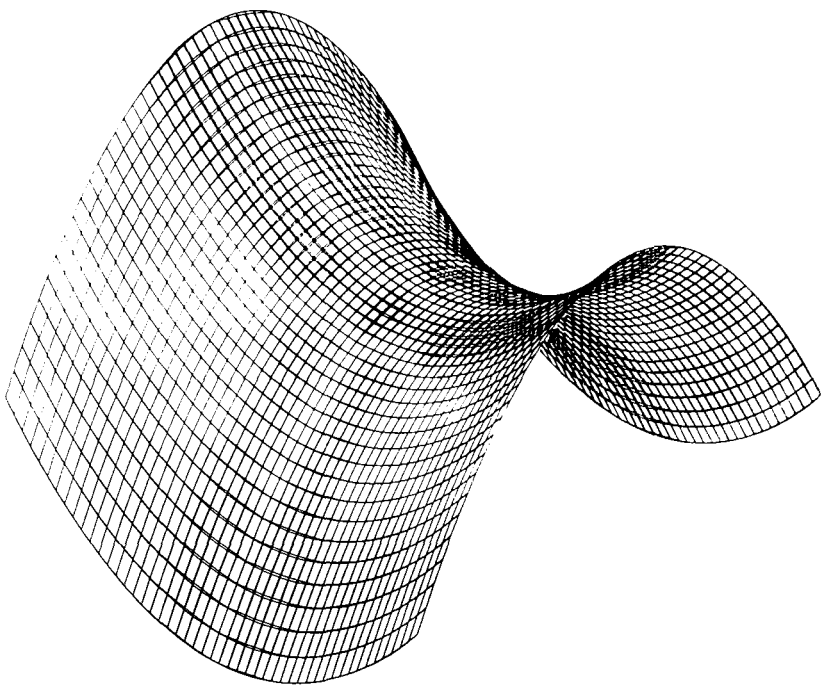

Figure 6.12 Combined plot by Modified algorithm with Butland and Frey Slopes.

Chapter 7

Conclusions and Future Work

7.1 Conclusions.

In this thesis we have presented several algorithms for generating shape-preserving interpolants to a set of arbitrary data using C^1 piecewise quadratic and cubic Hermite interpolation. These algorithms are local and have the distinct feature that the curve is constructed in a piecewise manner, and creation of each curve segment depends only on information related to a few neighbouring points. The most significant characteristic of algorithms of this category is that the slope at each data point is estimated before the actual generation of the curve segment takes place. The flexibility and versatility of these algorithms make them applicable to any situation which requires preservation of shape properties such as monotonicity and/or convexity. The main conclusions of this research work are summarized as follows:

- In Chapter 2, a survey of a number of the currently accepted shape-preserving interpolation methods based on piecewise cubic splines has been provided. Several methods including a number of formulae for specifying the derivative values at the data points have been reviewed and the intercomparisons have also been carried out by examining their performance on four data sets. Careful observation of the graphical results has suggested that the Brodlie, Fritsch-Butland, Huynh and Pruess methods are the best on the basis of producing visually pleasing

shapes of the interpolating curves. However, the Pruess method is not efficient as it inserts two additional knots in each interval and requires solving an optimization problem in the case of convex data sets, which results in a corresponding increase in the complexity of computer code and execution time than the other methods.

- In Chapter 3, algorithms proposed by McAllister-Roulier and Schumaker for the construction of interpolating curves which preserve the monotonicity and/or convexity of the data using C^1 quadratic splines have been analysed and proved to give precisely the same interpolants, if the slopes used in both algorithms are estimated by the Butland slope method. Furthermore, in the case of convex data, the slopes have been improved iteratively using the scheme proposed by Frey to produce more visually pleasing curves. The Schumaker algorithm has the merit that it is easy to implement, and that it needs fewer operations and less computation time than the McAllister-Roulier algorithm as it introduces an additional knot only in those intervals where $d_i + d_{i+1} \neq 2\delta_i$, whereas the latter algorithm always adds one extra knot in each interval.

- In Chapter 4, a new slope estimation method based on a generalized harmonic mean of chord slopes with a control parameter t has been introduced which provides a means for generating the slope at each data point. An advantage of this technique is that the slope at a data point is chosen within bounds given by the slopes of the two adjacent data segments to that data point. Based on this slope generation scheme, a new local automatic algorithm which has the property of preserving both monotonicity and/or convexity has been developed.

Necessary and sufficient conditions for a curve to be monotone and/or convex have been established. Compared to a number of current C^1 curve interpolation methods which have been presented in Chapter 2, this algorithm demonstrates its superiority in producing curves of high quality which are also visually pleasing.

- Two interactive algorithms have been described in Chapter 5 for the generation of interpolating curves which preserve the monotonicity and/or convexity of the data using C^1 quadratic and cubic splines respectively. Similar to above methods, these algorithms are also local and depend upon the new slope estimation formulae for determining the slope of the curve at each data point. Testing several sets of data has shown that changing the derivative value at the respective data point is an effective way of modifying the shape of the curve. These examples also illustrate that these algorithms give the user a control parameter t_i for the derivative value associated with each data point, which is utilized to modify the shape of the curve segment in an interval locally. These interactive algorithms are a powerful addition to the local one-pass algorithms and are capable of producing shape-preserving curves which are more visually pleasing.

- Finally in Chapter 6, the application of Schumaker's algorithm for shape-preserving curve for constructing a C^1 surface which interpolates given bivariate convex data defined on a rectangular grid is considered. An algorithm proposed by the Roulier is modified in which the above algorithm is used to generate curves along the grid lines. The Modified algorithm is much faster and gives more visually pleasing surfaces.

Given the summary of the methods listed above, it is now possible to analyse some of the main advantages of the new algorithms, in terms of general characteristics, when compared with other methods outlined in Chapter 2 and Chapter 3. The following overall conclusions can be drawn:

- An important advantage of the new algorithms is that they preserve *both* monotonicity and/or convexity of the data and produce visually pleasing curves. Undesirable features such as points of inflection and flat regions do not occur unless specified by the given data.

- Their defining formulae are quite simple and the generation of the curve segments is done on a local basis, in order to allow local changes in the curve. This local characteristic is desirable because adding, changing or removing a data point will only alter the curve in the vicinity of that point and is also important when storage requirements are critical as is the case for very large data sets.

- The specific advantage of the new automatic algorithm developed in Chapter 4 is that it is simple, fast, easy to code and from the standpoint of speed and storage; it does not require either the solution of an optimization problem as in the Pruess method or the insertion of additional knots between the original data points as seen in the methods described in Chapter 3.

- The main difference between the new interactive algorithms proposed in Chapter 5 is the size of storage required and the time of computation. The interactive algorithm based on C^1 quadratic splines adds one additional knot in each interval when $d_i + d_{i+1} \neq 2\delta_i$, whereas the other algorithm makes use of C^1 cubic splines and does not insert any extra

knot. Both algorithms provide the flexibility to design a large variety of visually pleasing shapes of the curves, by varying the parameter t_i which can be used interactively to give a means of local control over the shape of the curve.

7.2 Future Work.

While this research has attempted to gain insight into the shape-preserving algorithms issues associated with curve and surface design, many areas need further investigation. Even though the following list is suggested as an immediate extension to what has been achieved so far, it does not in any way exhaust all possiblities for future research. With this in mind, we would like to identify the following research topics which are highly relevant to the present work and deserve further study:

- We have considered the case of generating shape-preserving curves through C^1 quadratic and cubic splines. An attempt may be made to generalize this to higher degrees, i.e., quintic splines. In this regard, our new slope estimation method can be used in conjuction with the Ulrich-Watson [66] method, where the monotonicity region for quintic splines consists of the square $[0,5]\times[5,0]$ and derivative values must lie within it. The derivative values can be forced to lie in this region using the slope estimation formula (5.6) with $w_1 = 1$ and $1 \leq w_2 \leq 2$. A similar approach to ours (as described in Chapter 4) might be developed to construct an algorithm for generating the curves. Further investigations could also be carried out to build an interactive algorithm by varying the values of t to produce more visually pleasing curves.

- Our new slope estimation method has promising results when it is used with the piecewise rational splines of Stineman [65]. For example, Figure 7.1 shows the effect of progressively increasing the value of t. Further work is now being undertaken to investigate the possiblities of building automatic and interactive algorithms based on this rational representation.

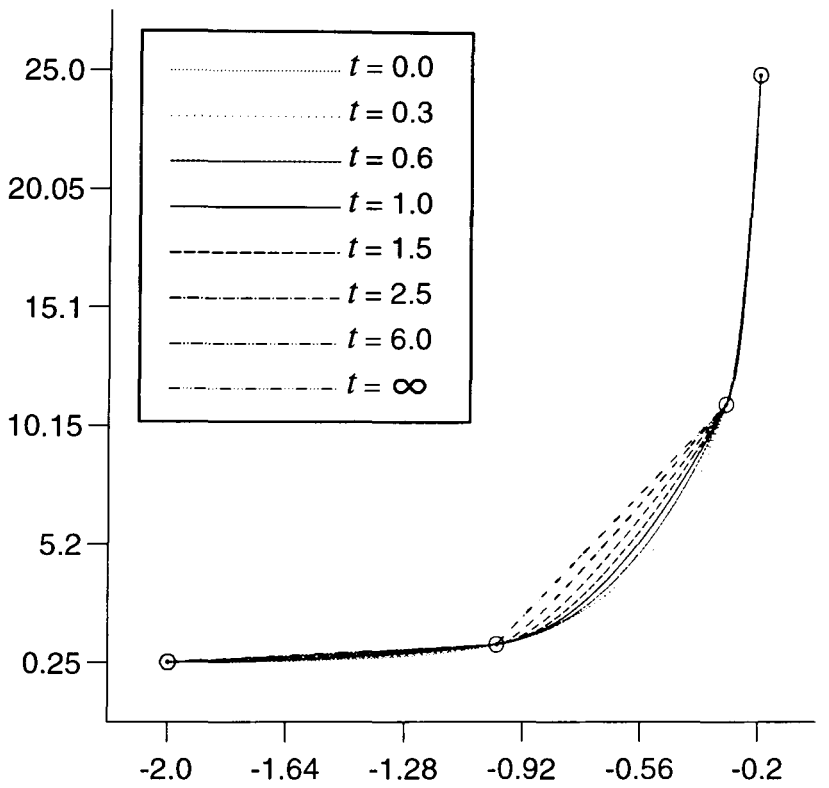

Figure 7.1 Interactive rational spline interpolant to Data 4.

- For surface interpolation by blending function schemes, the above rational scheme may be employed using Coon's technique as discussed in Gordon [38]. More analysis will be needed to investigate the possiblities of preserving the shape of the surface inside rectangles.

- An investigation into the possible modification of the Roulier approach

to construct an interactive algorithm using the new slope estimation method could be made. This may allow different values of the parameter t in different segments of the surface and different values of t in different directions.

The purpose of this research, as pointed out from the outset, was to develop shape-preserving algorithms in one and two dimensions. From the research that we have carried out and which is reported in this thesis, we can conclude that C^1 piecewise quadratic and cubic Hermite interpolation techniques are very powerful and flexible tools for the construction of shape-preserving algorithms for curves and surfaces. They offer smooth and good visual quality interpolants, in association with the new slope estimation formula which invloves a parameter t to control the size of estimated slope, and a simple representation of the underlying splines. It is hoped that this work will contribute to the development of systems for curve design and to the understanding of the mechanisms linking imposed shape-preserving constraints to properties of required shapes.

References

[1] J. H. Ahlberg, "The theory of splines and their applications", Academic Press, New York, 1967.

[2] H. Akima, "A new method of interpolation and smooth curve fitting based on local procedures", *J. ACM*, 17 (1970), pp. 589-602.

[3] S. Asaturyan and K. Unsworth, "A C^1 monotonicity preserving surface interpolation scheme", *The Mathematics of Surfaces III* , D.C. Handscomb, eds., Clarendon Press, Oxford, UK, 1989, pp. 243-263.

[4] B. A. Barsky, "Exponential and polynomial methods for applying tension to an interpolating spline curve", *Computer Vision, Graphics, and Image Processing*, 27 (1984), pp. 1-18.

[5] R. K. Beatson and H. Wolkowicz "Post-processing piecewise cubics for monotinicity", *SIAM J. Numer. Anal.*, 26 (1989), pp. 480-502.

[6] R. K. Beatson and Z. Ziegler, "Monotonicity preserving surface interpolation", *SIAM J. Numer. Anal.*, 22 (1985), pp. 401-411.

[7] C. de Boor, "A practical guide to splines", Springer-Verlag, New York, 1978.

[8] C. de Boor, "Package for calculating with B-splines", *SIAM J. Numer. Anal.*, 14 (1977), pp. 441-472.

[9] K. W. Brodlie, "A review of methods for curve and function drawing", in "Mathematical methods in computer graphics and design", (K. W. Brodlie ed.), pp. 1-37, Academic Press, New York and London, 1980.

[10] W. Burmeister, W. Heb and J. W. Schmidt, "Convex spline interpolants with minimal curvature", *Computing*, 35 (1985), pp. 219-229.

[11] J. Butland, "A method of interpolating reasonably-shaped curves through any data", *Proc. Computer Graphics 80*, Online Publication Ltd,

Middlesex, U.K, 1980, pp. 409-422.

[12] R. E. Carlson and F. N. Fritsch, "Monotone piecewise bicubic interpolation", *SIAM J. Numer. Anal.*, 22 (1985), pp. 386-400.

[13] R. E. Carlson and F. N. Fritsch, "An algorithm for monotone piecewise bicubic interpolation", *SIAM J. Numer.Anal.*, 26 (1989), pp. 1-9.

[14] A. K. Cline, "Scalar and planar-valued curve fitting using splines under tension", *Commun. ACM*, 17 (1974), pp. 218-220.

[15] P. Costantini, "An algorithm for computing shape-preserving splines of arbitrary degree", *J. Comput. Appl. Math.*, 22 (1988), pp. 89-136.

[16] P. Costantini, "Co-monotone interpolating splines of arbitrary degree. A local approach", *SIAM J. Sci. Stat. Comput.*, 8 (1987), pp. 1026-1034.

[17] P. Costantini, "On monotone and convex spline interpolation", *Math. Comp.*, 46 (1986), pp. 203-214.

[18] P. Costantini and F. Fontanella, "Shape-preserving bivariate interpolation", *SIAM J. Numer. Anal.*, 27 (1990), pp. 488-506.

[19] M. G. Cox, "An algorithm for spline interpolation", *J. Inst. Math. Appl.*, 15 (1975), pp. 95-108.

[20] R. Delbourgo, "Accurate C^2 rational interpolants in tension", *SIAM J. Numer. Anal.*, 30 (1993), pp. 595-607.

[21] R. Delbourgo, "Shape-preserving interpolation to convex data by rational functions with quadratic numerator and linear denominator", *IMA J. Numer. Anal.*, 9 (1989), pp. 123-136.

[22] R. Delbourgo and J. A. Gregory, "C^2 rational quadratic spline interpolation to monotonic data", *IMA J. Numer. Anal.*, 3 (1983), pp. 141-152.

[23] R. Delbourgo and J. A. Gregory, "Shape-preserving piecewise rational interpolation", *SIAM J. Sci. Stat. Comput.*, 6 (1985), pp. 967-876.

[24] S. L. Dodd, D. F. McAllister and J. A. Roulier, "Shape-presering spline

interpolation for specifying bivariate functions on grids", *IEEE Computer Graphics and Applications*, 3 (1983), pp. 70-79.

[25] R. L. Dougherty, A. Edelman and J. H. Hyman, "Nonnegativity, monotonicity or convexity preserving cubic and quintic Hermite interpolation", *Math. Comp.*, 52 (1989), pp. 471- 494.

[26] A. Edelman and C. A. Micchelli "Admissible slopes for monotone and convex interpolation", *Numer. Math.*, 51 (1987), pp. 441-458.

[27] S. C. Eisenstat, K. R. Jackson and J. W. Lewis, "The order of monotone piecewise cubic interpolation", *SIAM J. Numer. Anal.*, 22 (1985), pp. 1220-1237.

[28] J. C. Fiorot and J. Tabka, "Shape-preserving C^2 cubic polynomial interpolating splines", *Math. Comp.*, 57 (1991), pp. 291-298.

[29] Y. Fletcher and D. F. McAllister, "An analysis of tension methods for convexity-preserving interpolation", *IEEE Computer Graphics and Applications*, 7 (1987), pp. 7-14.

[30] Y. Fletcher and D. F. McAllister, "Automatic tension adjustment for interpolatory splines", *IEEE Computer Graphics and Applications*, 10 (1990), pp. 10-17.

[31] W. H. Frey, "A useful variant of McLaughlin's interpolant", *Technical Report GMR-5004*, General Motors Research Lab., Warren, Mich., May 1985.

[32] F. N. Fritsch and J. Butland, "A method of constructing local monotone piecewise cubic interpolants", *SIAM J. Sci. Stat. Comput.*, 5 (1984), pp. 300-304.

[33] F. N. Fritsch and R. E. Carlson, "Monotone piecewise cubic interpolation", *SIAM J. Numer. Anal.*, 17 (1980), pp. 238-246.

[34] M.G. Gasparo and R. Morandi, "Piecewise cubic monotone interpolation with assigned slopes", *Computing*, 46 (1991), pp. 355-365.

[35] J. A. Gregory, "Shape-preserving spline interpolation", *Computer Aided Design*, 18 (1986), pp. 53-57.

[36] J. A. Gregory, "Smooth interpolation without twist constraints", Computer Aided Geometric Design, R. E. Rarnhill and R. F. Riesenfeld. eds., Academic Press, New York and London, 1974, pp. 71-87.

[37] J. A. Gregory and R. Delbourgo, "Piecewise rational quadratic interpolation to monotonic data", *IMA J. Numer. Anal.*, 2 (1982), pp. 123-130.

[38] W. J. Gordon, "Spline blended surface interpolation through curve networks", *J. Math. and Mech.*, 18 (1969), pp. 931-951.

[39] G. H. Hardy, J. E. Littlewood and G. Polya, "Inequalities", Cambridge University Press, 1959.

[40] H.T. Huynh, "Accurate monotone cubic interpolation", *SIAM J. Numer. Anal.*, 30 (1993), pp. 57-100.

[41] R. Iqbal, "A one-pass algorithm for shape-preserving quadratic spline interpolation", *J. Sci. Comput.*, 7 (1992), pp. 359-376.

[42] R. Iqbal, "An algorithm for convexity-preserving surface interpolation", *J. Sci. Comput.*, 9 (1994), pp. 197-212.

[43] L. D. Irvine, S. P. Marin and P. W. Smith, "Constrained interpolation and smoothing", *Constr. Approx.*, 2 (1986), pp. 129-151.

[44] P. D. Kaklis and D. G. Pandelis, "Convexity-preserving polynomial splines of non-uniform degree", *IMA J. Numer. Anal.*, 10 (1990), pp. 223-234.

[45] A. Lahtinen, "On the construction of monotony preserving taper curves", *Acta For. Fennica*, 203 (1988), pp. 1-34.

[46] A. Lahtinen, "Shape-preserving interpolation by quadratic splines", *J. Comput. App. Math.*, 29 (1990), pp. 15-24.

[47] P. Lancaster and K. Salkauskas, "Curve and surface fitting: An introduction", Academic Press, London, 1986.

[48] E. T. Y. Lee, "A simplified B-spline computation routine", *Computing*, 29 (1982), pp. 365-371.

[49] D.F. McAllister, E. Passow and J.A. Roulier, "Algorithms for computing shape-preserving spline interpolation to data", *Math. Comp.*, 31 (1977), pp. 717-725.

[50] D.F. McAllister and J.A. Roulier, "An algorithm for computing a shape-preserving osculatory quadratic spline", *ACM Trans. Math. Software*, 7 (1981), pp. 331-347.

[51] B. J. McCartin, "Computation of Exponential splines", *SIAM J. Sci. Stat. Comput.*, 11 (1990), pp. 242-262.

[52] L. B. Montefusco, "An interactive procedure for shape-preserving cubic spline interpolation", *Computers and Graphics*, 11 (1987), pp. 389-392.

[53] S. Pruess, "An algorithm for computing smoothing splines in tension", *Computing*, 19 (1978), pp. 365-373.

[54] S. Pruess, "Shape preserving C^2 cubic spline interpolation", *IMA J. Numer. Anal.*, 13 (1993), pp. 493-507.

[55] V. Ramirez and J. Lorente, "C^1 rational quadratic spline interpolation to convex data", *Applied Numerical Mathematics*, 2 (1986), pp. 37-42.

[56] R. J. Renka, "Interpolatory tension splines with automatic selection of tension factors", *SIAM J. Sci. Stat. Comput.*, 8 (1987), pp. 393-415.

[57] P. Rentrop, "An algorithm for the computation of exponential spline", *Numer. Math.*, 35 (1980), pp. 81-93.

[58] J.A. Roulier, "A convexity-preserving grid refinement algorithm for interpolation of bivariate functions", *IEEE Computer Graphics and Applications*, 7 (1987), pp. 57-62.

[59] J. W. Schmidt and W. Heb, "An always successful method in univariate convex C^2 interpolation", *Numer. Math.*, 71 (1995), pp. 237-252.

[60] I. J. Schoenberg, "Contributions to the problem of approximation of equidistant data by analytic functions", *Quart. Appl. Math.*, 4 (1946), pp. 45-99.

[61] L.L. Schumaker, "On shape-preserving quadratic spline interpolation", *SIAM J. Numer. Anal.*, 20 (1983), pp. 854-864.

[62] L. L. Schumaker, "Spline functions : Basic theory", Wiley-Interscience, New York, 1981.

[63] D. G. Schweikert, "An interpolation curve using a spline in tension", *J. Math. Phys.*, 45 (1966), pp. 312-317.

[64] H. Spath, "Exponential spline interpolation", *Computing*, 4 (1969), pp. 225-233.

[65] R.W. Stineman, "A consistently well-behaved method of interpolation", *Creative Computing*, July 1980, pp. 54-57.

[66] G.Ulrich and L.T. Watson, "Positivity conditions for quintic polynomials", *SIAM J. Sci. Comput.*, 15 (1994), pp. 528-544.

[67] Z. Yan, "Piecewise cubic curve fitting algorithm", *Math. Comp.*, 49 (1987), pp. 203-213.

[68] Z. J. Zhang, Z. Q. Yang and C. M. Zhang, "Monotone piecewise curve fitting algorithms", *J. Comp. Math.*, 12 (1994), pp. 163-172.