MONOTONICITY PRESERVING PARAMETRIZATION FOR SPLINE INTERPOLATION

Capt. Opas Udomsomboonpon

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Applied Mathematics
Suranaree University of Technology
Academic Year 1999

ISBN 974-7359-30-8

การสร้างพารามิเตอร์เพื่อคงความโมโนโทนสำหรับการประมาณค่าในช่วงด้วยสไปลน์

ร.อ.โอภาส อุดมสมบูรณ์ผล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาคณิตศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีสุรนารี

ปีการศึกษา 2542

ISBN 974-7359-30-8

Thesis Title

Monotonicity Preserving Parametrization for Spline Interpolation

Suranaree University of Technology Council has approved this thesis submitted in partial fulfillment of the requirements for a Master's Degree

Thesis Examining Committee

[illegible]

(Assoc. Prof. Dr. Prapasri Asawakun)

Chairman

[illegible]

(Assoc. Prof. Dr. Boris I. Kvasov)

Thesis Advisor

[illegible]

(Assoc. Prof. Dr. Suwon Tangmanee)

Member

[illegible]

(Assist. Prof. Dr. Tavee Lertpanyavit)

Vice Rector for Academic Affairs

[illegible]

(Assoc. Prof. Dr. Tassanee Sukosol)

Dean / Institute of Science

ร.อ.โอภาส อุดมสมบูรณ์ผล: การสร้างพารามิเตอร์เพื่อคงความโมโนโทนสำหรับการประมาณค่าในช่วงด้วยสไปลน์ (MONOTONICITY PRESERVING PARAMETRIZATION FOR SPLINE INTERPOLATION) อ.ที่ปรึกษา : รศ. ดร.บอริส ไอ ควาซอร์ฟ, 90 หน้า.

ISBN 974-7359-30-8

ขั้นตอนวิธีการสร้างพารามิเตอร์แบบใหม่ สำหรับการประมาณค่าในช่วงด้วยสไปลน์ ได้ถูกสร้างขึ้นในวิทยานิพนธ์นี้ โดยรูปร่างของเส้นโค้งที่ได้จะดีกว่ารูปร่างของเส้นโค้ง ที่ได้จากการใช้พารามิเตอร์ด้วยความยาวคอร์ด การแบ่งที่เท่ากัน หรือการเข้าสู่ศูนย์กลาง การสร้างพารามิเตอร์แบบใหม่นี้ มุ่งเน้นการคงความโมโนโทนของข้อมูลเป็นหลัก โดยจะไม่แปรผันภายใต้การแปลงสัมพรรค ซึ่งสามารถเห็นความสอดคล้องได้จากเส้นโค้งกับข้อมูลเริ่มต้น ขั้นตอนวิธีนี้จะให้ค่าพารามิเตอร์เป็นจำนวนมาก เพื่อทำให้การประมาณค่าในบริเวณที่มีความชันสูงมีความเที่ยงมากขึ้น ตลอดจนสามารถนำไปใช้ประมาณค่าในกรณีพื้นผิว

สาขาวิชา คณิตศาสตร์ประยุกต์
ปีการศึกษา 2542

ลายมือชื่อนักศึกษา [illegible]
ลายมือชื่ออาจารย์ที่ปรึกษา [illegible]
ลายมือชื่ออาจารย์ที่ปรึกษาร่วม -

CAPT. OPAS UDOMSOMBOONPON: MONOTONICITY PRESERVING PARAMETRIZATION FOR SPLINE INTERPOLATION
THESIS ADVISOR: ASSOC. PROF. BORIS I. KVASOV, Ph.D. 90 PP.
ISBN 974-7359-30-8

computer aided geometric design, parametric spline interpolation, chord length, uniform, centripetal and monotonicity preserving parametrizations, mesh concentration in large gradient domains.

A new parametrization algorithm for interpolation by splines is given in this thesis which almost invariably results in better shapes than either chord length, uniform or centripetal parametrizations. The method is based on monotonicity preservation for the given data and is invariant under affine transformations. It enables one to improve the visual correspondence between curve and the initial data and gives a mesh concentration in large gradient domains. The algorithm can be generalized to approximate multivalued surfaces.

สาขาวิชา คณิตศาสตร์ประยุกต์	ลายมือชื่อนักศึกษา Capt. Opas.
ปีการศึกษา 2542	ลายมือชื่ออาจารย์ที่ปรึกษา [illegible]
	ลายมือชื่ออาจารย์ที่ปรึกษาร่วม -

Acknowledgment

First I would like to express my sincere thanks to my supervisor, Associate Professor Dr. Boris I. Kvasov, for his long-term guidance and patience, without which this thesis would not have been possible. His encouragement always keeps me going.

I am gratefully indebted to all the lectures who taught me in graduate courses at the School of Mathematics of Suranaree University of Technology and especially to the Chair, Associate Professor Dr. Prapasri Asawakun, for her whole-hearted support all these years.

I am also very thankful to Mr. Jessada Tanthanuch from graduate program in Applied Mathematics at SUT for his advice in computer programming and to Associate Professor Dr. Suwon Tangmanee, Associate Professor Dr. Pairote Sattayatham, Assistant Professor Dr. Eckart Schulz and Dr. Ajuna Chaiyasena for their help in many occasions during this study.

Finally, I wish to extend my appreciation and thanks to General Rawat Boontup and Lieutenant General Horm Holamyong for his continued support and encouragement.

Contents

Contents of Tables

Contents of Figures

Chapter I
Introduction

In many practical problems, we have to deal with interpolation of discrete data when geometric properties of the data such as positivity, monotonicity, convexity, presence of linear sections, the angles and the bends should be retained. Standard approaches such as spline interpolation, NURBS (nonuniform rational basis splines) and other usually fail in the treatment of this problem which is called *a shape preserving interpolation problem.* To obtain the required shape properties of the resulting curve/surface, different authors introduce some parameters into the structure of the spline to satisfy the geometric constraints. The key idea is to develop algorithms for an automatic choice of these parameters.

In the case of multivalued data, we need to combine the construction of shape preserving interpolatory functions with an appropriate parametrization.

Research on constructing shape preserving interpolatory functions started with the spline in tension of Schweikert (1966) where exponential splines were used as approximants. This was followed by the work of *Späth* (1969, 1974), Nielson (1974), Pruess (1976, 1979) and de Boor (1978) with various exponential and cubic spline interpolants containing "tension parameters" to control shape. All of these approximations were interpolatory and globally C^2, but strictly speaking were not local in the sense that changing data at one point meant the entire approximation had to be regenerated. This made automatic algorithm for choosing free parameters to control shape (especially monotonicity) fairly complicated. McAllister, Passow and Roulier (1977) derived a method for generating shape preserving curves of arbitrary smoothness based on the properties of Bernstein polynomials, but to achieve C^2 smoothness they had to use piecewise polynomials of degree at least four. There is also the possibility of using piecewise rational interpolants (e.g., see Delbourgo and

Gregory (1985), Gregory and Delbourgo (1982)) although these are usually only C^1 or they are intended for strictly monotone or strictly convex data.

In 1980, Fritsch and Carlson (1980) proposed a shape preserving interpolatory cubic spline which was only C^1 globally, but consequently was local, and admitted much simpler algorithms for the choice of free parameters to control shape (Fritsch and Butland 1984). Recently Renka (1987) working on the exponential spline has produced an algorithm for automatically choosing tension parameters in the C^1 case together with an iterative approach to extend this in a special manner to C^2. Costantini (1987) also has families of shape preserving interpolants based on Bernstein polynomials; these are very simple to use but are co-monotone, i.e., the spline on the *i*th data interval is increasing or decreasing as the data on that interval. Such splines have the disadvantage that they must have slope zero at a point where the neighboring secant lines have a sign change in their slope; hence, any local extrema of the underlying approximation are assumed to be in the data sample. Also, to get globally C^2 interpolants, one must use quintic splines. Other examples of C^1 of shape preserving spline interpolants, are found in Burmeister, Hess and Schmidt (1985), and Schmidt and Hess (1987). Finally, there is the work of Dougherty, Edleman and Hyman (1989) where C^2 quintic splines are used; a fairly complete algorithm is given there for preserving monotonicity and there is also a considerable discussion concerning convexity for the piecewise cubic case.

There are cases where the added smoothness of the C^2 splines is often needed and it is desirable to avoid the global dependence of the tension parameters in existing algorithms. In the paper by Pruess (1993), it was shown that if two extra break points are allowed between each data subintervals (similar to Pruess (1979)), then there are enough degrees of freedom to construct a globally C^2 cubic spline interpolant which is local and which has slopes and curvatures at the data points as free parameters. Another local algorithm for constructing monotone and convex C^2 splines was developed in Kvasov (1996) using generalized tension splines.

In many CAGD applications, a user wishes to construct a smooth and visually pleasing parametric curve passing through some given 2D or 3D control points.

The two most commonly used parametrizations or knots spacings are the chord length spacing and the uniform spacing. It is easy to construct examples where the uniform knot spacing yields poor results by having two data points near each other and the next point significally farther away. Examples are given in de Boor (1978) and Farin (1993) where the data are unequally spaced on gentle curves, yet loops or oscillations occur between data points. Several authors (for example Boor (1978) and Farin (1993)) have suggested using chord length knot spacing partially because it approximates the arc length of a parametric curve. Another motivating factor for chord length is the result of Epstein (1976) which guarantees that there will be no cusps for the case of a closed periodic curve. As noted in Foley (1986), the chord length knot spacing often produces visually poor results when the data is poorly scaled, or when the direction of the data changes abruptly.

A nonlinear optimization problem is solved in Marin (1984) in order to determine a knot sequence for interpolation, and another optimization problem is solved in Hoschek (1988) to select a parametrization for an approximation (not interpolation) problem. A knot spacing recently developed by Lee (1989) and discussed in Farin (1993), is termed the centripetal model. This model was motivated by the paradigm of a car traveling through the data points, and it works well on some data sets. An interesting note related to the car paradigm is that this knot sequence, (also the chord and uniform methods), disregard the angles formed by the control polygon. It would be useful to take these angles into consideration.

The three spline interpolation methods which use the chord length, uniform and centripetal parametrizations, are each invariant under rotations, translations and equal scaling in the x and y coordinates. However, the chord length and the centripetal methods are not scale invariant when the x and y coordinates of the data are scaled differently. The geometric properties of scale, rotation and translation invariance are important in character font applications where characters are stored as a sequence of control points which are transformed to the desired position, and a parametric spline interpolant is formed to represent the character. The shape of the characters should be consistent if the defining data points are scaled differently in x and y, rotated, translated or sheared. Another justification for wanting scale invariance is that the resulting curve should be the same regardless of whether the x-axis is measured in seconds, minutes or hours, and the y-axis is measured in inches, feet or meters, for example. An easy solution to having a scale invariant method would be to form the bounding box of the data points and map this rectangle to the unit square. Such a scheme, however, would not be rotation invariant.

In applications involving robot motion and animation, if $P_i \to P_{i+1}$ and the other data points remain fixed, then it is desirable for $h_i = t_{i+1} - t_i \to 0$, because h_i represents the time it takes to travel the distance between these two points. The chord length and centripetal parametrizations satisfy this property, but the uniform spacing does not. Uniform spacing often yields poor results when the data points are unevenly distributed because it does not take into account any measure of distance between points. The chord length knot spacing has a property which is useful in motion applications in that the average speed between data points is constant in linear regions. In terms of ratios, if $P_{i-2}, \ldots, P_{i+2}$ are collinear and the average speed from P_{i-1} to P_i is equal to the average speed from P_i to P_{i+1}, then $\left|P_i - P_{i-1}\right| / h_{i-1} = \left|P_{i+1} - P_i\right| / h_i$.Since there are possible corners to negotiate at P_{i-2} and P_{i+2}, we would not expect to have similar ratios involving h_{i-2} and h_{i+1}. The centripetal and uniform methods do not have this property, and these methods may yield extraneous oscillations in regions where the data points are not equally spaced and the points are linear or gently curved. A property that all three of these methods have is that the parametric spline curves are continuous with respect to small changes in the data.

We feel that in addition to distance, the relative shape of the control polygon should be considered when motivated by the car paradigm because you would like to slow down for corners and travel at a constant speed on linear stretches. Another aspect that we like to incorporate is the affine invariant metric described in Nielson (1987) and Nielson and Foley (1989). By using this metric, poorly scaled data generally cause no problems. More importantly, it should not matter if the x-axis is measured in seconds, minutes or hours, and the y-axis is measured in inches, feet or meters. The interpolating curve should be independent of these arbitrary choices.

In this thesis, a new algorithm for a mesh construction by C^2 parametric cubic spline interpolation is developed. The algorithm is based on monotonicity preservation for the given data and is invariant under affine transformations. The mesh knots can easily be calculated from explicit formulae and almost invariably result in better shapes than either the chord length or the uniform parametrizations. This enables one to improve the visual correspondence between curve and the initial data and gives a mesh concentration in large gradient domains. The algorithm is generalized for the case of surface approximation. Its possibilities are illustrated by the test examples.

A large variety of applications now requires the use of curve/surface description, especially in fields such as computer aided design and machining, and computer vision and inspection of manufactures parts. Other areas where the description of curves/surfaces is of interest include many fields of science, medicine, cartography, television and the film industry. This diversity and the wide range of applicability of the subject enables us to consider the problem of constructing monotonicity preserving curve/surface spline parametrization as very valuable.

Chapter II

Cubic Spline Interpolation

In this chapter, we describe some of the algorithms for computing cubic interpolating splines which are most often used in practical spline approximation. Cubic splines combine the smoothness properties required in many applications with the simplicity of their computer calculation and high accuracy of approximation. However, in a number of cases the behaviour of cubic splines does not conform to the properties of the initial data. Visually, this is evidenced by the presence of jumps, oscillations, and various deviations not characteristic of a given set of points. These features may be expressed mathematically as nonmonotonicity and the presence of inflection points of the spline on intervals where the data is monotone and convex. One can obtain "correct" behaviour of the spline by increasing the number of interpolation nodes. If this is impossible, then one should use the methods of shape preserving approximation described further in chapter 3.

2.1 Cubic Interpolating Splines

Suppose that we want to interpolate a function f given by the data (x_i, f_i), $i = 0,1,\ldots,N$, where $f_i = f(x_i)$ and the points x_i from an ordered sequence $a = x_0 < x_1 < \cdots < x_N = b$. This interpolation problem can be solved very efficiently by using cubic splines.

Definition 2.1. A cubic interpolating spline is a function $S \in C^2[a,b]$ such that

(i) on every interval $[x_i, x_{i+1}]$, the function S is a cubic polynomial (of order four)

$$S(x) \equiv S_i(x) = a_{i,0} + a_{i,1}(x - x_i) + a_{i,2}(x - x_i)^2 + a_{i,3}(x - x_i)^3$$
$$\text{for } x \in [x_i, x_{i+1}], \quad i = 0,1,\ldots,N-1;$$

(ii) the consecutive polynomials are smoothly adjusted

$$S^{(r)}(x_i - 0) = S^{(r)}(x_i + 0), \quad i = 1,\ldots,N-1,\ r = 0,1,2;$$

(iii) the interpolation conditions are satisfied

$$S(x_i) = f_i, \quad i = 0,\ldots,N.$$

According to this definition, a cubic interpolating spline S is an ordered set of cubic polynomials which match up smoothly and form a twice continuously differentiable function. The points x_i, $i = 1,\ldots,N-1$, where the polynomials are matched, are called the *knots* of the spline, and must not coincide with the interpolation nodes. The knots of the spline can also have different multiplicities depending on the number of the adjusted derivatives. In particular, a knot x_i has multiplicity k_i $(0 \le k_i \le 3)$ if the derivatives of two consecutive polynomials are matched in this knot up to the order 3-k_i. However, in this chapter we consider cubic splines with only simple knots (of multiplicity 1).

Each of the N polynomials forming a spline has 4 coefficients, that gives us a total of 4N parameters. From this number, one needs to subtract 3(N-1) conditions of smoothness and N+1 conditions of interpolation. The remaining two free parameters (4N-3(N-1)-N-1 = 2) are usually determined from the restrictions on the values of the spline and its derivatives at the endpoints of the interval $[a,b]$ (or near its ends). These restrictions are called the *endpoint constraints*. There exist several different types of endpoint constraints, among which the most common are the following four boundary conditions:

1. First derivative endpoint conditions:

$$S'(x_0) = f_0' \text{ and } S'(x_N) = f_N';$$

2. Second derivative endpoint conditions:

$$S''(x_0) = f_0'' \text{ and } S''(x_N) = f_N'';$$

3. Periodic endpoint conditions:

$$S^{(r)}(x_0) = S^{(r)}(x_N),\ r=0,1,2;$$

4. "Not-a-knot" endpoint conditions where adjacent polynomials nearest to the endpoints of the interval $[a,b]$ coincide: $S_0(x) \equiv S_1(x)$ and $S_{N-1}(x) \equiv S_N(x)$, that is,

$$S'''(x_i - 0) = S'''(x_i + 0), \quad i=1, N-1.$$

It is natural to consider periodic endpoint conditions by assuming that the interpolated function f is periodic with the period $b-a$.

2.2 Defining Relations for the Construction of a Cubic Interpolating Splines

The second derivative S'' of a cubic spline is a continuous piecewise linear function. Thus, using the notation $M_i = S''(x_i),\ i=0,\dots,N,$ one can write

$$S''(x) \equiv S_i''(x) = \frac{M_i}{h_i}(x_{i+1} - x) + \frac{M_{i+1}}{h_i}(x - x_i), \quad x \in [x_i, x_{i+1}], \tag{2.1}$$

where $h_i = x_{i+1} - x_i,\ \ i=0,\dots,N-1.$

Integrating (2.1) twice will introduce two constants of integration, and the result can be expressed in the form

$$S_i(x) = \frac{M_i}{6h_i}(x_{i+1} - x)^3 + \frac{M_{i+1}}{6h_i}(x - x_i)^3 + C_{1,i}(x_{i+1} - x) + C_{2,i}(x - x_i). \tag{2.2}$$

Substituting x_i and x_{i+1} into equation (2.2) and using the values $f_i = S_i(x_i)$ and $f_{i+1} = S_i(x_{i+1})$ yields the following equations involving $C_{1,i}$ and $C_{2,i}$, respectively:

$$f_i = \frac{M_i}{6}h_i^2 + C_{1,i}h_i \quad \text{and} \quad f_{i+1} = \frac{M_{i+1}}{6}h_i^2 + C_{2,i}h_i.$$

These two equations are easily solved for $C_{1,i}$ and $C_{2,i}$, and when these values are substituted into equation (2.2), the result is the following expression for the cubic function S on $[x_i, x_{i+1}]$:

$$S_i(x) = \frac{M_i}{6h_i}(x_{i+1} - x)^3 + \frac{M_{i+1}}{6h_i}(x - x_i)^3 + (\frac{f_i}{h_i} - \frac{M_i h_i}{6})(x_{i+1} - x) + (\frac{f_{i+1}}{h_i} - \frac{M_{i+1}h_i}{6})(x - x_i). \tag{2.3}$$

To find the unknown coefficients M_i, $i = 0, \ldots, N$, one must use the derivative of (2.3), which is

$$S_i'(x) = -\frac{M_i}{2h_i}(x_{i+1} - x)^2 + \frac{M_{i+1}}{2h_i}(x - x_i)^2 - (\frac{f_i}{h_i} - \frac{M_i h_i}{6}) + (\frac{f_{i+1}}{h_i} - \frac{M_{i+1}h_i}{6}) \tag{2.4}$$

Evaluating (2.4) at x_i and simplifying the result yields

$$S_i'(x_i + 0) = -\frac{M_i}{3}h_i - \frac{M_{i+1}}{6}h_i + \Delta_i f, \quad \Delta_i f = \frac{f_{i+1} - f_i}{h_i}.$$

Similarly, we can replace i by $i-1$ in (2.4) to get an expression for $S'_{i-1}(x)$ and evaluate it at x_i to obtain

$$S'_{i-1}(x_i-0)=\frac{M_i}{3}h_{i-1}+\frac{M_{i-1}}{6}h_{i-1}+\Delta_{i-1}f.$$

As $S'_i(x_i+0)=S'_{i-1}(x_i-0),\ \ i=1,\ldots,N-1,$ we obtain

$$h_{i-1}M_{i-1}+2(h_{i-1}+h_i)M_i+h_iM_{i+1}=6\delta_i f,\ \ \delta_i f=\Delta_i f-\Delta_{i-1}f,$$
$$i=1,2,\ldots,N-1. \qquad (2.5)$$

The system (2.5) is underdetermined as it contains only N-1 equations for finding N+1 unknowns coefficients M_i. In order to complete this system, one needs two additional equations. The standard strategy is to make use of one of the above stated four endpoint conditions.

2.3 Endpoint Constraints and the Resulting Systems of Linear Equations

Using formulae (2.1) and (2.4), one can rewrite the endpoint conditions given in section 2.1 in the following form:

1. $2M_0+M_1=\frac{6}{h_0}[\Delta_0 f-f'_0],\quad M_{N-1}+2M_N=\frac{6}{h_{N-1}}[f'_N-\Delta_{N-1}f];$
2. $M_0=f''_0$ and $M_N=f''_N;$
3. $f_{N+i}=f_i,\ M_{N+i}=M_i,\ h_{N+i}=h_i,$ for all i;
4. $\frac{M_{i+1}-M_i}{h_i}=\frac{M_i-M_{i-1}}{h_{i-1}},\ \ i=1,N-1.$

Let us consider the resulting systems of linear equations for calculating the unknowns $M_i,\, i=0,\ldots,N,$ in more detail.

1. For first derivative endpoint conditions, one obtains the following system

$$\begin{bmatrix} 2h_0 & h_0 & 0 & . & . & . & 0 \\ h_0 & 2(h_0+h_1) & h_1 & . & . & . & 0 \\ 0 & h_1 & 2(h_1+h_2) & . & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & . & . & . & 0 & h_{N-1} & 2h_{N-1} \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ . \\ . \\ . \\ M_N \end{bmatrix} = \overline{b}, \qquad (2.6)$$

where

$$\overline{b} = [6(\Delta_0 f - f_0'), 6\delta_1 f, 6\delta_2 f, \ldots, 6(f_N' - \Delta_{N-1} f)]^T$$

and T is the transposition operator.

2. For second derivative endpoint conditions, the system differs only by its first and last equations.

$$\begin{bmatrix} 1 & 0 & 0 & . & . & . & 0 \\ h_0 & 2(h_0+h_1) & h_1 & . & . & . & 0 \\ 0 & h_1 & 2(h_1+h_2) & . & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & . & . & . & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ . \\ . \\ . \\ M_N \end{bmatrix} = \begin{bmatrix} y_0'' \\ 6\delta_1 f \\ 6\delta_2 f \\ . \\ . \\ . \\ f_N'' \end{bmatrix} \qquad (2.7)$$

3. For the periodic endpoint conditions, equation (2.5) is also valid for $i = N\, (or\ \ i=0)$, that is, one has

$$h_{N-1} M_{N-1} + 2(h_{N-1} + h_N) M_N + h_N M_{N+1} = 6\delta_N f. \qquad (2.8)$$

Because $f_{N+i} = f_i$, $M_{N+i} = M_i$, $i = 0,1$, and $h_N = h_0$ we obtain

$$\Delta_N f = \frac{f_{N+1} - f_N}{h_N} = \frac{f_1 - f_0}{h_0} = \Delta_0 f$$

and equation (2.8) takes the form

$$h_0 M_1 + h_{N-1} M_{N-1} + 2(h_{N-1} + h_0) M_N = 6(\Delta_0 f - \Delta_{N-1} f).$$

We arrive at the following system of linear equations

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & 0 & . & . & . & h_0 \\ h_1 & 2(h_1+h_2) & h_2 & . & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ h_0 & . & . & . & 0 & h_{N-1} & 2(h_{N-1}+h_0) \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ . \\ . \\ . \\ M_N \end{bmatrix} = \bar{b}, \qquad (2.9)$$

where

$$\bar{b} = [6\delta_1 f, 6\delta_2 f, \ldots, 6(\Delta_0 f - \Delta_{N-1} f)]^T.$$

4. For "not-a-knot" endpoint conditions, one obtains the following system

$$\begin{bmatrix} h_1 & -(h_0+h_1) & h_0 & . & . & . & 0 \\ h_0 & 2(h_0+h_1) & h_1 & . & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & . & . & . & h_{N-2} & 2(h_{N-2}+h_{N-1}) & h_{N-1} \\ 0 & . & . & . & h_{N-1} & -(h_{N-2}+h_{N-1}) & h_{N-2} \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ . \\ . \\ . \\ M_{N-1} \\ M_N \end{bmatrix} = \bar{b}, \qquad (2.10)$$

where

$$\bar{b} = [0, 6\delta_1 f, 6\delta_2 f, \ldots, 6\delta_{N-1} f, 0]^T.$$

Systems (2.6), (2.7), and (2.9) have tridiagonal or "almost" tridiagonal matrices. This permits us to apply particularly efficient algorithms (Gaussian elimination without pivoting) for their solution. In order to obtain a system with tridiagonal matrix in case of "not-a-knot" endpoint conditions, one needs first to eliminate the unknowns M_0 and M_N from system (2.10). If one subtract from second equation of system (2.10), multiplied by h_1 by the first equation multiplied by h_0, then the resulting equation takes the form

$$(h_0+2h_1)(h_0+h_1)M_1+(h_1^2-h_0^2)M_2 = 6h_1\delta_1 f.$$

Analogously, if one subtracts from next to the last equation of the system (2.10), multiplied by h_{N-2}, the last equation multiplied by h_{N-1}, then the resulting equation will be

$$\begin{aligned}(h_{N-2}^2-h_{N-1}^2)M_{N-2}+(2h_{N-2}+h_{N-1})(h_{N-2}+h_{N-1})M_{N-1}\\ = 6h_{N-2}\delta_{N-1}f.\end{aligned}$$

We arrive at the following system of linear equations with three-diagonal matrix

$$\begin{bmatrix} (h_0+2h_1) & h_1-h_0 & 0 & . & . & . & 0 \\ h_1 & 2(h_1+h_2) & h_2 & . & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & . & . & . & 0 & h_{N-2}-h_{N-1} & 2h_{N-2}+h_{N-1}) \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ . \\ . \\ . \\ M_{N-1} \end{bmatrix}$$

$$=\bar{b}, \qquad (2.11)$$

where

$$\bar{b}=[6\lambda_0\delta_1 f, 6\delta_2 f, \ldots, 6\mu_{N-1}\delta_{N-1}f]^T$$

and $\lambda_0 = h_1/(h_0 + h_1)$, $\mu_{N-1} = h_{N-1}/(h_{N-2} + h_{N-1})$.

2.4 Diagonally Dominant Matrices. Existence and Uniqueness of the Solution

Let us investigate whether systems (2.6), (2.7), (2.9), and (2.11) have unique solutions. Obviously, this is the case if and only if the matrices in those systems are nonsingular.

Definition 2.2. A square matrix $A = \{a_{ij}\}_{i,j=1}^{n}$ is called a matrix with a *diagonal dominance*, if the following conditions are fulfilled

$$r_i = |a_{ii}| - \sum_{j=1, j\neq i}^{n} |a_{ij}| \geq 0, \quad i=1,\ldots,n. \tag{2.12}$$

A matrix A is called a matrix with strict diagonal dominance, if the inequalities (2.12) are strict.

Theorem 2.1. (Hadamard criterion). Every matrix with strict diagonal dominance is nonsingular.

Proof: Suppose to the contrary that a matrix A has strict diagonal dominance and is singular, that is, $\det(A) = 0$ and the homogeneous system of equations $Ax = 0$ or

$$\sum_{j=1}^{n} a_{ij} x_j = 0, \quad i=1,\ldots,n,$$

has a nontrivial solution $x=(x_1,\ldots,x_n)^T$.

One can find k such that $|x_k| \geq |x_i|$, $i=1,\ldots,n$. Then it follows from kth equation that

$$|a_{kk}||x_k| \le \sum_{j=1, j\ne k}^{n} |a_{kj}||x_j| \le |x_k| \sum_{j=1, j\ne k}^{n} |a_{kj}|$$

From here

$$|a_{kk}| \le \sum_{j=1, j\ne k}^{n} |a_{kj}|$$

which contradicts the assumption of strict diagonal dominance of A. This completes the proof.

It is easy to verify that the systems (2.6), (2.7), (2.9), and (2.11) have matrices with strict diagonal dominance. In the case of first derivative endpoint conditions, one has from system (2.6)

$$\begin{aligned}
r_0 &= 2h_0 - h_0 = h_0 > 0, \\
r_i &= 2(h_{i-1} + h_i) - h_{i-1} - h_i = h_{i-1} + h_i > 0, \quad i = 1, \dots, N-1, \\
r_N &= 2h_{N-1} - h_{N-1} = h_{N-1} > 0,
\end{aligned}$$

Therefore, the matrix of this system has strict diagonal dominance.

By looking at equations (2.7) and (2.9), one can easily see that for second derivative and periodic endpoint conditions, strict diagonal dominance also occurs. For "not-a-knot" endpoint conditions, one obtains from system (2.11) strict diagonal dominance of the matrix of this system as well

$$\begin{aligned}
r_1 &= h_0 + 2h_1 - |h_1 - h_0| > 0, \\
r_i &= 2(h_{i-1} + h_i) - h_{i-1} - h_i = h_{i-1} + h_i > 0, \quad i = 2, \dots, N-2, \\
r_{N-1} &= 2h_{N-2} + h_{N-1} - |h_{N-2} - h_{N-1}| > 0.
\end{aligned}$$

Now using Theorem 1.1, one concludes that systems (2.6), (2.7), (2.9), and (2.11) have unique solutions. As a consequence, there exists a unique cubic interpolating spline S satisfying any of the considered above four types of endpoint constraints.

2.5 Gaussian Elimination for Tridiagonal Systems

Let us consider a specially effective algorithm for solving linear systems with tridiagonal matrices. The algorithm given below is a special variant of Gaussian elimination. Keeping in mind the systems for a cubic interpolating spline with endpoint conditions of types 1, 2, and 4, we consider the following system.

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \cdots & 0 & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{2.13}$$

To start the elimination, one divide the first equation of this system by the diagonal element b_1 and uses notations $p_1 = c_1/b_1$ and $q_1 = d_1/b_1$. Now suppose that we have eliminated all nonzero subdiagonal elements in the first $i-1$ rows. In this case, the matrix (2.13) is transformed to the following form

$$\begin{bmatrix} 1 & p_1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & p_2 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ 0 & \cdots & 0 & 1 & p_{i-1} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & a_i & b_i & c_i & \cdots & 0 \\ \vdots & & & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \cdots & 0 & 0 & 0 & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{i-1} \\ x_i \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{i-1} \\ d_i \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

Now in order to eliminate the subdiagonal element a_i in the *i*th row, we have to multiply the (*i*-1)st row by a_i and subtract it from the *i*th row. As a result, the *i*th row of our system takes the following form

$$(b_i - a_i p_{i-1})x_i + c_i x_{i+1} = d_i - a_i q_{i-1}.$$

To obtain the unit on the main diagonal, one must divide the *i*th row by the coefficient $b_i - a_i p_{i-1}$. As a result, in the final form of the *i*th row, one obtains the following formulae for the elements p_i and q_i:

$$p_i = \frac{c_i}{b_i - a_i p_{i-1}}, \quad i=2,\dots,n-1, \quad p_1 = \frac{c_1}{b_1},$$
$$q_i = \frac{d_i - a_i q_{i-1}}{b_i - a_i p_{i-1}}, \quad i=2,\dots,n, \quad q_1 = \frac{d_1}{b_1}, \qquad (2.14)$$

Proceeding in this way, one arrives at the system

$$\begin{bmatrix} 1 & p_1 & 0 & 0 & \dots & 0 \\ 0 & 1 & p_2 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ 0 & \dots & 0 & 1 & p_{n-2} & 0 \\ 0 & \dots & 0 & 0 & 1 & p_{n-1} \\ 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{n-2} \\ q_{n-1} \\ q_n \end{bmatrix}$$

Now using the back substitution, one can compute the unknown x_i:

$$x_n = q_n,$$
$$x_i = -p_i x_{i+1} + q_i, \quad i=n-1,\dots,1. \qquad (2.15)$$

2.6 Correctness and Stability of Gaussian Elimination

Let us consider the correctness and stability of the calculations in the above described special variant of Gaussian elimination. By correctness, we mean the possibility to perform all necessary calculations in the algorithm, that is, in our case, that the denominators in formulae (2.14) do not vanish. The algorithm of Gaussian elimination will be also stable if we do not have any progressive accumulation of round-off errors by performing arithmetic operations (in our case by multiplications in formula (2.15)).

For system (2.13) with tridiagonal matrix, the conditions of strict diagonal dominance (2.12) take the form

$$|b_i| > |a_i| + |c_i|, \quad i = 1,\ldots,n, \tag{2.16}$$

with $a_1 = c_n = 0$.

Let us show that if the conditions of strict diagonal dominance (2.16) are fulfilled, then the algorithm of Gaussian elimination (2.14) and (2.15) is correct and stable. According to (2.14) and (2.16), one has $|p_1| = |c_1| / |b_1| \le 1$. Let us suppose by induction that $|p_j| \le 1$, $j = 1,\ldots,i-1$. Then using formula (2.14), one obtains

$$|p_i| = \frac{|c_i|}{|b_i - a_i p_{i-1}|} \le \frac{|c_i|}{|b_i| - |a_i||p_{i-1}|} \le \frac{|c_i|}{|b_i| - |a_i|} \le \frac{|c_i|}{|c_i|} = 1,$$

that is, $|p_i| \le 1$ for all i.

As

$$|b_i - a_i p_{i-1}| \ge |b_i| - |a_i||p_{i-1}| > |b_i| - |a_i| > 0, \qquad i = 2,\ldots,n-1,$$

then the denominators in formulae (2.14) are nonzero. This means the correctness of Gaussian elimination.

Suppose that while practically solving system (2.13) by applying formulae (2.14) and (2.15), one obtains $\overline{x}_i = x_i + \varepsilon_i$, $i = 1, \ldots, n$, where ε_i is a round-off error at the ith step. Then according to (2.15), one obtains

$$\overline{x}_i = -p_i \overline{x}_{i+1} + q_i, \quad i = n-1, \ldots, 1.$$

By subtracting formula (2.15) from this equation, one gets

$$\varepsilon_i = -p_i \varepsilon_{i+1}, \quad i = n-1, \ldots, 1,$$

or

$$|\varepsilon_i| \le |p_i| |\varepsilon_{i+1}| \le |\varepsilon_n|, \quad i = n-1, \ldots, 1,$$

that is, the calculations by formula (2.15) are stable.

It was shown above that for a cubic interpolating spline, the matrices of the systems (2.6), (2.7), (2.9) and (2.11) for all considered four types of endpoint conditions have strict diagonal dominance. Therefore, the systems (2.6), (2.7) and (2.11) can be stably solved by the algorithm of Gaussian elimination without pivoting (2.14) and (2.15). To solve system (2.9) one must use a slightly more complicated algorithm which is , however, another modification of Gaussian elimination.

2.7 Method of First Derivative

In some cases, it is more convenient to use a different algorithm for constructing cubic interpolating spline. Such an algorithm is based on the representation of the spline through endpoint values of its first derivative.

Let us denote $m_i = S'(x_i)$, $i = 0, \ldots, N$. On the interval $[x_i, x_{i+1}]$, one can write down the following formula for the cubic interpolating spline

$$S(x) \equiv S_i(x) = f_i(1-t) + f_{i+1}t + t(1-t)h_i[(m_i - \Delta_i f)(1-t) + (\Delta_i f - m_{i+1})t],$$
$$\Delta_i f = \frac{f_{i+1} - f_i}{h_i}, \qquad (2.17)$$

where $t = (x - x_i)/h_i$ and $h_i = x_{i+1} - x_i$.

It is easy to verify that $S_i(x_j) = f_j$, $S_i'(x_j) = m_j$, $j = i, i+1$. By differentiating the formula (2.17) two times with respect to x, one obtains

$$S_i''(x) = \frac{2}{h_i}[3(1-2t)\Delta_i f - (2-3t)m_i - (1-3t)m_{i+1}]. \qquad (2.18)$$

Using this expression for adjacent polynomials on the intervals $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$ in the joint point $x = x_i$, one finds

$$S_{i-1}''(x_i - 0) = \frac{2}{h_{i-1}}(-3\Delta_{i-1}f + m_{i-1} + 2m_i),$$
$$S_i''(x_i + 0) = \frac{2}{h_i}(3\Delta_i f - 2m_i - m_{i+1}).$$

From the condition of continuity $S_{i-1}''(x_i - 0) = S_i''(x_i + 0)$, $i = 1, \ldots, N-1$, one obtains the system of linear equations

$$\frac{1}{h_{i-1}}m_{i-1} + 2\left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right)m_i + \frac{1}{h_i}m_{i+1} = 3\left(\frac{\Delta_{i-1}f}{h_{i-1}} + \frac{\Delta_i f}{h_i}\right),$$
$$i = 1, \ldots, N-1, \qquad (2.19)$$

In order to complete this system of equations, one needs two additional restrictions which are usually given as endpoint conditions of the types considered in section 1.1. Using formulae (2.17) and (2.18), one can rewrite these endpoint conditions in the form:

1. First derivative endpoint conditions:

$$m_0 = f_0' \text{ and } m_N = f_N'; \tag{2.20}$$

2. Second derivative endpoint conditions:

$$\begin{aligned} 2m_0 + m_1 &= 3\Delta_0 f - f_0'' h_0/2, \\ m_{N-1} + 2m_N &= f_N'' h_{N-1}/2 + 3\Delta_{N-1} f; \end{aligned} \tag{2.21}$$

3. Periodic endpoint conditions:

$$f_{N+i} = f_i, \ m_{N+i} = m_i, \ h_{N+i} = h_i \text{ for all } i; \tag{2.22}$$

4. "Not-a-knot" endpoint conditions where adjacent polynomials nearest to the endpoints of the interval $[a,b]$ coincide: $S_0(x) \equiv S_1(x)$ and $S_{N-1}(x) \equiv S_N(x)$, that is,

$$S'''(x_i - 0) = S'''(x_i + 0), \quad i = 1, N-1,$$

or

$$\frac{m_i + m_{i+1}}{h_i^2} - \frac{m_{i-1} + m_i}{h_{i-1}^2} = \frac{\Delta_i f}{h_i^2} - \frac{\Delta_{i-1} f}{h_{i-1}^2}, \quad i = 1, N-1, \tag{2.23}$$

In the case of endpoint conditions of types 1 and 2, equations (2.20) and (2.21) permit one to directly complete the system (2.19). In the case of periodic endpoint conditions, one assumes that equation (2.19) is also valid for $i = N$. As according to the conditions (2.22) $y_{N+i} = y_i$, $m_{N+i} = m_i$ $i = 0,1$, and $h_N = h_0$, the system (2.19) is completed by the equation

$$\frac{1}{h_0}m_1 + \frac{1}{h_{N-1}}m_{N-1} + 2(\frac{1}{h_0} + \frac{1}{h_{N-1}})m_N = 3(\frac{\Delta_0 f}{h_0} + \frac{\Delta_{N-1} f}{h_{N-1}}).$$

One can easily verify that for the first three types of endpoint conditions, the corresponding systems of linear equations have matrices with strict diagonal dominance and therefore there exists a unique related cubic interpolating spline. In the case of "not-a-knot" endpoint conditions, in order to obtain a system whose matrix has strict diagonal dominance, one needs first to eliminate the unknowns m_0 and m_N from the system (2.19). As a result, taking into account the relations (2.23), the first and last equations of this system reduce to the form

$$(\frac{1}{h_0} + \frac{1}{h_1})m_1 + \frac{1}{h_1}m_2 = 2\lambda_1\frac{\Delta_0 f}{h_0} + (1+2\lambda_1)\frac{\Delta_1 f}{h_1},$$

$$\frac{m_{N-2}}{h_{N-2}} + (\frac{1}{h_{N-2}} + \frac{1}{h_{N-1}})m_{N-1} = (1+2\mu_{N-1})\frac{\Delta_{N-2} f}{h_{N-2}} + 2\mu_{N-1}\frac{\Delta_{N-1} f}{h_{N-1}},$$

where $\lambda_1 = h_1/(h_0 + h_1)$ and $\mu_{N-1} = h_{N-2}/(h_{N-2} + h_{N-1})$.

Thus, for all four types of endpoint conditions, the matrices of the corresponding linear systems in the unknown $m_i, i = 0,\ldots,N$, have strict diagonal dominance. This permits us to solve these systems efficiently by means of the above described algorithms of Gaussian elimination without pivoting.

In order to reduce the number of arithmetic operations performed in a practical evaluation of the spline and its derivatives on the interval $[x_i, x_{i+1}]$, one can rewrite formula (2.17) in the form

$$S_i(x) = f_i + (x - x_i)(\Delta_i f + (x - x_{i+1})(a_i + (x - x_i)b_i));$$

where

$$a_i = \frac{\Delta_i f - m_i}{h_i} \text{ and } b_i = \frac{m_i + 2\Delta_i f + m_{i+1}}{h_i^2}.$$

Chapter III

Convex and Monotone Spline Interpolation

In this chapter, we discuss the convex and monotone interpolation by splines $S \in C^2$. In practice, the C^2 cubic splines are very often used. We give sufficient conditions of monotonicity (convexity) for these splines, provided the interpolated data is monotone (convex). The conditions are formulated in terms of data divided differences and they are very easy for testing. The method that was used to deduce these conditions is based on the simple lemma about the tridiagonal system. It may be applied to any spline if the construction of the spline is reduced to solution of tridiagonal system with diagonal dominance.

3.1 Problem of Convex and Monotone Interpolation

Let us consider Lagrange interpolation for points $a = x_0 < x_1 < \cdots < x_N = b$,

$$S(x_i) = f_i \quad (= f(x_i)) \tag{3.1}$$

by spline functions S which preserve the shape of the data (x_i, f_i), $i = 0, \ldots, N$. For example, if the function f is monotone or convex on some interval $[x_j, x_k]$, we would like to have a spline S which also has these properties. To achieve this, we take splines S with knots at the points x_i, which have more parameters than necessary to satisfy (3.1). The additional parameters are then selected to ensure the desired properties of S. The main point, however, is to determine whether the error of approximation $\|f - S\|$ remain small under the proposed algorithms which describe S.

We introduce the notation for the first two divided differences

$$\Delta_i f = (f_{i+1} - f_i)/h_i,\ h_i = x_{i+1} - x_i,\ i = 0,\dots,N-1,$$
$$\delta_i f = \Delta_i f - \Delta_{i-1} f,\ i = 1,\dots,N-1.$$

The data $\{f_i\}$ are said to be *monotone* if

$$\Delta_i f \geq 0,\ i = 0,\dots,N-1, \tag{3.2}$$

and *convex* if

$$\delta_i f \geq 0,\ i = 1,\dots,N-1, \tag{3.3}$$

The problem of monotone (convex) spline interpolation consists of constructing a monotone (convex) spline interpolant to monotone (convex) data.

A cubic spline $S \in C^1[a,b]$ interpolant to $\{f_i\}$ can be written, for $x \in [x_i, x_{i+1}]$, in the form (see (2.17) in chapter 2)

$$\begin{aligned} S(x) \equiv S_i(x) = (1-t)^2(1+2t)f_i + t^2(3-2t)f_{i+1} \\ + h_i t(1-t)^2 m_i - h_i t^2(1-t)m_{i+1}, \end{aligned} \tag{3.4}$$

where $t = (x - x_i)/h_i,\ m_j = S'(x_j),\ j = i, i+1.$

Fritsch and Carlson (1980) have proved the following

Lemma 3.1. If $\Delta_i f \geq 0$ and

$$0 \leq m_j \leq 3\Delta_i f,\ \ j = i, i+1. \tag{3.5}$$

then $S_i'(x) \geq 0$ for $x \in [x_i, x_{i+1}]$.

Proof: This result can easily be obtained without appealing to Fritsch and Carlson

(1980). In fact, from (3.4), we have

$$S_i'(x) = s(t, m_i, m_{i+1}) = 6t(1-t)\Delta_i f$$
$$+(1-4t+3t^2)m_i + (-2t+3t^2)m_{i+1}. \tag{3.6}$$

The function $s(t, m_i, m_{i+1})$ is linear in variables m_i and m_{i+1}. Therefore, for proving the inequality $S_i'(x) \geq 0$, $x \in [x_i, x_{i+1}]$, under restrictions (3.5), it is sufficient to verify the inequalities: $s(t,0,0) \geq 0$, $s(t, 3\Delta_i f, 0) \geq 0$, $s(t, 0, 3\Delta_i f) \geq 0$, $s(t, 3\Delta_i f, 3\Delta_i f) \geq 0$ for $t \in [0,1]$. It is easy to convince oneself that these inequalities hold; hence, the desired result follows. This proves the lemma.

3.2 Monotone Matrix. Lemma on Tridiagonal Systems

Definition 3.1. A square matrix A is called monotone if $Ay \geq 0$ implies $y \geq 0$ and $Ay \leq 0$ implies $y \leq 0$. By $y \geq 0$ $(y \leq 0)$, we mean that all components of a vector y are nonnegative (nonpositive).

Let A be an invertible square matrix. Then A is monotone if and only if all elements of A^{-1} are nonnegative (see Collatz (1964)).

Lemma 3.2. If a square matrix $A = \{a_{ij}\}_{i,j=1}^{n}$ has a diagonal dominance and

$$a_{ii} > 0, \; a_{ij} < 0 \quad (j \neq i), \; i, j = 1, \ldots, n,$$

then it is monotone.

Proof: Show that $Ay \geq 0$ implies $y \geq 0$. Suppose to the contrary that a vector y has negative components and let y_k denotes the negative component of largest absolute value. We take a vector z whose components all equal to $|y_k|$. As

$$\sum a_{ij} z_j = |y_k| \sum_j a_{ij} > 0, \; i = 1, \dots, n,$$

then $Az > 0$ and thus $A(z + y) = Az + Ay > 0.$ However

$$\sum_j a_{jk}(y_j + z_j) = \sum_{j \neq k} a_{kj}(y_j + |y_k|) \leq 0.$$

The obtained contradiction proves the lemma.

Consider the system

$$\begin{aligned} b_0 z_0 + c_0 z_1 &= d_0, \\ a_i z_{i-1} + b_i z_i + c_i z_{i+1} &= d_i, \;\; i = 1, 2, \dots, N-1, \\ a_N z_{N-1} + b_N z_N &= d_N. \end{aligned} \tag{3.7}$$

Let the system be solvable and its righthand members be positive. By Lemma 3.2 if the matrix of system (3.7) is monotone then $z_i \geq 0, i = 0, \dots, N$. But the matrices that occur by the construction of splines are usually not monotone. In this case, we can apply the following.

Lemma 3.3. Let the coefficients of system (3.7) be such that

$$b_i > 0, \; i = 0, \dots, N; \; a_i \geq 0, \; c_i \geq 0, \; b_i > a_i + c_i, \; i = 1, \dots, N-1, \tag{3.8}$$

$$c_0 < b_0 b_1 / (a_1 + c_1), \;\; a_N < b_{N-1} b_N / (a_{N-1} + c_{N-1}). \tag{3.9}$$

If

$$d_i \geq 0, \; d_i - c_i d_{i+1} / b_{i+1} - a_i d_{i-1} / b_{i-1} \geq 0, \;\; i = 0, \dots, N \tag{3.10}$$

(here $a_0 = c_N = d_{-1} = d_{N+1} = 0, \;\; b_{-1} = b_{N+1} = 1$), then system (3.7) is solvable and

$$z_i \geq 0, \; i = 0,\dots,N. \tag{3.11}$$

Proof. First, we consider the case $c_0 \geq 0$ and $a_N \geq 0$. We add to (3.7) equations $b_i z_i = d_i$, with $b_i = d_i = 1$, $i = -1, N+1$, and suppose $a_0 = a_{N+1} = c_{-1} = c_{N+1} = 0$. For each $i = 0,1,2,\dots,N$, we take the linear combination of the (i-1)th, ith and (i+1)th equations of this system with corresponding coefficients $-a_i / b_{i-1}$, 1, $-c_i / b_{i+1}$. Then we have

$$\begin{aligned} B_0 z_0 - C_0 z_2 &= D_0, \\ -A_i z_{i-2} + B_i z_i - C_i z_{i+2} &= D_i, \quad i = 1,2,\dots,N-1. \\ -A_N z_{N-2} + B_N z_N &= D_N, \end{aligned} \tag{3.12}$$

Where $A_i = a_{i-1} a_i / b_{i-1}, B_i = b_i - a_i c_{i-1} / b_{i-1} - a_{i+1} c_i / b_{i+1}$,
$C_i = c_i c_{i+1} / b_{i+1}, D_i = d_i - a_i d_{i-1} / b_{i-1} - c_i d_{i+1} / b_{i+1}$,

and evidently $A_1 = C_{N-1} = 0$. Thus, (3.12) is the system with unknowns $z_0,\dots,z_N$. By using (3.8), we have $A_i \geq 0$, $C_i \geq 0$, $i = 0,\dots,N$, and $B_i \geq b_i - a_i - c_i > 0$, $i = 2, \dots, N-2$. Next, by using (3.9), we obtain

$$B_1 > b_1 - b_1 a_1 / (a_1 + c_1) - c_1 a_2 / b_2 = c_1 [b_1 / (a_1 + c_1) - a_2 / b_2] > 0,$$
$$B_0 = b_0 - a_1 c_0 / b_1 > b_0 - a_1 b_0 / (a_1 + c_1) > 0,$$

and $B_{N-1} > 0$, $B_N > 0$. Thus, all B_i in (3.12) are positive. Further, it is easy to show that system (3.12) has a matrix with diagonal dominance. As a result, this matrix is monotone and nonsingular. Therefore, (3.11) is hold.

Now, let $c_0 < 0, a_N < 0$ (the proof in the cases when $c_0 \geq 0, a_N < 0$ or $c_0 < 0$, $a_N \geq 0$ is similar). Eliminating z_0 and z_N from (3.7), we obtain the system

$$\begin{aligned} &\hat{b}_1 z_1 + c_1 z_2 = \hat{d}_1, \\ &a_i z_{i-1} + b_i z_i + c_i z_{i+1} = d_i, \quad i = 2,\dots,N-2 \\ &a_{N-1} z_{N-2} + \hat{b}_{N-1} z_{N-1} = \hat{d}_{N-1}, \end{aligned} \tag{3.13}$$

where $\hat{b}_1 = b_1 - a_1 c_0 / b_0, \hat{d}_1 = d_1 - a_1 d_0 / b_0,$

$\hat{b}_{N-1} = b_{N-1} - c_{N-1} a_N / b_N \ , \hat{d}_{N-1} = d_{N-1} - c_{N-1} d_N / b_N .$

It is easy to see that system (3.13) satisfies all of the hypotheses which we have used above in the study of the case $c_0 \ge 0, a_N \ge 0$. Thus, $z_i \ge 0, \quad i = 1,\dots, N-1$. But $z_0 = (d_0 - c_0 z_1)/b_1 > 0$ and, similarly, $z_N \ge 0$. This proof the lemma.

3.3 Convex cubic splines.

Let S be a C^2 cubic interpolating spline with endpoint conditions

$$S'(x_i) = f_i', \quad i = 0, N. \tag{3.14}$$

It is shown in chapter 1 that values of the second derivative of the spline $M_i = S''(x_i)$, $i = 0,\dots,N$, satisfy the system of linear equations (2.6) which can be rewritten in the form

$$\begin{aligned} &2M_0 + M_1 = d_0, \\ &\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = d_i, \quad i = 0,\dots,N-1, \\ &M_{N-1} + 2M_N = d_N, \end{aligned} \tag{3.15}$$

where

$$\mu_i = h_{i-1}/(h_{i-1} + h_i), \ \lambda_i = 1 - \mu_i, \ d_i = 6f[x_{i-1}, x_i, x_{i+1}],$$

$$d_0 = \frac{6}{h_0}(\Delta_0 f + f_0'), \ d_N = \frac{6}{h_{N-1}}(f_N' - \Delta_{N-1} f), \ \Delta_j f = \frac{f_{j+1} - f_j}{h_j}.$$

Theorem 3.1 Let a C^2 cubic splines S with endpoint conditions (3.14) interpolate the convex data $\{f_i\}$, $i = 0,\dots,N$. If the righthand elements of system (3.15) satisfy the following inequalities

$$d_0 \geq 0\,,\ d_N \geq 0,\ 2d_i - \lambda_i d_{i+1} - \mu_i d_{i-1} \geq 0,\quad i = 0,1,\dots,N\,, \qquad (3.16)$$

where $d_{-1} = d_{N+1} = 0$ and $\lambda_0 = \mu_N = 1$, then $S''(x) \geq 0$ for all $x \in [a,b]$, that is, S is convex on $[a,b]$.

Proof. It is easy to verify directly that system (3.15) satisfies the conditions of Lemma 3.3. Therefore, by the constraints (3.16) its solution is nonnegative: $M_i \geq 0$, $i = 0,\dots,N$. On each interval $[x_i, x_{i+1}]$, $i = 0,\dots,N-1$, we have

$$S''(x) = (1-t)M_i + tM_{i+1},\ t = (x - x_i)/h_i$$

and thus $S''(x) \geq 0$ for all $x \in [a,b]$. This proves the theorem.

Remark. Define $\overline{h} = \max\limits_i h_i$. If a C^2 cubic spline S interpolates a function $f \in C^2$ $[a,b]$ and $f''(x) > 0$ for all $x \in [a,b]$, then conditions (3.16) will be fulfilled provided that $\overline{h}$ is sufficiently small.

3.4 Monotone cubic splines.

Let a C^2 cubic spline S interpolate the monotone data $\{f_i\}$, $i = 0,1,\dots,N$, and satisfy the endpoint conditions

$$S''(x_i) = f_i'';\quad i = 0, N. \qquad (3.17)$$

For values of the first derivative of the spline $m_i = S'(x_i)$, $i = 0,\dots,N$, one has the following system of linear equations (see (2.19) and (2.21))

$$\begin{aligned} 2m_0 + m_1 &= c_0 , \\ \lambda_i m_{i-1} + 2m_i + \mu_i m_{i+1} &= c_i , \quad i = 1,\dots,N-1, \\ m_{N-1} + 2m_N &= c_N , \end{aligned} \tag{3.18}$$

where

$$c_i = 3\lambda_i \Delta_{i-1} f + 3\mu_i \Delta_i f, \ c_0 = 3\Delta_0 f - f_0'' h_0/2, \ c_N = 3\Delta_{N-1} f + f_N'' h_{N-1}/2.$$

Lemma 3.4. If the righthand element of system (3.18) satisfy the following inequalities

$$c_0 \ge 0 \ , c_N \ge 0 , \ 2c_i - \lambda_i c_{i-1} - \mu_i c_{i+1} \ge 0, \ i = 0,1,\dots,N , \tag{3.19}$$

where $c_{-1} = c_{N+1} = 0$ and $\mu_0 = \lambda_N = 1$, then $m_i \ge 0, \ i = 0,1,\dots,N$.

Prove: It is easy to verify that system (3.18) satisfies the conditions of lemma 3.3. Therefore, by the restrictions (3.19), its solution is nonnegative: $m_i \ge 0, \ i = 0,1,\dots,N$. This proves the lemma.

From the fact that $m_i \ge 0, i = 0,1,\dots,N,$ it does not necessarily follow that $S'(x) \ge 0$ for all $x \in [a,b]$. To obtain this assertion, one needs a stronger assumption than in Lemma 3.4.

Theorem 3.2. Let a C^2 cubic spline S with endpoint conditions (3.17) interpolate the monotone data $\{f_i\}, \ i = 0,1,\dots,N$. If the following inequalities are valid

$$c_1 \le 2c_0 \le 12\Delta_0 f, \tag{3.20}$$

$$c_{N-1} \le 2c_N \le 12\Delta_{N-1} f, \tag{3.21}$$

$$\lambda_i \Delta_{i-1} f \le (1+\lambda_i)\Delta_i f, \ i = 1,\dots,N-1, \tag{3.22}$$

$$\mu_i \Delta_i f \le (1+\mu_i)\Delta_{i-1} f, \ i = 1,\dots,N-1 \tag{3.23}$$

then $S'(x) \geq 0$ for all $x \in [a,b]$, that is, S is monotone on $[a,b]$.

Proof. It is easy to check that the hypotheses of Lemma 3.4 follow from conditions (3.20)-(3.23). Thus, $m_i \geq 0,\ i=0,1,\dots,N$. From (3.18), we conclude that

$$\begin{aligned}
&m_0 \leq c_0 / 2, \\
&m_i \leq c_i / 2 = 3(\lambda_i \Delta_{i-1} f + \mu_i \Delta_i f)/2, \quad i = 1,\dots,N-1, \\
&m_N \leq c_N / 2.
\end{aligned}$$

Taking into account (3.20)-(3.23), we obtain

$$\begin{aligned}
&0 \leq m_0 \leq 3\Delta_0 f,. \\
&0 \leq m_i \leq 3\Delta_i f, \quad j = i-1, i; \ \ i = 1,\dots,N-1, \\
&0 \leq m_N \leq 3\Delta_{N-1} f.
\end{aligned}$$

Now it follows from Lemma 3.1 that $S'(x) \geq 0$ for all $x \in [a,b]$, that is, the spline S is monotone on $[a,b]$. This proves the theorem.

3.5 Tension Generalized Splines. Conditions of Existence and Uniqueness

If the cubic spline does not preserve monotonicity or convexity of the data, we will use generalized tension splines (see *Späth* (1990), Zavyalov et al.(1980)). These splines include, as special cases, the C^2 cubic spline, various types of rational splines, the exponential spline, the cubic spline with additional nodes, etc. We give explicit formulae for the parameters of generalized splines in order to secure the preserving of monotonicity and convexity of the data.

Let us associate with a partition $\Delta : a = x_0 < x_1 < \cdots < x_N = b$ of the interval $[a,b]$, a apace of functions S_4^G whose restriction to a subinterval $[x_i, x_{i+1}]$, $i = 0,\dots,N-1$, is spanned by the system of four linearly independent functions

$\{1,x,\Phi_i,\Psi_i\}$, and where every function in S_4^G is continuous and has two continuous derivatives.

Definition 3.1. An interpolating tension generalized spline is a function $S\in S_4^G$ such that

(i) for any $x\in[x_i,x_{i+1}]$, $i=0,\dots,N-1$,

$$S(x)=[f_i-\Phi_i(x_i)M_i](1-t)+[f_{i+1}-\Psi_i(x_{i+1})M_{i+1}]t$$
$$+\Phi_i(x)M_i+\Psi_i(x)M_{i+1}, \tag{3.24}$$

where $t=(x-x_i)/h_i$, $M_j=S''(x_j)$, $j=i,i+1$, and the function Φ_i and Ψ_i are subject to the constraints

$$\Phi_i^{(r)}(x_{i+1})=\Psi_i^{(r)}(x_i)=0,\ r=0,1,2;\qquad \Phi_i''(x_i)=\Psi_i''(x_{i+1})=1.$$

(ii) $S\in C^2[a,b]$.

The functions Φ_i and Ψ_i depend on the tension parameters which influence the behaviour of S fundamentally. We call them the *defining functions*. In practice, one takes

$$\Phi_i(x)=\varphi_i(t)h_i^2=\psi_i(p_i,1-t)h_i^2,$$
$$\Psi_i(x)=\psi_i(t)h_i^2=\psi(q_i,t)h_i^2,\ 0\le p_{i,}q_i<\infty. \tag{3.25}$$

In the limiting case when $p_{i,}q_i\to\infty$, we require that $\lim_{p_i\to\infty}\Phi_i(p_i,x)=0$, $x\in(x_i,x_{i+1}]$, and $\lim_{q_i\to\infty}\Psi_i(q_i,x)=0$, $x\in[x_i,x_{i+1})$, so that the function S in formula (3.24) turns into a linear function. Additionally, we require that if $p_i=q_i=0$, for all i, we get a conventional cubic spline with $\varphi_i(t)=(1-t)^3/6$ and $\psi_i(t)=t^3/6$.

According to (3.24), we have

$$S_i'(x_i) = m_i = \Delta_i f - \overline{a}_i \frac{M_i}{h_i} - \Psi_i(x_{i+1}) \frac{M_{i+1}}{h_i},$$

$$S_{i+1}'(x_{i+1}) = m_{i+1} = \Delta_i f + \Phi_i(x_i) \frac{M_i}{h_i} + \overline{b}_i \frac{M_{i+1}}{h_i}, \tag{3.26}$$

where

$$\overline{a}_i = -\Phi_i(x_i) - h_i \Phi_i'(x_i),$$

$$\overline{b}_i = -\Psi_i(x_{i+1}) + h_i \Psi_i'(x_{i+1}). \tag{3.27}$$

The continuity condition for S' on Δ and the boundary relation $S'(a) = f_0'$ and $S'(b) = f_N'$, result in the following system of linear algebraic equations

$$\overline{a}_0 \frac{M_0}{h_0} + \Psi_0(x_1) \frac{M_1}{h_0} = \Delta_0 f - f_0',$$

$$\Phi_{i-1}(x_{i-1}) \frac{M_{i-1}}{h_{i-1}} + \left(\frac{\overline{b}_{i-1}}{h_{i-1}} + \frac{\overline{a}_i}{h_i}\right) M_i + \Psi_i(x_{i+1}) \frac{M_{i+1}}{h_i} = \delta_i f,$$

$$i = 1, \ldots, N-1, \tag{3.28}$$

$$\Phi_{N-1}(x_{N-1}) \frac{M_{N-1}}{h_{N-1}} + \overline{b}_{N-1} \frac{M_N}{h_{N-1}} = f_N' - \Delta_{N-1} f.$$

Let us find constraints on the defining functions Φ_i and Ψ_i which ensure that the interpolating tension generalized spline S exists and is unique.

Lemma 3.5. If the conditions

$$0 < \Phi_i(x_i) < \overline{b}_i, \; 0 < \Psi_i(x_{i+1}) < \overline{a}_i, \; i = 0, \ldots, N-1, \tag{3.29}$$

are satisfied, where $\overline{a}_i$ and $\overline{b}_i$ are as defined in (3.27), then the interpolating tension generalized spline S exists and is unique.

Proof: By virtue of conditions (3.29), the matrix of the system (3.28) is diagonally dominant:

$$r_0 = \frac{1}{h_0}[\bar{a}_0 - \Psi_0(x_1)] > 0,$$

$$r_i = \frac{1}{h_{i-1}}[\bar{b}_{i-1} - \Phi_{i-1}(x_{i-1})] + \frac{1}{h_i}[\bar{a}_i - \Psi_i(x_{i+1})] > 0, \qquad i = 1, \ldots, N-1,$$

$$r_N = \frac{1}{h_{N-1}}[\bar{b}_{N-1} - \Phi_{N-1}(x_{N-1})] > 0.$$

Thus, the Hadamard criterion (see Theorem 2.1) implies that the matrix of system (3.28) is nonsingular and that the interpolating tension generalized spline S exists and is unique. This proves the lemma.

The conditions of Lemma 3.5 can be weakened. From (3.26), we have

$$\begin{aligned} M_i &= \frac{h_i}{T_i}\{[\bar{b}_i + \Psi_i(x_{i+1})]\Delta_i f - \bar{b}_i m_i - \Psi_i(x_{i+1}) m_{i+1}\}, \\ M_{i+1} &= \frac{h_i}{T_i}\{-[\bar{a}_i + \Phi_i(x_i)]\Delta_i f + \Phi_i(x_i) m_i + \bar{a}_i m_{i+1}\}, \\ T_i &= \bar{a}_i \bar{b}_i - \Phi_i(x_i)\Psi_i(x_{i+1}). \end{aligned} \tag{3.30}$$

The continuity condition for S'' on Δ and the endpoint relations $S''(a) = f_0''$ and $S''(b) = f_N''$, result in the following system of equations

$$\bar{b}_0 m_0 + \Psi_0(x_1) m_1 = [\bar{b}_0 + \Psi_0(x_1)]\Delta_0 f - \frac{T_0}{h_0} f_0'';$$

$$\Phi_{i-1}(x_{i-1})\frac{h_{i-1}}{T_{i-1}} m_{i-1} + (\bar{a}_{i-1}\frac{h_{i-1}}{T_{i-1}} + \bar{b}_i \frac{h_i}{T_i}) m_i + \Psi_i(x_{i+1})\frac{h_i}{T_i} m_{i+1}$$

$$= [\bar{a}_{i-1} + \Phi_{i-1}(x_{i-1})]\frac{h_{i-1}}{T_{i-1}}\Delta_{i-1} f + [\bar{b}_i + \Psi_i(x_{i+1})]\frac{h_i}{T_i}\Delta_i f,$$

$$i = 1, \ldots, N-1, \tag{3.31}$$

$$\Phi_{N-1}(x_{N-1})m_{N-1} + \bar{a}_{N-1}m_N = \frac{T_{N-1}}{h_{N-1}} f_N'' + [\bar{a}_{N-1} + \Phi_{N-1}(x_{N-1})]\Delta_{N-1} f.$$

Lemma 3.6. If the conditions

$$0 < \Phi_i(x_i) < \bar{a}_i, \;\; 0 < \Psi_i(x_{i+1}) < \bar{b}_i, \;\; i = 0, \dots, N-1, \tag{3.32}$$

are satisfied, where $\bar{a}_i$ and $\bar{b}_i$ are as defined in (3.27), then the interpolating tension generalized spline S exists and is unique.

Proof: It follows from the conditions of the lemma that if

$$0 < \Phi_i(x_i)\Psi_i(x_{i+1}) < \bar{a}_i\bar{b}_i, \;\; i = 0, \dots, N-1,$$

then

$$T_i = \bar{a}_i\bar{b}_i - \Phi_i(x_i)\Psi_i(x_{i+1}), \;\; i = 0, \dots, N-1.$$

Therefore, by virtue of the conditions of the lemma, the matrix of system (3.31) is diagonally dominant:

$$\bar{r}_0 = \bar{b}_0 - \Psi_0(x_1) > 0,$$

$$\bar{r}_i = [\bar{a}_{i-1} - \Phi_{i-1}(x_{i-1})]\frac{h_{i-1}}{T_{i-1}} + [\bar{b}_i - \Psi_i(x_{i+1})]\frac{h_i}{T_i} > 0, \;\; i = 1, \dots, N-1,$$

$$\bar{r}_N = [\bar{a}_{N-1} - \Phi_{N-1}(x_{N-1})] > 0.$$

This ensure (see Theorem 2.1) that the spline S exists and is unique, and proves the lemma.

In practical examples the conditions of Lemma 3.6 are less restrictive than those formulated in Lemma 3.5. They are satisfied for the majority of tension generalized splines used in practice. One can readily verify that these conditions are fulfilled by all of the defining functions presented below in section 3.7. This allows one to construct the splines, that is, to solve the tridiagonal linear system (3.31), efficiently by a special version of Gaussian elimination that avoids pivoting (see chapter 2).

3.6 Convex Interpolation by Tension Generalized Splines

Let a tension generalized spline S given by (3.24) interpolate the convex data $\{f_i\}$, $i = 0,\ldots,N$, and satisfy the endpoint conditions $S''(x_i) = f_i''$, $i = 0, N$, where $f_0'' \geq 0$ and $f_N'' \geq 0$. The system (3.28) for values of the second derivative of the spline $M_i = S''(x_i)$, $i = 0,\ldots,N$, can be rewritten in the form

$$\begin{aligned} M_0 &= f_0'', \\ a_i M_{i-1} + b_i M_i + c_i M_{i+1} &= d_i, \quad i = 1,\ldots,N-1. \\ M_N &= f_N'', \end{aligned} \tag{3.33}$$

where

$$a_i = \frac{\Phi_{i-1}(x_{i-1})}{h_{i-1}}, \quad b_i = \frac{\bar{b}_{i-1}}{h_{i-1}} + \frac{\bar{a}_i}{h_i}, \quad c_i = \frac{\Psi_i(x_{i+1})}{h_i}, \quad d_i = \delta_i f.$$

Theorem 3.3. Let a tension generalized spline S interpolate the convex data $\{f_i\}$, $i = 0,\ldots,N$, and satisfy the endpoint conditions $S''(x_i) = f_i''$, $i = 0, N$. Suppose that, the defining functions Φ_i and Ψ_i in (3.24) given by (3.25) are convex on $[x_i, x_{i+1}]$, $i = 0,\ldots,N-1$, that is, $\Phi_i''(x) \geq 0$ and $\Psi_i''(x) \geq 0$ for all $x \in [x_i, x_{i+1}]$, and comply with constraints (3.29). If the righthand elements of system (3.33) satisfy the following inequalities

$$f_0'' \geq 0, \; f_N'' \geq 0, \; d_i + a_i d_{i-1}/b_{i-1} - c_i d_{i+1}/b_{i+1} \geq 0, \qquad i = 1,\ldots,N-1. \tag{3.34}$$

Where $b_0 = b_N = 1$, $d_0 = f_0''$, $d_N = f_N''$, then $S''(x) \geq 0$ for all $x \in [a,b]$, that is S is convex on $[a,b]$.

Proof: It is easy to verify that under constraints (3.29), the systems (3.33) satisfies the conditions of Lemma 3.3. Therefore, by the restrictions (3.34), its solution is nonnegative: $M_i \geq 0$, $i = 0,\ldots,N$. Using formula (3.24) on each interval $[x_i, x_{i+1}]$, $i = 0,\ldots,N-1$, we have

$$S''(x) = \Phi_i''(x) M_i + \Psi_i''(x) M_{i+1},$$

where by assumption $\Phi_i''(x) \geq 0$ and $\Psi_i''(x) \geq 0$. Thus $S''(x) \geq 0$ for all $x \in [a,b]$. This proves the theorem.

3.7. Monotone Interpolation by Tension Generalized Splines

Let a tension generalized spline S interpolate the monotone data $\{f_i\}$, $i = 0,\ldots,N$, and satisfy the endpoint conditions $S'(x_i) = f_i'$, $i = 0, N$, with $f_0' \geq 0$ and $f_N' \geq 0$. The system (3.31) for values of the first derivative of the spline $m_i = S'(x_i)$, $i = 0,\ldots,N$, can be rewritten in the form

$$\begin{gathered} m_0 = f_0', \\ \tilde{a}_i m_{i-1} + \tilde{b}_i m_i + \tilde{c}_i m_{i+1} = \tilde{d}_i, \ \ i = 0,\ldots,N-1. \\ m_N = f_N', \end{gathered} \tag{3.35}$$

where

$$\tilde{a}_i = \Phi_{i-1}(x_{i-1}) \frac{h_{i-1}}{T_{i-1}}, \ \tilde{b}_i = \bar{a}_{i-1} \frac{h_{i-1}}{T_{i-1}} + \bar{b}_i \frac{h_i}{T_i}, \ \tilde{c}_i = \Psi_i(x_{i+1}) \frac{h_i}{T_i},$$

$$\tilde{d}_i = [\bar{a}_{i-1} + \Phi_{i-1}(x_{i-1})] \frac{h_{i-1}}{T_{i-1}} \Delta_{i-1} f + [\bar{b}_i + \Psi_i(x_{i+1})] \frac{h_i}{T_i} \Delta_i f.$$

Lemma 3.7. Let the constraints (3.32) be fulfilled. If the righthand elements of system (3.35) satisfy the following inequalities

$$f_0' \geq 0,\ f_N' \geq 0,\ \tilde{d}_i + \tilde{a}_i \tilde{d}_{i-1} / \tilde{b}_{i-1} - \tilde{c}_i \tilde{d}_{i+1} / \tilde{b}_{i+1} \geq 0, \qquad i = 1, \ldots, N-1. \tag{3.36}$$

where $\tilde{b}_0 = \tilde{b}_N = 1,\ \tilde{d}_0 = f_0',\ \tilde{d}_N = f_N',$ then $m_i \geq 0,\ i = 0, \ldots, N.$

Proof: It is easy to verify that under constraints (3.32), the systems (3.35) satisfies the conditions of Lemma 3.3. Therefore, by the restrictions (3.36), its solution is nonnegative: $m_i \geq 0,\ i = 0, \ldots, N$. This proves the lemma.

Theorem 3.4. Let a tension generalized spline S interpolate the monotone data $\{f_i\}$, $i = 0, \ldots, N$, and satisfy the endpoint conditions $S'(x_i) = f_i',\ i = 0, N$, where $f_0' \geq 0$ and $f_N' \geq 0$. Suppose the restrictions (3.32) and (3.36) are fulfilled. If the defining functions Φ_i and Ψ_i in (3.24), given by (3.25), are convex on $[x_i, x_{i+1}]$ and

$$1 \leq \frac{-h_i \Phi_i'(x)}{\Phi_i(x_i)} + \frac{h_i \Psi_i'(x)}{\Psi_i(x_{i+1})}, \quad i = 0, \ldots, N-1, \tag{3.37}$$

then $S'(x) \geq 0$ for all $x \in [a, b]$, that is, S is monotone on $[a, b]$.

Proof: By Lemma 3.7, one has $m_i \geq 0,\ i = 0, \ldots, N$. Using constraints (3.32) from system (3.35), we obtain

$$m_i \leq \tilde{d}_i / \tilde{b}_i \leq 2 \max(\Delta_{i-1} f, \Delta_i f), \quad i = 1, \ldots, N-1. \tag{3.38}$$

Differentiating (3.24) gives us

$$S'(x) = s(x, m_i, m_{i+1}) = \frac{h_i}{T_i}[h_i\alpha_i(x)\Delta_i f + \beta_i(x)m_i + \gamma_i(x)m_{i+1}],$$

where

$$\alpha_i(x) = \Psi_i'(x_{i+1})\Phi_i'(x) + \Phi_i'(x_i)\Psi_i'(x) - \Phi_i'(x_i)\Psi_i'(x_{i+1}),$$
$$\beta_i(x) = -\overline{b}_i\Phi_i'(x) + \Phi_i(x_i)\Psi_i'(x) - \Phi_i(x_i)\Psi_i'(x_{i+1}),$$
$$\gamma_i(x) = -\Psi_i(x_{i+1})\Phi_i'(x) + \overline{a}_i(x_i)\Psi_i'(x) - \Phi_i'(x_i)\Psi_i(x_{i+1}).$$

The function $s(x, m_i, m_{i+1})$ is linear in variables m_i and m_{i+1}. Therefore, for proving the inequality $S'(x) \geq 0$ for all $x \in [x_i, x_{i+1}]$, under restrictions (3.38), it is sufficient to verify the inequalities $s(x,0,0) \geq 0$, $s(x,2\Delta_i f,0) \geq 0$, $s(x,0,2\Delta_i f) \geq 0$, $s(x,2\Delta_i f, 2\Delta_i f) \geq 0$, for $x \in [x_i, x_{i+1}]$. It can be easily done immediately by using properties of functions Φ_i and Ψ_i and inequalities (3.37); hence, the desired result follows. This proves the theorem.

3.8 Examples of Defining Functions

Let us consider the defining functions Φ_i and Ψ_i in (3.25), which are in most common use. In the examples given below, they depend on the parameters:

$$\Phi_i(x) = \varphi_i(t)h_i^2 = \psi(p_i, 1-t)h_i^2,$$
$$\Psi_i(x) = \psi_i(t)h_i^2 = \psi(q_i, t),$$

where $t = (x - x_i)/h_i$ and $0 \leq p_i, q_i < \infty$.

(1) Rational splines with linear denominator (see *Späth* (1990)):

$$\psi_i(t) = t^3/[1 + q_i(1-t)]Q_i, \quad Q_i^{-1} = 2(3 + 3q_i + q_i^2).$$

The conditions of Lemma 3.6 are satisfied at $-1 < p_i, q_i < \infty$, $i = 0, \ldots, N-1$, and thus the interpolation rational spline exists and is unique. Lemma 2.5 holds, if, for example, we demand additionally that $p_i = q_i$, $i = 0, \dots, N-1$.

(2) Rational splines with a quadratic denominator (see *Späth* (1990)):

$$\psi_i(t) = t^3 / [1 + q_i t(1-t)] Q_i, \; Q_i^{-1} = 2(1+q_i)(3+q_i).$$

Here the conditions for Lemmas 3.5 and 3.6 to hold are the same as in (1).

(3) Exponential splines (see *Späth* (1974, 1990)):

$$\psi_i(t) = t^3 e^{-q_i(1-t)} / (6 + 6q_i + q_i^2).$$

(4) Hyperbolic splines (see Koch and Lyche (1993)) and numerous references there:

$$\psi_i(t) = \frac{\sinh q_i t - q_i t}{q_i^2 \sinh(q_i)}.$$

(5) Splines with additional nodes (see Pruess (1979)):

$$\psi_i(t) = \frac{1+q_i}{6}\left(t - \frac{q_i}{1+q_i}\right)_+^3.$$

If we take $\alpha_i = (1+p_i)^{-1}$ and $\beta_i = 1 - (1+q_i)^{-1}$, then the points, $x_{i1} = x_i + \alpha_i h_i$ and $x_{i2} = x_i + \beta_i h_i$, fix the positions of two additional nodes of the spline on the interval $[x_i, x_{i+1}]$. By moving them, we can go from a cubic spline to a piecewise linear interpolation (see Pruess (1979)).

Chapter IV

Monotonicity Preserving Parametrization

In this chapter, a new parametrization algorithm for interpolation by splines is given which almost invariably results in better shapes than either parametrization by chord length, uniform or centripetal parametrizations. The method is based on monotonicity preservation for the given data and is invariant under affine transformations. It enables one to improve the visual correspondence between curve and the initial data and gives a mesh concentration in large gradient domains. The algorithm can be generalized to approximate multivalued surfaces.

4.1 Background and Statement of the Problem

Let $P_i = (x_i, y_i)$, $i = 0,1,\ldots,N$, be a sequence of pairwise different data points in the xy-plane. In order to draw a curve passing through these points, it is in general necessary to construct a mesh Δ: $a = t_0 < t_1 < \cdots < t_N = b$ and to define a continuous vector-function $C(t) = (C_x(t), C_y(t))$, $t \in [a,b]$, such that

$$C_x(t_i) = x_i, \; C_y(t_i) = y_i, \; i = 0,1,\ldots,N,$$

i.e.

$$C(t_i) = P_i, \; i = 0,1,\ldots,N.$$

The chosen parameter values are called the *interpolating nodes*. The shape of the curve is determined by choice of the interpolating nodes as well as the method of interpolation on this mesh. The choice of the nodes greatly influences the resulting curve. The problem of finding a good set of interpolating nodes is known as the *parametrization problem.*

The simplest and most widely used choice is the *uniform* parametrization, provided by

$$t_i = t_{i-1} + h,\ h = (b-a)/N,\ \ i = 1,2,\dots,N.$$

This is generally unsatisfactory for the obvious reason that the nodes do not relate to the distribution of the data points. The choice of the interpolating nodes should be based on the behaviour of the data, giving a *data dependent* parametrization. It is generally accepted that a better choice is the *cumulative chord length* parametrization

$$t_i \ = t_{i-1} + \frac{\left|P_i - P_{i-1}\right|}{\sum_{J=1}^{N}\left|P_j - P_{j-1}\right|}(b-a),\ \ j = 1,2,\dots,N,$$

with $|\bullet|$ denoting the usual Euclidean distance. Here, the term "better" refers to a rather vague quality of the curve: its "fairness". There is no a precise definition of this quality but it is customary to accept that a curve is fair if it reproduces the interpolation polygon well and has a high degree of smoothness (see Criterion A in Sapidis and Farin (1990)).

The theoretical foundation of parametrization by cumulative chord length for natural splines was laid by Epstein (1976). In this case, the curve has no corners. But it was shown by Lee (1992) that it may have cusps. Roughly, the distinction between a corner and a cusp (for a parametric curve) is this: the position of the tangent line changes discontinuously across a corner (at which the tangent line is undefined), whereas across a cusp, the tangent line varies continuously, but the unit tangent vector reverses its direction. Such a parametrization has sometimes been called the "natural parametrization". The main reason for this choice seems to be that it roughly approximates the *arc length* parametrization (see Boor (1978)):

$$t_i = t_{i-1} + \int_{t_{i-1}}^{t_i} \sqrt{[C'_x(t)]^2 + [C'_y(t)]^2}\, dt, \quad i = 1,2,\ldots,N,$$

which requires the iterations (see Spath (1974)). However, our aim is to create a curve through a given set of points, and it is not clear why one should strive for the arc length parametrization, nor is it clear that the suggested iterations should even converge.

One obtains the *exponential* parametrization of Lee (1989), if

$$t_i = t_{i-1} + \frac{\left|P_i - P_{i-1}\right|^e}{\sum_{j=1}^{N}\left|P_j - P_{j-1}\right|^e}(b-a), \quad j = 1,2,\ldots,N, \; 0 \le e \le 1$$

As a particular cases, as e = 0, 0.5, and 1, this gives uniform, *centripetal* (see Lee(1989)), and chord length parametrizations. With nearly equally spaced points, all these three parametrizations are roughly the same. In general, the centripetal parametrization gives better results than either the chord length or the uniform parametrizations.

The *affine invariant* parametrization by Foley and Nielson (1989) takes the geometry of the control points into consideration and produces quality results for a wide variety of curve/surface fitting problems. The interpolating nodes can also be derived through optimization techniques (see Greiner (1994), Kurchatov and Snigirev (1989), Marin (1984)). *Intrinsic* parametrization by Hoschek (1988) uses the minimization of the distance between given points P_i and an approximation curve which is a nonlinear problem. But optimization methods are expensive, and moreover, it is not entirely clear what objective function should be used.

The choice of the interpolating nodes can be based on the preservation of the data shape properties such as monotonicity, convexity, etc. We shall say that a curve $C(t)$ is monotonicity preserving for the given data in the interval $[t_k, t_l]$, $l > k$, provided the following conditions are fulfilled

$$C'_x(t)(x_{j+1}, x_j) > 0,\ C'_y(t)(y_{j+1} - y_j) > 0, \text{ if } t \in [t_j, t_{j+1}],$$
$$\text{for all } j = k, \dots, l-1. \qquad (4.1)$$

4.2 Affine Invariance of Polynomials and Splines in Parametric Space

Let $R = (-\infty, \infty)$ be a real axis. Let us consider an affine transformation of the parametric space $R \to \hat{R}: \hat{t} = qt + p,\ p, q = const.$ Then the mesh $\Delta: t_0 < t_1 < \dots < t_N$, is transformed into the mesh $\hat{\Delta}: \hat{t}_0 < \hat{t}_1 < \dots < \hat{t}_N$, where $\hat{t}_j = qt_j + p$, $j = 0, 1, \dots, N$.

Lemma 4.1. Interpolating Lagrange polynomials are invariant with respect to affine transformations of the parameter space R.

Proof: The Lagrange polynomial of degree n that interpolates the data (t_j, f_j), $j = i, \dots, i+n$, has the form

$$L_{i,n}(t) = \sum_{j=i}^{i+n} f_j l_j(t),\quad l_j(t) = \prod_{k=i, k\neq j}^{i+n} \frac{(t - t_k)}{(t_j - t_k)}.$$

Here

$$l_j(t) = \prod_{k=i, k\neq j}^{i+n} \frac{(t - t_k)}{(t_j - t_k)} = \prod_{k=i, k\neq j}^{i+n} \frac{(\hat{t} - \hat{t}_k)}{(\hat{t}_j - \hat{t}_k)} = \hat{l}_j(\hat{t}).$$

Hence

$$L_{i,n}(t) = \sum_{j=i}^{i+n} f_j l_j(t) = \sum_{j=i}^{i+n} f_j \hat{l}_j(\hat{t}) = \hat{L}_{i,n}(\hat{t}).$$

This proves the lemma.

Let us consider on the interval $[a,b]$, in addition to the mesh Δ another mesh $\delta : a<\zeta_1 <\cdots <\zeta_{N-n} <b$, such that $t_{i-1} <\zeta_i <t_{i+n}$, $i=1,2,\ldots,N-n$. By $S_{n,1}(\delta)$, we denote a set of polynomial splines of degree n with nodes of multiplicity 1 on the mesh δ. Then (see Zavyalov et al. (1980)) there exists a unique spline $S \in S_{n,1}(\delta)$, satisfying the interpolation conditions

$$S(t_i) = f_i, \ i=0,1,\ldots,N.$$

Let us extend the mesh δ by introducing additional nodes $\zeta_{-n} <\cdots <\zeta_0 \le a$, $b \le \zeta_{N-n+1} <\cdots <\zeta_{N+1}$. Any spline $S \in S_{n,1}(\delta)$, can be uniquely represented on the interval $[a,b]$ in the form (see Zavyalov et al. (1980))

$$S(t) = \sum_{i=-n}^{N-n} b_i B_{n,i}(t),$$

where

$$B_{n,i}(t) = (\zeta_{i+n+1} - \zeta_i) \sum_{k=i}^{i+n+1} \frac{(\zeta_k - t)^n{}_+}{\omega'_{n+1,i}(\zeta_k)},$$

$$\omega_{n+1,i}(t) = (t-\zeta_i)(t-\zeta_{i+1})\cdots(t-\zeta_{i+n+1}), \quad t_+ = \max(0,t),$$

is a normalized basis spline with support interval (ζ_i, ζ_{i+n+1}).

Under the above affine transformation of the parametric space, the meshes Δ and δ transform into the meshes $\hat{\Delta} : \hat{t}_j = qt_j + p$, $j=0,1,\ldots N$, and $\hat{\delta} : \hat{\zeta}_j = q\zeta_j + p$, $j=1,2,\ldots,N-n$. The additional nodes of the mesh $\delta : \zeta_j$, $j=-n,\ldots,0,N-n+1,\ldots,N+1$, are substituted by the nodes $\hat{\zeta}_j$, $j=-n,\ldots,0,N-n+1,\ldots,N+1$, of the extended mesh $\hat{\delta}$.

Lemma 4.2. Interpolating splines are invariant with respect to affine transformations of the parameter space R.

Proof: Let us show the equality $B_{n,1}(t) = \hat{B}_{n,1}(\hat{t})$. If $n = 0$, then by definition (see Zavyalov et al. (1980))

$$B_{0,i}(t) = \begin{cases} 1 & \text{if } t \in [\zeta_i, \zeta_{i+1}), \\ 0 & \text{otherwise.} \end{cases}$$

Therefore $\hat{B}_{0,i}(\hat{t}) = B_{0,i}(t)$. Suppose the required equality is fulfilled for $n-1=k$ $(k \geq 0)$. Then, by virtue of the recurrence relation for normalized B-splines (see Zavyalov et al. (1980)) and by induction

$$\begin{aligned} B_{n,i}(t) &= \frac{t-\zeta_i}{\zeta_{i+n}-\zeta_i} B_{n-1,i}(t) + \frac{\zeta_{i+n+1}-t}{\zeta_{i+n+1}-\zeta_{i+1}} B_{n-1,i+1}(t) \\ &= \frac{\hat{t}-\hat{\zeta}_i}{\hat{\zeta}_{i+n}-\hat{\zeta}_i} \hat{B}_{n-1,i}(\hat{t}) + \frac{\hat{\zeta}_{i+n+1}-\hat{t}}{\hat{\zeta}_{i+n+1}-\hat{\zeta}_{i+1}} \hat{B}_{n-1,i+1}(\hat{t}) = \hat{B}_{n,i}(\hat{t}). \end{aligned}$$

If now S and $\hat{S}$ are interpolating splines on the meshes δ and $\hat{\delta}$, respectively, then, by virtue of the uniqueness of the spline representation as a linear combination of the basis splines, the following representation holds:

$$S(t) = \sum_{i=-n}^{N-n} b_i B_{n,i}(t) = \sum_{i=-n}^{N-n} b_i \hat{B}_{n,i}(\hat{t}) = \hat{S}(\hat{t}).$$

This proves the lemma.

Note that, the invariance with respect to affine transformations of the paramatric space of cubic weighted ν-splines, and in particular conventional C^2 cubic splines, was shown in Foley (1987).

Let us obtain the relation between S' and $\hat{S}'$. It is known (see Zavyalov et al. (1980)) that

$$S'(t) = n\sum_{i=-n+1}^{N-n} b_i^{(1)} B_{n-1,i}(t)\ ,\ b_i^{(1)} = (b_i - b_{i-1})/(\zeta_{i+n} - \zeta_i).$$

From here

$$b_i^{(1)} = q(b_i - b_{i-1})/(\hat{\zeta}_{i+n} - \hat{\zeta}_i) = q\hat{b}_i^{(1)}.$$

Hence, by virtue of Lemma 4.2, we have

$$S'(t) = q\hat{S}'(\hat{t}). \tag{4.2}$$

Let $z = (z_1, \dots, z_M)$. By the vector $z > 0$ (< 0), we mean that $z_i > 0$ (< 0) for all $i = 1, 2, \dots, M$. We shall have to make the following refinement of the result from Miroshnichenko (1984).

Lemma 4.3. If $A = \left(a_{ij}\right)_{i,j=1}^{M}$ is a real matrix with diagonal dominance and

$$a_{ii} > 0,\ a_{ij} \le 0\ \ (j \ne i),\ i, j = 1, 2, \dots, M,$$

then from the condition $Az > 0$ (< 0), it follows that $z > 0$ (< 0).

Proof: In the conditions of the Lemma, the matrix A is of monotone type (see chapter 3), that is, from $Az \ge 0$ (≤ 0), it follows that $z \ge 0$ (≤ 0). Let $Az > 0$. Assuming $z_k = 0$, we have was shown in Foley (1987).

Let us obtain the relation between S' and $\hat{S}'$. It is known (see Zavyalov et al. (1980)) that

$$S'(t) = n\sum_{i=-n+1}^{N-n} b_i^{(1)} B_{n-1,i}(t)\ ,\ b_i^{(1)} = (b_i - b_{i-1})/(\zeta_{i+n} - \zeta_i).$$

From here

$$b_i^{(1)} = q\,(b_i - b_{i-1})/(\hat{\zeta}_{i+n} - \hat{\zeta}_i) = q\hat{b}_i^{(1)}.$$

Hence, by virtue of Lemma 4.2, we have

$$S'(t) = q\hat{S}'(\hat{t}). \tag{4.2}$$

Let $z = (z_1,\dots,z_M)$. By the vector $z > 0\ (<0)$, we mean that $z_i > 0\ (<0)$ for all $i=1,2,\dots,M$. We shall have to make the following refinement of the result from Miroshnichenko (1984).

Lemma 4.3. If $A = \left(a_{ij}\right)_{i,j=1}^{M}$ is a real matrix with diagonal dominance and

$$a_{ii} > 0,\ \ a_{ij} \le 0\ \ (j \ne i),\ \ i,j=1,2,\dots,M,$$

then from the condition $Az > 0\ (<0)$, it follows that $z > 0\ (<0)$.

Proof: In the conditions of the Lemma, the matrix A is of monotone type (see chapter 3), that is, from $Az \ge 0\ \ (\le 0)$, it follows that $z \ge 0\ (\le 0)$. Let $Az > 0$. Assuming $z_k = 0$, we have

$$\sum_{j=1}^{M} a_{kj} z_j = \sum_{j=1, j\neq k}^{M} a_{kj} z_j \leq 0.$$

This contradiction proves the lemma.

4.3 Algorithm of Parabolic Parametrization

Let us consider the behaviour of the parabola $L_{i,2}$ passing through the points (t_j, f_j), $j = i, i+1, i+2$, depending on the mesh choice. Let us seek the knot t_{i+1} such that the parabola should be monotonicity preserving for the initial data, that is, such that the following relations are met

$$L'_{i,2}(t)(f_{j+1} - f_j) > 0, \quad j = i, i+1.$$

We introduce the following notation,

$$\alpha_i = (t_{i+1} - t_i)/(t_{i+2} - t_i), \qquad T_i = t_{i+2} - t_i.$$

As a corollary of the Lemma 4.1 the knots $t_i = 0$ and $t_{i+2} = 1$ can be fixed. Then

$$\alpha = t_{i+1}, \quad T_i = 1,$$

$$\frac{d}{dt} L_{i,2}(t) = \frac{(1-\alpha_i)(f_{i+1} - f_i) + (2t - \alpha_i)[\alpha_i(f_{i+2} - f_i) - f_{i+1} + f_i]}{\alpha_i(1-\alpha_i)}. \tag{4.3}$$

Let $F_i \equiv (f_{i+1} - f_i)(f_{i+2} - f_{i+1})$. We consider three possible configurations of the initial data:

A. $F_i > 0$;

B. $F_i < 0$;

C. $F_i = 0$.

A. By assumption the sequence $f_i < f_{i+1} < f_{i+2}$ or $f_i > f_{i+1} > f_{i+2}$ is monotonic. Since $L'_{i,2}(t)$ is a linear function, the monotonicity of $L_{i,2}$ is equivalent to the fulfillment of two inequalities

$$L'_{i,2}(t_i)(f_{i+1} - f_i) > 0, \;\; L'_{i,2}(t_{i+2})(f_{i+1} - f_i) > 0.$$

Simple manipulations, based on the fact that $0<\alpha_i<1$, readily yield the inequalities

$$-\alpha_i^{\,2} + \frac{f_{i+1} - f_i}{f_{i+2} - f_i} > 0, \;\; -(1-\alpha)^2 + \frac{f_{i+2} - f_{i+1}}{f_{i+2} - fi} > 0.$$

From these inequalities, it follows that

$$\alpha^f_{\min} = 1 - \sqrt{\frac{f_{i+2} - f_{i+1}}{f_{i+2} - f_i}} < \alpha_i < \sqrt{\frac{f_{i+1} - f_i}{f_{i+2} - f_i}} = \alpha^f_{\max}. \qquad (4.4)$$

The same argument implies the following result:

Lemma 4.4. If the restriction imposed on the initial data $F_i = (f_{i+1} - f_i) \times (f_{i+2} - f_{i+1}) > 0$ holds, monotonicity of the parabola is equivalent to the knot t_{i+1} being in the interval $T_i^f = (t_i + \alpha^f_{\min} T_i, \; t_i + \alpha^f_{\max} T_i)$.

One value of the knot t_{i+1} in the range T_i^f can be obtained by minimizing the parabola length. Since $f_{i+2} \neq f_i$ the equality (4.3) can be transformated into the form

$$(f_{i+2} - f_i)^{-1} L'_{i,2}(t) = 1 + \frac{1-2t}{\alpha_i(1-\alpha_i)} \left(\frac{f_{i+1} - f_i}{f_{i+2} - f_i} - \alpha_i\right).$$

Hence, if

$$\alpha_i = (f_{i+1} - f_i) / (f_{i+2} - f_i), \tag{4.5}$$

then, we have the equality $(f_{i+2} - f_i)^{-1} L'_{i,2}(t) = 1$, that is, the parabola degenerates into a straight line and it is easily verified that $t_{i+1} = t_i + \alpha_i T_i \in T_i^f$.

B. In this case, the knot t_{i+1} is chosen so that

$$L'_{i,2}(t_{i+1}) = 0.$$

Since, by virtue of (4.3), it follows that

$$L'_{i,2}(t_{i+1}) = \alpha_i^{-1}(1-\alpha_i)^{-1}[\alpha_i^2(f_{i+2} - f_i) - 2\alpha_i(f_{i+1} - f_i) + f_{i+1} - f_i]$$
$$= \alpha_i^{-1}(1-\alpha_i)^{-1}[\alpha^2_i(f_{i+2} - f_{i+1}) + (1-\alpha_i)^2(f_{i+1} - f_i)]$$

we have

$$\alpha_i = \frac{\sqrt{|f_{i+1} - f_i|}}{\sqrt{|f_{i+1} - f_i| \quad + |f_{i+2} - f_{i+1}|}}. \tag{4.6}$$

C. There are three possible configurations of the initial data:

1. $f_i = f_{i+1} = f_{i+2}$;
2. $f_i = f_{i+1}, f_{i+1} \neq f_{i+2}$;
3. $f_i \neq f_{i+1}, f_{i+1} = f_{i+2}$.

In the first case, the position of the knot can be chosen arbitrarily because $L_{i,2}(t) = constant$. The selection range for t_{i+1} will be the interval $T_i^f = (0,1)$. In the second and third cases, the parabola is non-monotone for any position of the knot t_{i+1} (for case 2 in the interval $[t_i, t_{i+1}]$, and for case 3 in the interval $[t_{i+1}, t_{i+2}]$, respectively). For continuity of formulae (4.5) and (4.6), we set

$$\alpha_i = \begin{cases} \varepsilon & if \;\; f_i = f_{i+1}, \; f_{i+1} \neq f_{i+2}, \\ 1-\varepsilon & if \;\; f_i \neq f_{i+1}, \; f_{i+1} = f_{i+2}, \end{cases} \qquad (4.7)$$

where ε is a small number (e.g. ε =(computer precision)$\times$ 100). In the case B and C, the interval T_i^f degenerates into the point t_{i+1}.

Therefore, formulae (4.5)-(4.7), determine a location of the knot t_{i+1} for any possible configuration of the initial data.

Let us formally denote the set of parameters obtained from formulae (4.5)-(4.7), for the data $f_0, f_1, \ldots, f_N$, as $\alpha_i^f = \alpha_i, \;\; i = 0,1,\ldots,N-2.$

Using described above algorithm of parabolic parametrization, one can find two sets of parameter values α_i^x, $i = 0,1,\ldots,N-2$, and α_i^y, $i = 0,1,\ldots,N-2$, for data sets $x_0, x_1, \ldots, x_N$, and $y_0, y_1, \ldots, y_N$, correspondingly.

The mesh Δ for the curve $C(t) = (C_x(t), C_y(t))$ is constructed by the two found meshes for functions C_x and C_y. It follows from the above that

Lemma 4.5. The curve C will be rendered monotonicity preserving for the initial data if $T_i^x \cap T_i^y \neq \phi$ and $t_{i+1} \in T_i^x \cap T_i^y$, for all $i = 0,1\ldots,N-2$. Otherwise the conditions of monotonicity preservation are violated.

Let us specify the algorithm for parametrizing the curve C in the general case. We define the parameters

$$\hat{\alpha}_i = {d_i}\big/{(d_i + d_{i+1})}, \;\; d_j = ((x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2)^{1/2},$$
$$j = i, i+1; \;\; i = 0,1,\ldots,N-2,$$

which fix a mesh normed by cumulative chord length. Then we choose $t_{i+1} = t_i + \hat{\alpha}_i$, if $t_{i+1} \in T_i^x \cap T_i^y$. If $t_i + \hat{\alpha}_i \notin T_i^x \cap T_i^y$ or $T_i^x \cap T_i^y = \phi$, we set

$$t_{i+1} = t_i + \alpha_i ,$$

where

$$\alpha_i = \begin{cases} \alpha_i^x & \text{if} \quad T_i^x \subset T_i^y, \\ \alpha_i^y & \text{if} \quad T_i^y \subset T_i^x, \\ (\alpha_i^x + \alpha_i^y)/2 & \text{otherwise.} \end{cases} \tag{4.8}$$

The value $t_{i+1} \notin T_i^x \cap T_i^y \neq \phi$ is additionally corrected to the nearest point of the interval $T_i^x \cap T_i^y$. Thus, the mesh has been constructed.

One can say that we consider a parametrization which lies some where between uniform and cumulative chord length parametrizations, similar to centripetal parametrization but with very different algorithm.

Let us denote the step size of the mesh Δ by $h_i = t_{i+1} - t_i$, $i = 0,1,\ldots,N-1.$ For a curve $C(t)$ passing through the points p_i, $i = 0,1\ldots,N$, the mesh $\Delta : t_0 < t_1 < \cdots < t_N$, is uniquely determined by a set of parameters, $\alpha_0, \alpha_1, \cdots, \alpha_{N-2}$, assigning the relations of the mesh steps $h_i / h_{i-1}, i = 1,2,\ldots,N-1$, and values h_0 or h_{N-1} .

For a vector-function $C_3(t) = (C_x(t), C_y(t), C_z(t))$, which defines the curve in the xyz-space passing through the points $Q_i = (x_i, y_i, z_i)$, $i = 0,1,\ldots,N$, the general parametrization algorithm is similar to the 2-D case with the formal substitution of the interval $T_i^x \cap T_i^y \cap T_i^z$ for the interval $T_i^x \cap T_i^y$. In formula (4.8), instead of $(\alpha_i^x + \alpha_i^y)/2$, we choose $\alpha_i = (\alpha_i^x + \alpha_i^y + \alpha_i^z)/3.$

Furthermore, the parametrization will be called monotonicity preserving if the parameters, $\alpha_0, \alpha_1, \cdots, \alpha_{N-2}$, are chosen by the algorithms described above.

4.4 Parametrization for Cubic Splines

Let us consider a conventional cubic spline on the mesh $\Delta : t_0 < t_1 < \cdots < t_N$, satisfying the interpolation conditions

$$S(t_i) = f_i, \;\; i = 0,1,\ldots,N.$$

In addition, let us choose boundary conditions preferable for practical calculations (see Beatson and Chacko (1989)):

$$S'(t_i) = f_i', \ i = 0, N. \tag{4.9}$$

Note that in order to preserve the invariance of the spline under affine transformations of the parameter space, by virtue of (4.2), the value f_i' must be modified by $\hat{f}_i' = q^{-1} f_i', \quad i = 0, N.$

The given data will be called strictly monotonic if $f_0 < f_1 < \cdots < f_N$ or $f_0 > f_1 > \cdots > f_N$ and if in addition the following inequalities are met

$$f_0'(f_1 - f_0) > 0, \quad f_N'(f_N - f_{N-1}) > 0.$$

We will say that the spline S preserves the strict monotonicity of the initial data if $S'(t) > 0$, $t \in [t_0, t_N]$, and $f_0 < f_1 < \cdots < f_N$ or $S'(t) < 0$, $t \in [t_0, t_N]$, and $f_0 > f_1 > \cdots > f_N$. Sufficient conditions (see Miroshnichenko (1984)) for the initial data are established which ensure that given nondecreasing initial data $f_0 \le f_1 \le \cdots \le f_N$, and the derivative of the cubic spline is nonnegative $S'(t) \ge 0$, $t \in [t_0, t_N]$.By virtue of Lemma 4.3, the follows from Miroshnichenko (1984) that

Lemma 4.6. A cubic spline S with boundary conditions (4.9), preserves the strict monotonicity of the given data if the following conditions are fulfilled

$$|f_0'| < 3|f_1 - f_0| / h_0, \ |f_N'| < 3|f_N - f_{N-1}|,$$
$$h_i h_{i-1}^{-1} |f_i - f_{i-1}| < (h_{i-1} + 2h_i) h^{-1}{}_i |f_{i+1} - f_i|, \ i = 1,2,\ldots, N-1,$$
$$h_{i-1} h_i^{-1} |f_{i+1} - f_i| < (2h_{i-1} + h_i) h_{i-1}^{-1} |f_i - f_{i-1}|, \ i = 1,2,\ldots, N-1,$$

Let us denote again $\alpha_i = (t_{i+1} - t_i) / (t_{i+2} - t_i)$, $T_i = t_{i+2} - t_i$, $i = 0,1,\ldots, N-2$. Since according to the algorithm of monotonicity preserving parametrization $h_i/h_{i-1} = (1-\alpha_{i-1})/\alpha_{i-1}$, then from Lemma 4.6, the following restrictions on α_i follow immediately:

$$1-\sqrt{\frac{f_{i+2}-f_{i+1}}{f_{i+2}-f_i}} < \alpha_i < \sqrt{\frac{f_{i+1}-f_i}{f_{i+2}-f_i}}, \quad i = 0,1,\ldots,N-2,$$

which coincide with conditions (4.4) of monotonicity preserving parametrization. Thus, we have proved.

Theorem 4.1. Let the initial data be strictly monotonic and given on the mesh constructed by the algorithm of monotonicity preserving parametrization. Then a cubic spline S, that interpolates this data and satisfies the boundary conditions (4.9), preserves the strict monotonicity of the initial data if the following inequalities are met:

$$h_0 < 3(f_1 - f_0)/f_0', \; h_{N-1} < 3(f_N - f_{N-1})/f_N'. \tag{4.10}$$

The following two corollaries follow immediately from Theorem 4.1.

Corollary 4.1. Let the initial data be strictly monotonic. The interpolating parametric cubic spline, $C(t) = (C_x(t), C_y(t))$, with nodes on the mesh Δ constructed by the algorithm for monotonicity preserving parametrization is monotonicity preserving for the initial data if

1. $T_i^x \cap T_i^y \neq \phi, \quad i = 0,1\ldots,N-2;$
2. $h_0 < 3\min(\frac{x_1-x_0}{x_0'}, \frac{y_1-y_0}{y_0'}),$
3. $h_{N-1} < 3\min(\frac{x_N-x_{N-1}}{x_N'}, \frac{y_N-y_{N-1}}{y_N'}).$

Corollary 4.2. Let $x_0 < x_1 < \cdots < x_N$ and $x_0' > 0,\ x_N' > 0$. The interpolating parametric cubic spline with nodes on the mesh Δ such that $t_{i+1} \in T_i^x,\ i = 0,1,\ldots,\ N-2$, and $h_0 < 3(x_1 - x_0) / x_0',\ h_{N-1} < 3(x_N - x_{N-1}) / x_N'$, monotonically increases on the interval $[t_0, t_N]$.

If the restrictions of Corollary 4.2 are fulfilled, then for single-valued functional data, there is a one-to-one correspondence between the points of the x-axis and the curve points, that is, there exists a single-valued function $y = y(x)$ with a graph (C_x, C_y).

Let us consider the choice of h_0 and h_{N-1}. Conditions (4.10) can be fulfilled in two ways:

1. Starting from specified values f_0' and f_N', we find h_0 and h_{N-1};
2. We fix h_0 and h_{N-1} and correct f_0' and f_N'.

In the first case, we choose as $h_{i+1} = h_i(1-\alpha_i)/\alpha_i$,

$$R = \prod_{i=0}^{N-2} \frac{h_{i+1}}{h_i} = \prod_{i=0}^{N-2} \frac{1-\alpha_i}{\alpha_i}.$$

We have $h_{N-1} = h_0 R$. Furthermore, if

$$h_0 = 3\min\left(\frac{f_1 - f_0}{f_0'}, \frac{f_N - f_{N-1}}{R f_N'}\right),$$

then conditions (4.10) will be fulfilled.

The second case is necessary in particular if the knots t_0 and t_N of the mesh Δ are fixed, e.g. if we use the normed parametrization $\Delta: 0 = t_0 < t_1 < \cdots < t_N = 1$. We set

$$f_0' = \begin{cases} L_{0,2}'(t_0) & \text{if } L_{0,2}'(t_0)(f_1 - f_0) > 0; \\ \varepsilon\, sign(f_1 - f_0) & otherwise. \end{cases}$$

Analogously we find f_N'. In order to set f_0' and f_N', the cubic Lagrange polynomials can also be used.

Remark 4.1. By virtue of formula (4.5), the step size of the "monotonizing" mesh for monotonicity preserving parametrization is essentially increased in the domains of sharply increasing "gradient" of the initial data. Hence, the points of the spline whose values are obtained on a uniform partition of the interval, $[t_0, t_N]$, will be concentrated in such domains. This property of the suggested parametrization is useful in applications.

4.5 Examples and Numerical Results

The figures below illustrate the employment of monotonicity preserving parametrization (mp-parametrization for short) in interpolations by parametric cubic and generalized tension splines of chapter 3 with the defining functions

$$\psi_i(t) = t^3/[1 + q_i t(1-t)]Q_i, \quad Q_i^{-1} = 2(1+q_i)(3+q_i), \quad \varphi_i(t) = \psi_i(1-t).$$

For comparison, spline curves with centripetal, cumulative chord length, and uniform parametrizations, which are the most common, are given and are comparable with the method suggested in this chapter in terms of implementation complexity and computer resources consumed. The solid, dotted, dashed and dash-dotted lines show, respectively, the curves with mp-, centripetal, cumulative chord length, and uniform parametrizations. The bullet signs denote the data points. In the construction of the cubic and generalized tension splines boundary conditions of type (4.9) were used where the derivatives were computed by means of the second degree Lagrange interpolating polynomials: $S'(t_0) = L_{0,2}'(t_0)$ and $S'(t_N) = L_{N-2,2}'(t_N)$.

As our first example, we have interpolated Akima's data of Table 4.1. The effects of using four different parametrizations are depicted in Fig.4.1. Figs.4.1a and 4.1b are obtained setting $q_i = 0$, for all *i*, that is considering the parametric cubic splines interpolating the data. Uniform and cumulative chord length parametrizations are utterly unsatisfactory. The graph of the spline with centripetal parametrization fails in one-to-one correspondence between the points of the *x*-axis and the curve. The spline with mp-parametrization has small oscillations along the data, because the conditions 1 of Corollary 4.1 are violated. A magnification on Fig. 4.1b shows clearly this effect. In Figs. 4.1c and 4.1d, the new interpolants with tension parameters $q_0 = 1, q_1 = 11, q_2 = 10, q_3 = 16, q_4 = 35, q_5 = 5.5, q_6 = 3.8, q_7 = 4, q_8 = 5, q_9 = 2$ are displayed for the same data, and the stretching effect of the increase in tension parameters is evident.

Figure 4.2 illustrates an example from C. de Boor's book (1978). The data points have been obtained from the function $f(x) = (x - 03)^2$, with x = 0, 0.1, 0.2, 0.3, 0.301, 0.4,0.5, 0.6. Here, centripetal and uniform parametrizations give a cusp and a loop correspondently. Nonsymmetric and nonmonotone graph of the cubic spline ($q_i = 0$ for all *i*) with mp-parametrization on Fig. 4.2a and 4.2b can be easily improved by using tension with $q_0 = 1.5$, $q_1 = 2.5$, $q_2 = 0.75$, $q_3 = 1.12$, $q_4 = 7.5$, $q_5 = q_6 = 0$ (see Figs. 4.2c and 4.2d).

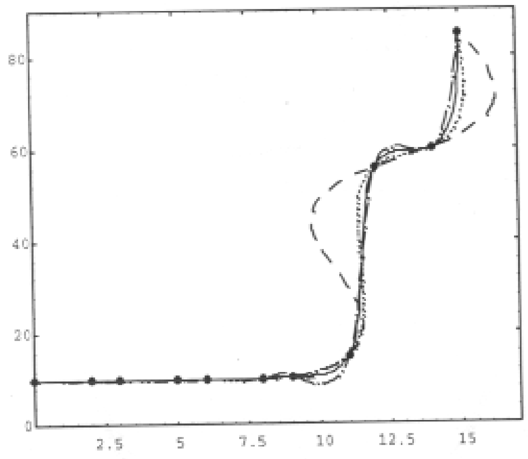
The data for Figure 4.3 (Table 4.2) has been taken from the book by *Späth* (1974). Figs. 4.3a and 4.3b show the plots of interpolating cubic splines produced by a uniform choice of tension parameters, namely, $q_i = 0$. In Figs. 4.3c and 4.3d to approximate the segment of a straight line passing through the last three data points, we have set $q_5 = q_6 = q_7 = 10$, while the remaining q_i are unchanged.

The data for Figure 4.4 (Table 4.3) has been taken from the function $f(x) = (x - 5)^4 + 2$ with $x = 2.5 + i$, $i = 0,1,\ldots,5$, considered by Goodman and Unsworth (1988). Figs. 4.4a and 4.4b (no tension) and 4.4c and 4.4d (with tension parameters $q_0 = 1.5, q_1 = 2.5, q_2 = 0.75,\ q_3 = 1.12,\ q_4 = 7.5$) show that we can easily transfer from nonmonotone to monotone graph of the spline with mp-parametrization

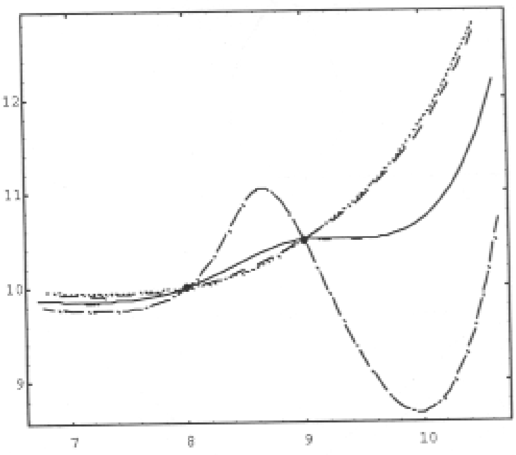
Figure 4.5 illustrates the behaviour of the splines for the data selected by us on the plane using a drawing: {x} = {0, 0.5, 1.3, 1.4, 1.1, 1.1, 0.5, 1.1, 1, 1.7, 1.7, 1, 1, 1.4, 0.5}, {y}= {0, 0.5, 0.6, 0.8, 1.5, 1.8, 2.3, 2.5, 3, 3.2, 3.5, 4.5, 4.7, 5, 7}. Figs. 4.5a and 4.5b are obtained setting $q_i = 0$ for all *i*. In Figs. 4.5c and 4.5d, we have used tension parameters $q_7 = 2$, $q_8 = 18.5$, $q_9 = 1$, $q_{10} = 3.8$, $q_{11} = 0.8$, while the remaining parameters q_i are unchanged.

Table 4.1. Akima's Data:

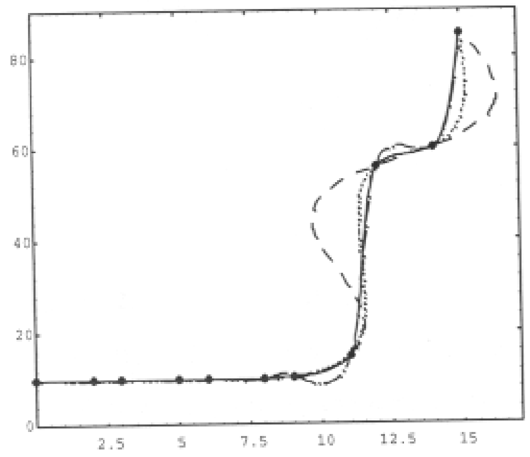
x_i	0	2	3	5	6	8	9	11	12	14	15
y_i	10	10	10	10	10	10	10.5	15	56	60	85



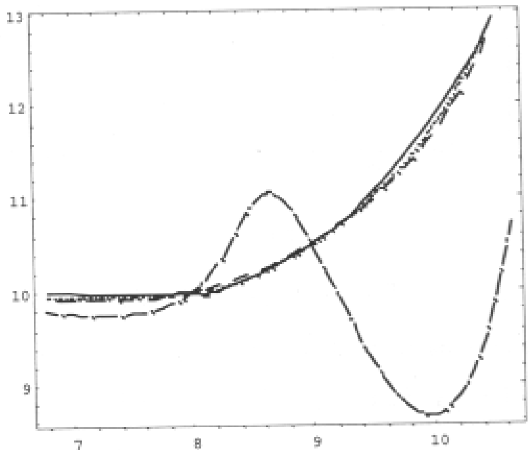
(a)



(b)



(c)



(d)

Figure 4.1. Akima's data with sharp gradient increase. (a) Interpolation using mp-, centripetal, cumulative chord length and uniform parametrizations ($q_i = 0$). (b) Magnification of the interval [8,9]. (c), (d) The same as (a),(b) but with $q_0 = 1$, $q_1 = 11$, $q_2 = 10$, $q_3 = 16$, $q_4 = 35$, $q_5 = 5.5$, $q_6 = 3.8$, $q_7 = 4$, $q_8 = 5$, $q_9 = 2$.

-·-·-·-·-·-·-·-·-	uniform parametrization
- - - - - - - - - - - - - - -	cumulative chord length parametrization
. . . .	centripetal parametrization
________________	monotonicity preserving parametrization

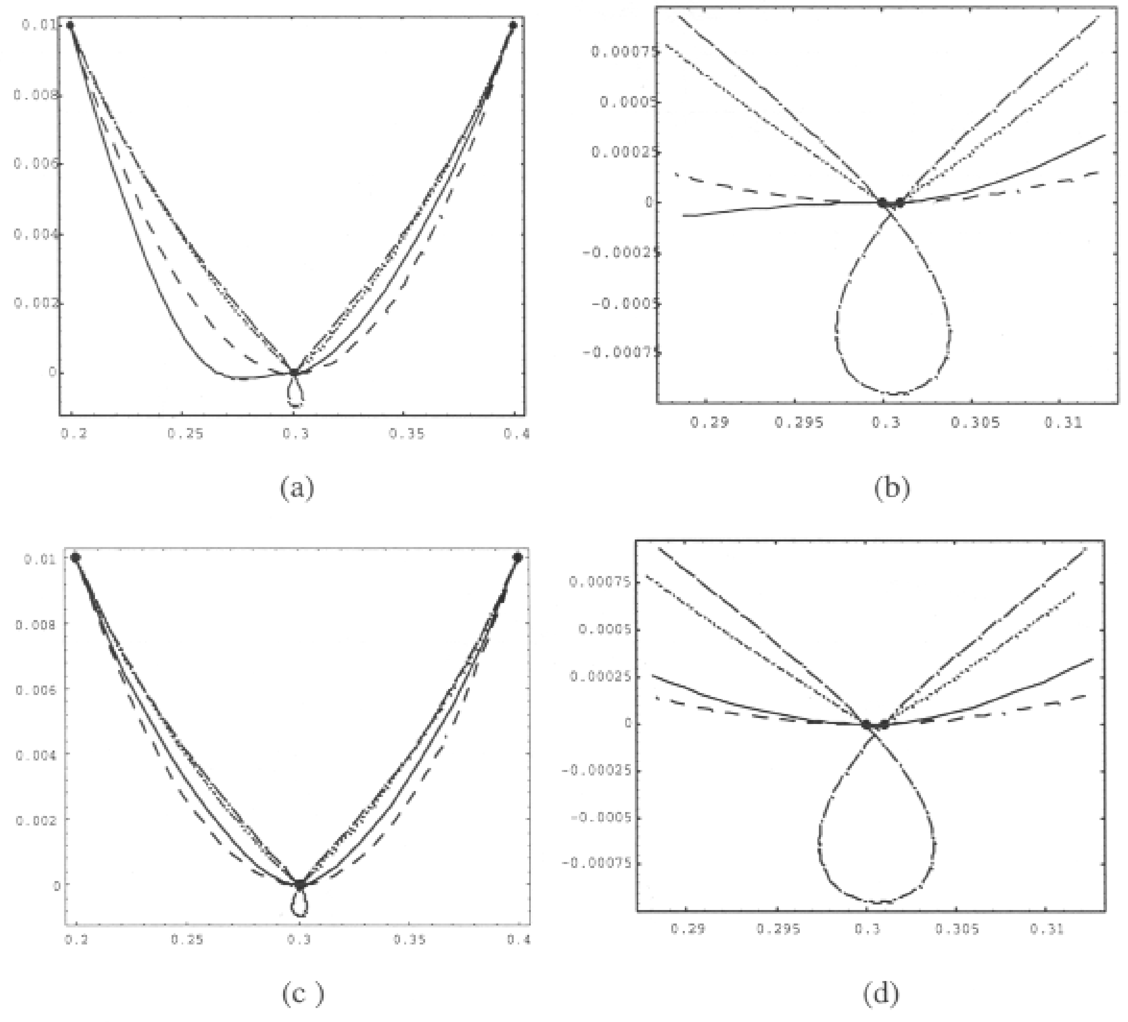


Figure 4.2. C. de Boor example. (a) Parametrizations for the parabola $f(x) = (x - 03)^2$ with x = 0, 0.1, 0.2, 0.3, 0.301, 0.4,0.5, 0.6 and magnification of the interval [0.2,0.4]. (b) Magnification of the interval [0.29,0.31]. (c),(d) The same as (a), (b) but with $q_0 = 1.5,\ q_1 = 2.5,\ q_2 = 0.75,\ q_3 = 1.12,$ $q_4 = 7.5,\ q_5 = q_6 = 0.$

-·-·-·-·-·-·-·-·-·- uniform parametrization

- - - - - - - - - - - - - - - cumulative chord length parametrization

. . . . centripetal parametrization

——————— monotonicity preserving parametrization

Table 4.2. *Späth* Data:

x_i	0	2	2.5	3.5	5.5	6	7	8.5	10
y_i	2	2.5	4.5	5	4.5	1.5	1	0.5	0

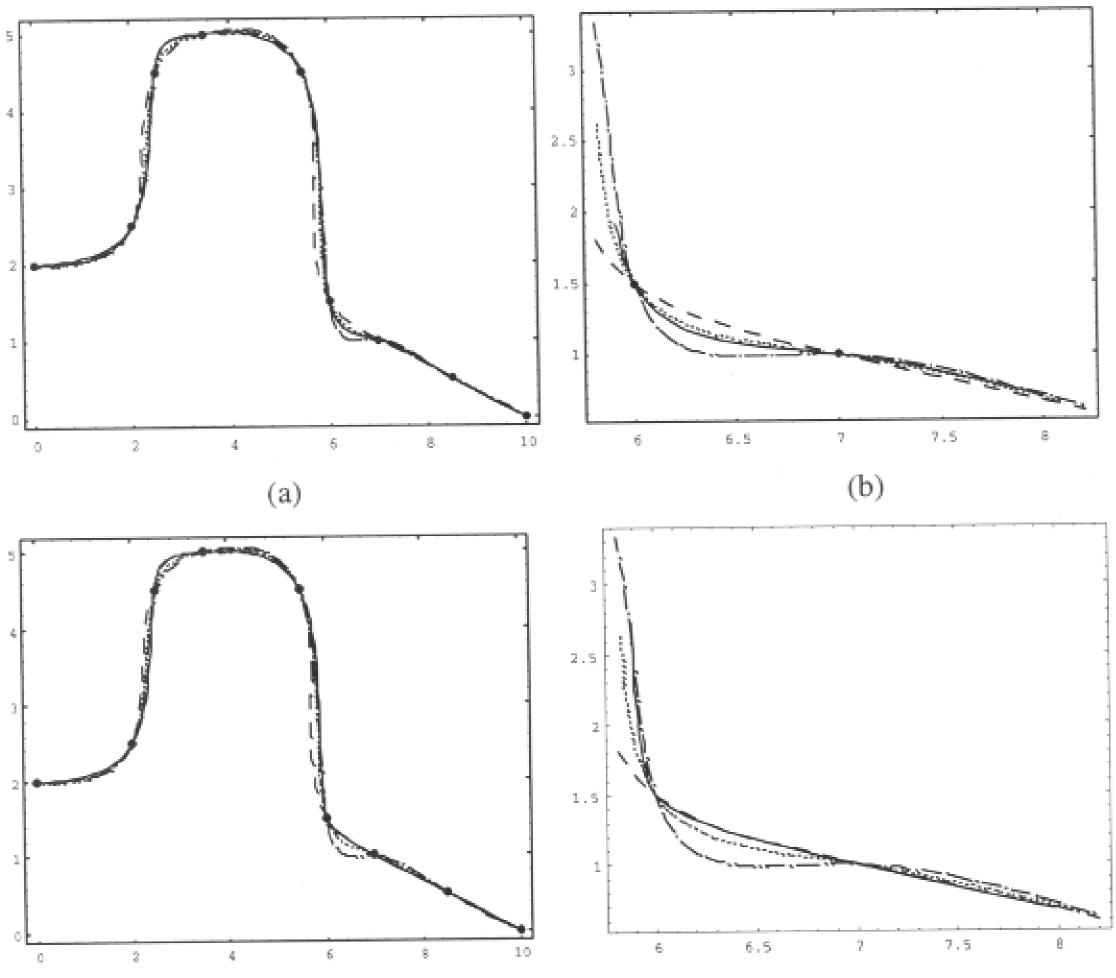


Figure 4.3. *Späth* data. (a) Interpolation by parametric cubic splines. (b) Magnification of the interval [6,8]. (c), (d) The same as (a), (b) but with $q_i = 10,\ i = 5,6,7$.

-·-·-·-·-·-·-·-·- uniform parametrization

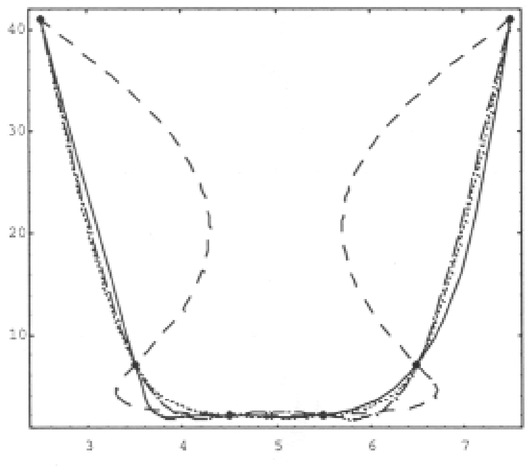
- - - - - - - - - - - - - - cumulative chord length parametrization

. . . . centripetal parametrization

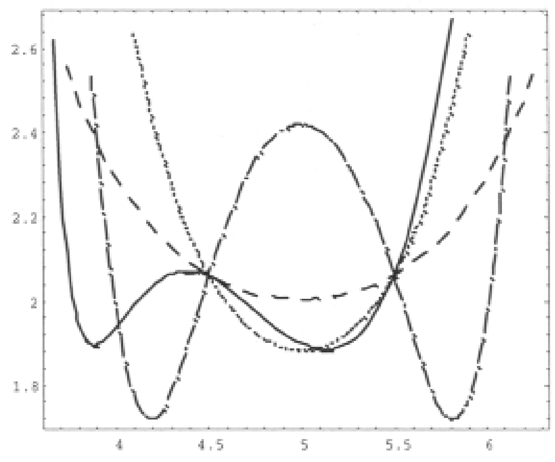
______________ monotonicity preserving parametrization

Table 4.3. Data for function $f(x) = (x-5)^4 + 2$ with $x = 2.5 + i,\ i = 0,1,\ldots,5.$:

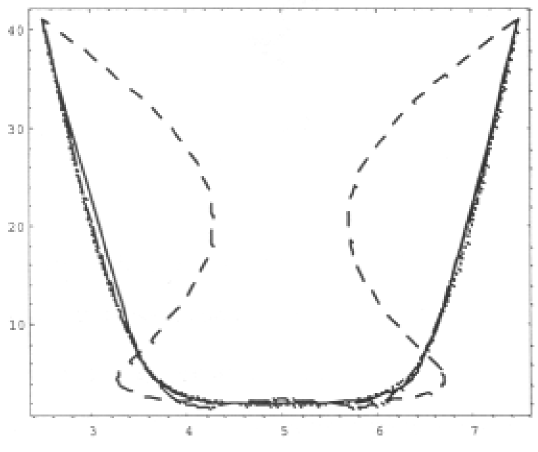
x_i	2.5	3.5	4.5	5.5	6.5	7.5
y_i	41.0625	7.0625	2.0625	2.0625	7.0625	41.0625



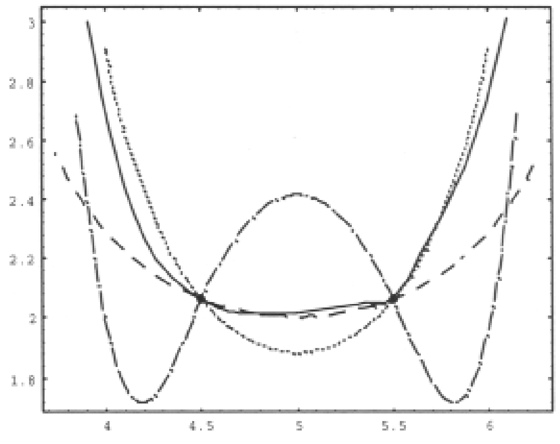
(a)



(b)



(c)



(d)

Figure 4.4. (a), (b) Parametrizations for the data obtained from the function $f(x) = (x-5)^4 + 2$ with $x = 2.5 + i,\ i = 0,1,\ldots,5.$ (c), (d) The same as (a), (b) but with $q_0 = 1.5,\ q_1 = 2.5,$ $q_2 = 0.75,\ q_3 = 1.12,\ q_4 = 7.5$.

-·-·-·-·-·-·-·-·-	uniform parametrization
- - - - - - - - - - - - - - -	cumulative chord length parametrization
. . . .	centripetal parametrization
———————	monotonicity preserving parametrization

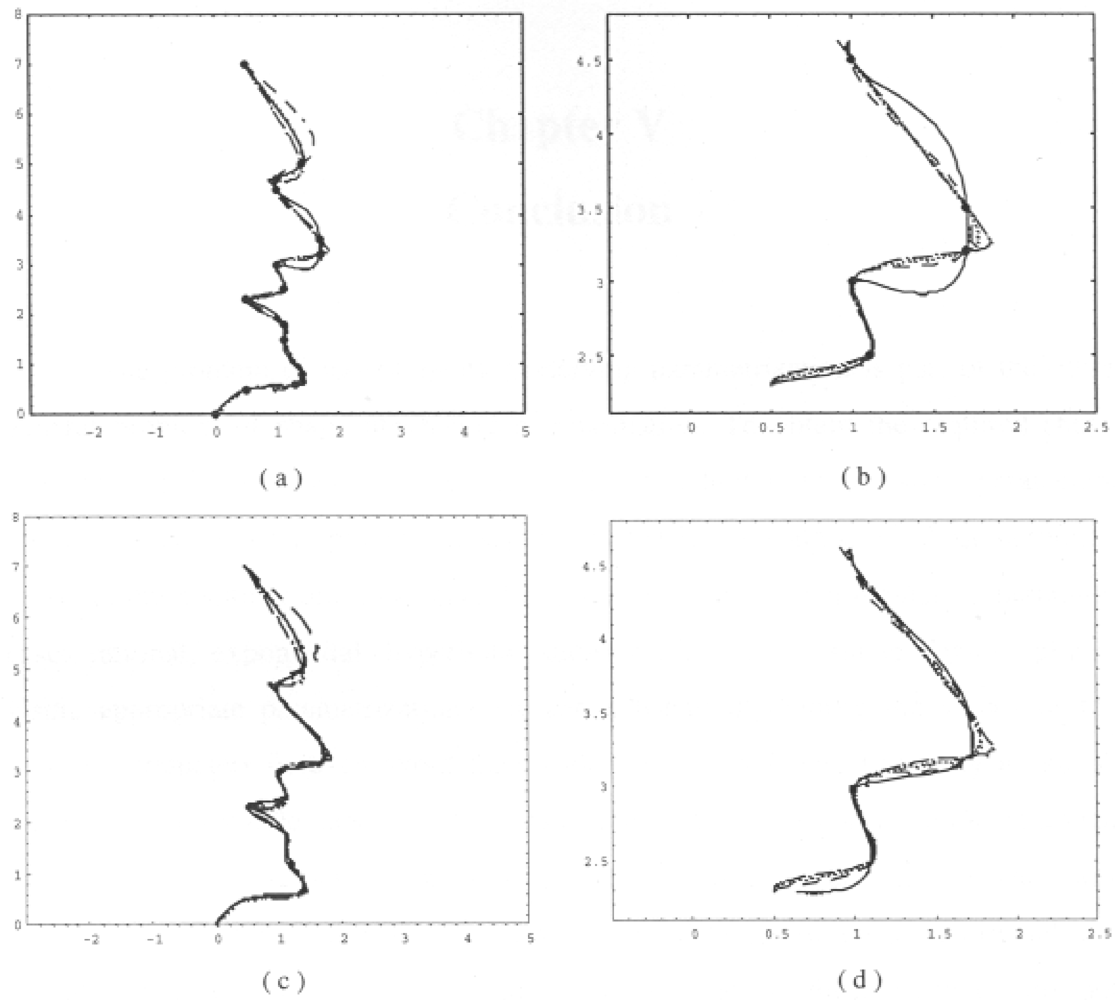


Figure 4.5. Example with the data of a "human face".(a), (b) No tension. (c),(d) The same as (a), (b) but with tension parameters $q_7 = 2$, $q_8 = 18.5$, $q_9 = 1$, $q_{10} = 3.8$, $q_{11} = 0.8$. while the remaining parameters q_i are unchanged.

-.-.-.-.-.-.-.-.-.-	uniform parametrization
- - - - - - - - - - - - - - -	cumulative chord length parametrization
. . . .	centripetal parametrization
———————	monotonicity preserving parametrization

Chapter V
Conclusion

The problem of monotonicity preserving parametrization is part of the more general problem of shape preserving approximation. To obtain the required shape properties of the resulting curve/surface, different authors introduce some parameters into the structure of the spline to satisfy the geometric constraints. For those purposes, most common splines are now generalized tension spline which include as particular cases rational, exponential, hyperbolic additional-knots, a variable degree splines. Using appropriate parametrization, one can substantially reduce the values of the tension parameters or even avoid their application completely. Evidently a "good" parametrization should depend on the data and retain properties of the initial data such as positivity, monotonicity, convexity, presence of linear sections, etc. The most common methods of parametrization for discrete data are uniform, centripetal and cumulative chord-length parametrizations. Unfortunately, these methods do not preserve the above mentioned data shape properties and very often do not have anything in common with the shape of the data.

This thesis presents one of the first attempts to develop a "true" shape preserving parametrization. To construct a parametrization method, only one shape property of the data has been chosen, that is, monotonicity. Very often this property is the most valuable one. If we can construct a monotone curve matching the monotone data then the property of positivity will be provided automatically. We consider only a "comonotone" approximation when the curve satisfies criterion (4.1). This approach permits us to construct a very simple algorithm of parametrization, but which is not always practical. As a result, in order to guarantee monotonicity of the resulting curve for monotone data, one still needs to apply tension splines. However, the values of the tension parameters can be reduced, when compared with other methods of parametrization.

In this thesis, a new parametrization algorithm for interpolation by splines is given which almost invariably results in better shapes than either parametrization by chord length, uniform or centripetal parametrizations. The method is based on monotonicity preservation for the given data and is invariant under affine transformations. It enables one to improve the visual correspondence between curve and the initial data and gives a mesh concentration in large gradient domains. The algorithm can be generalized to approximate multivalued surfaces.

A large variety of applications now requires the use of curve/surface description, especially in fields such as computer aided design and machining, and computer vision and inspection of manufactures parts. Other areas, where the description of curves/surfaces is of interest include many fields of science, medicine, cartography, television and the film industry. This diversity and the wide range of applicability of the subject enables us to consider the problem of constructing monotonicity preserving curve/surface spline parametrization as very valuable.

The results obtained in this thesis can also be used in many applied problems and first of all in computer aided geometric design (computer description and numerical modelling of aircraft surfaces, bodies of ships and cars, complex details of engines etc.). These algorithms can be applied also in many fields of science (mathematical models in mechanics and physics describing by differential equations), in high resolution TV system, in medical research (software for digital diagnostic equipment) etc.

References

References

Akima, H. (1970). A new method of interpolation and smooth curve fitting based on local procedures. J. Assoc. Comput. Mech. 17, 589-602.

Beatson, R. K. and Chacko, E. (1989). A quantative comparison of end conditions for cubic spline interpolation, in: *Approximation Theory VI: Proceedings of the Sixth International Symposium on Approximation Theory*. Vol. I. C. K. Chui, L. L. Schumaker and J. D. Ward (eds.), pp. 77-79, Academic Press, Boston.

Burmeister, W., Hess, W., and Schmidt, J. (1985). Convex spline interpolants with minimal curvature. Computing 35, 219-229.

Collatz, L. (1964). *Funktionalanalysis and Numerische Mathematik.* Springer Verlag, Berlin, Gottingen.

Costantini, P. (1987). Co-monotone interpolating splines of arbitrary degree a local approach. SIAM J. Sci. Stat. Comput. 8, 1026-1034.

De Boor, C. (1978). *A practical Guide to Splines.* Springer Verlag, New York.

Delbourgo, R., and Gregory, J. A.(1985). Shape preserving piecewise rational interpolation. SIAM J. Sci. and Statist. Comput. 6, 967-976.

Dougherty, R., Edelman, A., and Hyman, J. M. (1989) Positivity-, monotonicity-, or convexity-preserving cubic and quintic hermite interpolation. Math. Comput. 52, 471-494.

Epstein, M. P. (1976). On the Influence of Parametrization in Parametric Interpolation, SIAM J. Numer. Anal. 13, 261-268.

Farin, G., (1993) *Curves and Surfaces for Computer Aided Geometric Design. A Practical Guide*. Academic Press, San Diego.

Foley, T. A. (1986). Local control of interval tension using weighted splines, Computer Aided Geometric Design 3, 281-294.

Foley, T. A. (1987). Interpolation with interval and point tension control using cubic weighted ν-splines, ACM Trans. Math. Soft. 13, 68-96.

Foley, T. A. and Nielson, G. M. (1989). Knot selection for parametric spline interpolation. in: *Mathematical Methods in Computer Aided Geometric Design*. T. Lyche and L. L. Schumaker (eds.), pp. 261-271, Academic Press, Boston.

Fritsch, F. N. and Butland, J. (1984). A method for constrained local monotone piecewise cubic interpolants. SIAM J. Sci. Stat. Comput. 5, 300-304.

Fritsch, F. N. and Carlson, R. E. (1988). Monotone piecewise cubic interpolation. SIAM J. Numer. Anal. 17, 238-246.

Goodman, T. N. T. and. Unsworth, K. (1988). Shape-preserving interpolation by parametrically defined curves. SIAM J. Numer. Anal. 25, 1453-1465.

Gregory, J. A. and Delbourgo, R. (1982). Piecewise rational quadratic interpolation to monotone data. IMA J. Numer. Anal. 2, 123-130.

Greiner, G. (1994). Variational design and fairing of spline surfaces, Computer Graphics Forum 13, 144-154.

Hoschek, J. (1988). Intrinsic parametrization for approximation, Computer Aided Geometric Design 5, 27-31.

Koch, P. E. and Lyche, T. (1993). Interpolation with Exponential B-splines in Tension, in: *Geometric Modelling, Computing/Supplementum 8*. Farin G. et al. (eds.). Springer-Verlag, Wien, pp.173-190.

Kurchatov, E. Yu. and Snigirev, V. F. (1989). Best choice of spline knots in automation of contour design, in: *Mathematical and Experimental Methods of Technical System Synthesis*, Kazan, pp. 38-43 (in Russian).

Kvasov, B. I. (1996a). Shape preserving spline approximate via local algorithms, in: *Advanced Topics in Multivariate Approximation*. F. Fontanella, K. Jetter and P. J. Laurent (eds.), pp. 181-196, World Scientific Publishing Co., Inc., Singapore.

Kvasov, B. I. (1996b). GB-splines and their properties, Annals of Numerical Mathematics 3, 139-149.

Kvasov, B. I. (1996c). Isogeometric interpolation by generalized splines, Rus. J. Numer. Anal. Math. Modelling 10, 223-246.

Lee, E. T. Y. (1989). Choosing nodes in parametric curve interpolation. CAD 21 (1989), 363-370.

Lee, E. T. Y. (1992). Corners, Cusps, and Parametrization: Variations on a Theorem of Epstein, SIAM J. Numer. Anal. 29, 553-565.

Marin, S. P. (1984). An approach to data parametrization in parametric cubic spline interpolation problems, Approx. Theory 41, 64-86.

McAllister, D., Passow, E. and Roulier, J. (1977). Algorithms for computing shape preserving spline interpolations to data. Math. Comput. 31, 717-725.

Miroshnichenko, V. L. (1984). Convex and monotone spline interpolation, in: *Constructive Theory of Functions'84*, Sofia, pp. 610-620.

Nielson, G. M. (1974). Some piecewise polynomial alternatives to splines under tension, in: *Computer Aided Geometric Design* (Eds.). R. E. Barnhill and R. F. Riesenfeld). Academic Press, New York, pp. 209-235.

Nielson, G. M.(1987). Coordinate free scattered data interpolation, in: *Topics in Multivariate Approximation*, C. Chui, L. L. Schumaker and F. Utreras (eds.), Academic Press, New York, 175-184.

Nielson, G. M. and Foley, T. A. (1989). A survey of applications of an affine invariant norm, in: *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. L. Schumaker (eds.), pp.445-467, Academic Press, New York.

Pruess, S. (1976). Alternatives to the Exponential Spline in Tension, Math. Comp. 33, 1273-1281.

Pruess, S. (1993). Shape preserving C^2 cubic spline interpolation. IMA J. Num. Anal. 13, 493-507.

Renka, R. (1987). Interpolation tension splines with automatic selection of tension factors. SIAM J. Sci. Stat. Comp. 8, 393-415.

Sapidis, N. and G. Farin. (1990). Automatic fairing algorithm for B-spline curves, Computer Aided Design 22, 121-129.

Schmidt, W. J. and Hess, W. (1987). Quadratic and related exponential splines in shape preserving interpolation. J. Comput. Appl. Math. 18, 321-330.

Schweikert, D. G. (1966). An interpolating curve using splines in tension. J. Math. Phys. 45, 312-317.

Späth, H. (1969). Exponential spline interpolation. Computing 4, 225-233.

Späth, H. (1974). *Spline Algorithms for Curves and Surfaces.* Utilitas Mathematica Publishing, Inc., Winnipeg.

Späth, H.(1990). *Eindimensionale spline-Interpolations-Algorithmen*. R. Oldenbourg Verlag, *München* .

Zavyalov, Yu. S., Kvasov, B. I. and Miroshnichenko, V. L. (1980). *Methods of spline functions*. Nauka, Moscow, (in Russian).

Appendix A

Appendix A
Computer Program for Monotonicity Preserving Parametrization

```
{$A+,B-,D+,E+,F-,G-,I+,L+,N-,O-,P-,Q-,R+,S+,T-,V+,X+}{$M 65520,0,655360}
Program     Tension_Spline;
Uses   Crt, Graph;
const  N       = 11;
       NN      = N;
       max     = 20;
       epsilon = 1e-4;
type   arr1 = array[0..N-1] of real;
       arr2 = array[0..(N-1)*10] of real;
       arr3 = array[0..N-2] of real;
       RowType = Array[0..N] of Real;
var    T, X, Y, mi, mi_for_x, mi_for_y, del_x, del_y, xii, yi,PP,QQ : arr1;
       P,X1,Y1,X2,Y2,X3,Y3,X4,Y4 : arr2;
        pi, qi : arr3;
       N1 : integer;
       sum,a,b,l1,l2,l3,l4,U : real;
       mat : Array[0..N-1] of RowType;
       start,stop : integer;

 Procedure     RowReduce;
 Var    i : integer;
 Begin
        PP[0] := mat[0,1]/mat[0,0];
        QQ[0] := mat[0,N]/mat[0,0];
        For i := 1 to N-2 do
                begin
                        PP[i] := mat[i,i+1]/(mat[i,i] - mat[i,i-1]*PP[i-1]);
                        QQ[i] := (mat[i,N] - (mat[i,i-1]*QQ[i-1]))/
                                  (mat[i,i] - (mat[i,i-1]*PP[i-1]));

                end;
        QQ[N-1] := (mat[N-1,N] - (mat[N-1,N-2]*QQ[N-2]))/
                    (mat[N-1,N-1] - (mat[N-1,N-2]*PP[N-2]));

End;


Procedure       Back_Subsitution;
Var     i : Integer;
Begin
        mi[N-1] := QQ[N-1];
        For i := N-2 Downto 0 do
        begin
                mi[i] := - (PP[i]*mi[i+1]) + QQ[i];

        end;
End;


Procedure       Fine_delta_s(xi,xi_1,xi_2,Si,Si_1,Si_2:Real ; Var delta:Real);
Var     delta1,delta2 : Real;
Begin
        delta1 := (Si_1 - Si)/(xi_1 - xi);
        delta2 := (Si_2 - Si_1)/(xi_2 - xi_1);
        delta  := delta2 - delta1;
End;


Function        qq_i(qi1:real):Real;
Begin
```

```
        qq_i  := 1/(6 + 6*qi1 + 2*qi1*qi1);
End;

Function        Sine_(q4,t: real):real;
Var     qqq : real;
Begin
        qqq   := qq_i(q4);
        Sine_   := (((3*Sqr(t))*(1+q4*(1-t)) + q4*(t*t*t))/Sqr(1+q4*(1-t))) * qqq;
end;

Function        sin_(i:integer ; XX,q21,h:Real):Real;
Var     t4 : Real;
begin
        t4  := (XX - T[i])/h;
        sin_  := h*Sine_(q21,t4);
End;

Function        phi_(i:integer ; XX,p21,h:Real):Real;
Var     t3: Real;
Begin
        t3    := (XX - T[i])/h;
        phi_ := -Sine_(p21,1-t3) * h;
End;

Function        Sine(q3,t : Real) : Real;
Var     qqqq : Real;
Begin
        qqqq := qq_i(q3);
        Sine   := ((t*t*t)/(1+q3*(1-t)))*qqqq;
End;
```

```
Function        sin(i:integer ; XX,q11,h:Real):Real;
Var     t2 : Real;
Begin
        t2 := (XX - T[i])/h;
        sin:= Sine(q11,t2)*(h*h);
End;

Function        phi(i:integer ; XX,p11,h:Real):Real;
var     t5 : real;
Begin
        t5  := (XX - T[i])/h;
        phi  := Sine(p11,1-t5)*(h*h);
End;

Procedure       Fine_a_i(i:integer; XX,p1,h:real ; Var a_i:Real);
Begin
        a_i  := -phi(i,xx,p1,h) - h*phi_(i,xx,p1,h);
End;

Procedure       Fine_b_i(i:integer; XX,q1,h:real ; Var b_i : real);
Begin
        b_i := -sin(i,xx,q1,h) + h*sin_(i,xx,q1,h);
End;

Procedure       MakeDelta(N : Integer; T,X,Y : Arr1);
Var     i : Integer;
        delta_x,delta_y : Real;
Begin
        for i:=1 to N-2 do
        begin
                Fine_delta_s(T[i-1],T[i],T[i+1],X[i-1],X[i],X[i+1],delta_x);
```

```
            del_x[i] := delta_x;
            Fine_delta_s(T[i-1],T[i],T[i+1],Y[i-1],Y[i],Y[i+1],delta_y);
            del_y[i] := delta_y;
        end;
End;


Procedure      Spline(N:integer;T,X:Arr1;Delta:arr1;del_0,del_n_1:real ;Var mi:arr1);
var     h, h1, a_i, b_i : Real;
        i,j  : integer;
Begin
        qi := pi;
        FillChar(mat,Sizeof(mat),0);
        mat[0,0]     := 1;
        mat[0,N]     := del_0;

        mat[N-1,N-1] := 1;
        mat[N-1,N]   := del_n_1;

        For i:= 0 to N-3 do
        begin
             h  := T[i+1] - T[i];
             h1:= T[i+2] - T[i+1];

             Fine_a_i(i+1, T[i+1], pi[i], h1, a_i );
             Fine_b_i(i, T[i+1], qi[i], h, b_i );

             mat[i+1,i]     := phi(i, T[i], qi[i], h)/h;
             mat[i+1,i+1]  := (b_i/h + a_i/h1);
             mat[i+1,i+2]  := sin(i+1, T[i+2], qi[i], h1)/h1;
             mat[i+1,N]    := delta[i+1];
        end;
        h  := T[N-2] - T[N-3];
        h1:= T[N-1] - T[N-2];

        Fine_a_i(N-2, T[N-2], pi[N-2], h1, a_i );
        Fine_b_i(N-3, T[N-2], qi[N-2], h, b_i );

        mat[N-2,N-3]  := phi(N-3, T[N-3], qi[N-2], h)/h;
        mat[N-2,N-2]  := (b_i/h + a_i/h1);
        mat[N-2,N-1]  := sin(N-2, T[N-1], qi[N-2], h1)/h1;
        mat[N-2,N]    := delta[N-2];

        RowReduce;
        Back_Subsitution;
End;

Function      Ipoint(var U:real; N:integer; var X:arr1):integer;
Var     i,k: integer;
Begin {Ipoint}
        For I:= 0 to N-2 do
        If (x[i]<=U) and (U<=x[i+1]) then k:=i;
        Ipoint:=k;
End  {Ipoint};

Function      Seval(N:integer;  xx,qi:real ; TT,X,Mi : arr1) : real;
var     j  : integer;
        h1,t : Real;
const   i : integer = 0;
Begin { Seval }
        i := Ipoint(xx,N,TT);
        h1:= TT[i+1]-TT[i];
        t := (xx-TT[i])/h1;
        Seval := (X[i] - Phi(i,TT[i],qi,h1)*mi[i])*(1-t) + (X[i+1] -Sin(i,TT[i+1],qi,h1)
                 *mi[i+1])*t+Phi(i,xx,qi,h1)*mi[i]+Sin(i,xx,qi,h1)*mi[i+1];
End {Seval};

Procedure       Graphics(N:integer; X1,Y1,X2,Y2,X3,Y3,X4,Y4:arr2);
var     a,b,p,xv1,xv2,yv1,yv2,Ylow,Yhigh:real;
        ErrCode,grDriver,grMode,i,l  : integer;
        st : string;

        Function        I1(x:real):integer;
        begin
                I1 := trunc((xv2-xv1)*(x-a)/(b-a)+xv1)
        end;
        Function I2(y:real):integer;
        begin
                I2 := trunc((yv1-yv2)*(y-Ylow)/(Yhigh-Ylow)+yv2)
        end;

Begin {Graphics}{ Choice of scale }
        xv1:=0; xv2:=630; yv1:=0; yv2:=349;
        a := X1[Start]; b := a;
        {a := X2[Start]; b := a;}
        {a := X3[Start]; b := a;}
        {a := X4[Start]; b := a;}
        for i:= Start+1 to Stop do
        begin
                p:=X1[i];
                if p<a  then a:=p;
                if p>b  then b:=p;

                {p:=X2[i];
        if p<a  then a:=p;
        if p>b  then b:=p;}

        {p:=X3[i];
        if p<a  then a:=p;
        if p>b  then b:=p;}

        {p:=X4[i];
        if p<a  then a:=p;
        if p>b  then b:=p;}
end;
Ylow:=Y1[Start]; Yhigh:=Ylow;
{Ylow:=Y2[Start]; Yhigh:=Ylow;}
{Ylow:=Y3[Start]; Yhigh:=Ylow;}
{Ylow:=Y4[Start]; Yhigh:=Ylow;}
for i:= Start+1 to Stop do
begin
        p:=Y1[i];
        if p<Ylow  then Ylow :=p;
        if p>Yhigh then Yhigh:=p;

        {p:=Y2[i];
        if p<Ylow  then Ylow :=p;
        if p>Yhigh then Yhigh:=p;}

        {p:=Y3[i];
        if p<Ylow  then Ylow :=p;
        if p>Yhigh then Yhigh:=p;}

        {p:=Y4[i];
        if p<Ylow  then Ylow :=p;
        if p>Yhigh then Yhigh:=p;}
end;
Ylow  := Ylow-0.5;
Yhigh := Yhigh+0.5;

{ Initialization of Graphical Regime }
grDriver:=3; grMode:=1; InitGraph(grDriver,GrMode,'c:\tp\bgi');
ErrCode:=GraphResult; if Not (ErrCode=grOk) then
begin RestoreCrtMode;
        WriteLn('Graphics error:',GraphErrorMsg(ErrCode)); exit
end;
{ Clear of the screen }
SetBkColor(black); ClearDevice;
{ Drawing of the axis }
SetLineStyle(SolidLn,0,3); SetColor(white);
if(a*b<=0) then line(I1(0),I2(Ylow),I1(0),I2(Yhigh));
if(Ylow*Yhigh<=0) then begin SetLineStyle(SolidLn,0,1
Line(I1(a),I2(0),I1(b),I2(0)) end;
{ Drawing for the graph of the function }

SetLineStyle(SolidLn,0,0); SetColor(red);
MoveTo(I1(X1[Start]),I2(Y1[Start]));
for i:= Start+1 to Stop do
LineTo(I1(X1[i]),I2(Y1[i]));  Readkey;

{SetLineStyle(SolidLn,0,0); SetColor(white);
MoveTo(I1(X2[Start]),I2(Y2[Start]));
for i:= Start+1 to Stop do
LineTo(I1(X2[i]),I2(Y2[i]));  Readkey;}

{SetLineStyle(SolidLn,0,0); SetColor(green);
```

```
        MoveTo(I1(X3[Start]),I2(Y3[Start]));
         for i:= Start+1 to Stop do
         LineTo(I1(X3[i]),I2(Y3[i]));  Readkey;}

         {SetLineStyle(SolidLn,0,0); SetColor(yellow);
         MoveTo(I1(X4[Start]),I2(Y4[Start]));
         for i:= Start+1 to Stop do
         LineTo(I1(X4[i]),I2(Y4[i]));  Readkey;}

        for i:=0 to NN-1 do
        begin
                PutPixel(I1(X[i])-1,I2(Y[i]),LightRed);
                PutPixel(I1(X[i])+1,I2(Y[i]),LightRed);
                PutPixel(I1(X[i]),I2(Y[i]),LightRed);
                PutPixel(I1(X[i]),I2(Y[i])-1,LightRed);
                PutPixel(I1(X[i]),I2(Y[i])+1,LightRed);
        end;
         ReadKey;
         CloseGraph
End;  {of  Graphics}

Function        Norm(x1,y1,x2,y2,l:real):real;
Var     Temp : Real;
Begin
        Temp := Sqrt( sqr(x1-x2) + sqr(y1-y2) );
        Norm := exp(l*ln(temp));
End;

Procedure       Input_Tension;
Begin
        pi[0] := 0.;
        pi[1] := 0.;
        pi[2] := 0.;
        pi[3] := 0.;
        pi[4] := 0.;
        pi[5] := 0.;
        pi[6] := 0.;
        pi[7] := 0.;
        pi[8] := 0.;
        pi[9] := 0.;

        start := 0;
        stop := 100;
        qi := pi;
End;

Procedure       Input_Tension1;
Begin
        pi[0] := 1;
        pi[1] := 11;
        pi[2] := 10;
        pi[3] := 16;
        pi[4] := 35;
        pi[5] := 5.5;
        pi[6] := 3.8;
        pi[7] := 4;
        pi[8] := 5.;
        pi[9] := 2.;

        start := 0;
        stop := 100;
        qi := pi;
```

```
End;


Procedure Input_Data;
Begin
   X[0]  := 0;
   X[1]  := 2;
   X[2]  := 3;
   X[3]  := 5;
   X[4]  := 6;
   X[5]  := 8;
   X[6]  := 9;
   X[7]  := 11;
   X[8]  := 12;
   X[9]  := 14;
   X[10] := 15;

   Y[0]  := 10;
   Y[1]  := 10;
   Y[2]  := 10;
   Y[3]  := 10;
   Y[4]  := 10;
   Y[5]  := 10;
   Y[6]  := 10.5;
   Y[7]  := 15;
   Y[8]  := 56;
   Y[9]  := 60;
   Y[10] := 85;


     a := 0;
     b := 1;   { interval (a,b) }
End;
```

```
Procedure       MakeT(l:Real);
Var     i : Integer;
Begin
        Sum := 0;
        For i:= 0 to N-2 do
        Sum := Sum+Norm(X[i+1],Y[i+1],X[i],Y[i],l);
        T[0] := a;
        For i:=1 to N-1 do
        T[i] := T[i-1]+Norm(X[i],Y[i],X[i-1],Y[i-1],l)*(b-a)/Sum;
End;

Procedure       MakeSpline(l:real;Var S1,S2:Arr2);
Var     Temp : Real;
        I,J,Z  : Integer;
        FFF  : Text;
Begin
        MakeT(l);
        MakeDelta(N,T,X,Y);

        Spline (N, T, X, Del_x, del_x[0], del_x[N-1], mi);
        mi_for_x := mi;

        Spline (N, T, Y, Del_y, del_y[0], del_y[N-1], mi);
        mi_for_y := mi;

        N1:=10*(N-1);

        for j := 0 to N-2 do
        begin
                Temp := (T[J+1]-T[J])/10;
                for i:= 1 to 10 do
                begin
                        U := T[j]+(i-1)*Temp;
                        Z := j*10+i-1;
                        S1[Z] := Seval(N, U, qi[j], T, X, mi_for_x);
                        S2[Z] := Seval(N, U, qi[j], T, Y, mi_for_y);
                        P[Z]  := U;
                end;
      end;

      U := T[N-1];
      S1[N1] := Seval(N, U, qi[N-2], T, X, mi_for_x);
      S2[N1] := Seval(N, U, qi[N-2], T, Y, mi_for_y);
      P[N1]  := U;

      Assign(FFF,'UU1.DAT');
      Rewrite(FFF);
      For I:= 0 to 100 do
              Writeln(FFF,P[I],'  ',S1[I]);
      Close(FFF);

      Assign(FFF,'UU2.DAT');
      Rewrite(FFF);
      For I:= 44 to 67 do
              Writeln(FFF,P[I],'  ',S2[I]);
      Close(FFF);
End;
(*
{ Monotonicity Preserving Parametrizatin Algorithm }
Procedure       Find_Alpha(fi,fi_1,fi_2:real ; Var condition:char ; Var alpha:real);
Begin
   { Case A }
```

```
        If (fi_1 - fi)*(fi_2 - fi_1) > 0 then
                begin
                        alpha       := (fi_1 - fi)/(fi_2 - fi);
                        condition  := 'A';
                end  Else
    { Case B }
        If (fi_1 - fi)*(fi_2 - fi_1) < 0 then
                begin
                        Alpha      := sqrt(abs(fi_1 - fi))/sqrt(abs(fi_1-fi)+abs(fi_2-fi_1));
                        condition  := 'B';
                end  Else
    { Case C }
                begin
                        If (fi=fi_1) and(fi_1=fi_2)  then Alpha := 0.5 {any const} else
                        If (fi=fi_1) and(fi_1<>fi_2) then Alpha := epsilon  else
                        If (fi<>fi_1)and(fi_1=fi_2)  then Alpha := 1-epsilon ;
                        condition := 'C';
                end;
End;

Procedure       Find_min_max(fi,fi_1,fi_2,ti:real ; Var min,max,alpha:real);
Var     condition : char;
Begin
        Find_alpha(fi,fi_1,fi_2,condition,alpha);
        If Condition='A' Then
        begin
                min := 1-sqrt((fi_2 - fi_1)/(fi_2 - fi));
                max := sqrt((fi_1 - fi)/(fi_2 - fi));
                min := ti+min;
                max := ti+max;
        end Else
        begin
                min := ti+alpha;
                max := ti+alpha;
        end;
End;

Procedure       MakeT2(l:real);
Var     i  : Integer;
        Tx_min, Tx_max, Ty_min, Ty_max : Real;
        T_min, T_max                   : Real;
        a_, a__, di, di_1, ti_1        : real;
        a_x_i, a_y_i                   : real;
        condition                      : char;
Begin
        T[0] := a;
        a_ := 0;
        ti_1 := 0;
        For i:=0 to N-3 do
        begin
                di  :=Norm(X[i+1],Y[i+1],X[i],Y[i],l);
                di_1:=Norm(X[i+2],Y[i+2],X[i+1],Y[i+1],l);
                a__ :=di/(di+di_1);

                Find_min_max(x[i],x[i+1],x[i+2],t[i],Tx_min,Tx_max,a_x_i);
                {Find open set Txi and find alpha_x}
                Find_min_max(y[i],y[i+1],y[i+2],t[i],Ty_min,Ty_max,a_y_i);
                {Find open set Tyi and find alpha_y}

                If Tx_min > Ty_min Then T_min := Tx_min Else
                T_min := Ty_min;
                If Tx_max > Ty_max Then T_max := Ty_max Else
        T_max := Tx_max;
        { Find intersection of Txi and Tyi }
        ti_1:=T[i]+a__;

        If (T[i]+a__>=T_min) and (T[i]+a__<=T_max) Then ti_1:=T[i]+a__;

        If (T_min>T_max)or((T[i]+a__)<T_min)or((T[i]+a__)>T_max) Then
        begin
            If (Tx_min>=Ty_min)and(Tx_max<=Ty_max) Then
            { Txi is subset of Tyi } a_ := a_x_i else
            If (Ty_min>=Tx_min)and(Ty_max<=Tx_max) Then
            { Tyi is subset of Txi } a_ := a_y_i else
                                     a_ := (a_x_i + a_y_i)/2;
            ti_1 := T[i] + a_;
        end;
        T[i+1]:=ti_1;
    end;
    T[N-1] := T[N-3]+1;
End;

Procedure     MakeSpline1(l4:real; Var S1,S2:Arr2);
Var     Temp   : Real;
        I,J,Z  : Integer;
        FFF    : Text;
Begin
    MakeT2(l4);
    MakeDelta(N,T,X,Y);

    Spline (N, T, X, Del_x, del_x[0], del_x[N-1], mi);
    mi_for_x := mi;
```

```
Spline (N, T, Y, Del_y, del_y[0], del_y[N-1], mi);
mi_for_y := mi;


N1 := 10*(N-1);


for j := 0 to N-2 do
begin
        Temp := (T[J+1]-T[J])/10;
        for i:= 1 to 10 do
        begin
                U := T[j]+(i-1)*Temp;
                Z := j*10+i-1;
                S1[Z] := Seval(N, U, qi[j], T, X, mi_for_x);
                S2[Z] := Seval(N, U, qi[j], T, Y, mi_for_y);
                P[Z]  := U;
        end;
end;
U := T[N-1];
S1[N1] := Seval(N, U, qi[N-2], T, X, mi_for_x);
S2[N1] := Seval(N, U, qi[N-2], T, Y, mi_for_y);
P[N1]  := U;


 Assign(FFF,'h:\Prs3.DAT');
 Rewrite(FFF);
 For I:= 0 to N1 do
         Writeln(FFF,P[I],'  ',S1[I]
 Close(FFF);


 Assign(FFF,'h:\Prs4.DAT');
 Rewrite(FFF);
 For I:= 0 to N1 do
```

```
                Writeln(FFF,P[I],'  ',S2[I]);
        Close(FFF);
End;
                *)
BEGIN {main program}
        ClrScr;
        Write('Input l1 : '); Readln(l1);
        {Write('Input l2 : '); Readln(l2);}
        {Write('Input l3 : '); Readln(l3);}
        {Write('Input l4 : '); Readln(l4);}

        input_data;

        input_Tension;

        del_x[0]     := -1;
        del_x[N-1]  := -1;
        del_y[0]     :=  0;
        del_y[N-1]  := 21;
        MakeSpline(l1,X1,Y1);

        {del_x[0]    := -1;
        del_x[N-1]  := -1;
        del_y[0]     :=  0;
        del_y[N-1]  := 21;
        MakeSpline(l2,X2,Y2);}

        {del_x[0]     := -1;
        del_x[N-1]   := -1;
        del_y[0]      :=  0;
        del_y[N-1]   := 21;
        MakeSpline(l3,X3,Y3);}

        {input_Tension1;

        del_x[0]     := -1;
        del_x[N-1]  := -1;
        del_y[0]     :=  0;
        del_y[N-1]  := 21;
        MakeSpline1(l4,X4,Y4);}

        Graphics(N1,X1,Y1,X2,Y2,X3,Y3,X4,Y4);
END. {Program}
```

Biography

Biography

Captain Opas Udomsomboonpon was born on 11th December 1960 in Bangkok. He went to study in the Department of Physics, Faculty of Science, at Ramkhamhaeng University, where he graduated with a B.Sc degree Physics in 1985. After that he studied Aviation at the Army Aviation Center, Lopburi, and later became aviator at the Third Infantry Division Aviation Company, Nakornrachasima. He now works as an aviation teacher and also teaches mathematics to Privates in the Third Infantry Division. In 1996, he decided to study for a Master's degree the mathematics and began enrol in the School of Mathematics, Institute of Science, Suranaree University of Technology.

At present, he is the Deputy Battery Commander of the Third Infantry Division Aviation Company.