วิธีผลต่างสืบเนื่องสำหรับการประมาณค่าในช่วงด้วยสไปลน์ชนิดคงรูป

นายอนิรุธ ลวดทรง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต

สาขาวิชาคณิตศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีสุรนารี

ปีการศึกษา 2545

ISBN 974-533-181-3

FINITE-DIFFERENCE METHOD FOR SHAPE PRESERVING SPLINE INTERPOLATION

Mr. Anirut Luadsong

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in Applied Mathematics
Suranaree University of Technology
Academic Year 2002
ISBN 974-533-181-3

FINITE-DIFFERENCE METHOD FOR SHAPE PRESERVING SPLINE INTERPOLATION

Suranaree University of Technology has approved this thesis submitted in partial fulfillment of the requirements for a Doctoral Degree

อนิรุธ ลวดทรง: วิธีผลต่างสืบเนื่องสำหรับการประมาณค่าในช่วงด้วยสไปลน์ชนิดคงรูป (FINITE-DIFFERENCE METHOD FOR SHAPE PRESERVING SPLINE INTERPOLATION)
อ. ที่ปรึกษา: PROF. DR. BORIS I. KVASOV, 141 หน้า
ISBN 974-533-181-3

วิทยานิพนธ์ฉบับนี้เสนองานวิจัยเกี่ยวกับวิธีการหาผลเฉลยของปัญหาการประมาณค่าในช่วงด้วยสไปลน์ชนิดคงรูป ซึ่งศึกษาเกี่ยวกับการประมาณค่าของชุดข้อมูล เพื่อหาเส้นโค้งและพื้นผิว ที่ยังคงรักษาสมบัติทางเรขาคณิตของชุดข้อมูลเหล่านั้น เช่น ความเป็นค่าบวก การเพิ่มขึ้นหรือการลดลงในทิศทางเดียว ความนูน เป็นต้น งานวิจัยนี้ได้ศึกษาและพัฒนาวิธีการใหม่ด้วยปัญหาค่าขอบชนิดหลายจุดเชิงอนุพันธ์ หรือใช้ชื่อย่อว่า DMBVP ในการหาผลเฉลยของปัญหาการประมาณค่าในช่วงด้วยสไปลน์ชนิดคงรูปในกรณี 1 มิติ และ 2 มิติ วิธีการใหม่นี้ช่วยเพิ่มประสิทธิภาพการคำนวณด้วยคอมพิวเตอร์ สำหรับระเบียบวิธีเชิงตัวเลขของ DMBVP ได้ใช้ตัวดำเนินการผลต่างสืบเนื่องในการประมาณตัวดำเนินการเชิงอนุพันธ์ ผลลัพธ์ที่ได้คือ ระบบสมการเชิงเส้นซึ่งมีเมตริกซ์สัมประสิทธิ์ที่มีโครงสร้างเป็นรูปแบบอย่างชัดเจนและเป็นบวกแน่นอน ทำให้การหาผลเฉลยของระบบสมการเชิงเส้นนี้สามารถทำได้ทั้งวิธีทางตรงและวิธีการทำซ้ำ สำหรับวิธีทางตรงใช้วิธีการขจัดของเกาส์ ส่วนวิธีการทำซ้ำใช้วิธีผ่อนปรนเกินสืบเนื่อง (successive over-relaxation) และวิธีขั้นเศษส่วน (fractional steps) ระเบียบวิธีผลต่างสืบเนื่องในวิธีขั้นเศษส่วนของ DMBVP ช่วยเพิ่มประสิทธิภาพในการคำนวณเชิงตัวเลข และสามารถใช้งานกับคอมพิวเตอร์เชิงขนานได้ด้วย สำหรับการออกแบบโปรแกรมคอมพิวเตอร์เชิงขนานใช้การติดต่อประสานงานกันแบบส่งผ่านข้อมูล หรือใช้ชื่อย่อว่า MPI

ผลที่ได้จากงานวิจัยของวิทยานิพนธ์นี้สามารถนำไปประยุกต์กับปัญหาด้านการออกแบบกราฟฟิกด้วยคอมพิวเตอร์ ในด้านการออกแบบผลิตภัณฑ์ และในด้านอุตสาหกรรม เช่น การออกแบบรถยนต์ เครื่องบิน เรือ เป็นต้น นอกจากนี้แล้วยังสามารถนำไปใช้แก้ปัญหาทางวิทยาศาสตร์หลายสาขา การวิจัยทางการแพทย์ การวิเคราะห์ภาพ การเพิ่มความละเอียดของภาพในระบบโทรทัศน์ การทำแผนที่ และ อุตสาหกรรมภาพยนตร์ เป็นต้น

สาขาวิชาคณิตศาสตร์	ลายมือชื่อนักศึกษา....................................
ปีการศึกษา 2545	ลายมือชื่ออาจารย์ที่ปรึกษา..............................
	ลายมือชื่ออาจารย์ที่ปรึกษาร่วม..........................
	ลายมือชื่ออาจารย์ที่ปรึกษาร่วม..........................

ANIRUT LUADSONG : FINITE-DIFFERENCE METHOD FOR SHAPE PRESERVING SPLINE INTERPOLATION
THESIS ADVISOR: PROF. DR. BORIS I. KVASOV, Ph.D. 141 PP.
ISBN 974-533-181-3

SHAPE PRESERVING / CAGD / DMBVP / SOR / MPI / FINITE-DIFFERENCE / NUMERICAL ALGORITHM / PARALLEL ALGORITHM / BLOCK GAUSSIAN ELIMINATION / FRACTIONAL STEPS

This thesis addresses the solution of shape preserving spline interpolation problem which deals with the approximation of curves and surfaces with retaining certain shape properties such as positivity, monotonicity, convexity, etc. Based on formulation of this problem as a differential multipoint boundary value problem (DMBVP), we develop a new technique for solving shape preserving interpolation problem. This new approach has substantial computational advantages. We consider both 1-D and 2-D cases. For the numerical treatment of DMBVP, we replace the differential operator by its difference approximation. This gives us a system of linear equations with a positive definite matrix of a special structure which can be efficiently solved by direct and iterative methods. As a direct method we consider Gaussian elimination. For iterative solution of the obtained linear system, we apply successive over-relaxation method. Finite-difference schemes in fractional steps also prove their efficiency in the numerical treatment of our DMBVP. The developed algorithms can efficiently work on parallel-processing computers. The message-passing interface (MPI) is applied in the programming code of parallel version.

The results of the thesis can be used in many applied problems and first of all in CAGD (computer description and numerical modelling of aircraft surfaces, bodies of ships and cars, complex details of engines, etc.). The developed algorithms can also be applied in many fields of science (mathematical models in mechanics and physics, computer description of geological and physical phenomena). New applications include computer vision and inspection of manufactured parts, medical research (software for digital diagnostic equipment), image analysis, high resolution TV systems, cartography, the film industry, etc.

School of Mathematics	Student ____________
Academic Year 2002	Advisor ____________
	Co-Advisor ____________
	Co-Advisor ____________

Acknowledgements

I would like to express my sincere gratitude to my advisor Prof. Dr. Boris I. Kvasov, School of Mathematics, Institute of Science, Suranaree University of Technology for his kind guidance and support throughout the course of this research. I have had working under his invaluable supervision for about four years and learned a lot of unique experience for research which I will cherish throughout my life.

I am also very grateful to thank my co-advisor Dr. Markus Hegland, Mathematical Sciences Institute, Australian National University, the Chairman of the School of Mathematics of Suranaree University of Technology, Assoc. Prof. Dr. Prapasri Asawakun and all the lecturers who taught and helped me during my study in Suranaree University of Technology. They are Assoc. Prof. Dr. Suwan Tangmanee, Assoc. Prof. Dr. Pairote Sattayatham, Prof. Dr. Sergey V. Meleshko, Assoc. Prof. Dr. Nikolay P. Moshkin, Asst. Prof. Dr. Eckart Schulz, and Asst. Prof. Dr. Arjuna Chaiyasena for their help, advice and support.

I want to express my appreciation to all of my colleagues at Suranaree University of Technology, Australian National University, and King Mongkut's University of Technology Thonburi for their invaluable professional guidance and friendly encouragement.

In addition, I wish to press my special thanks to Thailand's Ministry of University Affairs for offering the scholarship which enabled me to continue my advanced studies in Suranaree University of Technology, and also at Mathematical Sciences Institute, Australian National University where I did my research.

Last but not least, I would like also to thank my parents, my relatives and my friends for their understanding, supporting and encouraging over the years of my study.

Anirut Luadsong

Contents

Contents (Continued)

List of Tables

List of Figures

List of Figures (Continued)

Figure **Page**

Chapter I

Introduction

Spline functions constitute the main tool in computer aided geometric design (CAGD for short) which is used for the approximation and representation of curves and surfaces that arises when these objects have to be processed by a computer. Applications of CAGD include not only geometric design of different products like car bodies, ship hulls, airplane fuselages, etc., but also computer vision and inspection of manufactured parts, medical research (software for digital diagnostic equipment), image analysis, high resolution TV systems, cartography, etc. In the majority of these applications, it is important to construct curves and surfaces which preserve certain properties of the data. For example, we may want the surface to be positive, monotone, or convex in some sense. Standard methods of spline functions do not retain these properties of the data. This problem is known as *the problem of shape preserving interpolation*. The purpose of this thesis is to develop new efficient methods for solving this problem.

In the theory of splines mainly two approaches are used: algebraic and variational. In the first approach (L. L. Schumaker (1981), Yu. S. Zavyalov *et al.* (1980)), splines are understood as smooth piecewise defined functions. In the second approach (P. J. Laurent (1972)), splines are solutions of some minimization problems for quadratic functionals with restrictions of equality and/or inequality type. But the third approach is also known (N. N. Janenko and B. I. Kvasov (1970)), where splines are defined as solutions of differential multipoint boundary value problems. In some important cases all three approaches give the same solutions. However, the third approach has substantial computational advantages. We develop this approach on the examples of hyperbolic and thin plate tension splines. It can be generalized to smoothing splines and even to scattered data in a straightforward manner.

Research on constructing shape-preserving interpolatory functions started with the spline in tension of D. G. Schweikert (1966) where exponential splines were used as approximants. This was followed by the work of H. Späth (1974, 1990), G. M. Nielson (1974), S. Pruess (1976, 1979), and C. De Boor (1978) with various exponential and cubic spline interpolants containing "tension parameter" 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. D. McAllister et al. (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 R. Delbourgo and J. A. Gregory (1985) and J. A. Gregory and R. Delbourgo (1982)) although these are usually only C^1 or they are intended for strictly monotone or strictly convex data.

In 1980 F. N. Fritsch and R. E. 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 (F. N. Fritsch and J. Butland (1984)). R. J. 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. P. Costantini (1987) also has families of shape-preserving interpolants based on Bernstein polynomials; these are very simple to use but are comonotone, i.e. the spline on the ith 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 shape preserving spline interpolants are found in W. Burmeister *et al.* (1985), and W. J. Schmidt and W. Hess (1987). There is the work of R. Dougherty *et al.* (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.

Generalized tension splines and GB-splines are widely used in solving problems of shape preserving interpolation (e.g., see J. C. Clement (1990), B. I. Kvasov (1996a, 2000), B. J. McCartin (1990), R. J. Renka (1987), N. S. Sapidis and P. D. Kaklis (1988) and N. S. Sapidis *et al.* (1988)). By introducing various parameters into the spline structure, one can preserve various characteristics of the initial data, including positivity, monotonicity, convexity, as well as linear and planar sections. The major idea is to find a reasonable compromise between a conventional cubic spline and piecewise linear interpolation of the data. The graph of a shape preserving spline should lie closest to the graph of a cubic interpolating spline, thus providing smooth curve with the best possible order of approximation specific to a cubic spline, while still retaining selected geometric properties of the data. Here, the main challenge is to develop algorithms that choose parameters automatically.

Introduced by D. G. Schweikert (1966), for solving shape preserving interpolation problem hyperbolic tension splines are still very popular (P. E. Koch and T. Lyche (1989, 1993), G. M. Nielson (1984), S. Pruess (1976), R. J. Renka (1987), P. Rentrop (1980), N. S. Sapidis and G. Farin (1990)). Unfortunately, it is difficult to work with hyperbolic tension splines for very small or very large values of the tension parameters. For this reason, in spite of the presence of refined algorithms for their calculation (P. Rentrop (1980)), hyperbolic tension splines were forced out by rational splines (R. Delbourgo and J. A. Gregory (1985), J. C. Fiorot and

P. Jeannin (1992), B. I. Kvasov (1996b) and H. Späth (1990)) in practical calculations. Recently, in P. Costantini *et al.* (1999) a difference method for constructing shape preserving hyperbolic tension splines as solutions of 1-D DMBVP was developed. Such an approach permits to avoid the computation of hyperbolic functions and has substantial other advantages. However, the extension of a mesh solution will be a discrete hyperbolic tension spline.

Discrete polynomial splines have been studied extensively. They were introduced by O. L. Mangasarian and L. L. Schumaker (1971) as solutions to certain minimization problems involving differences instead of derivatives. They are connected to best summation formulas (O. L. Mangasarian and L. L. Schumaker (1973)) and have been used by M. A. Malcolm (1977) for the computation of nonlinear splines by iteration. Approximation properties of discrete splines have been studied by T. Lyche (1976). While discrete polynomial splines are currently attracting widespread research interest (A. A. Melkman (1996), K. M. Mørken (1996), S. S. Rana and Y. P. Dubey (1996)), discrete tension splines have been less studied. The only results we know of regarding this topic are discrete exponential Box-splines (W. Dahmen and C. A. Micchelli (1989), A. Ron (1988)) and are therefore related to uniform partitions.

The content of the thesis is as follows. Chapters 2 and 3 present the basic tools of classical and shape preserving spline interpolation. This material together with Chapter 5 is important for understanding the two main chapters, 4 and 6.

Chapter 2 describes standard methods of cubic spline interpolation. Such splines remain very important tools in a multitude of applications involving curve fitting and design. The main reason for this is their excellent approximation properties. Cubic splines are easy to store, manipulate, and evaluate on computer. However, cubic interpolation splines do not ever retain the shape preserving properties of the data.

In Chapter 3 we consider problem of shape preserving spline interpolation and describe local and global algorithms of its solution. This chapter begins with the necessary and sufficient conditions which guarantee that a C^1 local cubic spline preserves the monotonicity of the data. We show that error of approximation remains small under the proposed algorithms. This follows by sufficient conditions of monotonicity and convexity for conventional C^2 global cubic interpolating splines which are formulated in terms of restrictions on the strictly monotone or strictly convex initial data. If these conditions are not satisfied then we provide algorithms of shape preserving interpolation of the same data by C^2 generalized tension splines with automatic choice of shape control parameters.

In Chapter 4 we formulate the 1-D problem of shape preserving spline interpolation as 1-D DMBVP. For the numerical treatment of this DMBVP we replace the differential operator by its difference approximation. This gives us a system of linear equations with a positive pentadiagonal matrix which can be solved efficiently by classical Cholesky factorization. It is substantially cheaper than performing calculations by the standard algorithm (P. Rentrop (1980)), which involves the solution of only a 3-diagonal system, but with hyperbolic coefficients, and then requires for interpolation the evaluation of hyperbolic functions with a much greater number of computations than the solution of our pentadiagonal system. We estimate the condition number of the matrix of the system of linear equations and show that it does not depend on the number of the data points but substantially depends on the refinement. We construct an extension of a mesh solution as a discrete hyperbolic tension spline and show how to split a large 5-diagonal system into a sequence of 3-diagonal systems which can be solved numerically by parallel computer. We find an upper bound for the distance between a discrete hyperbolic tension spline and the corresponding continuous spline interpolating the same set of data and having the same end conditions. Some practical aspects related to our discrete hyperbolic spline interpolants are discussed. We propose a possible generalization for a particular nonuniform subdivision of the main mesh and give some graphical examples to illustrate the main features of our difference method.

Chapter 5 investigates the tool of discrete generalized tension splines at which we naturally arrived at in Chapter 4. We prove sufficient conditions for their existence and uniqueness and construct a minimum length local support basis (whose elements are denoted as discrete GB-splines). Properties of GB-splines are discussed and the local approximation by discrete GB-splines of a given function from its samples is considered. We derive recurrence formulae for calculation with discrete GB-splines and study the properties of discrete GB-spline series. Given examples of defining functions conform to the sufficiency conditions derived earlier in this chapter.

In Chapter 6 we formulate the 2-D problem of shape preserving spline interpolation as 2-D DMBVP. For the numerical treatment of this DMBVP we replace the differential operator by its difference approximation. This gives us a system of linear equations with the matrix of a special structure. We show that this matrix is positive definite. So, we can solve efficiently this system of linear equations by direct or iterative methods. As a direct method, we suggest to consider a block Gaussian elimination. For iterative solution of the obtained linear system, we apply successive over-relaxation (SOR) method. Finite-difference schemes in fractional steps also prove their efficiency in the numerical treatment of our DMBVP. We give some graphical examples and discuss competing numerical algorithms.

In Chapter 7 we summarize the main results of the thesis emphasizing on the advantages of our finite difference approach. The programming codes for all algorithms were designed in FORTRAN 90 and their description is given in the appendix. For parallel version the *Massage-Passing Interface* (MPI for short), which is a library of functions and macros that can be used in C, FORTRAN, and C++ programs, was applied. This programming codes have been tested on different versions of Windows for PC and some Unix servers, e.g., the SUT's mathematical server and the ANU's supercomputer at the national facility of the Australian Partnership for Advanced Computing (APAC).

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 this thesis.

2.1 Cubic Interpolating Splines

Suppose that we want to interpolate a function f given by the data $(x_i, f_i),\ i = 0, \dots, N$, where $f_i = f(x_i)$ and the points x_i form 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)

$$S(x) \equiv S_i(x) = a_{i,0} + a_{i,1}(x - x_i) + a_{i,2}(x - x_i)^2 + a_{i,3}(x - x_i)^3$$

$$\text{for} \quad x \in [x_i, x_{i+1}], \quad i = 0, \dots, N-1;$$

(ii) consecutive polynomials are smoothly adjusted

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

(iii) the interpolation conditions are satisfied

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

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 $x_i,\ i = 1, \dots, N-1$, where the polynomials are matched, are called the *knots* of the spline, and can be coincided with the interpolation nodes. The knots of the spline can also have different multiplicities depending on the number of the adjusted derivations. 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 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 *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:
   $S'(x_0) = f'_0$ and $S'(x_N) = f'_N$;
2. Second derivative endpoint conditions:
   $S''(x_0) = f''_0$ 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, $S'''(x_i - 0) = S'''(x_i + 0),\ i = 1, N-1$.

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 Cubic Interpolating Spline

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

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

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

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

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:

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

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 $[x_i, x_{i+1}]$:

$$\begin{aligned} S_i(x) = M_i \frac{(x_{i+1} - x)^3}{6h_i} &+ M_{i+1} \frac{(x - x_i)^3}{6h_i} + \left(f_i - M_i \frac{h_i^2}{6} \right) \frac{x_{i+1} - x}{h_i} \\ &+ \left(f_{i+1} - M_{i+1} \frac{h_i^2}{6} \right) \frac{x - x_i}{h_i}. \end{aligned} \tag{2.3}$$

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

$$\begin{aligned} S_i'(x) = -M_i \frac{(x_{i+1} - x)^2}{2h_i} &+ M_{i+1} \frac{(x - x_i)^2}{2h_i} - \left(\frac{f_i}{h_i} - M_i \frac{h_i}{6} \right) \\ &+ \left(\frac{f_{i+1}}{h_i} - M_{i+1} \frac{h_i}{6} \right). \end{aligned} \tag{2.4}$$

Evaluating (2.4) at x_i and simplifying the result yields

$$S_i'(x_i + 0) = -M_i \frac{h_i}{3} - M_{i+1} \frac{h_i}{6} + f[x_i, x_{i+1}],$$

where $\quad f[x_i, x_{i+1}] = (f_{i+1} - f_i)/h_i$.

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

$$S_{i-1}'(x_i - 0) = M_{i-1} \frac{h_{i-1}}{6} + M_i \frac{h_{i-1}}{3} + f[x_{i-1}, x_i].$$

As $\ S_{i-1}'(x_i - 0) = S_i'(x_i + 0), \quad i = 1, \ldots, N - 1,\ $ we obtain

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

where $\quad \delta_i f = f[x_i, x_{i+1}] - f[x_{i-1}, x_i]$.

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. $2M_0 + M_1 = \dfrac{6}{h_0}(f[x_0,x_1] - f_0'),$
   $M_{N-1} + 2M_N = \dfrac{6}{h_{N-1}}(f_N' - f[x_{N-1},x_N]);$
2. $M_0 = f_0''$ and $M_N = f_N''$;
3. $f_{N+i} = f_i,\ M_{N+i} = M_i,\ h_{N+1} = h_i$ for all i;
4. $\dfrac{M_{i+1} - M_i}{h_i} = \dfrac{M_i - M_{i-1}}{h_{i-1}},\ i = 1, N-1.$

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

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

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

where

$$\bar{b} = [6(f[x_0,x_1] - f_0'), 6\delta_1, 6\delta_2 f, \dots, 6(f_N' - f[x_{N-1},x_N])]^T$$

and T is the transposition operator.

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

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \cdots & 0 \\ 0 & h_1 & 2(h_1+h_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_N \end{bmatrix} = \begin{bmatrix} f_0'' \\ 6\delta_1 f \\ 6\delta_2 f \\ \vdots \\ f_N'' \end{bmatrix}. \qquad (2.7)$$

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

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

Because $f_{N+i} = f_i,\ M_{N+i} = M_i,\ i = 0,1$, and $h_N = h_0$ then one obtains

$$f[x_N, x_{N+1}] = \frac{f_{N+1} - f_N}{h_N} = \frac{f_1 - f_0}{h_0} = f[x_0,x_1]$$

and equation (2.8) takes the form

$$h_0 M_1 + h_{N-1}M_{N-1} + 2(h_{N-1} + h_0)M_N = 6(f[x_0,x_1] - f[x_{N-1},x_N]).$$

We arrive at the following system of linear equations

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

where

$$\bar{b} = [6\delta_1 f, 6\delta_2 f, \ldots, 6(f[x_0,x_1] - f[x_{N-1},x_N])]^T.$$

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

$$\begin{bmatrix} h_1 & -(h_0+h_1) & h_0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & h_{N-2} & 2(h_{N-2}+h_{N-1}) & h_{N-1} \\ 0 & \cdots & h_{N-1} & -(h_{N-2}+h_{N-1}) & h_{N-2} \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{N-1} \\ M_N \end{bmatrix} = \bar{b}, \quad (2.10)$$

where

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

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 unknown M_0 and M_N from system (2.10). If one subtracts from the second equation of system (2.10), multiplied by h_1, 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 last equation of system (2.10), multiplied by h_{N-2} the last equation multiplied by h_{N-1}, then the resulting equation will be

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

We arrive at the following system of linear equations with tridiagonal matrix

$$\begin{bmatrix} (h_0+2h_1) & (h_1-h_0) & \cdots & 0 \\ h_1 & 2(h_1+h_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & h_{N-2}-h_{N-1} & 2h_{N-2}+h_{N-1} \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{N-1} \end{bmatrix} = \bar{b}, \quad (2.11)$$

where

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

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

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 diagonal dominance if the following conditions are fulfilled*

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

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 the 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

$$|a_{kk}||x_k| \leq \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{kj}||x_j| \leq |x_k| \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{kj}|.$$

From here

$$|a_{kk}| \leq \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{kj}|,$$

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)

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

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,

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

Now using Theorem 2.1 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 particularly efficient 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 divides 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, system (2.13) is transformed to the 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 ith row, we have to multiply the $(i-1)$st row by a_i and subtract it from the ith row. As a result, the ith 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 ith row by the coefficient $b_i - a_i p_{i-1}$. As a result, in the final form of the ith row one obtains the following formulae for the elements p_i and q_i:

$$\begin{aligned} p_i &= \frac{c_i}{b_i - a_i p_{i-1}}, \quad i = 2, \ldots, n-1, \quad p_1 = \frac{c_1}{b_1}, \\ q_i &= \frac{d_i - a_i q_{i-1}}{b_i - a_i p_{i-1}}, \quad i = 2, \ldots, n, \qquad q_1 = \frac{d_1}{b_1}. \end{aligned} \tag{2.14}$$

Proceeding in this way one arrives at the system

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

Now using the back substitution one can compute the unknowns x_i:

$$\begin{aligned} x_n &= q_n, \\ x_i &= -p_i x_{i+1} + q_i, \quad i = n-1, \ldots, 1. \end{aligned} \tag{2.15}$$

2.6 Correctness and Stability of Gaussian Elimination

Let us consider the correctness and stability of the calculations in the above discribed 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 formulae (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 $|p_1| = |c_1|/|b_1| < 1$. Let us suppose by induction that $|p_j| < 1$, $j = 1, \dots, i-1$. Then using formula (2.14) one obtains

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

that is, $|p_i| < 1$ for all i.

As

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

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 $\bar{x}_i = x_i + \varepsilon_i$, $i = 1, \dots, n$, where ε_i is the round-off error at the ith step. Then according to (2.15) one obtains

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

By subtracting formula (2.15) from this equation one gets

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

or

$$|\varepsilon_i| = |p_i|\,|\varepsilon_{i+1}| < |\varepsilon_{i+1}|, \quad i = n-1, \dots, 1,$$

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 First Derivative Algorithm

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

$$\begin{aligned} S(x) \equiv S_i(x) &= f_i(1-t) + f_{i+1}t + t(1-t)h_i[(m_i - f[x_i, x_{i+1}])(1-t) \\ &\quad + (f[x_i, x_{i+1}] - m_{i+1})t], \end{aligned} \tag{2.17}$$

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

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

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

$$\begin{aligned} S_{i-1}''(x_i - 0) &= \frac{2}{h_{i-1}}(-3f[x_{i-1}, x_i] + m_{i-1} + 2m_i), \\ S_i''(x_i + 0) &= \frac{2}{h_i}(3f[x_i, x_{i+1}] - 2m_i - m_{i+1}). \end{aligned}$$

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

$$\begin{aligned} \frac{1}{h_{i-1}}m_{i-1} + 2\Big(\frac{1}{h_{i-1}} + \frac{1}{h_i}\Big)m_i + \frac{1}{h_i}m_{i+1} &= 3\Big(\frac{f[x_{i-1}, x_i]}{h_{i-1}} + \frac{f[x_i, x_{i+1}]}{h_i}\Big), \\ & i = 1, \ldots, N-1. \end{aligned} \tag{2.19}$$

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 2.1. Using formulae (2.17) and (2.18) one can rewrite these endpoint conditions in the form:

1. First derivative endpoint conditions:
$$m_0 = f_0' \quad \text{and} \quad m_N = f_N'; \tag{2.20}$$
2. Second derivative endpoint conditions:
$$\begin{aligned} 2m_0 + m_1 &= 3f[x_0, x_1] - f_0''h_0/2, \\ m_{N-1} + 2m_N &= f_N''h_{N-1}/2 + 3f[x_{N-1}, x_N]; \end{aligned} \tag{2.21}$$
3. Periodic endpoint conditions:
$$f_{N+i} = f_i, \ m_{N+i} = m_i, \ h_{N+i} = h_i \quad \text{for all } i; \tag{2.22}$$
4. "Not-a-knot" endpoint conditions where adjacent polynomials nearest to the endpoints of the interval $[a, b]$ coincide:
$$S_0(x) \equiv S_1(x) \quad \text{and} \quad S_{N-1}(x) \equiv S_N(x),$$
that is,
$$S'''(x_i - 0) = S'''(x_i + 0), \quad i = 1, N-1,$$
or
$$\frac{m_i + m_{i+1}}{h_i^2} - \frac{m_{i-1} + m_i}{h_{i-1}^2} = \frac{f[x_i, x_{i+1}]}{h_i^2} - \frac{f[x_{i-1}, x_i]}{h_{i-1}^2}, \quad i = 1, N-1. \tag{2.23}$$

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

$$\frac{1}{h_0}m_1 + \frac{1}{h_{N-1}}m_{N-1} + 2\Big(\frac{1}{h_0} + \frac{1}{h_{N-1}}\Big)m_N = 3\Big(\frac{f[x_0, x_1]}{h_0} + \frac{f[x_{N-1}, x_N]}{h_{N-1}}\Big).$$

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

$$\begin{aligned}\Big(\frac{1}{h_0} + \frac{1}{h_1}\Big)m_1 + \frac{1}{h_1}m_2 &= 2\lambda_1\frac{f[x_0, x_1]}{h_0} + (1 + 2\lambda_1)\frac{f[x_1, x_2]}{h_1},\\ \frac{m_{N-2}}{h_{N-2}} + \Big(\frac{1}{h_{N-2}} + \frac{1}{h_{N-1}}\Big)m_{N-1} &= (1 + 2\mu_{N-1})\frac{f[x_{N-2}, x_{N-1}]}{h_{N-2}}\\ &\quad +2\mu_{N-1}\frac{f[x_{N-1}, x_N]}{h_{N-1}},\end{aligned}$$

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 unknowns 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

$$S_i(x) = f_i + (x - x_i)(f[x_i, x_{i+1}] + (x - x_{i+1})(a_i + (x - x_i)b_i)),$$

where

$$a_i = \frac{f[x_i, x_{i+1}] - m_i}{h_i} \quad \text{and} \quad b_i = \frac{m_i - 2f[x_i, x_{i+1}] + m_{i+1}}{h_i^2}.$$

2.8 Graphical Examples

As the first example we consider interpolating the function

$$f(x) = 1 - \frac{exp(100x) - 1}{exp(100) - 1}, \quad x \in [0, 1]$$

on the uniform mesh: $x_i = i/N$, $i = 0, \ldots, N$, where $N = 10, 40$. In Figures 2.1–2.5 the dashed and solid lines show the graphs of the cubic spline S of the C^2 class and the graph of function f, respectively. In both cases the boundary conditions $S'(0) = f'(0) = 0,\ S'(1) = f'(1) = -100$ are used. The graph of the cubic spline S approaches the graph of the function f when the number of interpolation points, $N + 1$, is increased.

In second example we consider interpolating the function

$$f(x) = e^{-x^4}, \qquad x \in [-4, 4]$$

on the uniform mesh $x_i = -4 + 8i/N,\ i = 0, \ldots, N$, where $N = 10, 16, 20$. In all Figures 2.3–2.5 we used the boundary conditions $S'(-4) = f'(-4) = 0,\ S'(4) = f'(4) = 0$. The behaviour of the cubic spline S is similar to that of the first example. The graph of the cubic spline S approaches the graph of the function f and preserves the monotonicity and convexity of the interpolation points, $N + 1$, is increased.

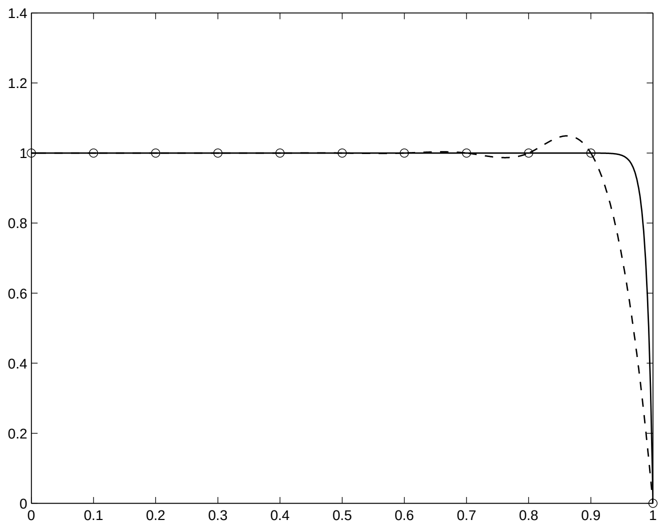

Figure 2.1: The graphs of the cubic spline S on the uniform mesh for $N = 10$ and of the function $f(x) = 1 - \frac{exp(100x)-1}{exp(100)-1}, \quad x \in [0, 1]$.

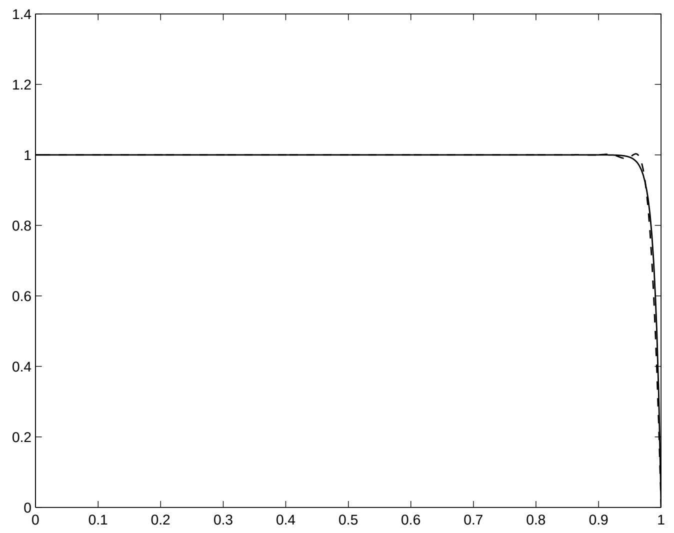

Figure 2.2: The same as Figure 2.1 but for $N = 40$.

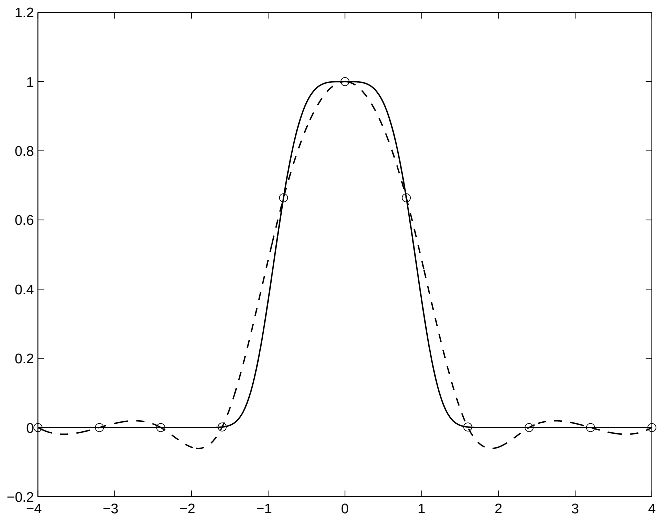

Figure 2.3: The graphs of the cubic spline S on the uniform mesh for $N = 10$ and of the function $f(x) = e^{-x^4}, \quad x \in [-4, 4]$.

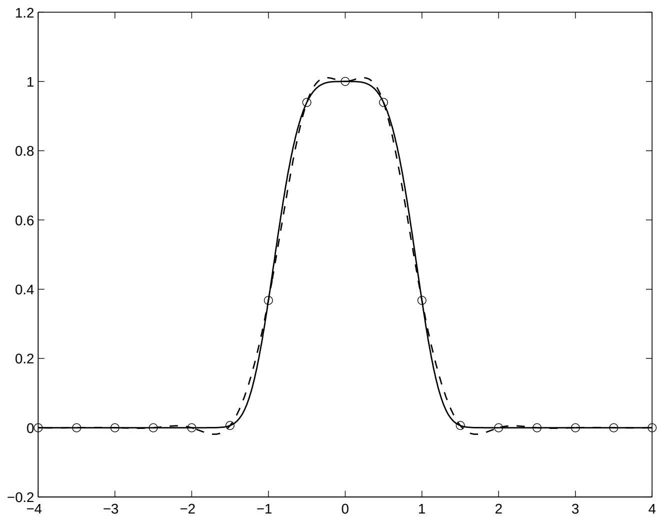

Figure 2.4: The same as Figure 2.3 but for $N = 16$.

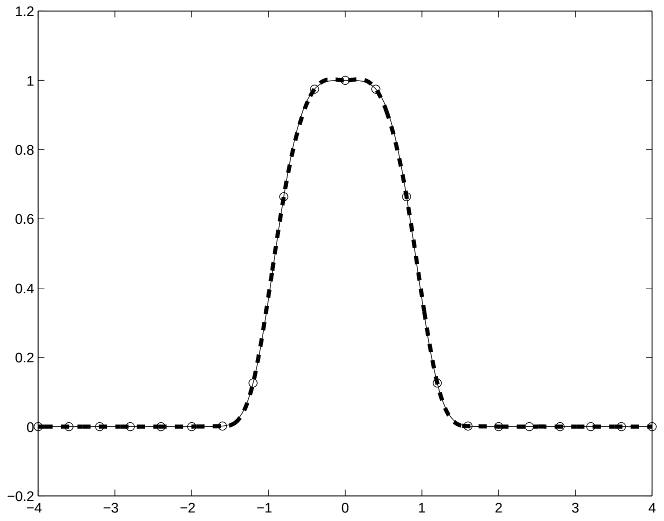

Figure 2.5: The same as Figure 2.3 but for $N = 20$.

Chapter III

Shape Preserving 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 The Problem of Shape Preserving Spline Interpolation

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

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

by spline functions S which preserve the shape of the data (x_i, f_i), $i = 0, \dots, N$. 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 is 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 $||f - S||$ remains small under the proposed algorithms which describe S.

We introduce the notation for the first two divided differences

$$f[x_i, x_{i+1}] = (f_{i+1} - f_i)/h_i, \quad h_i = x_{i+1} - x_i, \quad i = 0, \dots, N-1,$$

$$\delta_i f = f[x_i, x_{i+1}] - f[x_{i-1}, x_i], \quad i = 1, \dots, N-1.$$

The data f_i is said to be *monotone* if

$$f[x_i, x_{i+1}] \geq 0, \quad i = 0, \dots, N-1, \tag{3.2}$$

and *convex* if

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

The problem of monotone (convex) spline interpolation consists in 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)

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

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

F. N. Fritsch and R. E. Carlson (1980) have proved necessary and sufficient monotonicity conditions for cubic splines of class C^1. They yield the following

Lemma 3.1. *If $f[x_i, x_{i+1}] \geq 0$ and*

$$0 \leq m_j \leq 3f[x_i, x_{i+1}], \quad j = i, i+1, \tag{3.5}$$

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

Proof: This result can easily be obtained without appealing to F. N. Fritsch and R. E. Carlson (1980). In fact, from (3.4) we have

$$\begin{aligned} S_i'(x) &= s(t, m_i, m_{i+1}) = 6t(1-t)f[x_i, x_{i+1}] \\ &\quad + (1 - 4t + 3t^2)m_i + (-2t + 3t^2)m_{i+1}. \end{aligned} \tag{3.6}$$

The function $s(t, m_i, m_{i+1})$ is linear in the 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: $s(t,0,0) \geq 0,\ s(t,\alpha_i,0) \geq 0,\ s(t,0,\alpha_i) \geq 0,\ s(t,\alpha_i,\alpha_i) \geq 0$ for $t \in [0,1]$ and $\alpha_i = 3f[x_i, x_{i+1}]$. It is easy to verify oneself that these inequalities hold; hence, the desired result follows. This proves the lemma. $\square$

3.2 Monotone Matrix. Lemma on Tridiagonal System

Definition 3.1. *A real 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 (L. Collatz (1964)).

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

$$a_{ii} > 0,\ a_{ij} \leq 0\ (j \neq i), \quad i, j = 1, \dots, n,$$

then it is monotone, and the condition $Ay > 0$ (< 0) implies $y > 0$ (< 0).

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

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

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

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

This contradiction shows that A is monotone.

Suppose now $Ay > 0$. Assuming $y_k = 0$ we have

$$\sum_j a_{kj} y_j = \sum_{j \neq k} a_{kj} y_j \leq 0$$

which contradicts the assumption. This proves the lemma.. $\square$

Consider the system

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

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 $z_i \geq 0, \ i = 0, \ldots, N$. 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), where $N > 1$, be such that*

$$b_i > 0, \ i = 0, \ldots, N;$$

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

$$c_0 < \frac{b_0 b_1}{a_1 + c_1}, \quad a_N < \frac{b_{N-1} b_N}{a_{N-1} + c_{N-1}}, \tag{3.9}$$

then this system is nondegenerate. If the inequalities

$$d_i \geq 0, \quad d_i - \frac{a_i d_{i-1}}{b_{i-1}} - \frac{c_i d_{i+1}}{b_{i+1}} \geq 0, \quad i = 0, \ldots, N \tag{3.10}$$

where $a_0 = c_N = d_{-1} = d_{N+1} = 0, \ b_{-1} = b_{N+1} = 1$), hold, the solution of (3.7) is nonnegative:

$$z_i \geq 0, \quad i = 0, \ldots, N. \tag{3.11}$$

Proof: First we consider the case $c_0 \geq 0$ and $a_N \geq 0$. We add to (3.7) the equations $b_i z_i = d_i$, with $b_i = d_i = 1,\ i = -1, N+1$, and suppose that $a_0 = a_{N+1} = c_{-1} = c_{N+1} = 0$. For each $i = 0, \ldots, N$ 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

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

where

$$\begin{aligned} A_i &= \frac{a_{i-1} a_i}{b_{i-1}}, \quad B_i = b_i - \frac{a_i c_{i-1}}{b_{i-1}} - \frac{a_{i+1} c_i}{b_{i+1}}, \\ C_i &= \frac{c_i c_{i+1}}{b_{i+1}}, \quad D_i = d_i - \frac{a_i d_{i-1}}{b_{i-1}} - \frac{c_i d_{i+1}}{b_{i+1}}, \end{aligned}$$

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

$$\begin{aligned} B_1 &> b_1 - \frac{b_1 a_1}{a_1 + c_1} - \frac{c_1 a_2}{b_2} = c_1 \left[\frac{b_1}{a_1 + c_1} - \frac{a_2}{b_2} \right] > 0, \\ B_0 &= b_0 - \frac{a_1 c_0}{b_1} > b_0 - \frac{a_1 b_0}{a_1 + c_1} > 0, \end{aligned}$$

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) holds.

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

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

where

$$\widehat{b}_1 = b_1 - \frac{a_1 c_0}{b_0}, \quad \widehat{d}_1 = d_1 - \frac{a_1 d_0}{b_0},$$

$$\widehat{b}_{N-1} = b_{N-1} - \frac{c_{N-1} a_N}{b_N}, \quad \widehat{d}_{N-1} = d_{N-1} - \frac{c_{N-1} d_N}{b_N}.$$

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 $c_0 \geq 0,\ a_N \geq 0$. Thus, $z_i \geq 0,\ i = 1, \ldots, N-1$. But $z_0 = (d_0 - c_0 z_1)/b_1 > 0$ and, similarly, $z_N \geq 0$. This proves the lemma. $\square$

3.3 Convex Cubic Spline

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

$$S'(x_i) = f'_i, \qquad i = 0, N. \tag{3.14}$$

It was shown in Chapter 2 that values of the second derivative of the spline $M_i = S''(x_i),\ i = 0, \dots, N$ satisfy the system of linear equations (3.6) which can be rewritten in the form

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

where

$$\mu_i = h_{i-1}/(h_{i-1} + h_i), \quad \lambda_i = 1 - \mu_i, \quad d_i = 6f[x_{i-1}, x_i, x_{i+1}],$$
$$d_0 = \frac{6}{h_0}(f[x_0, x_1] - f'_0), \quad d_N = \frac{6}{h_{N-1}}(f'_N - f[x_{N-1}, x_N]).$$

Theorem 3.1. *Let a C^2 cubic spline 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*

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

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

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

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

Remark: Define $\overline{h} = \max_i h_i$. 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 Spline

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

$$S''(x_i) = f''_i, \quad i = 0, N. \tag{3.17}$$

For the 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

$$c_0 = 3f[x_0, x_1] - f_0'' h_0/2, \qquad c_i = 3\lambda_i f[x_{i-1}, x_i] + 3\mu_i f[x_i, x_{i+1}],$$
$$c_N = 3f[x_{N-1}, N] + f_N'' h_{N-1}/2.$$

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

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

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

Proof: 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: $m_i \geq 0,\ i = 0, \dots, N$. This proves the lemma. □

From the fact that $m_i \geq 0,\ i = 0, \dots, N$, it does not necessarily follow that $S'(x) \geq 0$ 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 $\{f_i\},\ i = 0, \dots, N$. If the following inequalities are valid*

$$\mu_0 c_1 \leq 2c_0 \leq 12f[x_0, x_1], \tag{3.20}$$
$$\lambda_N c_{N-1} \leq 2c_N \leq 12f[x_{N-1}, x_N], \tag{3.21}$$
$$\lambda_i f[x_{i-1}, x_i] \leq (1 + \lambda_i) f[x_i, x_{i+1}], \quad i = 1, \dots, N-1, \tag{3.22}$$
$$\mu_i f[x_i, x_{i+1}] \leq (1 + \mu_i) f[x_{i-1}, x_i], \quad i = 1, \dots, N-1, \tag{3.23}$$

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, $m_i \geq 0,\ i = 0, \dots, N$. From (3.8) we conclude that

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

Taking into account (3.20)–(3.23) we obtain

$$\begin{aligned} 0 &\leq m_0 &&\leq 3f[x_0, x_1], \\ 0 &\leq m_i &&\leq 3f[x_j, x_{j+1}], \quad j = i-1, i; \quad i = 1, \dots, N-1, \\ 0 &\leq m_N &&\leq 3f[x_{N-1}, x_N]. \end{aligned}$$

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 suppose to use generalized tension splines (see H. Späth (1990), Yu. S. 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 knots, 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 space of continuous 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 has two continuous derivatives.

Definition 3.2. *An interpolating tension generalized spline is a function $S \in S_4^G$ such that*

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

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

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

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

(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

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

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

$$\begin{aligned} S'(x_i) &= m_i = f[x_i, x_{i+1}] - \bar{a}_i \frac{M_i}{h_i} - \Psi_i(x_{i+1}) \frac{M_{i+1}}{h_i}, \\ S'(x_{i+1}) &= m_{i+1} = f[x_i, x_{i+1}] + \Phi_i(x_i) \frac{M_i}{h_i} + \bar{b}_i \frac{M_{i+1}}{h_i}, \end{aligned} \tag{3.26}$$

where

$$\begin{aligned} \overline{a}_i &= -\Phi_i(x_i) - h_i\Phi_i'(x_i), \\ \overline{b}_i &= -\Psi_i(x_{i+1}) + h_i\Psi_i'(x_{i+1}). \end{aligned} \tag{3.27}$$

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 linear algebraic equations

$$\begin{aligned} &\overline{a}_0\frac{M_0}{h_0} + \Psi_0(x_1)\frac{M_1}{h_0} = f[x_0, x_1] - f_0', \\ &\Phi_{i-1}(x_{i-1})\frac{M_{i-1}}{h_{i-1}} + \left(\frac{\overline{b}_{i-1}}{h_{i-1}} + \frac{\overline{a}_i}{h_i}\right) M_i + \Psi_i(x_{i+1})\frac{M_{i+1}}{h_i} = \delta_i f, \\ &\qquad\qquad\qquad\qquad\qquad\qquad i = 1, \ldots, N-1, \\ &\Phi_{N-1}(x_{N-1})\frac{M_{N-1}}{h_{N-1}} + \overline{b}_{N-1}\frac{M_N}{h_{N-1}} = f_N' - f[x_{N-1}, x_N]. \end{aligned} \tag{3.28}$$

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*

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

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:

$$\begin{aligned} r_0 &= \frac{1}{h_0}[\overline{a}_0 - \Psi_0(x_1)] > 0, \\ r_i &= \frac{1}{h_{i-1}}[\overline{b}_{i-1} - \Phi_{i-1}(x_{i-1})] + \frac{1}{h_i}[\overline{a}_i - \Psi_i(x_{i+1})] > 0, \quad i = 1, \ldots, N-1, \\ r_N &= \frac{1}{h_{N-1}}[\overline{b}_{N-1} - \Phi_{N-1}(x_{N-1})] > 0. \end{aligned}$$

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 weaken. From (3.26) we have

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

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

$$\begin{aligned}
&\overline{b}_0 m_0 + \Psi_0(x_1) m_1 = [\overline{b}_0 + \Psi_0(x_1)] f[x_0, x_1] - \frac{T_0}{h_0} f_0'', \\
&\Phi_{i-1}(x_{i-1}) \frac{h_{i-1}}{T_{i-1}} m_{i-1} + \left(\overline{a}_{i-1} \frac{h_{i-1}}{T_{i-1}} + \overline{b}_i \frac{h_i}{T_i} \right) m_i + \Psi_i(x_{i+1}) \frac{h_i}{T_i} m_{i+1} \\
&= [\overline{a}_{i-1} + \Phi_{i-1}(x_{i-1})] \frac{h_{i-1}}{T_{i-1}} f[x_{i-1}, x_i] + [\overline{b}_i + \Psi_i(x_{i+1})] \frac{h_i}{T_i} f[x_i, x_{i+1}], \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 1, \ldots, N-1, \\
&\Phi_{N-1}(x_{N-1}) m_{N-1} + \overline{a}_{N-1} m_N = \frac{T_{N-1}}{h_{N-1}} f_N'' \\
&\qquad\qquad\qquad + [\overline{a}_{N-1} + \Phi_{N-1}(x_{N-1})] f[x_{N-1}, x_N].
\end{aligned} \tag{3.31}$$

Lemma 3.6. *If the conditions*

$$0 < \Phi_i(x_i) < \overline{a}_i, \quad 0 < \Psi_i(x_{i+1}) < \overline{b}_i, \qquad i = 0, \ldots, N-1, \tag{3.32}$$

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: It follows from the conditions of the lemma that if

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

then

$$T_i = \overline{a}_i \overline{b}_i - \Phi_i(x_i)\Psi_i(x_{i+1}) > 0, \qquad i = 0, \ldots, N-1.$$

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

$$\begin{aligned}
\overline{r}_0 &= \overline{b}_0 - \Psi_0(x_1) > 0, \\
\overline{r}_i &= [\overline{a}_{i-1} - \Phi_{i-1}(x_{i-1})] \frac{h_{i-1}}{T_{i-1}} + [\overline{b}_i - \Psi_i(x_{i+1})] \frac{h_i}{T_i} > 0, \quad i = 1, \ldots, N-1, \\
\overline{r}_N &= \overline{a}_{N-1} - \Phi_{N-1}(x_{N-1}) > 0.
\end{aligned}$$

This ensure (see Theorem 3.1) 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 3.8. 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 $\{f_i\},\ i = 0, N$, and satisfy the endpoint conditions $S''(x_i) = f_i'',\ i = 0, \ldots, N$, 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*

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

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) system (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. $\square$

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 $m_i = S'(x_i),\ i = 0, \dots, N$, can be rewritten in the form

$$\begin{aligned} m_0 &= f'_0, \\ \widetilde{a}_i m_{i-1} + \widetilde{b}_i m_i + \widetilde{c}_i m_{i+1} &= \widetilde{d}_i, \quad i = 1, \dots, N-1, \\ m_N &= f'_N, \end{aligned} \tag{3.35}$$

where

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

$$\widetilde{d}_i = -\frac{h_{i-1}^2}{T_{i-1}}\Phi'_{i-1}(x_{i-1})f[x_{i-1}, x_i] + \frac{h_i^2}{T_i}\Psi'_i(x_{i+1})f[x_i, x_{i+1}].$$

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

$$f'_0 \geq 0, \quad f'_N \geq 0, \quad \widetilde{d}_i - \widetilde{a}_i\widetilde{d}_{i-1}/\widetilde{b}_{i-1} - \widetilde{c}_i\widetilde{d}_{i+1}/\widetilde{b}_{i+1} \geq 0, \quad i = 1, \dots, N-1, \tag{3.36}$$

where $\widetilde{b}_0 = \widetilde{b}_N = 1,\ \widetilde{d}_0 = f'_0,\ \widetilde{d}_N = f'_N$, *then* $m_i \geq 0,\ i = 0, \dots, N$.

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

Theorem 3.4. *Let a tension generalized spline* S *interpolate the monotone data* $\{f_i\},\ i = 0, \dots, N$, *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*

$$m_i \leq -\frac{h_i\Phi'_i(x_i)}{\Phi_i(x_i)}f[x_i, x_{i+1}]\tau, \quad m_{i+1} \leq \frac{h_i\Psi'_i(x_{i+1})}{\Psi_i(x_{i+1})}f[x_i, x_{i+1}](1-\tau), \quad 0 \leq \tau \leq 1,$$
$$i = 0, \dots, N-1, \tag{3.37}$$

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 $m_i \geq 0,\ i = 0, \dots, N$. Differentiating (3.24) and using (3.26) gives us

$$S'(x) = s(x, m_i, m_{i+1}) = \frac{h_i}{T_i}[h_i E_i(x) f[x_i, x_{i+1}] + F_i(x) m_i + G_i(x) m_{i+1}],$$

where

$$\begin{aligned} E_i(x) &= \Psi'_i(x_{i+1})\Phi'_i(x) + \Phi'_i(x_i)\Psi'(x) - \Phi'_i(x_i)\Psi'_i(x_{i+1}), \\ F_i(x) &= -\bar{b}_i\Phi'_i(x) + \Phi_i(x_i)\Psi'_i(x) - \Phi_i(x_i)\Psi'_i(x_{i+1}), \\ G_i(x) &= -\Psi_i(x_{i+1})\Phi'_i(x) + \bar{a}_i\Psi'_i(x) + \Phi'_i(x_i)\Psi_i(x_{i+1}). \end{aligned}$$

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 $s(x, 0, 0) \geq 0$, $s(x, \alpha_i, 0) \geq 0$, $s(x, 0, \beta_i) \geq 0$ for all $x \in [x_i, x_{i+1}]$, where $\alpha_i = -h_i\Phi_i'(x_i)/\Phi_i(x_i)f[x_i, x_{i+1}]$ and $\beta_i = h_i\Psi_i'(x_{i+1})/\Psi_i(x_{i+1})f[x_i, x_{i+1}]$. This can be done immediately by using properties of functions Φ_i and Ψ_i; hence, the desired result follows. This proves the theorem. □

3.8 Examples of Defining Functions

As was shown above the analysis of the conditions which provide monotone and convex interpolation is substantially based on the diagonal dominance of the system (3.28) and (3.31) for unknowns M_i and m_i respectively. The presence of diagonal dominance in these systems depends on the choice of the defining functions. Let us list 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 tension parameters:

$$\begin{aligned} \Phi_i(x) &= \varphi_i(t)h_i^2 &= \psi(p_i, 1-t)h_i^2, \\ \Psi_i(x) &= \psi_i(t)h_i^2 &= \varphi(q_i, t), \end{aligned}$$

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

(1) Rational splines with

$$\psi_i(t) = \frac{t^3 Q_i}{1 + q_i(1-t) + A_i(1-t)^2}, \quad Q_i^{-1} = 2(3 + 3q_i + q_i^2 - A_i),$$

where
(a) $A_i = 0$.
(b) $A_i = q_i^2$.
(c) $A_i = q_i(3 + q_i)$.

In the case $A_i = 0$ one has standard rational splines with linear denominator (H. Späth (1990)). The conditions of Lemma 3.6 are satisfied for $-1 < p_i, q_i < \infty$, $i = 0, \ldots, N-1$, and thus the interpolation rational spline exists and is unique. System (3.28) does not have a diagonal dominance. However Lemma 3.5 holds, if, for example, we demand additionally that $p_i = q_i$, $i = 0, \ldots, N-1$.

If $A_i = q_i^2$ or $A_i = q_i(3 + q_i)$ then both systems (3.28) and (3.31) have diagonal dominance. This choice of defining functions was given in V. L. Miroshnichenko (1997).

One can easily show that in all three cases (a)–(c)

$$-\frac{h_i\Phi_i'(x_i)}{\Phi_i(x_i)} = 3 + p_i, \quad \frac{h_i\Psi_i'(x_{i+1})}{\Psi_i(x_{i+1})} = 3 + q_i \tag{3.38}$$

and in order to provide convexity and/or monotonicity of the interpolation spline one can choose the tension parameters p_i, q_i as in B. I. Kvasov (2000).

(2) Rational splines with

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

where
(a) $A_i = 0$.
(b) $A_i = q_i(1+q_i)$.
(c) $A_i = q_i(4+q_i)$.

The case $A_i = 0$ corresponds to rational splines with quadratic denominator (H. Späth (1990)). Here the conditions for Lemmas 3.5 and 3.6 to hold are the same as in (1) for $A_i = 0$.

If $A_i = q_i(1+q_i)$ or $A_i = q_i(4+q_i)$ then both systems (3.28) and (3.31) have diagonal dominance and coincide with the corresponding system in (1) for (b) and (c). This choice of defining functions was considered in V. L. Miroshnichenko (1997).

In all three cases (a)–(c) the equalities (3.38) are valid as well.

(3) Exponential splines

$$\psi_i(t) = \frac{t^3 exp(-q_i(1-t) - A_i(1-t)^2)}{6 + 6q_i + q_i^2 - 2A_i},$$

where
(a) $A_i = 0$.
(b) $A_i = q_i^2/2$.
(c) $A_i = q_i(3 + q_i/2)$.

The case $A_i = 0$ was considered in H. Späth (1974, 1990). As above, only system (3.31) has diagonal dominance. Setting $p_i = q_i$ for all i one can provide diagonal dominance for system (3.28) also.

If $A_i = q_i^2/2$ or $A_i = q_i(3 + q_i/2)$ then both systems (3.28) and (3.31) have diagonal dominance. In order to choose tension parameters p_i, q_i one can again apply formulae (3.28).

(4) Hyperbolic splines (see P. E. Koch and T. Lyche (1991) and numerous references therein):

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

(5) Splines with additional knots (S. 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 knots of the spline on the interval $[x_i, x_{i+1}]$. By moving them, we can go from a cubic spline to piecewise linear interpolations (S. Pruess (1979)).

One has here

$$-\frac{h_i\Phi_i'(x_i)}{\Phi_i(x_i)} = 3(1+p_i), \quad \frac{h_i\Psi_i'(x_{i+1})}{\Psi_i(x_{i+1})} = 3(1+q_i).$$

(6) Splines of variable order (R. W. Soanes Jr. (1976)):

$$\psi_i(t) = \frac{t^{k_i}}{k_i(k_i-1)}, \quad k_i = q_i + 3.$$

The computational burden involved in using splines of variable order can be avoided by considering only integer values for the parameters k_i (and $l_i = p_i + 3$).

In cases (4)–(6) the condition for Lemmas 3.5 and 3.6 to hold are the same as in (1)–(3) for $A_i = 0$. In particular, Lemma 3.5 holds, it, for example, we demand additionally that $p_i = q_i, \ i = 0, \ldots, N-1$.

3.9 Graphical Examples

We tested the shape preserving interpolation algorithms described above on several examples. In order to construct the shape preserving function the hyperbolic splines due to D. G. Schweikert (1966) were used, which correspond to the choice in (3.25)

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

The data for the first example was taken from Yu. S. Zavyalov *et al.* (1980). We consider interpolating the function

$$f(x) = 1 - \frac{exp(100x) - 1}{exp(100) - 1}, \quad x \in [0,1]$$

on the uniform mesh: $x_i = i/10, \ i = 0, \ldots, 10$. In Figure. 3.1 (and in Figures 3.2 – 3.5) the dashed and solid lines show the graphs of the ordinary cubic spline S_3 of the C^2 class and the shape preserving hyperbolic spline S, respectively. In both cases the boundary conditions $S'(x_0) = 0$, $S'(x_{10}) = -100$ are used. The spline S_3 gives unacceptable oscillations. It is possible to decrease their amplitude by either introducing a nonuniform mesh that concentrates the knots in the domain having large gradient, or by the choosing an appropriate parameterization. Moreover, in this example the maximal deviation of the tension hyperbolic spline S from the interpolated function does not exceed 0.078 and S exhibits the same monotonicity and convexity as f.

In a number of papers devoted to shape preserving interpolation (see, for example, R. Delbourgo and J. A. Gregory (1985), S. E. Eisenstat *et al.* (1985)) the algorithms are tested by data taken from H. Akima (1970) and given in Table 3.1. Figure 3.2 shows the graphs of the splines S_3 and S for this data. The latter has the inflection point on $[x_8, x_9]$. In comparison with the profiles given in R. Delbourgo and J. A. Gregory (1985), J. A. Gregory (1984), in addition to retaining the monotonicity and convexity properties of the initial data the spline S approximates S_3 better.

Table 3.1: Data for Figure 3.2.

i	0	1	2	3	4	5	6	7	8	9	10
x_i	0	2	3	5	6	8	9	11	12	14	15
f_i	10	10	10	10	10	10	10.5	15	56	60	85

In J. A. Gregory (1984) the function $f(x) = 2 - \sqrt{x(2-x)}$, $0 \le x \le 2$, that defines the semicircle is considered. This function is interpolated on the mesh uniform in x (Fig. 3.3) and along the arc length (Fig. 3.4). In both cases 13 interpolation points and the boundary conditions $S'(x_0) = -50$, $S'(x_{12}) = 50$ were used. We see that the transition to the mesh with constant step along the arc length enables us to reduce the oscillations of the spline S_3 but it does not remove them. The hyperbolic spline S again retains the monotonicity and convexity properties of the initial data.

Figure 3.5 illustrates the interpolation of the data taken from H. Späth (1990) (Table 3.2). In version of the algorithm we used, the conditions $\delta_{N-1} f = 0$, $\delta_{N-2} f \neq 0$ and $f[x_{N-3}, x_{N-2}] f[x_{N-2}, x_{N-1}] > 0$ imply the linearity of the spline on the interval $[x_{N-2}, x_N]$.

Table 3.2: Data for Figure 3.5.

i	0	1	2	3	4	5	6	7	8
x_i	0	2.0	2.5	3.5	5.5	6.0	7	8.5	10
f_i	2	2.5	4.5	5.0	4.5	1.5	1	0.5	0

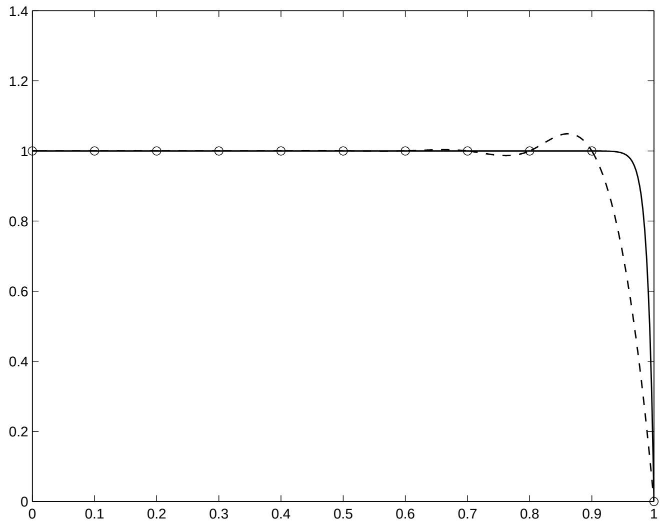

Figure 3.1: Exponential boundary layer type data (Yu. S. Zavyalov *et al.* (1980)). Profiles of interpolation and shape preserving splines.

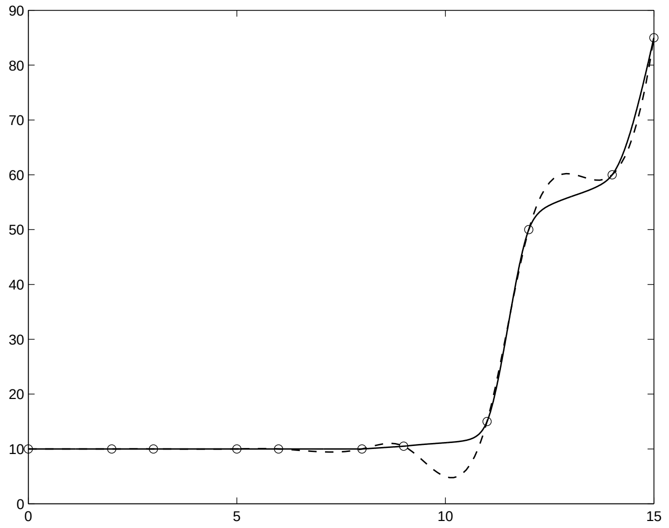

Figure 3.2: Data obtained by H. Akima (1970). Typical behaviour of interpolation and shape preserving splines on given fast- and slow-change sections of data.

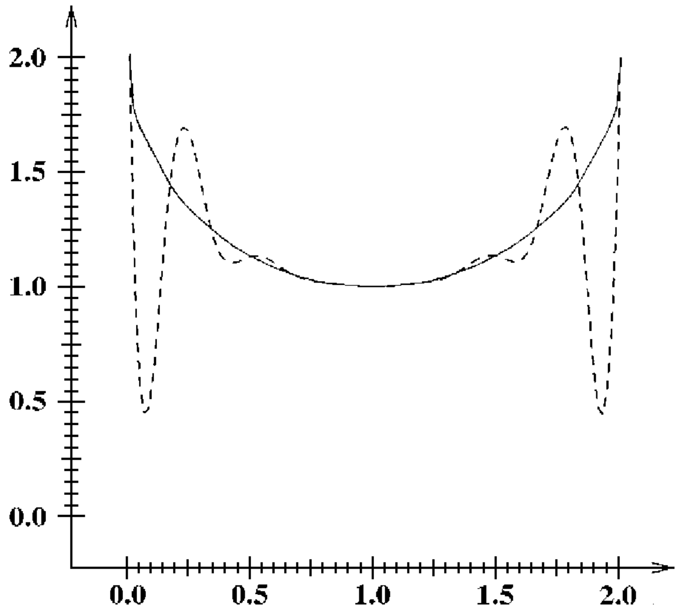

Figure 3.3: Interpolation of a semicircle by the data uniform in the x-coordinate.

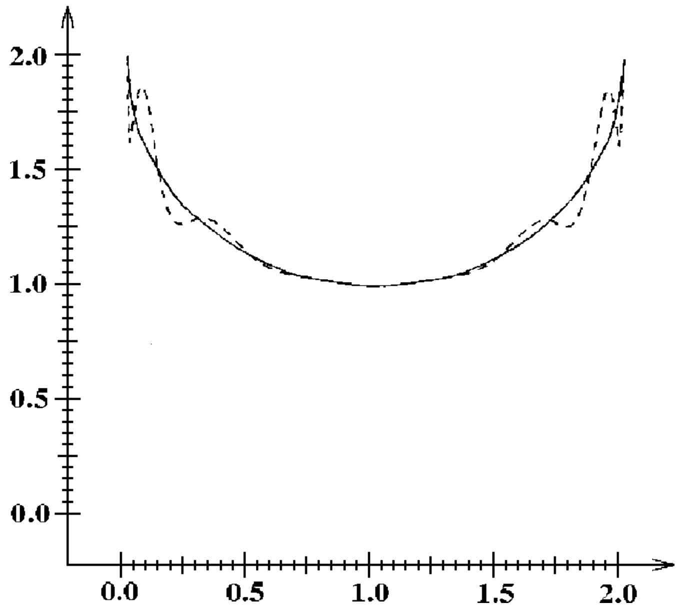

Figure 3.4: Interpolation of a semicircle by the data uniform in the arc length.

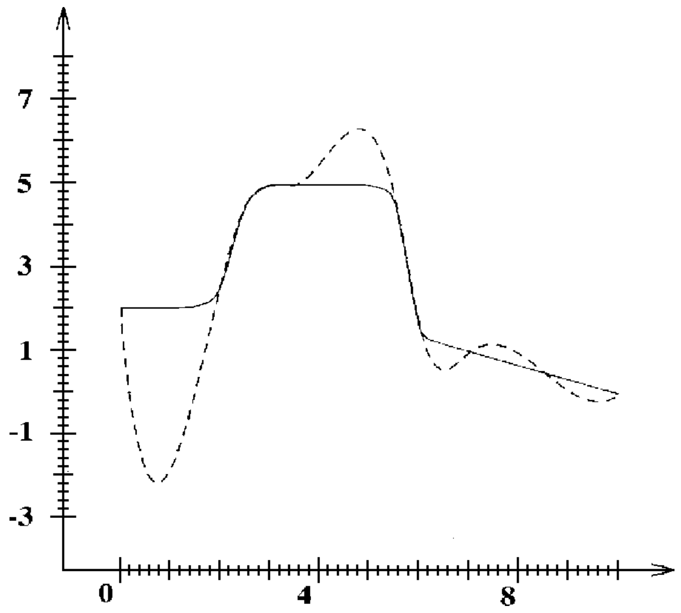

Figure 3.5: Data obtained by H. Späth (1990). The shape preserving spline is not sensitive to these outliers and automatically satisfies the boundary conditions.

Chapter IV

Finite-Difference Method for 1-D Shape Preserving Interpolation

In this chapter, tension hyperbolic splines are defined as solutions of differential multipoint boundary value problems by considering the 1-D case. For computations, we use difference approximations of such problems. This permits us to avoid calculations of hyperbolic functions, however, the extension of a mesh solution will be a discrete tension hyperbolic spline. We consider the basic computational aspects of this approach and illustrate its main advantages.

4.1 Problem Formulation. Finite Difference Approximation

Let the data

$$(x_i, f_i), \quad i = 0, \dots, N+1, \tag{4.1}$$

be given, where: $a = x_0 < x_1 < \cdots < x_{N+1} = b$. Let us put

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

An interpolating tension hyperbolic spline S with a set of tension parameters $\{p_i \ge 0 \mid i = 0, \dots, N\}$ is a solution of the differential multipoint boundary value problem (DMBVP for short)

$$\frac{d^4 S}{dx^4} - \left(\frac{p_i}{h_i}\right)^2 \frac{d^2 S}{dx^2} = 0, \quad x \in (x_i, x_{i+1}), \quad i = 0, \dots, N, \tag{4.2}$$

$$S \in C^2[a, b], \tag{4.3}$$

with the interpolation conditions

$$S(x_i) = f_i, \quad i = 0, \dots, N+1 \tag{4.4}$$

and the end constraints

$$S''(a) = f''_0 \quad \text{and} \quad S''(b) = f''_{N+1}. \tag{4.5}$$

For practical purposes it is often more interesting to know the values of the solution over a given tabulation of $[a, b]$ than its global analytic expression. In this paper we do not consider the tabulation of S directly but we study a natural discretization of the previous problem. We prove that the discretized problem has a unique solution, called *mesh solution*, and we study its properties. Of course, it turns out that the mesh solution is not a tabulation of S but can be extended on $[a,b]$ to a function u with properties very similar to those of S and which approaches S as the discretization step goes to zero. Due to these properties we will refer to u as a *discrete* hyperbolic tension spline interpolation of the data (4.1).

Let us assume (as a first step) that each h_i is an integer multiple of the same tabulation step, τ. Putting $n_i = h_i/\tau$, we look for a mesh solution $\bar{u} = \{u_{i,j} \mid j = -1,\dots,n_i+1, \quad i = 0,\dots,N\}$, satisfying the difference equations:

$$\left[\Lambda^2 - \left(\frac{p_i}{h_i}\right)^2 \Lambda\right] u_{i,j} = 0, \quad j = 1,\dots,n_i-1, \quad i = 0,\dots,N, \tag{4.6}$$

where

$$\Lambda u_{i,j} = \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\tau^2}.$$

The smoothness condition (4.3) changes to the equations

$$\begin{array}{rcl} u_{i-1,n_{i-1}} & = & u_{i,0}, \\ \dfrac{u_{i-1,n_{i-1}-1} - u_{i-1,n_{i-1}+1}}{2\tau} & = & \dfrac{u_{i,1} - u_{i,-1}}{2\tau}, \quad i = 1,\dots,N, \\ \Lambda u_{i-1,n_{i-1}} & = & \Lambda u_{i,0}, \end{array} \tag{4.7}$$

which are equivalent to

$$u_{i-1,n_{i-1}+j} = u_{i,j}, \quad j = -1,0,1. \tag{4.8}$$

The interpolation conditions (4.4) take the form

$$u_{i,0} = f_i, \quad u_{i,n_i} = f_{i+1}, \quad i = 0,\dots,N, \tag{4.9}$$

and for the end conditions (4.5) we have

$$\Lambda u_{0,0} = f''_0 \quad \text{and} \quad \Lambda u_{N,n_N} = f''_{N+1}. \tag{4.10}$$

4.2 An Algorithm for Solving the System of Difference Equations

The equalities (4.8) and (4.10) permit to eliminate the redundant unknowns in the difference equations (4.6). The values $u_{0,-1}$ and u_{N,n_N+1} are not explicitly computed but are introduced into the formulation to accommodate the two necessary end conditions. Putting $m = \sum_{i=0}^{N} n_i + 3$, the previous equations can be collected in the $m \times m$ linear system

$$A\hat{u} = b, \tag{4.11}$$

where

$$A = \left[\begin{array}{cccccccccccccc}
1 & -2 & 1 & 0 & & & & & & & & & & \\
0 & 1 & 0 & 0 & & & & & & & & & & \\
1 & a_0 & b_0 & a_0 & 1 & & & & & & & & & \\
 & 1 & a_0 & b_0 & a_0 & 1 & & & & & & & & \\
 & & & & \dots & & & & & & & & & \\
 & & 1 & a_0 & b_0 & a_0 & 1 & & & & & & & \\
 & & & 0 & 0 & 1 & 0 & 0 & & & & & & \\
 & & & & 1 & a_1 & b_1 & a_1 & 1 & & & & & \\
 & & & & & 1 & a_1 & b_1 & a_1 & 1 & & & & \\
 & & & & & & & \dots & & & & & & \\
 & & & & & & & & \dots & & & & & \\
 & & & & & & & & & \dots & & & & \\
 & & & & & & & & 1 & a_N & b_N & a_N & 1 & \\
 & & & & & & & & & 1 & a_N & b_N & a_N & 1 \\
 & & & & & & & & & & 0 & 0 & 1 & 0 \\
 & & & & & & & & & & 0 & 1 & -2 & 1
\end{array}\right],$$

$$a_i = -(4+\omega_i), \quad b_i = 6 + 2\omega_i, \quad \omega_i = \left(\frac{p_i\tau}{h_i}\right)^2, \quad i = 0, \dots, N,$$

and

$$\hat{u} = (u_{0,-1}, u_{00}, u_{01}, \dots, u_{0n_0-1}, u_{10}, \dots, u_{20}, \dots, u_{N0}, \dots, u_{Nn_N}, u_{Nn_N+1})^T,$$

$$b = (\tau^2 f_0'', f_0, 0, \dots, 0, f_1, \dots, f_2, \dots, f_N, \dots, f_{N+1}, \tau^2 f_{N+1}'')^T.$$

In the previous system the unknowns $u_{i,0}$, $i = 0, \dots, N+1$, can be immediately determined from the interpolation conditions while the expression of $u_{0,-1}$ (u_{N,n_N+1}) can be obtained from the first (last) equation and substituted in the third (third to last) equation. Then in practice we deal with the $m^* \times m^*$ linear system ($m^* = m - N - 4$)

$$A^* u^* = b^*, \tag{4.12}$$

where

$$A^* = \left[\begin{array}{ccccccccccc}
b_0 - 1 & a_0 & 1 & & & & & & & & \\
a_0 & b_0 & a_0 & 1 & & & & & & & \\
1 & a_0 & b_0 & a_0 & 1 & & & & & & \\
 & & \dots & & & & & & & & \\
 & 1 & a_0 & b_0 & a_0 & & & & & & \\
 & & 1 & a_0 & b_0 & 1 & & & & & \\
 & & & & 1 & b_1 & a_1 & 1 & & & \\
 & & & & & a_1 & b_1 & a_1 & 1 & & \\
 & & & & & & & \dots & & & \\
 & & & & & & 1 & a_N & b_N & a_N & 1 \\
 & & & & & & & 1 & a_N & b_N & a_N \\
 & & & & & & & & 1 & a_N & b_N - 1
\end{array}\right],$$

and u^*, b^* are correspondingly deduced from $\hat{u}$ and b.

Following M. A. Malcolm (1977) we observe that

$$A^* = C^* + D^*,$$

where both C^* and D^* are symmetric block diagonal matrices; to be more specific,

$$C^* = \begin{bmatrix} C_0 & & & & & & \\ & C_1 & & & & & \\ & & C_2 & & & & \\ & & & \cdot & & & \\ & & & & \cdot & & \\ & & & & & \cdot & \\ & & & & & & C_N \end{bmatrix},$$

$$C_i = \begin{bmatrix} b_i - 1 & a_i & 1 & & & & & \\ a_i & b_i & a_i & 1 & & & & \\ 1 & a_i & b_i & a_i & 1 & & & \\ & & & \cdots & & & & \\ & & & & \cdots & & & \\ & & 1 & a_i & b_i & a_i & 1 \\ & & & 1 & a_i & b_i & a_i \\ & & & & 1 & a_i & b_i - 1 \end{bmatrix}, \tag{4.13}$$

$$D^* = \begin{bmatrix} 0 & 0 & & & & & & & & & & & & & & \\ 0 & 0 & & & & & & & & & & & & & & \\ & & \ddots & & & & & & & & & & & & & \\ & & & 0 & & & & & & & & & & & & \\ & & & & 1 & 1 & & & & & & & & & & \\ & & & & 1 & 1 & & & & & & & & & & \\ & & & & & & 0 & & & & & & & & & \\ & & & & & & & \ddots & & & & & & & & \\ & & & & & & & & 0 & & & & & & & \\ & & & & & & & & & 1 & 1 & & & & & \\ & & & & & & & & & 1 & 1 & & & & & \\ & & & & & & & & & & & 0 & & & & \\ & & & & & & & & & & & & \ddots & & & \\ & & & & & & & & & & & & & 0 & & \\ & & & & & & & & & & & & & & 0 & 0 \\ & & & & & & & & & & & & & & 0 & 0 \end{bmatrix}$$

Since the eigenvalues of D^* are 0 and 2, from a corollary of the Courant-Fisher theorem (see G. H. Golub and C. F. Van Loan (1996)) we have that the eigenvalues of A^*, $\lambda_k(A^*)$, satisfy the following inequalities

$$\lambda_k(A^*) \geq \lambda_k(C^*), \quad k = 1, \ldots, m^*.$$

The eigenvalues of C^* are the collection of the eigenvalues of C_i and we have

$$C_i = B_i^2 - \omega_i B_i,$$

where B_i is the $(n_i - 1) \times (n_i - 1)$ tridiagonal matrix

$$B_i = \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & & \cdots & & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix}.$$

Because the eigenvalues of B_i are well known (see also M. A. Malcolm (1977)),

$$\lambda_j(B_i) = -2\Big(1 - \cos\frac{j\pi}{n_i}\Big), \quad j = 1, \dots, n_i - 1,$$

the eigenvalues of C_i are

$$\lambda_j(C_i) = 4\Big(1 - \cos\frac{j\pi}{n_i}\Big)^2 + 2\omega_i\Big(1 - \cos\frac{j\pi}{n_i}\Big).$$

It follows that

$$\lambda_k(A^*) \geq \min_{i,j} \lambda_j(C_i) = \min_i \Big[4\Big(1 - \cos\frac{\tau\pi}{h_i}\Big)^2 + 2\omega_i\Big(1 - \cos\frac{\tau\pi}{h_i}\Big)\Big].$$

Hence, A^* is a positive matrix and the linear system (4.12) (and (4.11) as well) has unique solution.

In addition, from Gershgorin's theorem, $\lambda_k(A^*) \leq \max_i[16 + 4\omega_i]$, so that for the condition number, $\mu_2(A^*)$, with respect to the 2-norm of A^*, we have the following upper bound not depending on the number of data points, $N + 2$:

$$\begin{aligned} \mu_2(A^*) &\leq \frac{\max_i \left[16 + 4(\frac{\tau p_i}{h_i})^2\right]}{\min_i \left[4(1 - \cos\frac{\tau\pi}{h_i})^2 + 2(\frac{\tau p_i}{h_i})^2(1 - \cos\frac{\tau\pi}{h_i})\right]} \\ &\leq \frac{\max_i \left[16 + 4(\frac{\tau p_i}{h_i})^2\right]}{\min_i(\frac{\tau}{h_i})^4[\pi^4 + (\pi p_i)^2]}. \end{aligned} \tag{4.14}$$

We remark that, for $p_i = 0, \ i = 0, \dots, N$, we recover the results presented in M. A. Malcolm (1977).

From the structure of A^*, the linear system (4.12) can be solved efficiently using a direct method for band matrices. Since A^* is positive band matrix of band width 2, the classical Cholesky factorization , $A^* = LL^T$, provides a lower triangular band matrix L of band 2 and it can be performed in $O(m)$ operations, G. H. Golub and C. F. Van Loan (1996).

Here we have presented the analysis for the linear system obtained considering, as end conditions, given values for the second divided differences at the end knots. A similar analysis can be performed for different types of end conditions.

For example, in practical applications, it seems sometimes more convenient to consider given values of the first divided differences at $x_0,\ x_{N+1}$. Even in this case, following the previous strategy, it is possible to prove the existence and uniqueness of the solution and to find a bound for the condition number of the matrix of the linear system exactly as in (4.14).

4.3 System Splitting and Mesh Solution Extension

In order to solve the DMBVP (4.2)–(4.5) numerically, we consider the system of difference equations (4.6) completed with the smoothness conditions (4.7) (or (4.8)), interpolation conditions (4.9) and end conditions (4.10).

With the notation

$$M_{ij} = \Lambda u_{ij}, \quad j = 0, \dots, n_i, \quad i = 0, \dots, N, \tag{4.15}$$

on the interval $[x_i, x_{i+1}]$ the system (4.6) takes the form

$$\begin{array}{rcl} M_{i0} & = & M_i, \\ \dfrac{M_{ij-1} - 2M_{ij} + M_{ij+1}}{\tau^2} - \left(\dfrac{p_i}{h_i}\right)^2 M_{ij} & = & 0,\ j = 1, \dots, n_i - 1, \\ M_{i,n_i} & = & M_{i+1}, \end{array} \tag{4.16}$$

where M_i and M_{i+1} are prescribed numbers. The system (4.16) has a unique solution, which can be represented as follows

$$M_{ij} = m_i(x_{ij}), \quad x_{ij} = x_i + j\tau, \quad j = 0, \dots, n_i$$

with

$$m_i(x) = M_i \frac{\sinh k_i(1-t)}{\sinh(k_i)} + M_{i+1} \frac{\sinh k_i t}{\sinh(k_i)}, \quad t = \frac{x - x_i}{h_i},$$

and where the parameters k_i are the solutions of the transcendental equations

$$\frac{2}{\hat{\tau}_i} \sinh \frac{k_i \hat{\tau}_i}{2} = p_i \quad (p_i \geq 0, \quad \hat{\tau}_i = \frac{\tau}{h_i}).$$

From the equation (4.15) and the interpolation conditions (4.9) we have

$$\begin{array}{rcl} u_{i0} & = & f_i, \\ \dfrac{u_{ij-1} - 2u_{ij} + u_{ij+1}}{\tau^2} & = & M_{ij},\ j = 1, \dots, n_i - 1, \\ u_{i,n_i} & = & f_{i+1}. \end{array} \tag{4.17}$$

Let us consider the function

$$u_i(x) = f_i(1-t) + f_{i+1}t + \hat{\varphi}_i(1-t)h_i^2 M_i + \hat{\varphi}_i(t)h_i^2 M_{i+1}, \tag{4.18}$$

where

$$\hat{\varphi}_i(t) = \frac{\sinh(k_i t) - t\sinh(k_i)}{p_i^2 \sinh(k_i)}.$$

The function u_i satisfies to the conditions

$$u_i(x_j) = f_j, \quad \Lambda u_i(x_j) = M_j, \quad j = i, i+1,$$

where

$$\Lambda u_i(x) = \frac{u_i(x-\tau) - 2u_i(x) + u_i(x+\tau)}{\tau^2}.$$

The mesh restriction of the function u_i gives us the solution of the system (4.17) with $u_{ij} = u_i(x_{ij}),\ j = 0, \ldots, n_i$. The smoothness conditions (4.7) can be rewritten as

$$\begin{aligned} u_{i-1}(x_i) &= u_i(x_i), \\ \frac{u_{i-1}(x_i+\tau) - u_{i-1}(x_i-\tau)}{2\tau} &= \frac{u_i(x_i+\tau) - u_i(x_i-\tau)}{2\tau}, \\ \Lambda u_{i-1}(x_i) &= \Lambda u_i(x_i), \end{aligned} \tag{4.19}$$

which are equivalent to

$$u_{i-1}(x_i + j\tau) = u_i(x_i + j\tau), \quad j = -1, 0, 1.$$

Using (4.18) and the second condition (4.19) we obtain a linear system with 3-diagonal matrix

$$\begin{aligned} M_0 &= f_0'', \\ \alpha_{i-1}h_{i-1}M_{i-1} + (\beta_{i-1}h_{i-1} + \beta_i h_i)M_i + \alpha_i h_i M_{i+1} &= d_i,\ i = 1, \ldots, N, \\ M_{N+1} &= f_{N+1}'', \end{aligned} \tag{4.20}$$

where

$$\begin{aligned} d_i &= \frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}}, \\ \alpha_i &= -\frac{\hat{\varphi}_i(\hat{\tau}_i) - \hat{\varphi}_i(-\hat{\tau}_i)}{2\hat{\tau}_i} &&= -\frac{\sinh(k_i\hat{\tau}_i) - \hat{\tau}_i \sinh(k_i)}{p_i^2 \hat{\tau}_i \sinh(k_i)}, \\ \beta_i &= \frac{\hat{\varphi}_i(1+\hat{\tau}_i) - \hat{\varphi}_i(1-\hat{\tau}_i)}{2\hat{\tau}_i} &&= \frac{\cosh(k_i)\sinh(k_i\hat{\tau}_i) - \hat{\tau}_i \sinh(k_i)}{p_i^2 \hat{\tau}_i \sinh(k_i)}. \end{aligned}$$

Using an expansion of the hyperbolic functions in the above expressions as power series we obtain

$$\beta_i > 2\alpha_i > 0, \quad i = 0, \ldots, N, \quad \text{for all} \quad \tau > 0, \quad p_i > 0.$$

Therefore the system (4.20) is diagonal dominant and has a unique solution.

We can now conclude that the function u which coincides with u_i for $x \in [x_i, x_{i+1}]$, $i = 0, 1, \ldots, N$, is *a discrete tension interpolation spline.* A mesh restriction of the spline u gives us a solution of the system (4.6). The spline u can also be easy recovered from the solution of the system (4.6).

In addition we observe that

$$\lim_{p_i \to 0} \alpha_i = \frac{1}{6}\Big[1 - \Big(\frac{\tau}{h_i}\Big)^2\Big], \quad \lim_{p_i \to 0} \beta_i = \frac{1}{6}\Big[2 + \Big(\frac{\tau}{h_i}\Big)^2\Big], \tag{4.21}$$

$$\lim_{p_i \to 0} \hat{\varphi}_i(t) = \frac{t(t^2-1)}{6},$$

hence we recover the result of M. A. Malcolm's paper (1977) for discrete cubics.

Instead of looks for a direct solution of the system (4.6) we recommend the following algorithm.

Step 1. Solve 3-diagonal system (4.20) for M_i, $i = 1, \ldots, N$.

Step 2. Solve $N+1$ 3-diagonal systems (4.16) for M_{ij}, $j = 1, \ldots, n_i - 1$, $i = 0, \ldots, N$,

Step 3. Solve $N+1$ 3-diagonal systems (4.17) for u_{ij}, $j = 1, \ldots, n_i - 1$, $i = 0, \ldots, N$.

In this algorithm, hyperbolic functions need to be computed in step 1 only, but not in steps 2 and 3. Furthermore, the solution of any system (4.16) or (4.17) requires $8m$ arithmetic operations, namely, $3m$ additions, $3m$ multiplications, and $2m$ divisions, Yu. S. Zavyalov *et al.* (1980), and is thus substantially cheaper than direct computation by formula (4.18).

Steps 2 and 3 can be replaced by a direct splitting of the system (4.11) into $N+1$ systems with 5-diagonal matrices

$$C_i u_i = c_i, \quad i = 0, \ldots, N, \tag{4.22}$$

where the $(n_i - 1) \times (n_i - 1)$ matrix $C_i = B_i^2 - \omega_i B_i$ and B_i has the form (4.13),

$$\begin{aligned} u_i &= (u_{i1}, u_{i2}, \ldots, u_{i,n_i-1})^T, \\ c_i &= \big((2+\omega_i)f_i - M_i, -f_i, 0, \ldots, 0, -f_{i+1}, (2+\omega_i)f_{i+1} - M_{i+1}\big)^T. \end{aligned}$$

The calculations for solving the systems (4.16) and (4.17) or (4.22) can be performed by using a multi-processing parallel computer system. If $n_i = n$ for all i, we can first store a triangular factorization of the matrices of the systems and then use parallel computations.

4.4 Error Estimates

In this section we bound the distance between a discrete tension hyperbolic spline and the corresponding continuous one interpolating the same set of data and having the same end conditions. For the sake of simplicity we only detail the case where second divided differences (derivatives) are prescribed at x_0 and x_{N+1}. Similar results hold for other standard end conditions.

It has been shown in the previous section how the mesh solution obtained from system (4.11) can be extended to a continuous function u in the interval $[a, b]$ considering the expression (4.18) in each subinterval $[x_i, x_{i+1}]$ where the constants M_i are solution of the system (4.6).

On the other hand, as mentioned in section 4.2, the classical smooth tension hyperbolic spline interpolating the data (4.1) is a function S, satisfying (4.2)–(4.5). It is well known that, setting

$$\tilde{M}_i := \frac{d^2 S}{dx^2}(x_i), \quad i = 0, \ldots, N+1,$$

we can express $S_i(x) := S(x)|_{[x_i, x_{i+1}]}$ as follows

$$S_i(x) = f_i(1-t) + f_{i+1}t + \tilde{\varphi}_i(1-t)h_i^2 \tilde{M}_i + \tilde{\varphi}_i(t)h_i^2 \tilde{M}_{i+1}, \tag{4.23}$$

where

$$\tilde{\varphi}_i(t) = \frac{\sinh(p_i t) - t\sinh(p_i)}{p_i^2 \sinh(p_i)},$$

and the constants $\tilde{M}_i$ are solutions of the linear system

$$\begin{aligned} \tilde{M}_0 &= f_0'', \\ \tilde{\alpha}_{i-1}h_{i-1}\tilde{M}_{i-1} + (\tilde{\beta}_{i-1}h_{i-1} + \tilde{\beta}_i h_i)\tilde{M}_i + \tilde{\alpha}_i h_i \tilde{M}_{i+1} &= d_i, \ i = 1, \ldots, N, \\ \tilde{M}_{N+1} &= f_{N+1}'', \end{aligned} \tag{4.24}$$

where

$$\begin{aligned} \tilde{\alpha}_i &= -\tilde{\varphi}_i'(0) = \frac{\sinh(p_i) - p_i}{p_i^2 \sinh(p_i)}, \\ \tilde{\beta}_i &= \tilde{\varphi}_i'(1) = \frac{p_i \cosh(p_i) - \sinh(p_i)}{p_i^2 \sinh(p_i)}. \end{aligned}$$

It is easy to verify that the 3-diagonal linear system (4.24) is diagonal dominant, more precisely

$$\tilde{\beta}_i \geq 2\tilde{\alpha}_i > 0, \quad \forall p_i \geq 0$$

and that, in the limit case $\tau \to 0$, systems (4.20) and (4.24) coincide since

$$\lim_{\tau \to 0} \alpha_i = \tilde{\alpha}_i, \quad \lim_{\tau \to 0} \beta_i = \tilde{\beta}_i.$$

Eliminating in (4.20) and (4.24) the unknowns M_0, M_{N+1}, and $\tilde{M}_0$, $\tilde{M}_{N+1}$ respectively we obtain two systems of N equations: their matrices will be denoted by T and $\tilde{T}$ respectively. Then (4.20) reduces to

$$TM = d, \quad d = (d_1 - \alpha_0 h_0 f_0'', \ d_2, \ \ldots, d_N - \alpha_N h_N f_{N+1}'')^T,$$

while (4.24) reduces to

$$\tilde{T}\tilde{M} = \tilde{d}, \quad \tilde{d} = (d_1 - \tilde{\alpha}_0 h_0 f_0'', \ d_2, \ \ldots, d_N - \tilde{\alpha}_N h_N f_{N+1}'')^T.$$

We have

$$\tilde{T} = T + \delta T,$$

where

$$\begin{aligned}
\delta T &= \tau^2 \begin{bmatrix} \frac{b_0}{h_0} + \frac{b_1}{h_1} & \frac{a_1}{h_1} & & & \\ \frac{a_1}{h_1} & \frac{b_1}{h_1} + \frac{b_2}{h_2} & \frac{a_2}{h_2} & & \\ & & \cdots & & \\ & \frac{a_{i-1}}{h_{i-1}} & \frac{b_{i-1}}{h_{i-1}} + \frac{b_i}{h_i} & \frac{a_i}{h_i} & \\ & & & \cdots & \\ & & \frac{a_{N-1}}{h_{N-1}} & \frac{b_{N-1}}{h_{N-1}} + \frac{b_N}{h_N} & \end{bmatrix}, \\
a_i &= \frac{1}{\hat{\tau}_i^2 p_i}\Big[\frac{\cosh\frac{k_i\hat{\tau}_i}{2}}{\sinh(k_i)} - \frac{1}{\sinh(p_i)}\Big], \\
b_i &= -\frac{1}{\hat{\tau}_i^2 p_i}\Big[\frac{\cosh(k_i)\cosh\frac{k_i\hat{\tau}_i}{2}}{\sinh(k_i)} - \frac{\cosh(p_i)}{\sinh(p_i)}\Big].
\end{aligned}$$

After some computations we obtain that a_i, b_i are continuous bounded functions of τ, $\tau \geq 0$, more precisely

$$|a_i|, \ |b_i| \leq \mathcal{A}_i(p_i) := \lim_{\hat{\tau}_i \to 0} |b_i| = \frac{3p_i\cosh(p_i)\sinh(p_i) + p_i^2}{24\sinh^2(p_i)}. \tag{4.25}$$

Then, putting $\tilde{M} = M + \delta M$, we can rewrite the linear system obtained reducing (4.24) as

$$(T + \delta T)(M + \delta M) = \tilde{d},$$

that is, following M. A. Malcolm (1977) and T. Lyche (1976)

$$\begin{aligned}
-\delta M &= (T^{-1} - \tilde{T}^{-1})d + \tilde{T}^{-1}(d - \tilde{d}) \\
&= \tilde{T}^{-1}(\tilde{T} - T)T^{-1}d + \tilde{T}^{-1}(d - \tilde{d}) \\
&= -\tilde{T}^{-1}\delta T M + \tilde{T}^{-1}(d - \tilde{d})
\end{aligned}$$

so that

$$\|\delta M\|_\infty \leq \|\delta M\|_2 \leq \|\tilde{T}^{-1}\|_2 \|\delta T\|_2 \|M\|_2 + \|\tilde{T}^{-1}(d - \tilde{d})\|_2.$$

Since, from Gershgorin's theorem,

$$\begin{aligned}
\|\delta T\|_2 &\leq \tau^2 4 \max_{i=0,\ldots,N} \frac{\mathcal{A}_i}{h_i}, \\
\|T^{-1}\|_2 &\leq \max_{i=1,\ldots,N} \frac{1}{(\beta_i - \alpha_i)h_i + (\beta_{i-1} - \alpha_{i-1})h_{i-1}}, \\
\|\tilde{T}^{-1}\|_2 &\leq \max_{i=1,\ldots,N} \frac{1}{(\tilde{\beta}_i - \tilde{\alpha}_i)h_i + (\tilde{\beta}_{i-1} - \tilde{\alpha}_{i-1})h_{i-1}},
\end{aligned}$$

we have

$$\|M\|_2 \leq \|d\|_2 \max_{i=1,\ldots,N} \frac{1}{(\beta_i-\alpha_i)h_i+(\beta_{i-1}-\alpha_{i-1})h_{i-1}}.$$

In addition, since from (4.25)

$$\|d-\tilde{d}\|_2 \leq \tau^2 \max_{i=0,N} \frac{\mathcal{A}_i}{h_i}\|g\|_2, \quad g=(f_0'', f_{N+1}'')^T$$

we obtain

$$\|\tilde{T}^{-1}(d-\tilde{d})\|_2 \leq \tau^2\|\tilde{T}^{-1}\|_2 \max_{i=0,N}\frac{\mathcal{A}_i}{h_i}\|g\|_2.$$

Therefore

$$\begin{aligned}\|\tilde{M}-M\|_2 \quad &\leq \quad \tau^2 \max_i \frac{1}{(\tilde{\beta}_i-\tilde{\alpha}_i)h_i+(\tilde{\beta}_{i-1}-\tilde{\alpha}_{i-1})h_{i-1}} \\ &\quad\quad \times\left[4\max_{i=0,\ldots,N}\frac{\mathcal{A}_i}{h_i}\|M\|_2+\max_{i=0,N}\frac{\mathcal{A}_i}{h_i}\|g\|_2\right].\end{aligned} \tag{4.26}$$

Then from the expressions of S_i and u_i, see (4.23) and (4.18), we have

$$\begin{aligned}\|S_i-u_i\| \quad &:= \quad \max_{x\in[x_i,x_{i+1}]}|\tilde{S}_i(x)-u_i(x)| \\ &\leq \quad h_i^2\max_{t\in[0,1]}\Big|\tilde{M}_i\tilde{\varphi}_i(1-t)-M_i\hat{\varphi}_i(1-t)+\tilde{M}_{i+1}\tilde{\varphi}_i(t)-M_{i+1}\hat{\varphi}_i(t)\Big|.\end{aligned}$$

Then putting

$$\mathcal{B}_i := 2\max_{t\in[0,1]}|\tilde{\varphi}_i(t)|, \quad \mathcal{C}_i := 2\max_{t\in[0,1]}\left|\frac{\tilde{\varphi}_i(t)-\hat{\varphi}_i(t)}{\hat{\tau}_i^2}\right|,$$

we have that $\mathcal{C}_i$ is a continuous bounded function of τ, $\tau\geq 0$ and

$$\|S_i-u_i\| \leq h_i^2\left[\|\delta M\|_2\mathcal{B}_i+\|M\|_2\mathcal{C}_i\frac{\tau^2}{h_i^2}\right],$$

that is from (4.26)

$$\begin{aligned}\|\tilde{S}_i-u_i\| \quad &\leq \quad h_i^2\tau^2\Big\{\max_{i=1,\ldots,N}\frac{\mathcal{B}_i}{(\tilde{\beta}_i-\tilde{\alpha}_i)h_i+(\tilde{\beta}_{i-1}-\tilde{\alpha}_{i-1})h_{i-1}} \\ &\quad\quad \times\left[\max_{i=0,\ldots,N}\frac{\mathcal{A}_i}{h_i}4\|M\|_2+\max_{i=0,N}\frac{\mathcal{A}_i}{h_i}\|g\|_2\right]+\frac{\mathcal{C}_i}{h_i^2}\|M\|_2\Big\}.\end{aligned} \tag{4.27}$$

From (4.27), for each fixed sequences of the values $p_0,\ldots,p_N$, we have a second order convergence of the discrete tension hyperbolic splines to the corresponding continuous one. The results agree with the order of approximation of the discretization which we have used for the first, second and fourth derivatives. For a better understanding of the upper bound (4.27) let us detail the behaviour of $\mathcal{B}_i$, $\mathcal{C}_i$ for different values of the tension parameter p_i. We have

$$\mathcal{B}_i = \frac{2}{p_i^2}\max_{t\in[0,1]}\left|\frac{\sinh(p_it)-t\sinh(p_i)}{\sinh(p_i)}\right| \leq \frac{1}{p_i^2},$$

and using an expansion of the hyperbolic functions in the previous expression as power series, as p_i approaches 0

$$\mathcal{B}_i \leq \frac{2}{9\sqrt{3}}. \tag{4.28}$$

Concerning $\mathcal{C}_i$, we have:

$$\begin{aligned} \lim_{\tau\to 0} \mathcal{C}_i &= \frac{p_i}{12\sinh^2(p_i)} \max_{t\in[0,1]} |\cosh(p_i)\sinh(tp_i) - t\sinh(p_i)\cosh(tp_i)| \\ &=: \mathcal{D}_i \leq p_i, \end{aligned}$$

$$\lim_{p_i\to\infty} \mathcal{D}_i p_i^q = 0, \quad \forall\, q \geq 0,$$

and from the series expansion of the hyperbolic functions, as p_i approaches 0

$$\mathcal{C}_i \leq \frac{p_i^2}{54\sqrt{3}}. \tag{4.29}$$

Finally let us consider in detail the limit case $p_i = 0$, $i = 0, \ldots, N$. From (4.21), (4.25), (4.28), (4.29) we obtain

$$\lim_{p_i\to 0} \mathcal{A}_i = \frac{1}{6}, \quad \lim_{p_i\to 0} (\tilde{\beta}_i - \tilde{\alpha}_i) = \frac{1}{6}, \quad \lim_{p_i\to 0} \mathcal{B}_i \leq 1, \quad \lim_{p_i\to 0} \mathcal{C}_i = 0,$$

so that from (4.27)

$$\|S_i - u_i\| \leq h_i^2\tau^2 \max_{i=1,\ldots,N} \frac{1}{h_i + h_{i-1}} \left[4 \max_{i=0,\ldots,N} \frac{1}{h_i}\|M\|_2 + \max_{i=0,N} \frac{1}{h_i}\|g\|_2 \right],$$

and we recover, with some improvements, the corresponding result of M. A. Malcolm (1977).

Finally, we observe that (4.27) can be used to estimate the rate of convergence of a discrete tension hyperbolic splines towards a function generating the interpolation points as $\max_i h_i \to 0$. To do that it suffices to combine, via triangle inequality, (4.27) with the results of M. Marušić and M. Rogina (1995) where the convergence of a continuous tension hyperbolic spline to a function generating the interpolation points is studied.

4.5 Practical Aspects and Generalizations

The aim of this section is to investigate the practical aspects related to our discrete spline interpolants and to propose a possible generalization for nonuniform subdivisions of the main mesh.

The first problem we want to address is the numerical evaluation of the mesh function $\bar{u}$ defined in section 4.2. The simplest approach is obviously given by a direct computation, using the linear system (4.11). With this choice, we have a symmetric, pentadiagonal, positive definite system and therefore, can use specialized algorithms, with a computational cost of $17m$ arithmetic operations, namely,

$7m$ additions, $7m$ multiplications, and $3m$ divisions H. Späth (1990), where m is the number of unknowns. This is substantially cheaper than performing calculations by the standard algorithm P. Rentrop (1980), which involves the solution of only a 3-diagonal system, but with hyperbolic coefficients, and which requires the evaluaation of hyperbolic functions in formula (4.23). The evaluation of these hyperbolic functions demands a much greater number of computations than the solution of our pentadiagonal system. We also recall that the upper bound for the condition number of the matrix A, given in (4.14), does not depend on the number of interpolation points and thus such methods can be used with some confidence.

We can observe, however, that m can sometimes be very large when the points are irregularly distributed and the tabulation step τ is small (for instance, for the radio chemical data of table 4.1, having chosen τ as the largest step such that h_i/τ is integer, we get $m = 1193$). In some cases, for example for generating a grid in bivariate interpolation, even the linear computational cost may prove to be too expensive, but if we have a parallel machine, we can easily share the computation among the processors as outlined below.

The basic idea is to transform the matrix A of (4.11), which, for $N = 2$, $n_i = 18$ has the form shown in Figure 4.1, into the form K of Figure 4.2. Note that the ith interpolation condition is settled in the r_ith row, where $r_i = 2 + \sum_{\nu=0}^{i-1} n_\nu$. If we extract he first and the last rows from K (which correspond to boundary conditions), then the second and the second last, and, for $i = 0, 1, \ldots, N$ the rows $r_i, \ldots, r_i + 4$, we get a block matrix E of the form shown in Figure 4.3. The corresponding linear system has few equations, and having solved it, it is possible to solve in parallel the $N+1$ linear systems obtained from the "remaining" matrix F of Figure 4.4 by extracting its independent blocks.

The problem now is how to move from A to K. From Sections 4.2, 4.3 and 4.4 we have the following two facts. Having in mind (4.11) and the corresponding Figure 4.1, let us consider the section given by rows $r_i, \ldots, r_{i+1} - 1$. We note that the entries of the columns with index $r_i + 3, \ldots, r_{i+1} - 3$ are $1, a_i, b_i, a_i, 1$ which are the coefficients of the difference equation (4.6). On the other hand, it is shown in Section 4.4 that any function of the form

$$\Upsilon_i(x) = c_1(1-t) + c_2 t + c_3 \widehat{\varphi}_i(1-t) + c_4 \widehat{\varphi}_i(t), \tag{4.30}$$

is a solution for (4.6); therefore if we multiply the row of index $r_i + \nu$, $\nu = 1, \ldots, n_i - 1$, by $\Upsilon_i(x_{i,\nu}) = \Upsilon_i(x_i + \nu\tau)$ and then add all these rows, then the contribution of all the columns from $r_i + 3$ to $r_{i+1} - 3$ sums up to zero. The idea for obtaining the matrix K from A is the following: we replace the four rows of index $r_i + 1, r_i + 2, r_i + 3, r_i + 4$ with the sum of the rows from $r_i + 1$ to $r_{i+1} - 1$ multiplied by the values assumed in $x_{i,\nu}$ by four linearly independent functions of the form (4.30). The remaining question is how to choose these functions. Several numerical experiments have shown that the lowest condition number of the matrix K (which is in general larger than that of A) is achieved when we use the cardinal functions for Lagrange interpolation at the points $x_{i,\nu}$ closest to $x_i, x_i + h_i/3, x_{i+1} - h_i/3, x_{i+1}$.

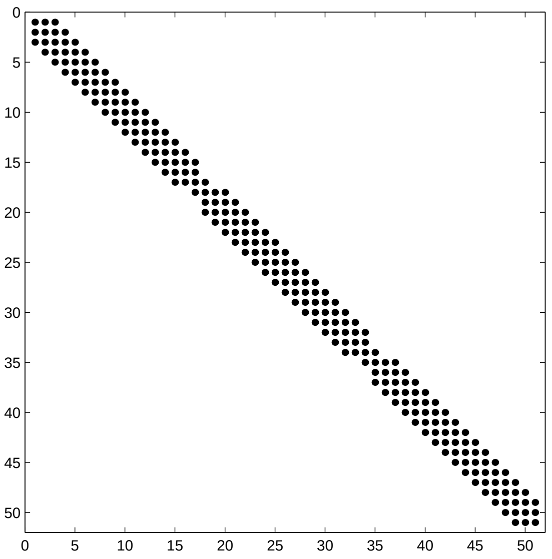

Figure 4.1: The form of the matrix of the system (4.11) for $N = 2,\ n_i = 18$.

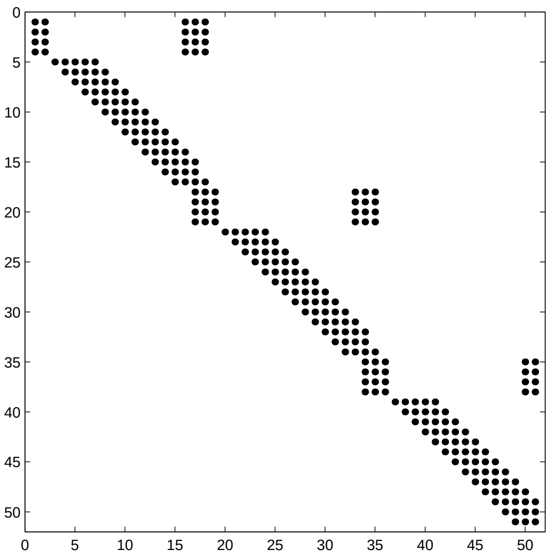

Figure 4.2: The matrix K obtained from the matrix of Figure 4.1 by substitution of the rows as indicated in the text.

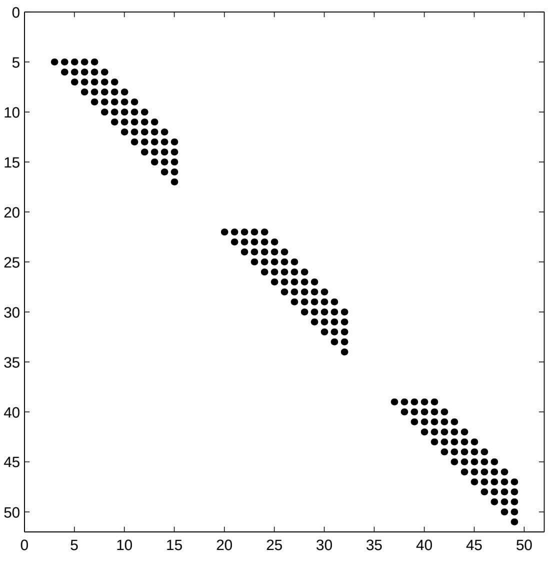

Figure 4.3: The block matrix E.

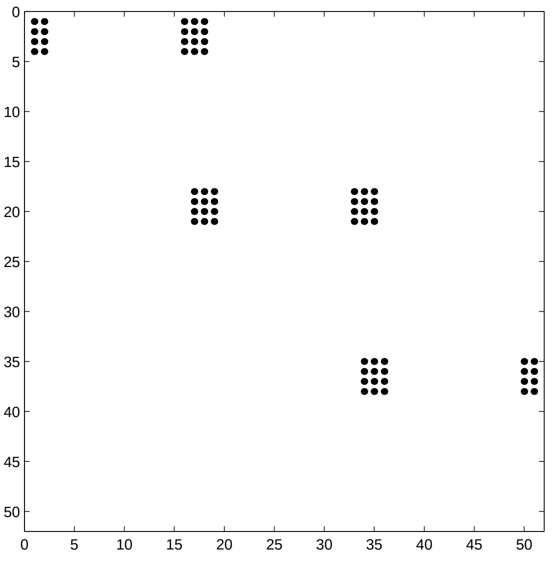

Figure 4.4: The block matrix F.

Another possibility is to directly use the results of section 4.4, that is: first we solve the tridiagonal system (4.20) obtaining the divided differences at the knots $M_0, \dots, M_{N+1}$ and then we compute the refined solution within each interval of the main mesh either solving the $N+1$ tridiagonal systems (4.16), (4.17) or simply evaluating the function of (4.18). We note that this latter approach is the only one possible if we want a continuous extension of the discrete solution beyond the mesh point. At first sight, this approach based on the solution of a tridiagonal system seems preferable because of the limited waste of computational time and the good estimates for the condition number of the matrix in (4.20). However, it should be observed that we have to face the numerical computation of hyperbolic functions of the form $\sinh(k_i t)$ and $\cosh(k_i t)$ - a very difficult task, both for cancellation errors (when $k_i \to 0$) and for overflow problems (when $k_i \to \infty$). A stable computation of the hyperbolic functions was proposed by P. Rentrop (1980), where different formulas for the cases $k_i \leq 0.5$ and $k_i > 0.5$ were considered and a specialized polynomial approximation for $\sinh(\cdot)$ was used. This approach is effective but somewhat complicated, and it is the author's opinion that all these difficulties could be more easily overcome using the features of some new programming languages which allow numerical data type with arbitrary (and user selected) lengths for the fractional and the exponent parts of the floating point representation.

So far, we have considered only *uniform* subdivisions of the main mesh, given by suitable step sizes τ such that for any $i,\ h_i = n_i \tau$. However, it is clear that nonuniform subdivisions are necessary in many applications, for example when we have big changes in the main mesh size or when we want to zoom the solution into a neighborhood of some particular points.

For these reasons, we want to restate – as far as possible – the above theory for a particular nonuniform mesh, that is for the grid

$$\{x_{i,\nu} = x_i + \nu\tau_i;\ i = 0, \dots, N;\ \nu = 0, 1, \dots, n_i \text{ where } n_i = h_i/\tau_i\},$$

in such a way that:

(a) the solution coincides with the uniform one when $\tau_i = \tau$ and

(b) within any interval $[x_i, x_{i+1}]$, the solution can be seen as the mesh restriction of a function u_i of the form (4.18) (having substituted $\hat{\tau}_i = \tau_i/h_i$).

In the uniform case the solution u_i given in (4.18) satisfies the difference equation (4.6) centered at the points $x_{i,\nu},\ \nu = 1, \dots, n_i - 1$ and thus the points involved range from $x_{i,-1} = x_i - \tau$ to $x_{i,n_i+1} = x_{i+1}, \tau$. The smoothness conditions (4.5) involve central divided differences and therefore use evaluations of u_i at $x_i - \tau, x_i, x_i + \tau$. We want to do exactly the same for nonuniform meshes. We start by defining a *redundant* mesh

$$\check{x}_{i,\nu} = x_i + \nu\tau_i;\ i = 0, \dots, N;\ \nu = -1, \dots, n_i + 1$$

and we seek a mesh solution $\check{u}_{i,\nu};\ i = 0, \dots, N;\ \nu = -1, \dots, n_i + 1$ such that

$$\left[\Lambda_i^2 - \left(\frac{p_i}{h_i}\right)^2 \Lambda_i\right] \check{u}_{i,\nu} = 0, \quad \nu = 1, \ldots, n_i - 1, \quad i = 0, \ldots, N, \tag{4.31}$$

where

$$\Lambda_i \check{u}_{i,j} = \frac{\check{u}_{i,\nu-1} - 2\check{u}_{i,\nu} + \check{u}_{i,\nu+1}}{\tau_i^2}.$$

which satisfy the following smoothness conditions

$$\begin{aligned} \check{u}_{i-1,n_{i-1}} &= \check{u}_{i,0}, \\ \frac{\check{u}_{i-1,n_{i-1}+1} - \check{u}_{i-1,n_{i-1}-1}}{2\tau_{i-1}} &= \frac{\check{u}_{i,1} - \check{u}_{i,-1}}{2\tau_i}, \quad i = 1, \ldots, N, \\ \Lambda_{i-1}\check{u}_{i-1,n_{i-1}} &= \Lambda_i \check{u}_{i,0}, \end{aligned} \tag{4.32}$$

and are interpolant, that is

$$\begin{array}{ll} \check{u}_{i,0} = f_i, \quad i = 0, \ldots, N; & \check{u}_{N,n_N} = f_{N+1}, \\ \Lambda_0 \check{u}_{0,0} = f_0''; & \Lambda_N \check{u}_{N,n_N} = f_{N=1}''. \end{array} \tag{4.33}$$

Our discrete solution will then be defined as

$$u_{i,\nu} := \check{u}_{i,\nu}, \ i = 0, \ldots, N; \ \nu = 0, \ldots, n_i, \tag{4.34}$$

and one seems immediately that since $\check{u}_{i,n_i-1} = \check{u}_{i+1,-1}$ and $\check{u}_{i,n_i+1} = \check{u}_{i+1,1}$ for $\tau_i = \tau$ the discrete solution given by (4.31)–(4.34) does coincide with that defined by (4.6)–(4.10) for the uniform subdivision.

It is possible to repeat all the arguments developed so far for this nonuniform mesh. Having added to (4.34) the values $u_{0,-1}$ and u_{N,n_N+1} to satisfy the boundary conditions, we can collect (4.31)–(4.33) in a linear system $A\hat{u} = b$, identical to (4.11) with the exception of the matrix A which has now the form:

$$\left[\begin{array}{cccccccccccccc}
1 & -2 & 1 & 0 & & & & & & & & & & \\
0 & 1 & 0 & 0 & & & & & & & & & & \\
1 & a_0 & b_0 & a_0 & 1 & & & & & & & & & \\
 & 1 & a_0 & b_0 & a_0 & 1 & & & & & & & & \\
 & & & & \cdots & & & & & & & & & \\
 & & 1 & \alpha_{0,n_0-1} & \beta_{0,n_0-1} & \gamma_{0,n_0-1} & \delta_{0,n_0-1} & & & & & & & \\
 & & & 0 & 0 & 1 & 0 & 0 & & & & & & \\
 & & & & \delta_{1,1} & \gamma_{1,1} & \beta_{1,1} & \alpha_{1,1} & 1 & & & & & \\
 & & & & & 1 & a_1 & b_1 & a_1 & 1 & & & & \\
 & & & & & & & \cdots & & & & & & \\
 & & & & & & & & \cdots & & & & & \\
 & & & & & & & & & \cdots & & & & \\
 & & & & & & & & 1 & a_N & b_N & a_N & 1 & \\
 & & & & & & & & & 1 & a_N & b_N & a_N & 1 \\
 & & & & & & & & & & 0 & 0 & 1 & 0 \\
 & & & & & & & & & & 0 & 1 & -2 & 1
\end{array}\right],$$

$$a_i = -(4+\omega_i), \quad b_i = 6+2\omega_i, \quad \omega_i = \left(\frac{p_i\tau_i}{h_i}\right)^2, \quad i=0,\ldots,N,$$

and

$$\begin{aligned}
&\alpha_{i-1,n_{i-1}-1} = -(4+\omega_{i-1}), \quad \beta_{i-1,n_{i-1}-1} = 6+2\omega_{i-1}+\frac{1-\rho_i}{1+\rho_i},\\
&\gamma_{i-1,n_{i-1}-1} = -(4+\omega_{i-1}+2\frac{1-\rho_i}{\rho_i}), \quad \delta_{i-1,n_{i-1}-1} = \frac{2}{\rho_i(\rho_i+1)},\\
&\delta_{i,1} = 2\frac{\rho_i^2}{\rho_i+1}, \quad \gamma_{i,1} = -(4+\omega_i+2(\rho_i-1)),\\
&\beta_{i,1} = 6+2\omega_i+\frac{\rho_i-1}{\rho_i+1}, \quad \alpha_{i,1} = -(4+\omega_i); \quad i=1,\ldots,N,
\end{aligned}$$

with $\rho_i = h_i/h_{i-1}$.

We emphasize that the results of sections four and five can be restated for this nonuniform case without any change; in particular there exist functions u_i of the form (4.18) which are continuous extensions of the discrete solution, that is

$$u_i(\check{x}_{i,\nu}) = \check{u}_{i,\nu}, \quad i=0,\ldots,N; \quad \nu=-1,0,\ldots,n_i+1.$$

We can also repeat the algorithmic considerations developed in the first part of this section; the only substantional difference lies in the fact that now the matrix A is no longer symmetric, and an analysis of its condition number cannot, is contrast with section 4.3, be carried out analytically. However, several numerical experiments have shown that the condition number is not influenced by the nonsymmetric structure, but does depend on the minimum step-size h_i, exactly as in the symmetric case. In other words, symmetric and nonsymmetric matrices, with the same dimension and produced by difference equations with the same smallest step-size, produce very close condition numbers. The nonuniform discrete tension hyperbolic splines have in fact been used for the graphical test of the following section.

4.6 Graphical Examples

The aim of this final section is to illustrate the tension features of discrete tension hyperbolic splines with some (famous) examples. Before, we want to notice that the continuous form u_i of our solution given in (4.18) has the good shape-preserving properties of cubics (see, e.g., P. Rentrop (1980)) in the sense that u_i is convex (concave) in $[x_i, x_{i+1}]$ if and only if $M_{i+j} \geq 0$ (≤ 0), $j=0,1$, and has at most one inflection point in $[x_i, x_{i+1}]$. In order to preserve the shape of the data, we therefore simply have to analyze the values $\Lambda_i u_{i,0}$ and $\Lambda_i u_{i,n_i}$ and increase the tension parameters if necessary. All the strategies proposed for the automatic choice of tension parameters in continuous tension hyperbolic spline interpolation can be used in our discrete context, see, e.g., R. J. Renka (1987), P. Rentrop (1980).

In our first example we have interpolated the *radio chemical* data reported in Table 4.1. The effects of changing the tension values $\omega_i = (p_i\tau_i/h_i)^2$ are depicted in Figures 4.5–4.7. We have adopted a nonuniform mesh, assigning the same number of points (30) to each interval of the main mesh, and imposed *natural* end conditions, that is, following formulae (4.6), $M_0 = M_{N+1} = 0$.

Table 4.1: Radio chemical data:

x_i	7.99	8.09	8.19	8.7	9.2
f_i	0	2.76429E-5	4.37498E-2	0.169183	0.469428
x_i	10	12	15	20	
f_i	0.943740	0.998636	0.999916	0.999994	

Figure 4.5 is obtained setting $p_i = 0$, and the zoom displayed at Figure 4.6 clearly shows the deep decrease – extraneous to data – in the first part of the discrete solution. In Figure 4.7 a new discrete interpolant with $p_0 = p_1 = 300$, $p_i = 15$, $i = 2, \ldots, 7$, is displayed for the same data, and the zoom displayed at Figure 4.8, and the stretching effect of the increase in tension parameters is evident.

Table 4.2: Akima's data:

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

In second example we have taken *Akima's* data of Table 4.2 and constructed discrete interpolants with 20 points for each interval, with the values of M_0 and M_{N+1} equal to the second divided differences of data. Figure 4.9 shows the plot produced by an uniform choice of tension factors, namely $\omega_i = 0$ and Figure 4.10 a second one, which perfectly reproduces the data shape, where we have set $\omega_5 = \omega_6 = \omega_8 = 0.1$.

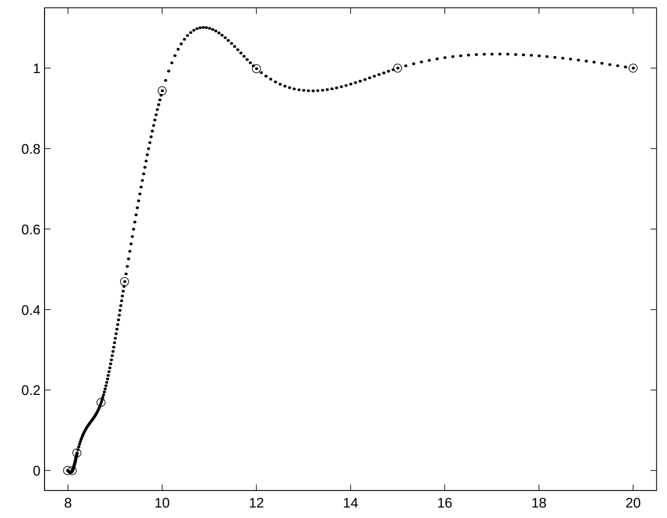

Figure 4.5: The radio chemical data with natural end conditions $M_0 = M_{N+1} = 0$. Interpolation by discrete cubic spline ($p_i = 0$).

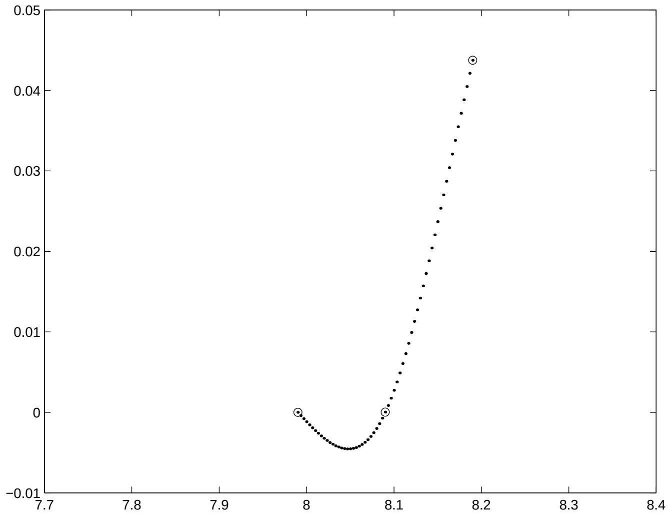

Figure 4.6: A magnification of the lower left corner of Figure 4.5 showing a decreasing part of the curve which contradicts the data.

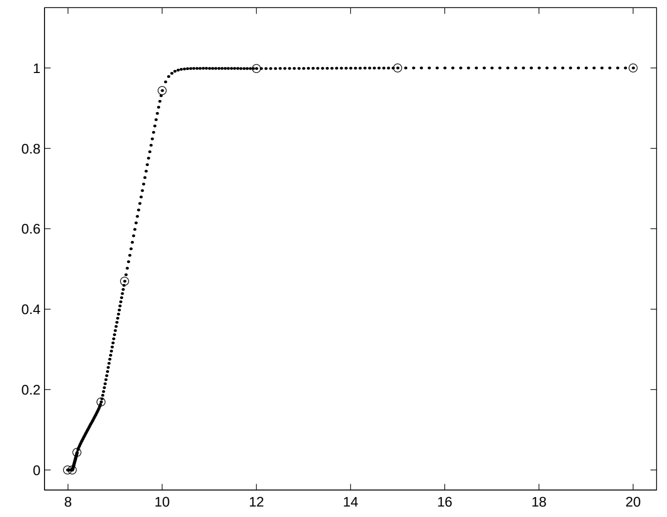

Figure 4.7: The same as Figure 4.5 with $p_0 = p_1 = 300,\ p_i = 15,\ i = 2, \ldots, 7$.

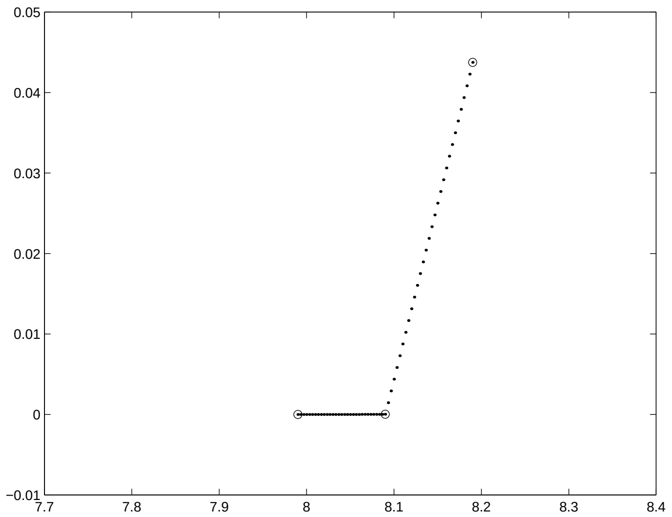

Figure 4.8: A magnification of the lower left corner of Figure 4.7.

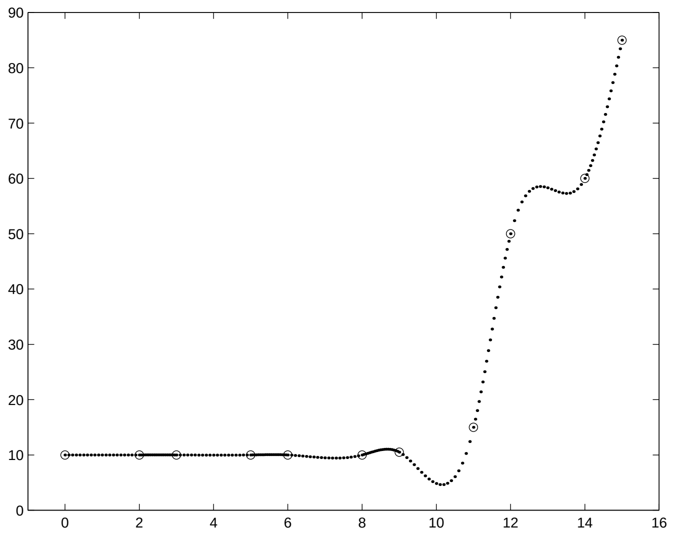

Figure 4.9: Akima's data. Interpolation by discrete tension hyperbolic spline with choice of tension parameters $\omega_i = 0$ for all i and values of M_0 and M_{N+1} equal to the second divided differences of the data.

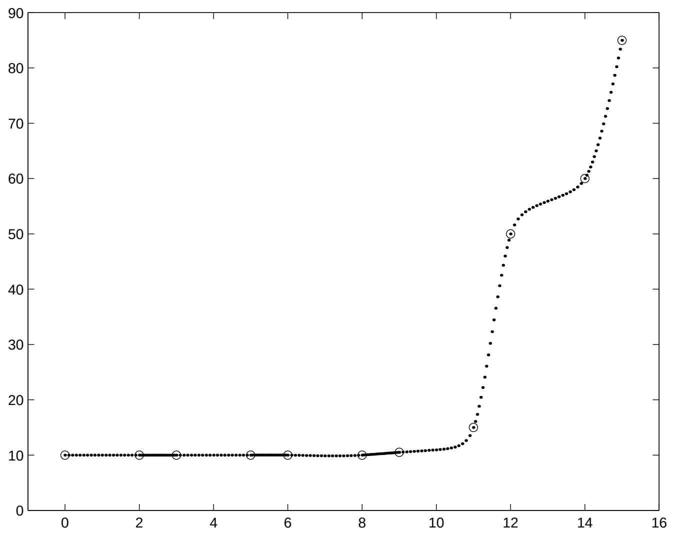

Figure 4.10: Same as Figure 4.9 but with $\omega_5 = \omega_6 = \omega_8 = 0.1$.

Chapter V

Discrete Tension Generalized Splines

This chapter addresses the definition and the study of discrete generalized splines. Discrete generalized splines are continuous piecewise defined functions which meet some smoothness conditions for the first and second divided differences at knots. They provide a generalization both of smooth tension generalized splines and the classical discrete cubic splines. Completely general configurations for steps in divided differences are considered. Direct algorithms are proposed for constructing discrete tension generalized splines and discrete tension generalized B-splines (discrete GB-splines for short). Explicit formulae and recurrence relations are obtained for discrete GB-splines. Properties of discrete GB-splines and their series are studied. It is shown that discrete GB-splines form weak Chebyshev systems and that series of discrete GB-splines have a variation diminishing property.

5.1 Discrete Tension Generalized Splines. Conditions of Existence and Uniqueness

Let a partition $\Delta : a = x_0 < x_1 < \cdots < x_N = b$ of the interval $[a, b]$ be given, to which we associate a space of functions S_4^{DG} whose restriction to a subinterval $[x_i, x_{i+1}],\ i = 0, \ldots, N-1$ is spanned by the system of linearly independent functions $\{1, x, \Phi_i, \Psi_i\}$ and where every function in S_4^{DG} is continuous and for given $\tau_i^{L_j} > 0$ and $\tau_i^{R_j} > 0$, $j = i-1, i$, the values of its first and second divided differences with respect to the points $x_i - \tau_i^{L_{i-1}}$, x_i, $x_i + \tau_i^{R_{i-1}}$ and $x_i - \tau_i^{L_i}$, x_i, $x_i + \tau_i^{R_i}$ coincide.

Given a continuous function S we introduce the linear difference operators

$$\begin{aligned}
D_1S(x) \equiv D_{i,1}S(x) &= (\lambda_i^{R_i} S[x - \tau_i^{L_i}, x] + \lambda_i^{L_i} S[x, x + \tau_i^{R_i}])(1 - t) \\
&\quad + (\lambda_{i+1}^{R_i} S[x - \tau_{i+1}^{L_i}, x] + \lambda_{i+1}^{L_i} S[x, x + \tau_{i+1}^{R_i}])t, \\
D_2S(x) \equiv D_{i,2}S(x) &= 2S[x - \tau_i^{L_i}, x, x + \tau_i^{R_i}](1 - t) + 2S[x - \tau_{i+1}^{L_i}, x, x + \tau_{i+1}^{R_i}]t \\
&\quad x \in [x_i, x_{i+1}) \quad i = 0, \ldots, N-1,
\end{aligned}$$

where $\lambda_j^{R_i} = 1 - \lambda_j^{L_i} = \tau_j^{R_i}/(\tau_j^{L_i} + \tau_j^{R_i}),\ j = i, i+1$ and $t = (x - x_i)/h_i$, $h_i = x_{i+1} - x_i$.

The square parentheses denote the usual first and second divided differences of the function S by the argument values $x_j - \tau_j^{L_i},\ x_j,\ x_j + \tau_j^{R_i},\ j = i, i+1$.

Definition 5.1. *A discrete generalized spline is a function $S \in S_4^{DG}$ such that*

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

$$\begin{aligned} S(x) \equiv S_i(x) &= [S(x_i) - \Phi_i(x_i)M_i](1-t) + [S(x_{i+1}) - \Psi_i(x_{i+1})M_{i+1}]t \\ &\quad + \Phi_i(x)M_i + \Psi_i(x)M_{i+1}, \end{aligned} \tag{5.1}$$

where $M_j = D_{i,2}S_i(x_j),\ j = i, i+1$, and the functions Φ_i and Ψ_i are subject to the constraints

$$\begin{array}{ll} \Phi_i(x_{i+1} - \tau_{i+1}^{L_i}) = \Phi_i(x_{i+1}) = \Phi_i(x_{i+1} + \tau_{i+1}^{R_i}) = 0, & D_{i,2}\Phi_i(x_i) = 1, \\ \Psi_i(x_i - \tau_i^{L_i}) = \Psi_i(x_i) \quad = \Psi_i(x_i + \tau_i^{R_i}) \quad = 0, & D_{i,2}\Psi_i(x_{i+1}) = 1; \end{array} \tag{5.2}$$

(ii) S satisfies the continuity conditions

$$\begin{aligned} S_{i-1}(x_i) &= S_i(x_i), \\ D_{i-1,1}S_{i-1}(x_i) &= D_{i,1}S_i(x_i), \quad i = 1, \dots, N-1. \\ D_{i-1,2}S_{i-1}(x_i) &= D_{i,2}S_i(x_i), \end{aligned} \tag{5.3}$$

This definition generalizes the notion of a discrete polynomial spline in L. L. Schumaker (1981) and of a tension generalized spline in B. I. Kvasov (1995, 1996b). The latter type can be obtained by setting $\tau_j^{L_i} = \tau_j^{R_i} = 0,\ j = i, i+1$ for all i. If $\tau_i^{L_j} = \tau_i^L$ and $\tau_i^{R_j} = \tau_i^R,\ j = i-1, i$ then according to smoothness conditions (5.3) the values of the functions S_{i-1} and S_i at the three consecutive points $x_i - \tau_i^L,\ x_i,\ x_i + \tau_i^R$ coincide. Setting $\tau_j^{L_i} = \tau_j^{R_i} = \tau_i,\ j = i, i+1$ we obtain $D_{1,i}S(x) = S[x - \tau_i, x + \tau_i]$ and $D_{2,i}S(x) = S[x - \tau_i, x, x + \tau_i]$ which is the case discussed in P. Costantini *et al.* (1999).

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 $\Phi_i(x) = \Phi_i(p_i, x),\ \Psi_i(x) = \Psi_i(q_i, x),\ 0 \le p_i, q_i < \infty$. 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 (5.1) turns into a linear function. Additionally, we require that if $p_i = q_i = 0$ for all i we get a discrete cubic spline with

$$\begin{aligned} \Phi_i(x) &= \frac{1}{2} \frac{(x_{i+1} - x - \tau_{i+1}^{L_i})(x_{i+1} - x)(x_{i+1} - x + \tau_{i+1}^{R_i})}{3h_i + \varepsilon_{i+1} - \varepsilon_i}, \\ \Psi_i(x) &= \frac{1}{2} \frac{(x - x_i + \tau_i^{L_i})(x - x_i)(x - x_i - \tau_i^{R_i})}{3h_i + \varepsilon_{i+1} - \varepsilon_i}, \\ \varepsilon_j &= \tau_j^{R_i} - \tau_j^{L_i}, \quad j = i, i+1. \end{aligned} \tag{5.4}$$

If $\tau_i^{L_j} = \tau_i^{R_j} = \tau_i,\ j = i-1, i$ for all i then this spline coincides with a discrete cubic spline of Yu. S. Zavyalov *et al.* (1980). The case $\tau_i = \tau$ for all i was considered in T. Lyche (1976).

Introducing the notation $m_j \equiv D_{i,1}S_i(x_j),\ j = i, i+1,\ i = 0, \dots, N-1$, we can obtain less restrictive conditions on the functions Φ_i and Ψ_i to guarantee existence and uniqueness of the discrete hyperbolic spline S than could be achieved by using the unknowns M_i.

From the equations $D_{i,1}S_i(x_j) = m_j,\ j = i, i+1$, we find using (5.1)

$$\begin{aligned} M_i &= \frac{h_i}{T_i}\{[\bar{b}_i + \Psi_i(x_{i+1})]S[x_i, x_{i+1}] - \bar{b}_i m_i - \Psi_i(x_{i+1})m_{i+1}\}, \\ M_{i+1} &= \frac{h_i}{T_i}\{-[\bar{a}_i + \Phi_i(x_i)]S[x_i, x_{i+1}] + \Phi_i(x_i)m_i - \bar{a}_i m_{i+1}\}, \\ T_i &= \bar{a}_i\bar{b}_i - \Phi_i(x_i)\Psi_i(x_{i+1}), \end{aligned} \tag{5.5}$$

where

$$\begin{aligned} \bar{a}_i &= -\Phi_i(x_i) - h_i D_{i,1}\Phi_i(x_i), \\ \bar{b}_i &= -\Psi_i(x_{i+1}) + h_i D_{i,1}\Psi_i(x_{i+1}). \end{aligned} \tag{5.6}$$

The third equation in the smoothness conditions (5.3) and the boundary relations $M_0 = D_{0,2}S(a)$ and $M_N = D_{0,2}S(b)$ result in the following system of linear algebraic equations

$$\begin{aligned} &\bar{b}_0 m_0 + \Psi_0(x_1)m_1 = [\bar{b}_0 + \Psi_0(x_1)]S[x_0, x_1] - D_{0,2}S(a)\frac{T_0}{h_0}, \\ &\Phi_{i-1}(x_{i-1})\frac{h_{i-1}}{T_{i-1}}m_{i-1} + \left(\bar{a}_{i-1}\frac{h_{i-1}}{T_{i-1}} + \bar{b}_i\frac{h_i}{T_i}\right)m_i + \Psi_i(x_{i+1})\frac{h_i}{T_i}m_{i+1} \\ &= [\bar{a}_{i-1} + \Phi_{i-1}(x_{i-1})]\frac{h_{i-1}}{T_{i-1}}S[x_{i-1}, x_i] + [\bar{b}_i + \Phi_i(x_{i+1})]\frac{h_i}{T_i}S[x_i, x_{i+1}], \\ &\qquad\qquad\qquad\qquad\qquad\qquad i = 1, \dots, N-1, \\ &\Phi_{N-1}(x_{N-1})m_{N-1} + \bar{a}_{N-1}m_N = D_{N,2}S(b)\frac{h_{N-1}}{T_{N-1}} \\ &\qquad\qquad +[\bar{a}_{N-1} + \Phi_{N-1}(x_{N-1})]S[x_{N-1}, x_N]. \end{aligned} \tag{5.7}$$

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

Lemma 5.1. *If the conditions*

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

are satisfied, where $\bar{a}_i$ and $\bar{b}_i$ are as defined in (5.6), then the discrete tension generalized spline S exists and is unique.

Proof: It follows from the conditions of the lemma that

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

Then

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

Therefore, by virtue of the conditions of the lemma, the matrix of system (5.7) is diagonally dominant:

$$\begin{aligned}\bar{r}_0 &= \bar{b}_0 - \Psi_0(x_1) > 0,\\ \bar{r}_i &= [\bar{a}_{i-1} - \Phi_{i-1}(x_{i-1})]\frac{h_{i-1}}{T_{i-1}} + [\bar{b}_i - \Psi_i(x_{i+1})]\frac{h_i}{T_i} > 0, \quad i = 1, \ldots, N-1,\\ \bar{r}_N &= \bar{a}_{N-1} - \Phi_{N-1}(x_{N-1}) > 0.\end{aligned}$$

This ensures (e.g., see Yu. S. Zavyalov *et al.* (1980)) that the spline S exists and is unique, and proves the lemma. $\square$

In practical examples, the conditions of Lemma 5.1 are satisfied for the majority of discrete tension generalized splines. This allows one to construct the splines, that is, to solve the tridiagonal linear system (5.7), efficiently by a special version of Gaussian elimination that avoids pivoting.

5.2 Construction of Discrete GB-Splines

Let us construct a basis for the space of discrete tension generalized splines S_4^{DG} by using functions which have local supports of minimum length. Since $dim(S_4^{DG}) = 4N - 3(N-1) = N+3$ we extend the grid Δ by adding the points x_j, $j = -3, -2, -1, N+1, N+2, N+3$, such that $x_{-3} < x_{-2} < x_{-1} < a$, $b < x_{N+1} < x_{N+2} < x_{N+3}$.

We demand that the discrete GB-splines B_i, $i = -3, \ldots, N-1$ have the properties

$$\begin{aligned}\mathrm{B}_i(x) &> 0, \quad x \in (x_i + \tau_i^{R_i}, x_{i+4} - \tau_{i+4}^{L_{i+3}}), &(5.8)\\ \mathrm{B}_i(x) &\equiv 0, \quad x \notin (x_i, x_{i+4}),\\ \sum_{j=-3}^{N-1} \mathrm{B}_j(x) &\equiv 1, \quad x \in [a, b]. &(5.9)\end{aligned}$$

According to (5.1), on the interval $[x_j, x_{j+1}]$, $j = i, \ldots, i+3$, the discrete GB-spline B_i has the form

$$\mathrm{B}_i(x) \equiv \mathrm{B}_{j,i}(x) = P_{i,j}(x) + \Phi_j(x) M_{j,\mathrm{B}_i} + \Psi_j(x) M_{j+1,\mathrm{B}_i}, \qquad (5.10)$$

where $P_{i,j}$ is a polynomial of the first degree and $M_{l,\mathrm{B}_i} = D_{j,2} B_i(x_l)$, $l = j, j+1$ are constants to be determined. The smoothness conditions (5.3) and constraints (5.2) give the following relations

$$\begin{aligned}P_{i,j}(x_j) &= P_{i,j-1}(x_j) + z_j M_{j,\mathrm{B}_i},\\ D_{j,1} P_{i,j}(x_j) &= D_{j-1,1} P_{i,j-1}(x_j) + c_{j-1,2} M_{j,\mathrm{B}_i},\end{aligned}$$

where

$$\begin{aligned}z_j &\equiv z_j(x_j) = \Psi_{j-1}(x_j) - \Phi_j(x_j),\\ c_{j-1,2} &= D_{j-1,1}\Psi_{j-1}(x_j) - D_{j,1}\Phi_j(x_j).\end{aligned}$$

Thus

$$P_{i,j}(x) = P_{i,j-1}(x) + [z_j + c_{j-1,2}(x - x_j)]M_{j,\mathrm{B}_i} \tag{5.11}$$

By repeated use of this formula we get

$$P_{i,j}(x) = \sum_{l=i+1}^{j} [z_l + c_{l-1,2}(x - x_l)]M_{l,\mathrm{B}_i} = -\sum_{l=j+1}^{i+3} [z_l + c_{l-1,2}(x - x_l)]M_{l,\mathrm{B}_i}.$$

As B_i vanishes outside the interval (x_i, x_{i+4}), we have from (5.11) that $P_{i,j} \equiv 0$ for $j = i, i+3$. In particular, the following identity is valid

$$\sum_{j=i+1}^{i+3} [z_j + c_{j-1,2}(x - x_j)]M_{j,\mathrm{B}_i} \equiv 0,$$

from which one obtains the equalities

$$\sum_{j=i+1}^{i+3} c_{j-1,2} y_j^r M_{j,\mathrm{B}_i} = 0, \quad r = 0, 1, \quad y_j = x_j - \frac{z_j}{c_{j-1,2}}. \tag{5.12}$$

Thus the formula for the discrete GB-spline B_i takes the form

$$\mathrm{B}_i(x) = \begin{cases} \Psi_i(x)M_{i+1,\mathrm{B}_i}, & x \in [x_i, x_{i+1}), \\ (x - y_{i+1})c_{i,2}M_{i+1,\mathrm{B}_i} + \Phi_{i+1}(x)M_{i+1,\mathrm{B}_i} + \Psi_{i+1}(x)M_{i+2,\mathrm{B}_i}, & \\ & x \in [x_{i+1}, x_{i+2}), \\ (y_{i+3} - x)c_{i+2,2}M_{i+3,\mathrm{B}_i} + \Phi_{i+2}(x)M_{i+2,\mathrm{B}_i} + \Psi_{i+2}(x)M_{i+3,\mathrm{B}_i}, & \\ & x \in [x_{i+2}, x_{i+3}), \\ \Phi_{i+3}(x)M_{i+3,\mathrm{B}_i}, & x \in [x_{i+3}, x_{i+4}), \\ 0, & \text{otherwise.} \end{cases} \tag{5.13}$$

Substituting formula (5.13) into the normalization condition (5.9) written for $x \in [x_i, x_{i+1}]$, we obtain

$$\begin{aligned} \sum_{j=i-3}^{i} \mathrm{B}_j(x) &= \Phi_i(x) \sum_{j=i-3}^{i-1} M_{i,\mathrm{B}_j} + \Psi_i(x) \sum_{j=i-2}^{i} M_{i+1,\mathrm{B}_j} \\ &\quad + (y_{i+1} - x)c_{i,2}M_{i+1,\mathrm{B}_{i-2}} + (x - y_i)c_{i-1,2}M_{i,\mathrm{B}_{i-1}} \equiv 1. \end{aligned}$$

As according to (5.9)

$$\sum_{j=i-3}^{i-1} M_{i,\mathrm{B}_j} = \sum_{j=i-2}^{i} M_{i+1,\mathrm{B}_j} = 0 \tag{5.14}$$

the following identity is valid

$$(y_{i+1} - x)c_{i,2}M_{i+1,\mathrm{B}_{i-2}} + (x - y_i)c_{i-1,2}M_{i,\mathrm{B}_{i-1}} \equiv 1.$$

From here one gets the equalities

$$y_{i+1}^r c_{i,2}M_{i+1,\mathrm{B}_{i-2}} - y_i^r c_{i-1,2}M_{i,\mathrm{B}_{i-1}} \equiv \delta_{1,r}, \quad r = 0, 1,$$

where $\delta_{1,r}$ is the Kronecker symbol. Solving this system of equations and using (5.12) or (5.14), we obtain

$$\begin{aligned} M_{j,\mathrm{B}_i} &= \frac{y_{i+3}-y_{i+1}}{c_{j-1,2}\omega'_{i+1}(y_j)}, \quad j=i+1,i+2,i+3, \\ \omega_{i+1}(x) &= (x-y_{i+1})(x-y_{i+2})(x-y_{i+3}) \end{aligned}$$

or with the notation $c_{j,3}=y_{j+2}-y_{j+1}$, $j=i,i+1$,

$$\begin{aligned} M_{i+1,\mathrm{B}_i} &= \frac{1}{c_{i,2}c_{i,3}}, \\ M_{i+2,\mathrm{B}_i} &= -\frac{1}{c_{i+1,2}}\Big(\frac{1}{c_{i,3}}+\frac{1}{c_{i+1,3}}\Big), \\ M_{i+3,\mathrm{B}_i} &= \frac{1}{c_{i+2,2}c_{i+1,3}}. \end{aligned} \tag{5.15}$$

5.3 Properties of Discrete GB-Splines

If the inequalities of Lemma 5.1 are satisfied at only the interior nodes of the support interval of the discrete GB-spline B_i, then we obtain the following result.

Lemma 5.2. *If the conditions*

$$0<\Psi_{j-1}(x_j)<\bar b_{j-1}, \quad 0<\Phi_j(x_j)<\bar a_j, \quad j=i+1,i+2,i+3$$

are satisfied, where $\bar b_{j-1}$ and $\bar a_j$ are as defined in (5.6), then in (5.15) $c_{j,k}>0$, $j=i,\dots,i+3-k$; $k=1,2$, and

$$(-1)^{j-i-1}M_{j,\mathrm{B}_i}>0, \quad j=i+1,i+2,i+3. \tag{5.16}$$

Proof: The conditions of the lemma can be rewritten in the form

$$\begin{aligned} 0 &< \frac{2}{h_{j-1}}\Psi_{j-1}(x_j) < D_{j-1,1}\Psi_{j-1}(x_j), \\ 0 &< \frac{2}{h_j}\Phi_j(x_j) < -D_{j,1}\Phi_j(x_j), \qquad j=i+1,i+2,i+3. \end{aligned} \tag{5.17}$$

Taking the sum of these equations we obtain

$$c_{j-1,2}=D_{j-1,1}\Psi_{j-1}(x_j)-D_{j,1}\Phi_j(x_j)>0, \quad j=i+1,i+2,i+3.$$

The inequalities

$$x_j-h_{j-1}/2<y_j<x_j+h_j/2, \quad j=i+1,i+2,i+3 \tag{5.18}$$

are equivalent to the relations

$$\begin{aligned} 0 &< [-\Psi_{j-1}(x_j)+\frac{h_{j-1}}{2}D_{j-1,1}\Psi_{j-1}(x_j)]+[\Phi_j(x_j)-\frac{h_{j-1}}{2}D_{j,1}\Phi_j(x_j)], \\ 0 &< [\Psi_{j-1}(x_j)+\frac{h_j}{2}D_{j-1,1}\Psi_{j-1}(x_j)]-[\Phi_j(x_j)+\frac{h_j}{2}D_{j,1}\Phi_j(x_j)], \\ &\qquad j=i+1,i+2,i+3, \end{aligned}$$

which are obviously satisfied by the conditions (5.17). From (5.18) we obtain $c_{j,3} = y_{j+2} - y_{j+1} > 0$, $j = i, i+1$. Now it follows from (5.15) that if the conditions of the lemma are satisfied, then the inequalities (5.16) hold. This proves the lemma. □

The functions B_j, $j = -3, \ldots, N-1$ possess many of the properties inherent in the usual discrete polynomial B-splines. To provide inequality (5.8) in what follows we will assume in addition that $D_{j,2}\Phi_j$ and $D_{j,2}\Psi_j$ are strictly monotone functions on the interval $[x_j, x_{j+1}]$.

Theorem 5.1. *Let the conditions of Lemma 5.1 be satisfied, the functions Φ_j and Ψ_j be convex and $D_{j,2}\Phi_j$ and $D_{j,2}\Psi_j$ be strictly monotone on the interval $[x_j, x_{j+1}]$. Then the functions B_j, $j = -3, \ldots, N-1$ have the following properties:*

1. $\mathrm{B}_j(x) > 0$ *for* $x \in (x_j + \tau_j^{R_j}, x_{j+4} - \tau_{j+4}^{L_{j+3}})$, *and* $\mathrm{B}_j(x) \equiv 0$ *if* $x \notin (x_j, x_{j+4})$;
2. B_j *satisfies the smoothness conditions (5.3);*
3. $\sum_{j=-3}^{N-1} y_{j+2}^r \mathrm{B}_j(x) \equiv x^r$, $r = 0, 1$ *for* $x \in [a, b]$, $\quad \Phi_j(x) = c_{j-1,2}c_{j-2,3}\mathrm{B}_{j-3}(x)$,

   $\Psi_j(x) = c_{j,2}c_{j,3}\mathrm{B}_j(x)$ *for* $x \in [x_j, x_{j+1}]$, $j = 0, \ldots, N-1$.

Proof: By assumption, the functions $D_{j,2}\Phi_j$ and $D_{j,2}\Psi_j$ are strictly monotone. Since Φ_j and Ψ_j satisfy the constraints (5.2) we conclude that the functions Φ_j and Ψ_j are convex. Furthermore, in virtue of the restriction (5.17) we obtain that Φ_j is positive and monotonically decreasing on the interval $[x_j, x_{j+1} - \tau_{j+1}^{L_j})$ and Ψ_j is positive and monotonically increasing on the interval $(x_j + \tau_j^{R_j}, x_{j+1}]$.

Using formula (5.13) and inequalities (5.16) we see that the function B_j is positive and increases (decreases) monotonically on the interval $(x_j + \tau_j^{R_j}, x_{j+1}]$ (on the interval $[x_{j+3}, x_{j+4} - \tau_{j+4}^{L_{j+3}})$).

For $x \in [x_l, x_{l+1}]$, $l = j+1, j+2$, we have

$$D_2\mathrm{B}_j(x) = D_{l,2}\Phi_l(x)M_{l,\mathrm{B}_j} + D_{l,2}\Psi_l(x)M_{l+1,\mathrm{B}_j}.$$

As the sum of two strictly monotonically decreasing (increasing) functions the function $D_2\mathrm{B}_j$ decreases monotonically on $[x_{j+1}, x_{j+2}]$ and increases monotonically on $[x_{j+2}, x_{j+3}]$. Then there exist points $\zeta_l \in [x_l, x_{l+1}]$, $l = j+1, j+2$, such that $D_2\mathrm{B}_j(\zeta_l) = 0$. Since by (5.13), (5.16), and (5.17) $D_{j+1,1}\mathrm{B}_j(x_{j+1}) > 0$ and $D_{j+3,1}\mathrm{B}_j(x_{j+3}) < 0$ we find that the discrete GB-spline B_j is positive, increases monotonically on $[x_{j+1}, \zeta_{j+1}]$, and decreases monotonically on $[\zeta_{j+2}, x_{j+3}]$. The function B_j is concave and also positive on the interval $[\zeta_{j+1}, \zeta_{j+2}]$.

All the remaining properties of the discrete GB-splines B_j, $j = -3, \ldots, N-1$ which are formulated in (2)–(3) of Theorem 5.1, follow directly from formula (5.13). This proves the theorem. □

Lemma 5.3. *The function B_i has support of minimum length.*

Proof: It is clear that the function B_i cannot be different from zero on only a part of the interval $[x_j, x_{j+1}]$, $j = i, i+3$. If we suppose that B_i vanishes outside the interval (x_{i+1}, x_{i+4}), then due to the continuity of $D_2\mathrm{B}_i$, we have $M_{i+1,\mathrm{B}_i} = 0$. But then it follows from the equalities (5.12) that $M_{j,\mathrm{B}_i} = 0$, $j = i+2, i+3$, and according to (5.13) we obtain $\mathrm{B}_i \equiv 0$. If we suppose that B_i vanishes outside the interval (x_i, x_{i+3}), we arrive at the same result. This proves the lemma. □

Theorem 5.2. *The functions* B_i, $i = -3, \ldots, N-1$, *are linearly independent and form a basis of the space* S_4^{DG} *of discrete generalized splines.*

Proof: Let us assume to the contrary that there exist constants $\bar{c}_i$, $i = -3, \ldots, N-1$, which are not all equal to zero and such that

$$\bar{c}_{-3}\mathrm{B}_{-3}(x) + \cdots + \bar{c}_{N-1}\mathrm{B}_{N-1}(x) = 0, \quad x \in [a, b]. \tag{5.19}$$

Because the functions B_i, $i = -3, \ldots, N-1$ have finite supports, in the sum (5.19) only the four terms with subscripts $i-3, \ldots, i$ do not vanish on the interval $[x_i, x_{i+1}]$. Hence taking into account formulae (5.13) and (5.15), we have

$$\begin{aligned}\sum_{j=i-3}^{i} \bar{c}_j B_j(x) &= \Phi_i(x) \sum_{j=i-3}^{i-1} \bar{c}_j M_{i,B_j} + \Psi_i(x) \sum_{j=i-2}^{i} \bar{c}_j M_{i+1,B_j} \\ &\quad + \bar{c}_{i-2}\frac{y_{i+1} - x}{y_{i+1} - y_i} + \bar{c}_{i-1}\frac{x - y_i}{y_{i+1} - y_i} = 0, \quad x \in [x_i, x_{i+1}].\end{aligned}$$

Since the functions $\{t, x, \Phi_i, \Psi_i\}$ are linearly independent on the interval $[x_i, x_{i+1}]$ and $M_{k,B_j} \neq 0$, $k = i, i+1$, it follows that $\bar{c}_{i-3} = \cdots = \bar{c}_i = 0$. Continuing this process, we find that $\bar{c}_i = 0$ for all i.

Since $\dim(S_4^{DG}) = N+3$, we see that the discrete GB-splines B_i, $i = -3, \ldots, N-1$, all of which are elements of the space S_4^{DG}, form a basis of this space. This proves the theorem. □

5.4 Local Approximation by Discrete GB-Splines

According to Theorem 5.2, any discrete generalized spline $S \in S_4^{DG}$ can be uniquely written in the form

$$S(x) = \sum_{j=-3}^{N-1} b_j B_j(x) \tag{5.20}$$

for some constant coefficients b_j.

If the coefficients b_j in (5.20) are known, then by virtue of formula (5.13) we can write out an expression for the discrete generalized spline S on the interval $[x_i, x_{i+1}]$, which is convenient for calculations,

$$S(x) = b_{i-2} + b_{i-1}^{(1)}(x - y_i) + b_{i-1}^{(2)}\Phi_i(x) + b_i^{(2)}\Psi_i(x), \tag{5.21}$$

where

$$b_j^{(k)} = \frac{b_j^{(k-1)} - b_{j-1}^{(k-1)}}{c_{j,4-k}}, \quad k = 1, 2;\ b_j^{(0)} = b_j. \tag{5.22}$$

The representations (5.20) and (5.21) allow us to find a simple and effective way to approximate a given function f from its samples.

Theorem 5.3. *Let a function $f \in C[a,b]$ be given by its samples $f(y_j)$, $j = -1, \dots, N+1$. Then for $b_j = f(y_{j+2})$, $j = -3, \dots, N-1$, formula (5.20) is exact for polynomials of the first degree and provides a formula for local approximation.*

Proof: It suffices to prove that the identities

$$\sum_{j=-3}^{N-1} y_{j+2}^r \mathrm{B}_j(x) \equiv x^r, \quad r = 0, 1 \tag{5.23}$$

hold for $x \in [a,b]$. Using formula (5.21) with the coefficients $b_{j-2} = 1$ and $b_{j-2} = y_j$, $j = i-1, i, i+1, i+2$, for an arbitrary interval $[x_i, x_{i+1}]$, we find that identities (5.23) hold.

For $b_{j-2} = f(y_j)$, formula (5.21) can be rewritten as

$$\begin{aligned} S(x) \;\; = \;\; & f(y_i) + f[y_i, y_{i+1}](x - y_i) + (y_{i+1} - y_{i-1}) f[y_{i-1}, y_i, y_{i+1}] c_{i-1,2}^{-1} \Phi_i(x) \\ & + (y_{i+2} - y_i) f[y_i, y_{i+1}, y_{i+2}] c_{i,2}^{-1} \Psi_i(x), \quad x \in [x_i, x_{i+1}]. \end{aligned}$$

This is the formula of local approximation. The theorem is proved. □

Corollary 5.1. *Let a function $f \in C[a,b]$ be given by its samples $f_j = f(x_j)$, $j = -2, \dots, N+2$. Then by setting*

$$b_{j-2} = f_j - \frac{1}{c_{j-1,2}} \Big[\Psi_{j-1}(x_j) f[x_j, x_{j+1}] - \Phi_j(x_j) f[x_{j-1}, x_j] \Big] \tag{5.24}$$

in (5.20), we obtain a formula of three-point local approximation, which is exact for polynomials of the first degree.

Proof: To prove the corollary, it is sufficient to take the monomials 1 and x as f. Then according to (5.24), we obtain $b_{j-2} = 1$ and $b_{j-2} = y_j$ and it only remains to make use of identities (5.24). This proves the corollary. □

Equation (5.21) permits us to write the coefficients of the spline S in its representation (5.20) of the form

$$b_{j-2} = \begin{cases} S(y_j) - D_{j-1,2} S(x_{j-1}) \Phi_{j-1}(y_j) - D_{j,2} S(x_j) \Psi_{j-1}(y_j), & y_j < x_j \\ S(y_j) - D_{j,2} S(x_j) \Phi_j(y_j) - D_{j+1,2} S(x_{j+1}) \Psi_j(y_j), & y_j \geq x_j \end{cases}$$

According to this formula we have $b_{j-2} = S(y_j) + O(\bar{h}_j^2)$, $\bar{h}_j = \max(h_{j-1}, h_i)$. Hence it follows that the control polygon (e.g., see P. E. Koch and T. Lyche (1989)) converges quadratically to the function f for $b_{j-2} = f(y_j)$, or if the formula (5.24) is used.

5.5 Recurrence Formulae for Discrete GB-Splines

Let us define functions

$$\mathrm{B}_{j,2}(x) = \begin{cases} D_{j,2}\Psi_j(x), & x \in [x_j, x_{j+1}), \\ D_{j+1,2}\Phi_{j+1}(x), & x \in [x_{j+1}, x_{j+2}], \\ 0, & \text{otherwise}, \end{cases} \quad j = i, i+1, i+2. \tag{5.25}$$

We assume that the functions $D_{j,2}\Psi_j$ and $D_{j+1,2}\Phi_{j+1}$ are strictly monotone on $[x_j, x_{j+1})$ and $[x_{j+1}, x_{j+2}]$ respectively. The splines $\mathrm{B}_{j,2}$ are a generalization of the "hat-functions" for polynomial B-splines. They are nonnegative and, furthermore, $\mathrm{B}_{j,2}(x_{j+l}) = \delta_{1,l}$, $l = 0, 1, 2$.

According to (5.13), (5.15) and (5.25) the function $D_2\mathrm{B}_i$ can be written in the form

$$\begin{aligned} D_2\mathrm{B}_i(x) &= \sum_{j=i+1}^{i+3} M_{j,\mathrm{B}_i}\mathrm{B}_{j-1,2}(x) \\ &= \frac{1}{c_{i,3}}\Big(\frac{\mathrm{B}_{i,2}(x)}{c_{i,2}} - \frac{\mathrm{B}_{i+1,2}(x)}{c_{i+1,2}}\Big) - \frac{1}{c_{i+1,3}}\Big(\frac{\mathrm{B}_{i+1,2}(x)}{c_{i+1,2}} - \frac{\mathrm{B}_{i+2,2}(x)}{c_{i+2,2}}\Big). \end{aligned} \tag{5.26}$$

The function $D_1\mathrm{B}_i$ satisfies to the relation

$$D_1\mathrm{B}_i(x) = \frac{\mathrm{B}_{i,3}(x)}{c_{i,3}} - \frac{\mathrm{B}_{i+1,3}(x)}{c_{i+1,3}}, \tag{5.27}$$

where

$$\mathrm{B}_{j,3}(x) = \begin{cases} \dfrac{D_{j,1}\Psi_j(x)}{c_{j,2},} & x \in [x_j, x_{j+1}), \\ 1 + \dfrac{D_{j+1,1}\Phi_{j+1}(x)}{c_{j,2}} - \dfrac{D_{j+1,1}\Psi_{j+1}(x)}{c_{j+1,2}}, & x \in [x_{j+1}, x_{j+2}), \\ -\dfrac{D_{j+2,1}\Phi_{j+2}(x)}{c_{j+1,2}}, & x \in [x_{j+2}, x_{j+3}), \\ 0, & \text{otherwise}. \end{cases} \tag{5.28}$$

Using formula (5.28) it is easy to show that functions $\mathrm{B}_{j,3}$, $j = -2, \dots, N-1$ satisfy the first and second smoothness conditions in (5.3), have supports of minimum length, are linearly independent and form a partition of unity,

$$\sum_{j=1}^{N-1} \mathrm{B}_{j,3}(x) \equiv 1, \quad x \in [a, b].$$

Figures 5.1 and 5.2 show the graphs of discrete GB-splines $\mathrm{B}_{j,2}$, $\mathrm{B}^L_{j,3}$, and B_j (from left to right) on a uniform mesh with step size $h = 1$ and discretization parameter $\tau = 0.1$ (Figure 5.1, left and Figure 5.2, right), $\tau = 0.33$ (Figure 5.1, right) and $\tau = 0.5$ (Figure 5.2, left) for

$$\begin{aligned} \Psi_i(x) &= \psi(p_i, \hat\tau_i, t)h_i^2 = \frac{\hat\tau_i \sinh p_i t - t\sinh(p_i\hat\tau_i)}{\frac{4}{\hat\tau_i}\sinh^2\frac{p_i\hat\tau_i}{2}\sinh p_i}h_i^2, \quad \hat\tau_i = \frac{\tau}{h_i}, \\ \Phi_i(x) &= \psi(p_i, \hat\tau_i, 1-t)h_i^2. \end{aligned}$$

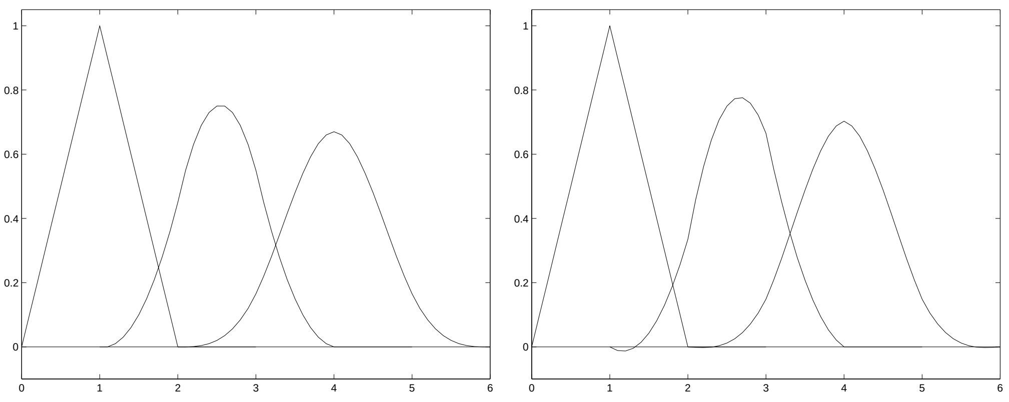

Figure 5.1: The discrete GB-splines $\mathrm{B}_{j,2}$, $\mathrm{B}^L_{j,3}$, and B_j (from left to right) on a uniform mesh with step size $h = 1$, no tension and discretization parameter $\tau = 0.1$ (left) and $\tau = 0.33$ (right).

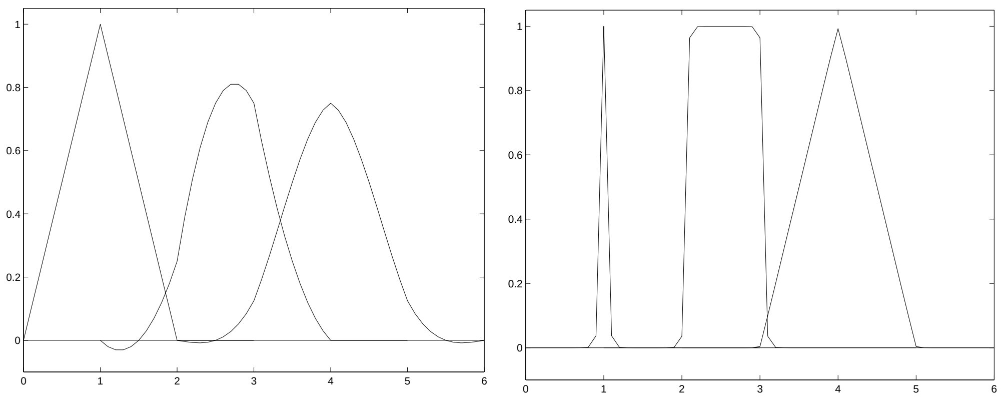

Figure 5.2: Same as Figure 5.1, but with discretization parameter $\tau = 0.5$ (left) and with tension parameters $p_i = 50$ for all i (right).

In figures 5.1 and 5.2 (left) we have parameters $p_i = 0$, that is, we have conventional discrete cubic B-splines (e.g., see T. Lyche (1976)). Visually, the presence of intervals where the B-splines B_j are negative is more visible with growing discretization parameter τ. In figure 5.2 (right) the tension parameters are $p_i = 50$ for all i, whence the shape of the graphs is practically unchanged when τ increases from 0.1 to 0.5. As the limit for $p_i \to \infty$ we obtain the pulse function for $\mathrm{B}_{j,2}$, the "step-function" for $\mathrm{B}^L_{j,3}$ and the "hat-function" for B_j (all of height 1).

Applying formulae (5.26) and (5.27) to the representation (5.20) we obtain

$$D_1^L S(x) = \sum_{j=-2}^{N-1} b_{j,3}\mathrm{B}^L_{j,3}(x), \quad D_2 S(x) = \sum_{j=-1}^{N-1} b_{j,2}\mathrm{B}_{j,2}(x), \tag{5.29}$$

where coefficients $b_{j,k}$, $k = 3, 2$ are calculated by formula (5.22).

5.6 Series of Discrete GB-Splines (uniform case)

Let us suppose that each step size $h_i = x_{i+1} - x_i$ of the mesh $\Delta : a = x_0 < x_1 < \cdots < x_N = b$ is an integer multiple of the same tabulation step, τ, of some detailed uniform refinement on $[a, b]$.

For $\theta \in \mathbb{R}$, $\tau > 0$ define

$$\mathbb{R}_{\theta\tau} = \{\theta + i\tau : \; i \text{ is an integer}\}$$

and let $\mathbb{R}_{\theta 0} = \mathbb{R}$. For any $a, b \in \mathbb{R}$ and $\tau > 0$ let

$$[a, b]_\tau = [a, b] \cap \mathbb{R}_{a\tau}.$$

The functions $\mathrm{B}_{j,2}$, $\mathrm{B}^L_{j,3}$, and B_j are nonnegative on the discrete interval $[a, b]_\tau$. This permits us to reprove the main results of discrete polynomial splines of L. L. Schumaker (1981) for series of discrete generalized splines. Even more, one can obtain the results for tension generalized splines of B. I. Kvasov (1995) from the corresponding statements for discrete tension generalized splines as a limiting case when $\tau \to 0$.

In particular, if in (5.20) and (5.29) we have the coefficients $b_{j,4-k} > 0$, $k = 0, 1, 2$, $j = -3+k, \ldots, N-1$, then the spline S will be a positive, monotonically increasing and convex function on $[a, b]_\tau$.

Let f be a function defined on the discrete set $[a, b]_\tau$. We say that f has a zero at the point $x \in [a, b]_\tau$ provided

$$f(x) = 0 \quad \text{or} \quad f(x - \tau) \cdot f(x) < 0.$$

When f vanishes at a set of consecutive points of $[a, b]_\tau$, say f is 0 at $x, \ldots, x + (r-1)\tau$, but $f(x - \tau) \cdot f(x + r\tau) \neq 0$, then we call the set $X = \{x, x + \tau, \ldots, x + (r-1)\tau\}$ a *multiple zero* of f, and we define its multiplicity by

$$Z_X(f) = \begin{cases} r, & \text{if } f(x-\tau) \cdot f(x+r\tau) < 0 \text{ and } r \text{ is odd,} \\ r, & \text{if } f(x-\tau) \cdot f(x+r\tau) > 0 \text{ and } r \text{ is even,} \\ r+1, & \text{otherwise.} \end{cases}$$

This definition assures that f changes sign at a zero if and only if the zero is of odd multiplicity.

Let $Z_{[a,b]_\tau}(f)$ be the number of zeros of a function f on the discrete set $[a, b]_\tau$, counted according to their multiplicity.

Theorem 5.4. *(Rolle's Theorem For Discrete Generalized Splines.) For any $S \in S^{DG}_4$,*

$$Z_{[a,b]_\tau}(D^L_1 S) \geq Z_{[a,b]_\tau}(S) - 1. \tag{5.30}$$

Proof: First, if S has a z-tuple zero on the set $X = \{x, \ldots, x + (r-1)\tau\}$, it follows that $D^L_1 S$ has a $(z-1)$-tuple zero on the set $X' = \{x + \tau, \ldots, x + (r-1)\tau\}$. Now if X^1 and X^2 are two consecutive zero sets of S, then it is trivially true that $D^L_1 S$ must have a sign change at some point between X^1 and X^2. Counting all of these zeros, we arrive at the assertion (5.30). This completes the proof. $\square$

Lemma 5.4. *For every $S \in S_4^{DG}$ which is not identically zero on any subinterval $[x_i, x_{i+1}]_\tau$, $i = 0, \dots, N-1$,*

$$Z_{[a,b]_\tau}(S) \le N + 2.$$

Proof: According to (5.25) and (5.29), the function $D_2 S$ has no more than one zero on $[x_i, x_{i+1}]$, because the functions $D_2\Phi_i$ and $D_2\Psi_i$ are strictly monotone and nonnegative on this subinterval. Hence $Z_{[a,b]_\tau}(D_2 S) \le N$. Then according to the Rolle's Theorem 5.4, we find $Z_{[a,b]_\tau}(S) \le N + 2$. This completes the proof.

□

Denote by $supp_\tau \mathrm{B}_i = \{x \in \mathbb{R}_{a,\tau} | \mathrm{B}_i(x) > 0\}$ the discrete support of the spline B_i, that is, the discrete set $(x_i + \tau, x_{i+4} - \tau)_\tau$.

Theorem 5.5. *Assume that $\zeta_{-3} < \zeta_{-2} < \cdots < \zeta_{N-1}$ are prescribed points on the discrete line $\mathbb{R}_{a,\tau}$. Then*

$$D = \det(\mathrm{B}_i(\zeta_j)) \ge 0, \quad i, j = -3, \dots, N-1$$

and strict positivity holds if and only if

$$\zeta_i \in supp_\tau \mathrm{B}_i, \quad i = -3, \dots, N-1. \tag{5.31}$$

Proof: Let us prove the theorem by induction. It is clear that the theorem holds for one basis function. Assume that it also holds for $l-1$ basis functions. Let us show that if (5.31) is satisfied, then $D \ne 0$ for l basis functions.

Let $\zeta_l \notin supp_\tau \mathrm{B}_l$. If ζ_l lies to the left (right) with respect to the discrete support of B_l then the last column (row) of the determinant D consists of zeros, that is, $D = 0$. If $\zeta_l \in supp_\tau \mathrm{B}_l$ and $D = 0$, then there exists a nonzero vector $\mathbf{c} = (c_{-3}, \dots, c_{l-4})$ such that

$$S(\zeta_k) = \sum_{j=-3}^{l-4} c_j \mathrm{B}_j(\zeta_k), \quad k = -3, \dots, l-4,$$

that is, the spline S has l zeros. But this contradicts Lemma 5.4, which states that S can have no more than $l-1$ zeros. Hence $\mathbf{c} = \mathbf{0}$ and $D \ne 0$.

Now it only remains to prove that $D > 0$ if (5.31) is satisfied. Let us choose $x_k + \tau < \zeta_k < x_{k+1} - \tau$ for all k. Then the diagonal elements of D are positive and all the elements above the main diagonal are zero, that is, $D > 0$. It is clear that D depends continuously on ζ_k, $k = -3, \dots, l-4$, and $D \ne 0$ for $\zeta_k \in supp_\tau \mathrm{B}_k$. Hence the determinant D is positive, if condition (5.31) is satisfied. This completes the proof. □

The following three statements follow immediately from Theorem 5.5.

Corollary 5.2. *The system of discrete GB-splines $\{\mathrm{B}_j\}$, $j = -3, \dots, N-1$, associated with knots on $\mathbb{R}_{a,\tau}$ is a weak Chebyshev system according to the definition given in L. L. Schumaker (1981), that is, for any $\zeta_{-3} < \zeta_{-2} < \cdots < \zeta_{N-1}$ in $\mathbb{R}_{a,\tau}$ we have $D \ge 0$ and $D > 0$ if and only if condition (5.31) is satisfied. In the latter case the discrete generalized spline $S(x) = \sum_{j=-3}^{N-1} b_j \mathrm{B}_j(x)$ has no more than $N+2$ zeros.*

Corollary 5.3. *If the conditions of Theorem 5.5 are satisfied, then the solution of the interpolation problem*

$$S(\zeta_i) = f_i, \quad i = -3, \dots, N-1, \quad f_i \in \mathbb{R} \tag{5.32}$$

exists and is unique.

Let $A = \{a_{ij}\}$, $i = 1, \dots, m$, $j = 1, \dots, n$, be a rectangular $m \times n$ matrix with $m \leq n$. The matrix A is said to be totally nonnegative (totally positive) (see, e.g., S. Karlin (1968)) if the minors of all order of the matrix are nonnegative (positive), that is, for all $1 \leq p \leq m$ we have

$$det(a_{i_k j_l}) \geq 0 \ (> 0) \quad \text{for all} \quad 1 \leq i_1 < \cdots < i_p \leq m, \ 1 \leq j_1 < \cdots < j_p \leq n.$$

Corollary 5.4. *For arbitrary integers $-3 \leq \nu_{-3} < \cdots < \nu_{p-4} \leq N-1$ and $\zeta_{-3} < \zeta_{-2} < \cdots < \zeta_{p-4}$ in $\mathbb{R}_{a,\tau}$ we have*

$$D_p = det\{\mathrm{B}_{\nu_i}(\zeta_j)\} \geq 0, \quad i,j = -3, \dots, p-4$$

and strict positivity holds if and only if

$$\zeta_i \in supp_\tau \, \mathrm{B}_{\nu_i}, \quad i = -3, \dots, p-4$$

that is, the matrix $\{\mathrm{B}_j(\zeta_i)\}$, $i,j = -3, \dots, N-1$ is totally nonnegative.

The last statement is proved by induction based on Theorem 5.5 and the recurrence relations for the minors of the matrix $\{\mathrm{B}_j(\zeta_i)\}$. The proof does not differ from that of Theorem 8.67 described by L. L. Schumaker (1981) p. 356.

Since the supports of discrete GB-splines are finite, the matrix of system (5.32) is banded and has seven nonzero diagonals in general. The matrix is tridiagonal if $\zeta_i = x_{i+2}$, $i = -3, \dots, N-1$.

An important particular case of the problem, in which $S'(x_i) = f'_i$, $i = 0, N$, can be obtained by passing to the limit as $\zeta_{-3} \to \zeta_{-2}$, $\zeta_{N-1} \to \zeta_{N-2}$.

C. De Boor and A. Pinkus (1977) proved that linear systems with totally nonnegative matrices can be solved by Gaussian elimination without choosing a pivot element. Thus, the system (5.32) can be solved effectively by the conventional Gauss method.

Denote by $S^-(\mathbf{v})$ the number of sign changes (variations) in the sequence of components of the vector $\mathbf{v} = (v_1, \cdots, v_n)$, with zeros being neglected. S. Karlin (1968) showed that if a matrix A is totally nonnegative then it decreases the variation, that is,

$$S^-(A\mathbf{v}) \leq S^-(\mathbf{v}).$$

By virtue of Corollary 5.4, the totally nonnegative matrix $\{\mathrm{B}_j(\zeta_i)\}$, $i,j = -3, \dots, N-1$, formed by discrete GB-splines decreases the variation.

For a bounded real function f, let $S^-(f)$ be the number of sign changes of the function f on the real axis $\mathbb{R}$, without taking into account the zeros

$$S^-(f) = \sup_n S^-[f(\zeta_1), \dots, f(\zeta_n)], \quad \zeta_1 < \zeta_2 < \cdots < \zeta_n.$$

Theorem 5.6. *The discrete generalized spline $S(x) = \sum_{j=-3}^{N-1} b_j \mathrm{B}_j(x)$ is a variation diminishing function, that is, the number of sign changes of S does not exceed that in the sequence of its coefficients:*

$$S^-_{\mathbb{R}}\Big(\sum_{j=-3}^{N-1} b_j \mathrm{B}_j\Big) \le S^-(\mathbf{b}), \quad \mathbf{b} = (b_{-3}, \dots, b_{N-1}).$$

The proof of this statement does not differ from that of Theorem 8.68 for discrete polynomial B-splines in L. L. Schumaker (1981).

By Theorem 5.6, the spline

$$S_f(x) = \sum_{j=-3}^{N-1} f(y_{j+2}) \mathrm{B}_j(x)$$

is a variation diminishing function. It enables us to write the inequalities

$$S^-(S_f) \le S^-(\overline{\mathbf{f}}) \le S^-(f),$$

where $\overline{\mathbf{f}} = (f(y_{-1}), \dots, f(y_{N+1}))$.

Since in addition by Theorem 5.3, the locally approximating discrete generalized spline S_f is also exact for polynomials l of first order, we arrive at the inequality

$$S^-(S_f - l) = S^-(S_{f-l}) \le S^-(f - l).$$

Thus, the following statement is true.

Theorem 5.7. *If $b_j = f(y_{j+2})$, $j = -3, \dots, N-1$, then the locally approximating discrete generalized spline S_f intersects an arbitrary straight line at most as often as the function f.*

5.7 Examples of Defining Functions

Let us give some choices of the defining functions Φ_i and Ψ_i for discrete generalized splines that conform to the sufficiency conditions derived earlier in the chapter.

Putting

$$\Psi_i(x) = \psi_i(t)h_i^2 = \psi(p_i, \hat{\tau}_i^{L_i}, \hat{\tau}_i^{R_i}, t)h_i^2, \quad \Phi_i(x) = \psi(q_i, \hat{\tau}_{i+1}^{R_i}, \hat{\tau}_{i+1}^{L_i}, 1-t)h_i^2,$$
$$\hat{\tau}_j^{L_i} = \tau_j^{L_i}/h_i, \quad \hat{\tau}_j^{R_i} = \tau_j^{R_i}/h_i; \quad j = i, i+1; \quad 0 \le p_i, q_i < \infty,$$

we consider some possibilities for choosing the function ψ_i which, by the definition 5.1, satisfies the conditions

$$\psi_i(-\hat{\tau}_i^{L_i}) = \psi_i(0) = \psi_i(\hat{\tau}_i^{R_i}) = 0, \ D_{2,i+1}\psi_i(1) = h_i^{-2}. \qquad (5.33)$$

1. Discrete rational spline with linear denominator:

$$\psi_i(t) = C_i \frac{(t + \hat{\tau}_i^{L_i})t(t - \hat{\tau}_i^{R_i})}{1 + p_i(1-t)}.$$

2. Discrete rational spline with quadratic denominator:

$$\psi_i(t) = C_i \frac{(t+\hat{\tau}_i^{L_i})t(t-\hat{\tau}_i^{R_i})}{1+p_i t(1-t)}.$$

3. Discrete exponential spline:

$$\psi_i(t) = C_i(t+\hat{\tau}_i^{L_i})t(t-\hat{\tau}_i^{R_i})e^{-p_i(1-t)}.$$

4. Discrete hyperbolic spline:

$$\psi_i(t) = C_{i,1}\big[\sinh p_i t - t\frac{\sinh p_i\hat{\tau}_i^{R_i}}{\hat{\tau}_i^{R_i}}\big] + C_{i,2}\big[\cosh p_i t - 1 - t\frac{\cosh p_i\hat{\tau}_i^{R_i} - 1}{\hat{\tau}_i^{R_i}}\big].$$

5. Discrete cubic spline with variable additional knots:

$$\begin{aligned} \psi_i(t) &= \frac{1}{2}\frac{(t-\alpha_i+\hat{\tau}_i^{L_i})(t-\alpha_i)_+(t-\alpha_i-\hat{\tau}_i^{R_i})}{3(1-\alpha_i)+\hat{\varepsilon}_{i+1}-\hat{\varepsilon}_i}, \\ \hat{\varepsilon}_j &= \hat{\tau}_j^{R_i} - \hat{\tau}_j^{L_i}, \quad j=i,i+1; \quad \alpha_i = (1+p_i)^{-1}. \end{aligned}$$

The points $x_i+\alpha_i h_i$ and $x_i+\beta_i h_i$ ($\beta_i = 1-(1+q_i)^{-1}$) fix the position of two additional knots of the spline on the interval $[x_i, x_{i+1}]$. By moving these knots one can perform a transfer from a discrete cubic spline to piecewise linear interpolation.

6. Discrete spline of variable order:

$$\psi_i(t) = C_i(t+\hat{\tau}_i^{L_i})t^{k_i}(t-\hat{\tau}^{R_i}), \quad k_i = 1+q_i.$$

The constants C_i in the expressions for the function ψ_i above are calculated from the condition (5.33) for the second divided difference of ψ_i. To find $C_{i,k}$, $k=1,2$, one needs additionally use the condition $\psi_i(-\hat{\tau}_i^{L_i}) = 0$. It is easy to check that in all cases 1.–6. we get the corresponding defining functions of B. I. Kvasov (1996a) by setting $\hat{\tau}_j^{L_i} = \hat{\tau}_j^{R_i} = 0$, $j=i,i+1$.

Chapter VI

Finite-Difference Method for 2-D Shape Preserving Interpolation

In this chapter 2-D problem of shape-preserving interpolation is formulated as Differential Multipoint Boundary Value Problem (DMBVP for short) for thin plate tension splines. For a numerical treatment of this problem, we consider its finite-difference approximation. This gives a system of linear algebraic equations which can be solved either by direct and iterative methods. As a direct method, we suggest to consider a block Gaussian elimination. For iterative solution of the obtained linear system, we apply Successive Over-Relaxation (SOR) method. Finite-difference schemes in fractional steps also prove their efficiency in the numerical treatment of our DMBVP.

6.1 Problem Formulation

In this section, we introduce necessary notations and define 2-D problem of shape-preserving interpolation as DMBVP which consists of differential equation, smoothness, interpolation, and boundary conditions.

Let us consider a rectangular domain $\overline{\Omega} = \Omega \cup \Gamma$ where

$$\Omega = \{(x, y) \mid a < x < b,\ c < y < d\}$$

and Γ is the boundary of Ω. We consider on $\overline{\Omega}$ a rectangular mesh $\Delta = \Delta_x \times \Delta_y$ with

$$\begin{aligned}\Delta_x : a &= x_0 < x_1 < \cdots < x_{N+1} = b,\\ \Delta_y : c &= y_0 < y_1 < \cdots < y_{M+1} = d,\end{aligned}$$

which divides the domain $\overline{\Omega}$ into the rectangles $\overline{\Omega}_{ij} = \Omega_{ij} \cup \Gamma_{ij}$ where

$$\Omega_{ij} = \{(x, y) \mid x \in (x_i, x_{i+1}),\ y \in (y_j, y_{j+1})\},$$

and Γ_{ij} is the boundary of Ω_{ij}, $i = 0, \ldots, N, \quad j = 0, \ldots, M.$

Let us associate to the mesh Δ the data

$$\left.\begin{array}{lll}(x_i, y_j, f_{ij}), & i = 0, \ldots, N+1, & j = 0, \ldots, M+1,\\ f_{ij}^{(2,0)}, & i = 0, N+1, & j = 0, \ldots, M+1,\\ f_{ij}^{(0,2)}, & i = 0, \ldots, N+1, & j = 0, M+1,\\ f_{ij}^{(2,2)}, & i = 0, N+1, & j = 0, M+1,\end{array}\right\} \tag{6.1}$$

where

$$f_{ij}^{(r,s)} = \frac{\partial^{r+s} f(x_i, y_j)}{\partial x^r \partial y^s}; \quad r, s = 0, 2.$$

It is convenient for us to collect this data in the following table.

$f^{(2,2)}_{0,M+1}$	$f^{(0,2)}_{0,M+1}$	$f^{(0,2)}_{1,M+1}$	$\cdots$	$f^{(0,2)}_{N,M+1}$	$f^{(0,2)}_{N+1,M+1}$	$f^{(2,2)}_{N+1,M+1}$
$f^{(2,0)}_{0,M+1}$	$f_{0,M+1}$	$f_{1,M+1}$	$\cdots$	$f_{N,M+1}$	$f_{N+1,M+1}$	$f^{(2,0)}_{N+1,M+1}$
$f^{(2,0)}_{0,M}$	$f_{0,M}$	$f_{1,M}$	$\cdots$	$f_{N,M}$	$f_{N+1,M}$	$f^{(2,0)}_{N+1,M}$
$\vdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\vdots$
$f^{(2,0)}_{0,1}$	$f_{0,1}$	$f_{1,1}$	$\cdots$	$f_{N,1}$	$f_{N+1,1}$	$f^{(2,0)}_{N+1,1}$
$f^{(2,0)}_{0,0}$	$f_{0,0}$	$f_{1,0}$	$\cdots$	$f_{N,0}$	$f_{N+1,0}$	$f^{(2,0)}_{N+1,0}$
$f^{(2,2)}_{0,0}$	$f^{(0,2)}_{0,0}$	$f^{(0,2)}_{1,0}$	$\cdots$	$f^{(0,2)}_{N,0}$	$f^{(0,2)}_{N+1,0}$	$f^{(2,2)}_{N+1,0}$

We introduce the following notations for divided differences:

$$\begin{aligned}
f[x_i; y_j] &= f(x_i, y_j) = f_{ij}, \\
f[x_i, \ldots, x_{i+k}; y_j] &= \frac{f[x_{i+1}, \ldots, x_{i+k}; y_j] - f[x_i, \ldots, x_{i+k-1}; y_j]}{x_{i+k} - x_i}, \\
&\quad k = 1, \ldots, N+1, \\
&\quad i = 0, \ldots, N+1-k, \quad j = 0, \ldots, M+1, \\
f[x_i; y_j, \ldots, y_{j+l}] &= \frac{f[x_i; y_{j+1}, \ldots, y_{j+l}] - f[x_i; y_j, \ldots, y_{j+l-1}]}{y_{j+l} - y_j}, \\
&\quad l = 1, \ldots, M+1, \\
&\quad i = 0, \ldots, N+1, \quad j = 0, \ldots, M+1-l.
\end{aligned}$$

In particular, one has for the first order divided differences:

$$\begin{aligned}
f[x_i, x_{i+1}; y_j] &= (f_{i+1,j} - f_{ij})/h_i; \quad h_i = x_{i+1} - x_i, \\
&\qquad i = 0, \ldots, N, \qquad j = 0, \ldots, M+1, \\
f[x_i; y_j, y_{j+1}] &= (f_{i,j+1} - f_{ij})/l_j, \quad l_j = y_{j+1} - y_j, \\
&\qquad i = 0, \ldots, N+1, \quad j = 0, \ldots, M.
\end{aligned}$$

Definition 6.1. *The data f_{ij} is said to be*

positive (negative) *if*

$$f_{ij} > 0 \ (< 0), \quad \text{for all} \quad i \text{ and } j,$$

monotonically increasing (decreasing) by x *if*

$$f[x_i, x_{i+1}; y_j] > 0 \ (< 0), \quad i = 0, \ldots, N, \quad j = 0, \ldots, M+1,$$

monotonically increasing (decreasing) by y *if*

$$f[x_i; y_j, y_{j+1}] > 0 \ (< 0), \quad i = 0, \ldots, N+1, \quad j = 0, \ldots, M,$$

convex (concave) by x *if*

$$f[x_i, x_{i+1}; y_j] - f[x_{i-1}, x_i; y_j] > 0 \ (< 0), \quad i = 1, \ldots, N, \quad j = 0, \ldots, M+1,$$

***convex (concave) by y** if*

$$f[x_i; y_j, y_{j+1}] - f[x_i; y_{j-1}, y_j] > 0\ (< 0), \quad i = 0, \dots, N+1, \quad j = 1, \dots, M.$$

We denote by $C^{2,2}[\overline{\Omega}]$ the set of all continuous on $\overline{\Omega}$ functions f having continuous partial and mixed derivatives up to the order 2. We say that the problem of searching for a function $S \in C^{2,2}[\overline{\Omega}]$ such that $S(x_i, y_j) = f_{ij};\ i = 0, \dots, N+1;\ j = 0, \dots, M+1,$ and that S preserves the form of the initial data is *the shape-preserving interpolation problem.* This means that where the data increases (decreases) monotonically, S has the same behavior, and S is convex (concave) over intervals where the data is convex (concave).

Evidently, the solution of the shape-preserving interpolation problem is not unique. We are looking for a solution of this problem as a thin plate tension spline.

Definition 6.2. *An interpolating thin plate tension spline S with two sets of tension parameters $\{p_{ij} \ge 0 \mid i = 0, \dots, N;\ j = 0, \dots, M+1\}$ and $\{q_{ij} \ge 0 \mid i = 0, \dots, N+1;\ j = 0, \dots, M\}$ is a solution of the DMBVP*

$$LS \equiv \frac{\partial^4 S}{\partial x^4} + 2\frac{\partial^4 S}{\partial x^2 \partial y^2} + \frac{\partial^4 S}{\partial y^4} - \left(\frac{p_{ij}}{h_i}\right)^2 \frac{\partial^2 S}{\partial x^2} - \left(\frac{q_{ij}}{l_j}\right)^2 \frac{\partial^2 S}{\partial y^2} = 0 \tag{6.2}$$

in each $\Omega_{ij}, \quad i = 0, \dots, N;\ j = 0, \dots, M,$

$$\frac{\partial^4 S}{\partial x^4} - \left(\frac{p_{ij}}{h_i}\right)^2 \frac{\partial^2 S}{\partial x^2} = 0, \quad x \in (x_i, x_{i+1}), \quad y = y_j, \tag{6.3}$$

$$i = 0, \dots, N;\ j = 0, \dots, M+1,$$

$$\frac{\partial^4 S}{\partial y^4} - \left(\frac{q_{ij}}{l_j}\right)^2 \frac{\partial^2 S}{\partial y^2} = 0, \quad x = x_i, \quad y \in (y_j, y_{j+1}), \tag{6.4}$$

$$i = 0, \dots, N+1;\ j = 0, \dots, M,$$

$$S \in C^{\,2,2}\left[\Omega\right], \tag{6.5}$$

with the interpolation conditions

$$S\left(x_i, y_j\right) = f_{ij}, \quad i = 0, \dots, N+1, \quad j = 0, \dots, M+1, \tag{6.6}$$

and the boundary conditions

$$\begin{array}{rcll} D^{(2,0)}S\left(x_i, y_j\right) & = & D^{(2,0)}f\left(x_i, y_j\right), & i = 0, N+1, \qquad j = 0, \dots, M+1, \\ D^{(0,2)}S\left(x_i, y_j\right) & = & D^{(0,2)}f\left(x_i, y_j\right), & i = 0, \dots, N+1, \quad j = 0, M+1, \\ D^{(2,2)}S\left(x_i, y_j\right) & = & D^{(2,2)}f\left(x_i, y_j\right), & i = 0, N+1, \qquad j = 0, M+1, \end{array} \tag{6.7}$$

where

$$D^{(r,s)}f\left(x_i, y_j\right) = \frac{\partial^{r+s} f\left(x_i, y_j\right)}{\partial x^r \partial y^s}; \quad r, s = 0, 2.$$

If all tension parameters of the thin plate tension spline S are zero then one obtains a smooth thin plate spline, see J. Hoschek and D. Lasser (1993), interpolating the data $(x_i, y_j, f_{ij});\ i = 0, \ldots, N;\ j = 0, \ldots, M$. If tension parameters p_{ij} and q_{ij} approach to the infinity then in the rectangle $\overline{\Omega}_{ij};\ i = 0, \ldots, N;\ j = 0, \ldots, M;$ thin plate spline S turns into a linear function separately by x and y, and obviously preserves on $\overline{\Omega}_{ij}$ shape properties of the data. So, by changing values of the shape control parameters p_{ij} and q_{ij} one can preserve various characteristics of the data including positivity, monotonicity, convexity, as well as linear and planar sections. By increasing one or more of these parameters the surface is pulled towards an inherent shape at the same time keeping its smoothness. Thus, DMBVP gives a reasonable mathematical formulation of the shape-preserving interpolation problem.

6.2 Finite-Difference Approximation of DMBVP

For practical purposes, it is often necessary to know the values of solution S of a DMBVP only over a prescribed grid instead of its global analytic expression. In this section, we consider a finite-difference approximation of the DMBVP. This provides a linear system which solution is called *a mesh solution.* It turns out that the mesh solution is not a tabulation of S but it can be extended on $\overline{\Omega}$ to a smooth function U which has shape properties very similar to those of S and which provides a second order approximation of S as the discretization step goes to zero. Due to these properties, we will refer to U as *a discrete thin plate tension spline.*

Let $n_i, m_j \in \mathbb{N},\ i = 0, \ldots, N;\ j = 0, \ldots, M$, be given such that $h_i/n_i = l_j/m_j = h$. We are looking for a mesh function

$$\Big\{ u_{ik;jl} \mid k = -1, \ldots, n_i + 1;\ i = 0, \ldots, N;\ l = -1, \ldots, m_j + 1;\ j = 0, \ldots, M \Big\}$$

satisfying the difference equations

$$\Lambda u_{ik;jl} \equiv \left[\Lambda_1^2 + 2\Lambda_1\Lambda_2 + \Lambda_2^2 - \left(\frac{p_{ij}}{h_i}\right)^2 \Lambda_1 - \left(\frac{q_{ij}}{l_j}\right)^2 \Lambda_2 \right] u_{ik;jl} = 0, \tag{6.8}$$

$$k = 1, \ldots, n_i - 1;\ i = 0, \ldots, N;\ l = 1, \ldots, m_j - 1;\ j = 0, \ldots, M,$$

$$\left[\Lambda_1^2 - \left(\frac{p_{ij}}{h_i}\right)^2 \Lambda_1 \right] u_{ik;jl} = 0, \tag{6.9}$$

$$k = 1, \ldots, n_i - 1;\ i = 0, \ldots, N; \quad l = \begin{cases} 0 & \text{if } j = 0, \ldots, M-1, \\ 0, m_M & \text{if } j = M, \end{cases}$$

$$\left[\Lambda_2^2 - \left(\frac{q_{ij}}{l_j}\right)^2 \Lambda_2 \right] u_{ik;jl} = 0, \tag{6.10}$$

$$k = \begin{cases} 0 & \text{if } i = 0, \dots, N-1, \\ 0, n_N & \text{if } i = N, \end{cases} \quad ; \quad l = 1, \dots, m_j - 1; \; j = 0, \dots, M,$$

where

$$\Lambda_1 u_{ik;jl} = \frac{u_{i,k-1;jl} - 2u_{ik;jl} + u_{i,k+1;jl}}{h^2}, \tag{6.11}$$

$$\Lambda_2 u_{ik;jl} = \frac{u_{ik;j,l-1} - 2u_{ik;jl} + u_{ik;j,l+1}}{h^2}, \tag{6.12}$$

$$\Lambda_1^2 u_{ik;jl} = \frac{1}{h^4}\Big[u_{i,k-2;jl} - 4u_{i,k-1;jl} + 6u_{ik;jl} - 4u_{i,k+1;jl} + u_{i,k+2;jl}\Big], \tag{6.13}$$

$$\Lambda_2^2 u_{ik;jl} = \frac{1}{h^4}\Big[u_{ik;j,l-2} - 4u_{ik;j,l-1} + 6u_{ik;jl} - 4u_{ik;j,l+1} + u_{ik;j,l+2}\Big], \tag{6.14}$$

$$\begin{aligned} \Lambda_1\Lambda_2 u_{ik;jl} = \frac{1}{h^4}\Big[& u_{i,k-1;j,l-1} + u_{i,k-1;j,l+1} + u_{i,k+1;j,l-1} + u_{i,k+1;j,l+1} \\ & -2u_{i,k-1;jl} - 2u_{i,k+1;jl} - 2u_{ik;j,l-1} - 2u_{ik;j,l+1} + 4u_{ik;jl}\Big]. \end{aligned} \tag{6.15}$$

The smoothness condition (6.5) is changed into

$$\begin{aligned} u_{i-1,n_{i-1};jl} &= u_{i0;jl}, \\ \frac{u_{i-1,n_{i-1}+1;jl} - u_{i-1,n_{i-1}-1;jl}}{2h} &= \frac{u_{i1;jl} - u_{i,-1;jl}}{2h}, \\ \Lambda_1 u_{i-1,n_{i-1};jl} &= \Lambda_1 u_{i0;jl}, \end{aligned} \tag{6.16}$$

$$i = 1, \dots, N; \; l = 0, \dots, m_j, \; j = 0, \dots, M,$$

$$\begin{aligned} u_{ik;j-1,m_{j-1}} &= u_{ik;j0}, \\ \frac{u_{ik;j-1,m_{j-1}+1} - u_{ik;j-1,m_{j-1}-1}}{2h} &= \frac{u_{ik;j1} - u_{ik;j,-1}}{2h}, \\ \Lambda_2 u_{ik;j-1,m_{j-1}} &= \Lambda_2 u_{ik;j0}, \end{aligned} \tag{6.17}$$

$$k = 0, \dots, n_i, \; i = 0, \dots, N; \; j = 1, \dots, M.$$

Conditions (6.6) and (6.7) take the form

$$\begin{aligned} u_{i0;j0} &= f_{ij}, & u_{N,n_N;j0} &= f_{N+1,j}, \\ u_{i0;M,m_M} &= f_{i,M+1}, & u_{N,n_N;M,m_M} &= f_{N+1,M+1}, \end{aligned} \tag{6.18}$$

$$i = 0, \dots, N, \quad j = 0, \dots, M$$

and

$$\begin{aligned} \Lambda_1 u_{00;j0} &= f^{(2,0)}_{0j}, \quad j = 0, \dots, M; & \Lambda_1 u_{00;M,m_M} &= f^{(2,0)}_{0,M+1}; \\ \Lambda_1 u_{N,n_N;j0} &= f^{(2,0)}_{N+1,j}, \quad j = 0, \dots, M; & \Lambda_1 u_{N,n_N;M,m_M} &= f^{(2,0)}_{N+1,M+1}; \end{aligned} \tag{6.19}$$

$$\begin{aligned} \Lambda_2 u_{i0;00} &= f^{(0,2)}_{i0}, \quad i = 0, \dots, N; & \Lambda_2 u_{N,n_N;00} &= f^{(0,2)}_{N+1,0}; \\ \Lambda_2 u_{i0;M,m_M} &= f^{(0,2)}_{i,M+1}, \quad i = 0, \dots, N; & \Lambda_2 u_{N,n_N;M,m_M} &= f^{(0,2)}_{N+1,M+1}; \end{aligned} \tag{6.20}$$

$$\begin{aligned} \Lambda_1\Lambda_2 u_{00;00} &= f^{(2,2)}_{00}; & \Lambda_1\Lambda_2 u_{N,n_N;00} &= f^{(2,2)}_{N+1,0}; \\ \Lambda_1\Lambda_2 u_{00;M,m_M} &= f^{(2,2)}_{0,M+1}; & \Lambda_1\Lambda_2 u_{N,n_N;M,m_M} &= f^{(2,2)}_{N+1,M+1}. \end{aligned} \tag{6.21}$$

The conditions (6.16) and (6.17) are equivalent to the relations

$$u_{i-1,n_{i-1}+k;jl} = u_{ik;jl}, \tag{6.22}$$

$$k=-1,0,1; \quad i=1,\dots,N; \quad l=0,\dots,m_j; \quad j=0,\dots,M;$$

$$u_{ik;j-1,m_{j-1}+l} = u_{ik;jl}, \tag{6.23}$$

$$k=0,\dots,n_i; \quad i=0,\dots,N; \quad l=-1,0,1; \quad j=1,\dots,M.$$

Difference equations (6.9) and (6.10) can be solved separately as 1-D problems. Then, we obtain

$$\begin{aligned}
\Lambda_1 u_{00;M,m_M} &= \frac{u_{0,-1;M,m_M} - 2u_{00;M,m_j} + u_{01;M,m_M}}{h^2} = f^{(2,0)}_{00;M,m_M},\\
\Lambda_1 u_{N,n_N;M,m_M} &= \frac{u_{N,n_N-1;M,m_M} - 2u_{N,n_N;M,m_M} + u_{N,n_N+1;M,m_M}}{h^2} = f^{(2,0)}_{N,n_N;M,m_M},\\
\Lambda_1 u_{00;jl} &= \frac{u_{0,-1;jl} - 2u_{00;jl} + u_{01;jl}}{h^2} = f^{(2,0)}_{00;jl},\\
\Lambda_1 u_{N,n_N;jl} &= \frac{u_{N,n_N-1;jl} - 2u_{N,n_N;jl} + u_{N,n_N+1;jl}}{h^2} = f^{(2,0)}_{N,n_N;jl},\\
& l=0,\dots,m_j-1, \quad j=0,\dots,M,\\
\Lambda_2 u_{N,n_N;00} &= \frac{u_{N,n_N;0,-1} - 2u_{N,n_N;00} + u_{N,n_N;01}}{h^2} = f^{(0,2)}_{N,n_N;00};\\
\Lambda_2 u_{N,n_N;M,m_M} &= \frac{u_{N,n_N;M,m_M-1} - 2u_{N,n_N;M,m_M} + u_{N,n_N;M,m_M+1}}{h^2} = f^{(0,2)}_{N,n_N;M,m_M};\\
\Lambda_2 u_{ik;00} &= \frac{u_{ik;0,-1} - 2u_{ik;00} + u_{ik;01}}{h^2} = f^{(0,2)}_{ik;00};\\
\Lambda_2 u_{ik;M,m_M} &= \frac{u_{ik;M,m_M-1} - 2u_{ik;M,m_M} + u_{ik;M,m_M+1}}{h^2} = f^{(0,2)}_{ik;M,m_M};\\
& k=0,\dots,n_i-1; \quad i=0,\dots,N.
\end{aligned}$$

Therefore, the above equations can be rewritten in the simple form

$$\left.\begin{aligned}
u_{0,-1;M,m_M} &= h^2 f^{(2,0)}_{00;M,m_M} + 2u_{00;M,m_M} - u_{01;M,m_M},\\
u_{N,n_N+1;M,m_M} &= h^2 f^{(2,0)}_{N,n_N;M,m_M} + 2u_{N,n_N;M,m_M} - u_{N,n_N-1;M,m_M},\\
u_{0,-1;jl} &= h^2 f^{(2,0)}_{00;jl} + 2u_{00;jl} - u_{01;jl},\\
u_{N,n_N+1;jl} &= h^2 f^{(2,0)}_{N,n_N;jl} + 2u_{N,n_N;jl} - u_{N,n_N-1;jl},\\
& l=0,\dots,m_j-1, \quad j=0,\dots,M,\\
u_{N,n_N;0,-1} &= h^2 f^{(0,2)}_{N,n_N;00} + 2u_{N,n_N;00} - u_{N,n_N;01},\\
u_{N,n_N;M,m_M+1} &= h^2 f^{(0,2)}_{N,n_N;M,m_M} + 2u_{N,n_N;M,m_M} - u_{N,n_N;M,m_M-1},\\
u_{ik;0,-1} &= h^2 f^{(0,2)}_{ik;00} + 2u_{ik;00} - u_{ik;01},\\
u_{ik;M,m_M+1} &= h^2 f^{(0,2)}_{ik;M,m_M} + 2u_{ik;M,m_M} - u_{ik;M,m_M-1},\\
& k=0,\dots,n_i-1, \quad i=0,\dots,N.
\end{aligned}\right\} \tag{6.24}$$

If we substitute relations (6.24) into equations (6.8) then the unknowns outside $\overline{\Omega}$ are eliminated. Relations (6.22) and (6.23) permit us to eliminate the following unknowns

$$u_{ik;jl}, \quad k=-1,n_i+1, \quad i=0,\dots,N, \quad l=0,\dots,m_j, \quad j=0,\dots,M,$$

and

$$u_{ik;jl}, \quad k=0,\dots,n_i, \quad i=0,\dots,N, \quad l=-1,m_j+1, \quad j=0,\dots,M.$$

Eliminating interpolation conditions (6.18), we obtain the following number of unknowns

$$\nu = \nu_x\nu_y - \nu_I, \tag{6.25}$$

where $\nu_x = 1+\sum_{i=0}^N n_i$, $\nu_y = 1+\sum_{j=0}^M m_j$ and $\nu_I = (N+2)(M+2)$.

Now, we are going to show that the number of unknowns and the number of equations in the above difference equations is the same. The number of conditions from (6.8), (6.9) and (6.10) are $\mu_T = \sum_{j=0}^M\sum_{i=0}^N(n_i-1)(m_j-1)$, $\mu_{Tx} = (M+2)\big[\sum_{i=0}^N n_i - (N+1)\big]$ and $\mu_{Ty} = (N+2)\big[\sum_{j=0}^M m_j - (M+1)\big]$ with respectively. Therefore, the total number of conditions for solving this problem is

$$\mu = \mu_T + \mu_{Tx} + \mu_{Ty}. \tag{6.26}$$

Now, we want to show $\nu = \mu$. Let us consider

$$\begin{aligned}
\nu_x\nu_y &= (1+\sum_{i=0}^N n_i)(1+\sum_{j=0}^M m_j)\\
&= 1+\sum_{i=0}^N n_i + \sum_{j=0}^M m_j + \sum_{i=0}^N n_i\sum_{j=0}^M m_j\\
\sum_{i=0}^N n_i\sum_{j=0}^M m_j &= \Big[\sum_{i=0}^N(n_i-1)+(N+1)\Big]\Big[\sum_{j=0}^M(m_j-1)+(M+1)\Big]\\
&= \sum_{i=0}^N(n_i-1)\sum_{j=0}^M(m_j-1) + (M+1)\sum_{i=0}^N(n_i-1)\\
&\quad +(N+1)\sum_{j=0}^M(m_j-1) + (M+1)(N+1)\\
&= \mu_T + (M+1)\Big[\sum_{i=0}^N n_i - (N+1)\Big]\\
&\quad +(N+1)\Big[\sum_{j=0}^M m_j - (M+1)\Big] + (M+1)(N+1)\\
&= \mu_T + (M+2)\Big[\sum_{i=0}^N n_i - (N+1)\Big] - \Big[\sum_{i=0}^N n_i - (N+1)\Big]\\
&\quad +(N+2)\Big[\sum_{j=0}^M m_j - (M+1)\Big] - \Big[\sum_{j=0}^M m_j - (M+1)\Big]\\
&\quad +(M+1)(N+1)
\end{aligned}$$

$$
\begin{aligned}
&= \mu_T + \mu_{Tx} + \mu_{Ty} - \sum_{i=0}^{N} n_i + (N+1) - \sum_{j=0}^{M} m_j + (M+1) \\
&\quad +(M+1)(N+1)
\end{aligned}
$$

$$
\begin{aligned}
\nu_x \nu_y &= \mu_T + \mu_{Tx} + \mu_{Ty} + (M+1) + (N+1) + (M+1)(N+1) + 1 \\
&= \mu + (M+1)(N+2) + (N+1) + 1 \\
&= \mu + (M+1)(N+2) + (N+2) = \mu + (M+2)(N+2) \\
&= \mu + \nu_I \\
\nu = \nu_x \nu_y - \nu_I &= \mu.
\end{aligned}
$$

6.3 Matrix Formulation

Finite-difference equations (6.9) and (6.10) can be solved separately as 1-D problems. Therefore, there are $(n_i - 1)(m_j - 1)$ unknowns and same number of difference equations (6.8) in each $\Omega_{ij},\ i = 0, \ldots, N,\ j = 0, \ldots, M$. Then the above system of difference equations can be written in matrix form as follows

$$Au = b, \tag{6.27}$$

where

$u = \{u_{ik;jl}\},\ b = \{b_{ik;jl}\},$

$k = 1, \ldots, n_i - 1;\ i = 0, \ldots, N;\ l = 1, \ldots, m_j - 1;\ j = 0, \ldots, M,$

$$
A = \left[\begin{array}{ccccc|cccccc}
\overline{A}_0 - I & \overline{B}_0 & I & & & & & & & & \\
\overline{B}_0 & \overline{A}_0 & \overline{B}_0 & I & & & & & & & \\
I & \overline{B}_0 & \overline{A}_0 & \overline{B}_0 & I & & & & & & \\
& \ddots & \ddots & \ddots & & & & & & & \\
& I & \overline{B}_0 & \overline{A}_0 & \overline{B}_0 & & & & & & \\
& & I & \overline{B}_0 & \overline{A}_0 & I & & & & & \\
\hline
& & & & I & \overline{A}_1 & \overline{B}_1 & I & & & \\
& & & & & \overline{B}_1 & \overline{A}_1 & \overline{B}_1 & I & & \\
& & & & & & \ddots & \ddots & \ddots & & \\
& & & & & & I & \overline{B}_M & \overline{A}_M & \overline{B}_M & I \\
& & & & & & & I & \overline{B}_M & \overline{A}_M & \overline{B}_M \\
& & & & & & & & I & \overline{B}_M & \overline{A}_M - I
\end{array}\right],
$$

$$\overline{A}_j = \left[\begin{array}{cccccc|cccccc}
\alpha_{0j}-1 & \beta_{0j} & 1 & & & & & & & & & \\
\beta_{0j} & \alpha_{0j} & \beta_{0j} & 1 & & & & & & & & \\
1 & \beta_{0j} & \alpha_{0j} & \beta_{0j} & 1 & & & & & & & \\
 & \ddots & \ddots & \ddots & & & & & & & & \\
 & 1 & \beta_{0j} & \alpha_{0j} & \beta_{0j} & & & & & & & \\
 & & 1 & \beta_{0j} & \alpha_{0j} & 1 & & & & & & \\
\hline
 & & & & 1 & \alpha_{1j} & \beta_{1j} & 1 & & & & \\
 & & & & & \beta_{1j} & \alpha_{1j} & \beta_{1j} & 1 & & & \\
 & & & & & & \ddots & \ddots & \ddots & & & \\
 & & & & & & 1 & \beta_{Nj} & \alpha_{Nj} & \beta_{Nj} & 1 & \\
 & & & & & & & 1 & \beta_{Nj} & \alpha_{Nj} & \beta_{Nj} & \\
 & & & & & & & & 1 & \beta_{Nj} & \alpha_{Nj}-1 &
\end{array}\right],$$

$$\overline{B}_j = 2I + \left[\begin{array}{ccccc|ccccc}
-\gamma_{0j} & 1 & & & & & & & & \\
1 & -\gamma_{0j} & 1 & & & & & & & \\
 & \ddots & \ddots & \ddots & & & & & & \\
 & & 1 & -\gamma_{0j} & 1 & & & & & \\
 & & & 1 & -\gamma_{0j} & & & & & \\
\hline
 & & & & & -\gamma_{1j} & 1 & & & \\
 & & & & & 1 & -\gamma_{1j} & 1 & & \\
 & & & & & & \ddots & \ddots & \ddots & \\
 & & & & & & & 1 & -\gamma_{Nj} & 1 \\
 & & & & & & & & 1 & -\gamma_{Nj}
\end{array}\right],$$

where $\alpha_{ij} = 20 + 2\left(\frac{p_{ij}}{n_i}\right)^2 + 2\left(\frac{q_{ij}}{m_j}\right)^2$, $\beta_{ij} = 8 + \left(\frac{p_{ij}}{n_i}\right)^2$ and $\gamma_{ij} = 8 + \left(\frac{q_{ij}}{m_j}\right)^2$, $i = 0, \ldots, N$; $j = 0, \ldots, M$, and the description of the right side b is given in Appendix A.

For solving equation (6.27), A must be well defined. So, we are going to investigate the properties of A. Positive definiteness of A is one property that we are interested to have. If A is positive definite matrix then equation (6.27) can be efficiently solved by direct or iterative methods.

Any linear operator in a finite dimensional space can be represented in a matrix form. Let the matrices, $A^{(1)}$, $A^{(2)}$, and $A^{(3)}$ represent the difference operators $\Lambda_1^2 - \left(\frac{p_{ij}}{h_i}\right)^2 \Lambda_1$, $\Lambda_2^2 - \left(\frac{q_{ij}}{l_j}\right)^2 \Lambda_2$, and $2\Lambda_1\Lambda_2$, respectively.

To show that the matrix A in (6.27) is positive definite, let us represent this matrix in the following form

$$A = A^{(1)} + A^{(2)} + A^{(3)},$$

where

$$A^{(1)} = \begin{bmatrix} \widehat{U}_0^{(1)} & & \\ & \ddots & \\ & & \widehat{U}_M^{(1)} \end{bmatrix}, \qquad \widehat{U}_j^{(1)} = \begin{bmatrix} U_j^{(1)} & & \\ & \ddots & \\ & & U_j^{(1)} \end{bmatrix}_{(m_j-1)},$$

$$U_j^{(1)} = \left[\begin{array}{ccccc|cccccc}
\alpha_{0j}^{(1)}-1 & \beta_{0j}^{(1)} & 1 & & & & & & & & \\
\beta_{0j}^{(1)} & \alpha_{0j}^{(1)} & \beta_{0j}^{(1)} & 1 & & & & & & & \\
1 & \beta_{0j}^{(1)} & \alpha_{0j}^{(1)} & \beta_{0j}^{(1)} & 1 & & & & & & \\
 & \ddots & \ddots & \ddots & & & & & & & \\
 & 1 & \beta_{0j}^{(1)} & \alpha_{0j}^{(1)} & \beta_{0j}^{(1)} & & & & & & \\
 & & 1 & \beta_{0j}^{(1)} & \alpha_{0j}^{(1)} & 1 & & & & & \\
\hline
 & & & & 1 & \alpha_{1j}^{(1)} & \beta_{1j}^{(1)} & 1 & & & \\
 & & & & & \beta_{1j}^{(1)} & \alpha_{1j}^{(1)} & \beta_{1j}^{(1)} & 1 & & \\
 & & & & & & \ddots & \ddots & \ddots & & \\
 & & & & & & 1 & \beta_{Nj}^{(1)} & \alpha_{Nj}^{(1)} & \beta_{Nj}^{(1)} & 1 \\
 & & & & & & & 1 & \beta_{Nj}^{(1)} & \alpha_{Nj}^{(1)} & \beta_{Nj}^{(1)} \\
 & & & & & & & & 1 & \beta_{Nj}^{(1)} & \alpha_{Nj}^{(1)}-1
\end{array}\right],$$

$$\alpha_{ij}^{(1)} = 6 + 2\left(\frac{p_{ij}}{n_i}\right)^2; \quad \beta_{ij}^{(1)} = -\left(4 + \left(\frac{p_{ij}}{n_i}\right)^2\right), \qquad i = 0, \ldots, N;\ j = 0, \ldots, M,$$

$$A^{(2)} = \left[\begin{array}{ccccc|cccccc}
\widehat{U}_0^{(2)}-I & \widehat{V}_0^{(2)} & I & & & & & & & & \\
\widehat{V}_0^{(2)} & \widehat{U}_0^{(2)} & \widehat{V}_0^{(2)} & I & & & & & & & \\
I & \widehat{V}_0^{(2)} & \widehat{U}_0^{(2)} & \widehat{V}_0^{(2)} & I & & & & & & \\
 & \ddots & \ddots & \ddots & & & & & & & \\
 & I & \widehat{V}_0^{(2)} & \widehat{U}_0^{(2)} & \widehat{V}_0^{(2)} & & & & & & \\
 & & I & \widehat{V}_0^{(2)} & \widehat{U}_0^{(2)} & I & & & & & \\
\hline
 & & & & I & \widehat{U}_1^{(2)} & \widehat{V}_1^{(2)} & I & & & \\
 & & & & & \widehat{V}_1^{(2)} & \widehat{U}_1^{(2)} & \widehat{V}_1^{(2)} & I & & \\
 & & & & & & \ddots & \ddots & \ddots & & \\
 & & & & & & I & \widehat{V}_M^{(2)} & \widehat{U}_M^{(2)} & \widehat{V}_M^{(2)} & I \\
 & & & & & & & I & \widehat{V}_M^{(2)} & \widehat{U}_M^{(2)} & \widehat{V}_M^{(2)} \\
 & & & & & & & & I & \widehat{V}_M^{(2)} & \widehat{U}_M^{(2)}-I
\end{array}\right],$$

$$\widehat{U}_j^{(2)} = \begin{bmatrix} U_{0j}^{(2)} & & \\ & \ddots & \\ & & U_{Nj}^{(2)} \end{bmatrix}, \qquad U_{ij}^{(2)} = \begin{bmatrix} \alpha_{ij}^{(2)} & & \\ & \ddots & \\ & & \alpha_{ij}^{(2)} \end{bmatrix}_{(n_i-1)},$$

$$\widehat{V}_j^{(2)} = \begin{bmatrix} V_{0j}^{(2)} & & \\ & \ddots & \\ & & V_{Nj}^{(2)} \end{bmatrix}, \qquad V_{ij}^{(2)} = \begin{bmatrix} \beta_{ij}^{(2)} & & \\ & \ddots & \\ & & \beta_{ij}^{(2)} \end{bmatrix}_{(n_i-1)},$$

$$\alpha_{ij}^{(2)} = 6 + 2\left(\frac{q_{ij}}{m_j}\right)^2; \quad \beta_{ij}^{(2)} = -4 - \left(\frac{q_{ij}}{m_j}\right)^2, \qquad i = 0, \ldots, N;\ j = 0, \ldots, M,$$

$$A^{(3)} = \begin{bmatrix} \widehat{U}^{(3)} & & \\ & \ddots & \\ & & \widehat{U}^{(3)} \end{bmatrix}_{M+1},$$

$$\widehat{U}^{(3)} = \begin{bmatrix} \overline{U}^{(3)} & \overline{V}^{(3)} & & & \\ \overline{V}^{(3)} & \overline{U}^{(3)} & \overline{V}^{(3)} & & \\ & \ddots & \ddots & \ddots & \\ & & \overline{V}^{(3)} & \overline{U}^{(3)} & \overline{V}^{(3)} \\ & & & \overline{V}^{(3)} & \overline{U}^{(3)} \end{bmatrix}_{(m_j-1)},$$

$$\overline{V}^{(3)} = \begin{bmatrix} V^{(3)} & & \\ & \ddots & \\ & & V^{(3)} \end{bmatrix}_{N+1}; \quad V^{(3)} = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}_{(n_i-1)},$$

$$\overline{U}^{(3)} = -2\overline{V}^{(3)}.$$

Since $U_j^{(1)}$ has the same pattern as the coefficient matrix of 1-D DMBVP which is positive definite matrix then for all eigenvalues of this matrix we have

$$\lambda(U_j^{(1)}) > 0, \quad j = 0, \ldots, M.$$

Therefore,

$$\lambda(A^{(1)}) > 0 \tag{6.28}$$

because $\lambda(U_j^{(1)}) > 0, \ j = 0, \ldots, M$ is a collection of eigenvalues $\lambda(\widehat{U}_j^{(1)}) > 0, \ j = 0, \ldots, M$. And $\lambda(\widehat{U}_j^{(1)}) > 0, \ j = 0, \ldots, M$ is a collection of eigenvalues $\lambda(A^{(1)}) > 0$.

For any $u \in \mathbb{R}^n$, there exists an unique permutation matrix $P \in \mathbb{R}^{n\times n}$ and a vector $v \in \mathbb{R}^n$ such that

$$u = Pv,$$

where $n = (n_i - 1)(m_j - 1)(N + 1)(M + 1), \ PP^T = P^T P = I,$

$$v_{jl;ik} = u_{ik;jl},$$

$$k = 1, \ldots, n_i - 1; \ i = 0, \ldots, N; \ l = 1, \ldots, m_j - 1; \ j = 0, \ldots, M.$$

Let us consider

$$\begin{aligned} A^{(2)}u &= A^{(2)}Pv \\ \lambda(A^{(2)})u &= \lambda(A^{(2)})Pv = P\lambda(A^{(2)})v. \end{aligned}$$

Then

$$P^T A^{(2)} P v = \lambda(A^{(2)})v.$$

Therefore, $B^{(2)} = P^T A^{(2)} P$ and $A^{(2)}$ have the same eigenvalues $\lambda(A^{(2)})$ and $B^{(2)}$ has the same structure as $A^{(1)}$. In the same way as for $A^{(1)}$, we obtain that

$$\lambda(B^{(2)}) > 0.$$

Therefore,

$$\lambda(A^{(2)}) > 0. \tag{6.29}$$

Since $\lambda(V^{(3)}) < 0$ (see M. A. Malcolm (1977)) and $\lambda(\overline{V}^{(3)})$ is a collection of eigenvalues $\lambda(V^{(3)})$ then

$$\lambda(V^{(3)}) < 0.$$

For any $u \in \mathbb{R}^n$ where $n = (n_i - 1)(m_j - 1)(N + 1)(M + 1)$, we obtain

$$\begin{aligned}
\lambda(\widehat{U}^{(3)})u &= \widehat{U}^{(3)}u \\
&= \begin{bmatrix} \overline{V}^{(3)} & & & & \\ & \overline{V}^{(3)} & & & \\ & & \ddots & & \\ & & & \overline{V}^{(3)} & \\ & & & & \overline{V}^{(3)} \end{bmatrix} \begin{bmatrix} -2I & I & & & \\ I & -2I & I & & \\ & \ddots & \ddots & \ddots & \\ & & I & -2I & I \\ & & & I & -2I \end{bmatrix} u \\
&= W^{(3)}\overline{W}^{(3)}u \\
&= W^{(3)}\lambda(\overline{W}^{(3)})u \\
&= \lambda(\overline{W}^{(3)})W^{(3)}u \\
&= \lambda(\overline{W}^{(3)})\lambda(W^{(3)})u.
\end{aligned}$$

Therefore,

$$\lambda(\widehat{U}^{(3)}) = \lambda(\overline{W}^{(3)})\lambda(W^{(3)}).$$

As $\lambda(W^{(3)})$ is a collection of eigenvalues $\lambda(\overline{V}^{(3)})$ then applying the idea of M. A. Malcolm (1977), we obtain $\lambda(W^{(3)}) < 0$ and $\lambda(\overline{W}^{(3)}) < 0$. Therefore, $\lambda(\widehat{U}^{(3)}) > 0$. Since $\lambda(A^{(3)})$ is a collection of eigenvalues $\lambda(\widehat{U}^{(3)}) > 0$ then

$$\lambda(A^{(3)}) > 0. \tag{6.30}$$

Since $A = A^{(1)} + A^{(2)} + A^{(3)}$, matrices $A^{(1)}$, $A^{(2)}$, and $A^{(3)}$ are symmetric matrices, and $\lambda(A^{(1)}) > 0$, $\lambda(A^{(2)}) > 0$, $\lambda(A^{(3)}) > 0$ then

$$\lambda(A) \geq \lambda(A^{(1)}) + \lambda(A^{(2)}) + \lambda(A^{(3)}) > 0.$$

Therefore, A is a positive definite matrix.

6.4 Successive Over-Relaxation (SOR) Algorithm

From previous section, we know already that the coefficient matrix A of our linear system is symmetric positive definite matrix. Therefore, iterative methods such as Jacobi, Gauss-Seidel, etc. should ever converge to the solution of this system. In general, the convergence of Gauss-Seidel method can be accelerated by using a relaxation parameter ω. This give us successive over-relaxation (SOR for short) method which is chosen for solving the system of difference equations.

For the numerical treatment of the finite difference problem (6.8)–(6.21) first we solve 1-D difference equations (6.9) and (6.10) on the main mesh Δ by algorithm of Chapter 4.

Then on the refinement we define a mesh function

$$\{u^{(0)}_{ik;jl}\,|\,k=1,\dots,n_i,\ i=0,\dots,N,\ l=1,\dots,m_j,\ j=0,\dots,M\}$$

by a piecewise linear interpolation of the initial data either in x or y directions.

In each subdomain $\Omega_{ij},\ i=0,\dots,N,\ j=0,\dots,M$, difference equations (6.8) can be rewritten in the componentwise form

$$\begin{aligned} u_{ik;j,l-2} + 2u_{i,k-1;j,l-1} - \gamma_{ij}u_{ik;j,l-1} + 2u_{i,k+1;j,l-1} + u_{i,k-2;jl} \\ -\beta_{ij}u_{i,k-1;jl} + \alpha_{ij}u_{ik;jl} - \beta_{ij}u_{i,k+1;jl} + u_{i,k+2;jl} + 2u_{i,k-1;j,l+1} \\ -\gamma_{ij}u_{ik;j,l+1} + 2u_{i,k+1;j,l+1} + u_{ik;j,l+2} &= 0, \end{aligned}$$

or

$$\begin{aligned} u_{ik;jl} &= \frac{1}{\alpha_{ij}}\Bigg\{\beta_{ij}\Big[u_{i,k-1;jl}+u_{i,k+1;jl}\Big]+\gamma_{ij}\Big[u_{ik;j,l-1}+u_{ik;j,l+1}\Big] \\ &\quad -2\Big[u_{i,k-1;j,l-1}+u_{i,k-1;j,l+1}+u_{i,k+1;j,l-1}+u_{i,k+1;j,l+1}\Big] \\ &\quad -u_{ik;j,l-2}-u_{ik;j,l+2}-u_{i,k-2;jl}-u_{i,k+2;jl}\Bigg\}, \end{aligned} \tag{6.31}$$

where $\alpha_{ij} = 20 + 2\left(\frac{p_{ij}}{n_i}\right)^2 + 2\left(\frac{q_{ij}}{m_j}\right)^2$, $\beta_{ij} = 8 + \left(\frac{p_{ij}}{n_i}\right)^2$ and $\gamma_{ij} = 8 + \left(\frac{q_{ij}}{m_j}\right)^2$.

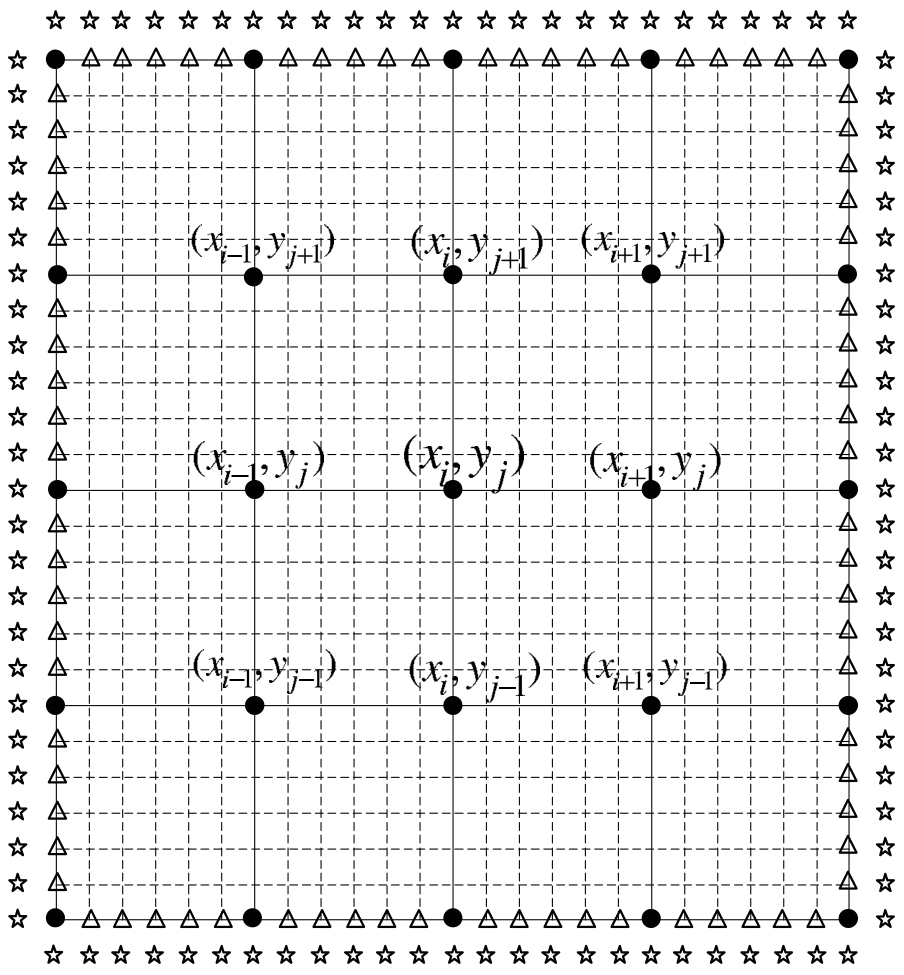

Figure 6.1: Interpolation domain with main mesh and a refinement.

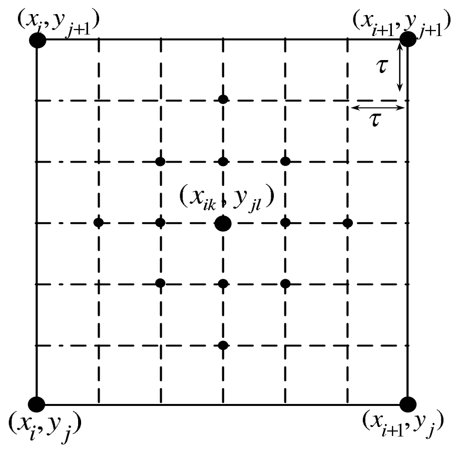

Figure 6.2: Subdomain Ω_{ij} with a refinement.

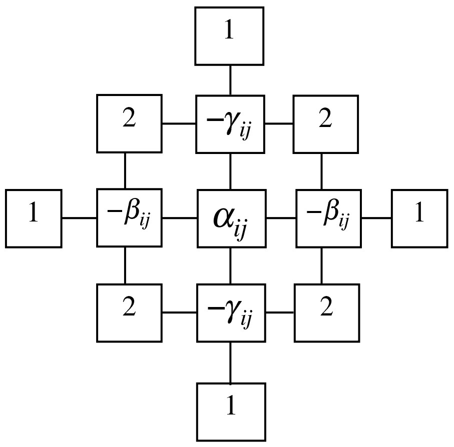

Figure 6.3: Grid stencil in subdomain Ω_{ij}.

Now to obtain a numerical solution on the refinement we apply in each subdomain $\Omega_{ij},\ i=0,\dots,N,\ j=0,\dots,M$, SOR method:

$$\begin{aligned}
\overline{u}_{ik;jl} &= \frac{1}{\alpha_{ij}}\Bigg\{\beta_{ij}\left[u^{(\nu+1)}_{i,k-1;jl}+u^{(\nu)}_{i,k+1;jl}\right]+\gamma_{ij}\left[u^{(\nu+1)}_{ik;j,l-1}+u^{(\nu)}_{ik;j,l+1}\right] \\
&\quad -2\left[u^{(\nu+1)}_{i,k-1;j,l-1}+u^{(\nu)}_{i,k-1;j,l+1}+u^{(\nu+1)}_{i,k+1;j,l-1}+u^{(\nu)}_{i,k+1;j,l+1}\right] \\
&\quad -u^{(\nu+1)}_{ik;j,l-2}-u^{(\nu)}_{ik;j,l+2}-u^{(\nu+1)}_{i,k-2;jl}-u^{(\nu)}_{i,k+2;jl}\Bigg\}, \qquad (6.32)\\
u^{(\nu+1)}_{ik;jl} &= u^{(\nu)}_{ik;jl}+\omega(\overline{u}_{ik;jl}-u^{(\nu)}_{ik;jl}), \qquad (6.33)
\end{aligned}$$

where $1<\omega<2,\quad k=1,\dots,n_i-1,\quad i=0,\dots,N,$
$\qquad l=1,\dots,m_j-1,\quad j=0,\dots,M.$

The formula (6.32) gives an approximation by Gauss-Seidel method while the next step (6.33) is used for the acceleration of the convergence.

Note that near the border of the domain Ω the extra unknowns $u_{0,-1;jl}$, $u_{N+1,1;jl}$, $j=0,\dots,M$, $l=1,\dots,m_j$, and $u_{ik;0,-1}$, $u_{ik;M+1,1}$, $i=0,\dots,N$, $k=1,\dots,n_i$ are eliminated using the equations (6.19) and (6.20) and do not participate in the iterations.

The described above approach can be formalized as the following algorithm.

Algorithm 6.1. SOR method for solving the system of difference equations. Let u be the approximate solution of the system (6.8)–(6.21), ε be an error bound of exact solution of this system and ν be an iteration number.

1. Input the data
   $N,\ M,\ h,\ \omega,\ \varepsilon,$
   $(x_i,y_j,f_{ij}),\quad i=0,\dots,N+1,\ j=0,\dots,M+1,$
   $f^{(2,0)}_{ij},\quad i=0,\dots,N+1,\ j=0,M+1,$
   $f^{(0,2)}_{ij},\quad i=0,N+1,\ j=0,\dots,M+1,$
   $f^{(2,2)}_{ij},\quad i=0,N+1,\ j=0,M+1.$

2. Calculate the following quantities:
   $h_i,\ n_i,\ i=0,\dots,N,$
   $l_j,\ m_j,\ j=0,\dots,M,$
   $\alpha_{ij},\ \beta_{ij}$, and γ_{ij} for each $i=0,\dots,N,\ j=0,\dots,M.$

3. Define tension parameters $p_{ij},\ q_{ij},\quad i=0,\dots,N,\ j=0,\dots,M$ by one of 1-D algorithms of shape preserving interpolation.

4. Solve the difference equations (6.9) and (6.10) as 1-D problems to find:
   $u_{ik;jl},\ \ k=\begin{cases}0 & \text{if } i=0,\dots,N-1,\\ 0,n_N & \text{if } i=N,\end{cases}\qquad l=1,\dots,m_j-1,\ j=0,\dots,M,$

$$u_{ik;jl},\ \ k=1,\dots,n_i-1,\ i=0,\dots,N,\quad l=\begin{cases}0 & \text{if } j=0,\dots,M-1,\\ 0,m_M & \text{if } j=M,\end{cases}$$

$$u^{(2,0)}_{ik;jl},\ \ k=\begin{cases}0 & \text{if } i=0,\\ n_N & \text{if } i=N,\end{cases}\qquad l=1,\dots,m_j-1,\quad j=0,\dots,M,$$

$$u^{(0,2)}_{ik;jl},\ \ k=1,\dots,n_i-1,\quad i=0,\dots,N,\qquad l=\begin{cases}0 & \text{if } j=0,\\ m_M & \text{if } j=M.\end{cases}$$

5. Apply the equations (6.24) to find
   $u_{0,-1;jl},\ u_{N,n_N+1;jl},\qquad l=1,\dots,m_j,\ j=0,\dots,M;$
   $u_{ik;0,-1},\ u_{ik;M,m_M+1},\qquad k=1,\dots,n_i,\ i=0,\dots,N.$

6. Find the initial values
   $u^{(0)}_{ik,jl},\ k=1,\dots,n_i-1,\ i=0,\dots,N,\ l=1,\dots,m_j-1,\ j=0,\dots,M$
   by using a piecewise linear interpolation.

7. Make the iterations:

```
$\nu = 0$
DO WHILE   $\max_{i,k;j,l} |u^{(\nu+1)}_{ik;jl} - u^{(\nu)}_{ik;jl}| \geq \varepsilon$
  DO $j = 0,\dots,M$
    DO $i = 0,\dots,N$
      DO $l = 1,\dots,m_j - 1$
        DO $k = 1,\dots,n_i - 1$
          Find $\overline{u}$ by the equation (6.32).
          Find $u^{(\nu+1)}$ by the equation (6.33).
        END DO
      END DO
    END DO
  END DO
  $\nu = \nu + 1$
END DO
```

8. Print the output
   $u^{(\nu)}_{00;00},\quad u^{(\nu)}_{00;jl},\quad u^{(\nu)}_{ik;00}$, and $u^{(\nu)}_{ik;jl}$,
   $$k=1,\dots,n_i,\quad i=0,\dots,N,\quad l=1,\dots,m_j,\quad j=0,\dots,M.$$

In this algorithm the computer time, especially the operation count, is around $18n$, i.e. $O(n)$, for each iteration where n is the number of unknowns in the interpolation problem. The required memory is about n. The algorithm can be easily converted into a programming code.

6.5 Method of Fractional Steps

In recent years computers' evolution is going dramatically fast. Computers have been improved a lot and became much more powerful. One of the new types of computers is a multi-processing computer. So, we should develop algorithms that support and could be suitable for this evolution. In this section we introduce the method of fractional steps (see N. N. Yanenko (1971)) for solving 2-D shape preserving interpolation problem. This method splits the original 2-D problem into a set of 1-D problems. At each step one has to solve m linearly independent 1-D systems of linear equations where m is the number of 1-D problems in appropriate direction. Therefore, we can solve each of these systems of linear equations at every step by m independent parallel processors. This method is preferable for multi-processing computers.

Let us consider along with the 2-D DMBVP (6.2)–(6.7) the unsteady problem

$$\begin{aligned} \frac{\partial s}{\partial t} + Ls &= 0 \quad \text{in each} \quad \Omega_{ij},\ i=0,\dots,N,\ j=0,\dots,M, && (6.34) \\ s(x,y,0) &= s_0(x,y) && (6.35) \end{aligned}$$

with the same smoothness, interpolation and boundary conditions (6.3)–(6.7). Denote by $S(x,y)$ the solution of the problem (6.2)–(6.7) and by $s(x,y,t)$ the solution of problem (6.34), (6.35), (6.3)–(6.7). Then

$$v(x,y,t) = s(x,y,t) - S(x,y)$$

satisfies the equation (6.34) with the initial conditions

$$v(x,y,0) = v_0(x,y) = s_0(x,y) - S(x,y),$$

the smoothness conditions (6.5), and zero boundary conditions

$$v_0(x,y) = 0 \qquad \text{for} \quad (x,y) \in \Delta,$$

$$\left.\frac{\partial^2 v_0}{\partial n^2}\right|_{\Gamma} = 0,$$

where $\frac{\partial^2 v_0}{\partial n^2}$ is the second normal derivative of v_0 across the boundary Γ.

The function $v(x,y,t)$ is represented in the form

$$v(x,y,t) = \sum_{k_1=1}^{\infty}\sum_{k_2=1}^{\infty} A_{ijk_1k_2}(t) \sin\left(k_1 \frac{x-x_i}{h_i}\pi\right) \sin\left(k_2 \frac{y-y_j}{l_j}\pi\right), \qquad (6.36)$$

$$(x,y) \in \Omega_{ij},\ i=0,\dots,N,\ j=0,\dots,M,$$

where

$$A_{ijk_1k_2}(t) = a_{ijk_1k_2} e^{-\pi^2 b_{ij} t}, \qquad (6.37)$$

$$b_{ij} = \pi^2 \left[\left(\frac{k_1}{h_i}\right)^2 + \left(\frac{k_2}{l_j}\right)^2\right]^2 + \left(\frac{p_{ij}}{h_i}\right)^2 \left(\frac{k_1}{h_i}\right)^2 + \left(\frac{q_{ij}}{l_j}\right)^2 \left(\frac{k_2}{l_j}\right)^2$$

is the Fourier coefficient of the function $v(x,y,t)$; $a_{ijk_1k_2}$ is the Fourier coefficient of the function $v_0(x,y)$ on Ω_{ij}.

Formulae (6.36) and (6.37) can be represented in operator form

$$v = B(t)v_0.$$

The operator $B(t)$ in space $L_2(\overline{\Omega})$ has the norm

$$\|B(t)\| = e^{-\pi^2 bt}, \quad b = \max_{i,j} b_{ij}.$$

It follows from this that

$$\|B(t)\| \to 0, \qquad t \to \infty.$$

This means that

$$\|v(x,y,t)\| = \|s(x,y,t) - S(x,y)\| \to 0 \quad \text{as } t \to \infty,$$

i.e., the solution of the unsteady problem approaches the solution of the steady problem with the same smoothness, interpolation, and boundary conditions, regardless of the choice of the initial data.

For the numerical solution of the generalized thin plate equation (6.2) we consider in each rectangular $\Omega_{ij},\ i = 0,\dots,N,\ j = 0,\dots,M$, the following splitting scheme (see N. N. Yanenko (1971))

$$\frac{u^{n+1/2} - u^n}{\tau} + \Psi_1 u^{n+1/2} + \Psi_3 u^n \qquad = 0, \tag{6.38a}$$

$$\frac{u^{n+1} - u^{n+1/2}}{\tau} + \Psi_2 u^{n+1} + \Psi_3 u^{n+1/2} = 0, \tag{6.38b}$$

where

$$u = \Big\{ u_{ik;jl} \mid k = 1,\dots,n_i - 1;\ i = 0,\dots,N;\ l = 1,\dots,m_j - 1;\ j = 0,\dots,M \Big\},$$
$$\Psi_1 = \Lambda_1^2 - p\Lambda_1,\ \Psi_2 = \Lambda_2^2 - q\Lambda_2,\ \Psi_3 = \Lambda_1\Lambda_2,\ p = (p_{ij}/h_i)^2,\ q = (q_{ij}/l_j)^2.$$

Equation (6.38) can be rewritten in the following form

$$\left.\begin{array}{rcl} (I + \tau\Psi_1)u^{n+1/2} & = & (I - \tau\Psi_3)u^n, \\ (I + \tau\Psi_2)u^{n+1} & = & (I - \tau\Psi_3)u^{n+1/2}, \end{array}\right\} \tag{6.39}$$

where I is an identity operator.

Eliminating from the equations (6.39) the fractional step $u^{n+1/2}$ yields

$$(I + \tau\Psi_1)(I + \tau\Psi_2)u^{n+1} = (I - \tau\Psi_3)^2 u^n$$

or

$$\left[I + \tau\left(\Psi_1 + \Psi_2\right) + \tau^2\Psi_1\Psi_3\right] u^{n+1} = \left[I - 2\tau\Psi_3 + \tau^2\Psi_3^2\right] u^n.$$

After some simple transformations we obtain the following scheme, equivalent to the scheme (6.38),

$$\frac{u^{n+1} - u^n}{\tau} + (\Psi_1 + \Psi_2)u^{n+1} + 2\Psi_3 u^n + \tau(\Psi_1\Psi_2 u^{n+1} - \Psi_3^2 u^n) = 0. \tag{6.40}$$

It follows from this that the scheme (6.40) and the equivalent scheme (6.38) approximate the unsteady generalized thin plate equation (6.34) with the same accuracy $O(\tau + h^2)$ as the scheme

$$\frac{u^{n+1} - u^n}{\tau} + (\Psi_1 + \Psi_2)u^{n+1} + 2\Psi_3 u^n = 0.$$

Let us prove the unconditional stability of the scheme (6.38) or, what is equivalent, the scheme (6.40). Using usual harmonic analysis (see N. N. Yanenko (1971)), assume that

$$u^n = \eta_n e^{i\pi z}, \quad u^{n+1/2} = \eta_{n+1/2} e^{i\pi z}, \quad z = k_1 \frac{x - x_i}{h_i} + k_2 \frac{y - y_j}{l_j}. \tag{6.41}$$

Substituting equations (6.41) into equations (6.38) we obtain for amplification factors

$$\varrho_1 = \frac{\eta_{n+1/2}}{\eta_n} = \frac{1 - a_1 a_2}{1 - p\sqrt{\tau} a_1 + a_1^2}, \quad \varrho_2 = \frac{\eta_{n+1}}{\eta_{n+1/2}} = \frac{1 - a_1 a_2}{1 - q\sqrt{\tau} a_2 + a_2^2},$$

$$\varrho = \varrho_1 \varrho_2 = \frac{(1 - a_1 a_2)^2}{(1 - p\sqrt{\tau} a_1 + a_1^2)(1 - q\sqrt{\tau} a_2 + a_2^2)},$$

where

$$a_1 = -\frac{4\sqrt{\tau}}{h^2} \sin^2\left(\frac{k_1 h}{2} \frac{\pi}{h_i}\right), \quad k_1 = 1, \dots, n_i - 1, \quad n_i h = h_i,$$
$$a_2 = -\frac{4\sqrt{\tau}}{h^2} \sin^2\left(\frac{k_2 h}{2} \frac{\pi}{l_j}\right), \quad k_2 = 1, \dots, m_j - 1, \quad m_j h = l_j.$$

It follows from this that

$$\begin{aligned} 0 \le \varrho &= \frac{(1 - a_1 a_2)^2}{(1 - p\sqrt{\tau} a_1 + a_1^2)(1 - q\sqrt{\tau} a_2 + a_2^2)} \\ &\le \frac{(1 - a_1 a_2)^2}{(1 + a_1^2)(1 + a_2^2)} = \frac{(1 - a_1 a_2)^2}{1 + a_1^2 + a_2^2 + a_1^2 a_2^2} \\ &\le \left(\frac{1 - a_1 a_2}{1 + a_1 a_2}\right)^2 \quad < 1 \end{aligned}$$

for any τ. The absolute stability of the scheme (6.38) is proved.

Thus, with the above proven approximation property and strong stability of the scheme (6.38) the solutions u^n of the problem (6.34), (6.35), (6.3)–(6.7) approaches the solution u of the problem (6.2)–(6.7) if $n \to \infty$. However the scheme (6.38) has the property of incomplete approximation (see N. N. Yanenko (1971)). By this reason in iterations we have to use small values of the iteration parameter τ, e.g., $\sqrt{\tau}/h^2 =$ constant.

Algorithm 6.2. Method of fractional steps for solving difference equations.

Let u be the approximate solution of the finite-difference problem (6.8)–(6.21), ε be an error bound of exact solution of this system and ν be an iteration number.

1. Input the data
   $N,\ M,\ \tau,\ h,\ \varepsilon,$
   $(x_i, y_j, f_{ij}),\quad i = 0,\dots,N+1,\ j = 0,\dots,M+1,$
   $f_{ij}^{(2,0)},\quad i = 0,\dots,N+1,\ j = 0, M+1,$
   $f_{ij}^{(0,2)},\quad i = 0, N+1,\ j = 0,\dots,M+1,$
   $f_{ij}^{(2,2)},\quad i = 0, N+1,\ j = 0, M+1.$

2. Calculate the following quantities:
   $h_i,\ n_i,\ i = 0,\dots,N,$
   $l_j,\ m_j,\ j = 0,\dots,M.$

3. Define tension parameters $p_{ij},\ q_{ij},\quad i = 0,\dots,N,\ j = 0,\dots,M$, by one of 1-D algorithms of shape preserving interpolation.

4. Solve the difference equations (6.9) and (6.10) as 1-D problems to find:
   $$u_{ik;jl},\ \ k = \begin{cases} 0 & \text{if } i = 0,\dots,N-1, \\ 0, n_N & \text{if } i = N, \end{cases} \qquad l = 1,\dots,m_j-1,\ j = 0,\dots,M,$$
   $$u_{ik;jl},\ \ k = 1,\dots,n_i-1,\ i = 0,\dots,N,\quad l = \begin{cases} 0 & \text{if } j = 0,\dots,M-1, \\ 0, m_M & \text{if } j = M, \end{cases}$$
   $$u_{ik;jl}^{(2,0)},\ \ k = \begin{cases} 0 & \text{if } i = 0, \\ n_N & \text{if } i = N, \end{cases} \qquad l = 1,\dots,m_j-1,\quad j = 0,\dots,M,$$
   $$u_{ik;jl}^{(0,2)},\ \ k = 1,\dots,n_i-1,\quad i = 0,\dots,N, \qquad l = \begin{cases} 0 & \text{if } j = 0, \\ m_M & \text{if } j = M. \end{cases}$$

5. Apply the equations (6.24) to find
   $u_{0,-1;jl},\ u_{N,n_N+1;jl},\qquad l = 1,\dots,m_j,\ j = 0,\dots,M;$
   $u_{ik;0,-1},\ u_{ik;M,m_M+1},\qquad k = 1,\dots,n_i,\ i = 0,\dots,N.$

6. Find the initial values
   $u_{ik,jl}^{(0)},\ k = 1,\dots,n_i-1,\ i = 0,\dots,N,\ l = 1,\dots,m_j-1,\ j = 0,\dots,M$
   by using a piecewise linear interpolation.

7. Make the iterations:

```
$\nu = 0$
DO WHILE $\max_{i,k;j,l} |u_{ik;jl}^{(\nu+1)} - u_{ik;jl}^{(\nu)}| \geq \varepsilon$
    DO $j = 0, \ldots, M$
        DO $i = 0, \ldots, N$
            DO $l = 1, \ldots, m_j - 1$
                Find $u_{ik,jl}^{(\nu+1/2)}$, $k = 1, \ldots, n_i - 1$, $i = 0, \ldots, N$,
                by the equation (6.38a).
            END DO
            DO $k = 1, \ldots, n_i - 1$
                Find $u_{ik,jl}^{(\nu+1)}$, $l = 1, \ldots, m_j - 1$, $j = 0, \ldots, M$,
                by the equation (6.38b).
            END DO
        END DO
    END DO
    $\nu = \nu + 1$
END DO
```

8. Print the output
$u_{00;00}^{(\nu)}$, $u_{00;jl}^{(\nu)}$, $u_{ik;00}^{(\nu)}$, and $u_{ik;jl}^{(\nu)}$,

$$k = 1, \ldots, n_i, \quad i = 0, \ldots, N, \quad l = 1, \ldots, m_j, \quad j = 0, \ldots, M.$$

The computer time, especially the operation count, in this algorithm is around $52n$, i.e., $O(n)$, for each iteration where n is the number of unknowns in the interpolation problem. However, since the numerical solution is updated at each fractional step, the number of iterations is usually smaller than that in SOR iterative method. The required memory is again about n. In general, the method of fractional steps seems to be more preferable and more flexible than SOR iterative method. It can work on both single and multi-processing computers.

6.6 Numerical Experiments and Examples

The algorithms introduced in the previous sections works well on more general data than used in examples given below. If the algorithm fails in the data monotonicity and/or convexity on some intervals then we have to increase the values of the corresponding tension parameters using algorithm of automatic selection of shape control parameters. This provides the properties of monotonicity and convexity for any data.

Example 6.1. We tried to reconstruct the surface which interpolates the data $(x_i, y_j, \widetilde{f}_{ij})$ obtained by taking of Akima's data in Table 4.1 both in x and y directions and using the formula $\widetilde{f}_{ij} = f_i + f_j$. Figures 6.4 and 6.5 show data points and the three-dimensional view of the initial data. As shown in Figure 6.6, the usual thin plate spline with zero tension parameters does not preserve the monotonicity and convexity properties of the initial data. On the other hand, the shape preserving thin plate spline in Figure 6.7 perfectly reproduces the data shape and simultaneously keeps a smooth surface.

Table 6.1 shows the numerical results obtained by applying SOR iterative method with optimal value of the relaxation parameter ω and by the method of fractional steps with $\varepsilon = 0.0005$ on a single processor computer. The method of fractional steps converges about three times faster than the SOR iterative method. But as was observed in the previous sections, the operation count at each step of the SOR iterative method is about three times less than that of the method of fractional steps. Therefore, the performance of both methods on a single processor computer is about the same. However, on a multi-processing computer the method of fractional steps should be faster than the SOR iterative method. Therefore, the method of fractional steps seems to be more preferable and flexible because it supports and works on both platforms.

Table 6.1: The numerical results in Example 6.1.

Method	Tension Parameters	Iteration Count	
		$\overline{n} = 5$	$\overline{n} = 10$
SOR	0	37	342
Splitting	0	13	119
SOR	Optimal	28	319
Splitting	Optimal	12	109

Remark: We call optimal tension parameters the smallest possible values of tension parameters which permit to preserve the data shape. In Table 6.1 (and in Tables 6.2–6.6 below) we denote by $\overline{n}$ $(= n_i = m_j)$ the number of nodes of the refinement in each rectangular Ω_{ij} in x and y direction.

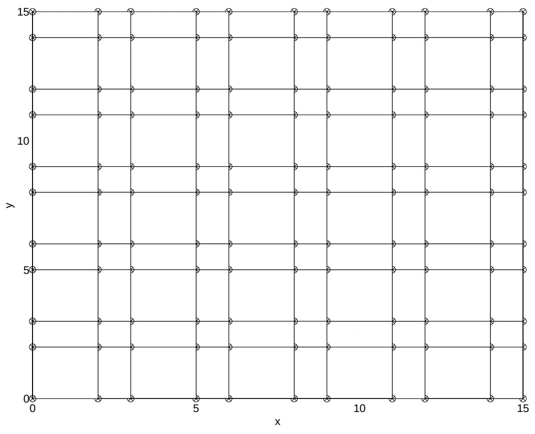

Figure 6.4: The data points.

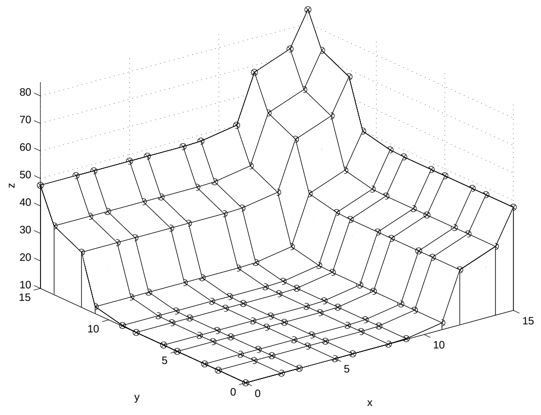

Figure 6.5: 3-D view of the initial data.

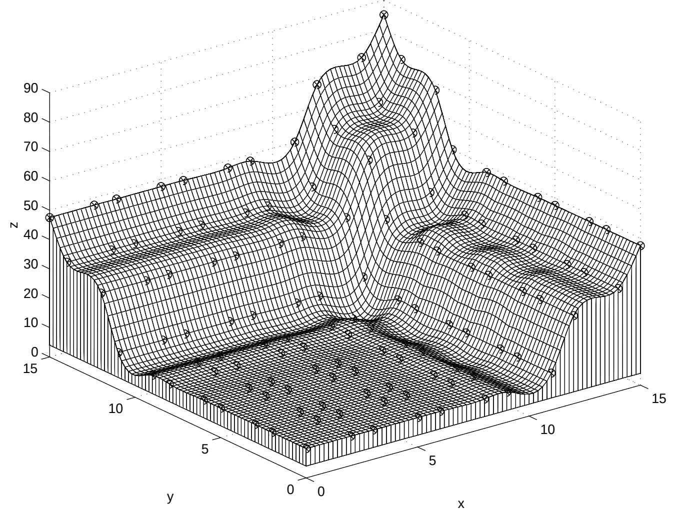

Figure 6.6: Thin plate surface with zero tension parameters.

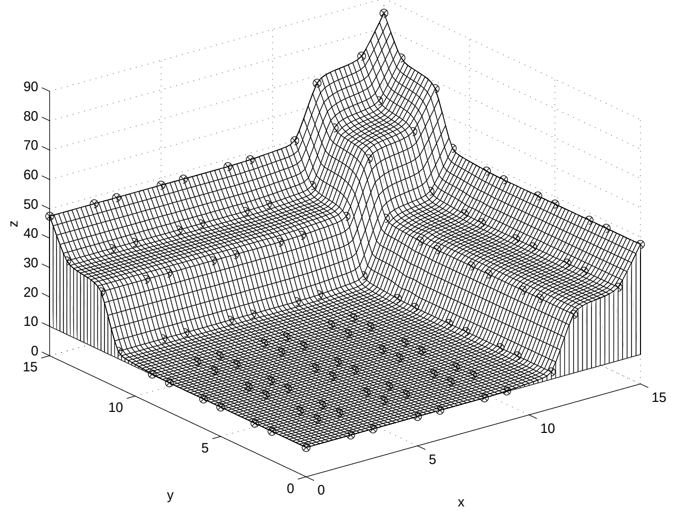

Figure 6.7: Thin plate surface with optimal tension parameters.

Example 6.2. We tried to reconstruct the surface with the initial data shown in Figure 6.8. Figure 6.9 was obtained by setting all tension parameters to zero. This surface does not preserve the shape properties of the data. The new surface shown in Figure 6.10 was obtained by adjusting the tension parameters.

Table 6.2 shows the numerical results obtained by using SOR iterative method with optimal value of the relaxation parameter ω and by the method of fractional steps with $\varepsilon = 0.0005$. The rate of convergence and the results for both methods are similar as in the previous example.

Table 6.2: The numerical results in Example 6.2.

Method	Tension Parameters	Iteration Count	
		$\overline{n} = 5$	$\overline{n} = 10$
SOR	0	18	78
Splitting	0	7	36
SOR	Optimal	8	17
Splitting	Optimal	3	9

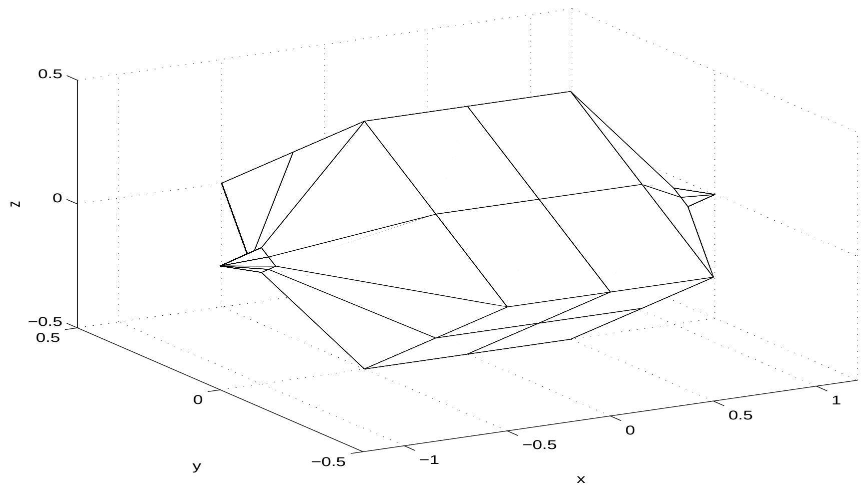

Figure 6.8: The initial data.

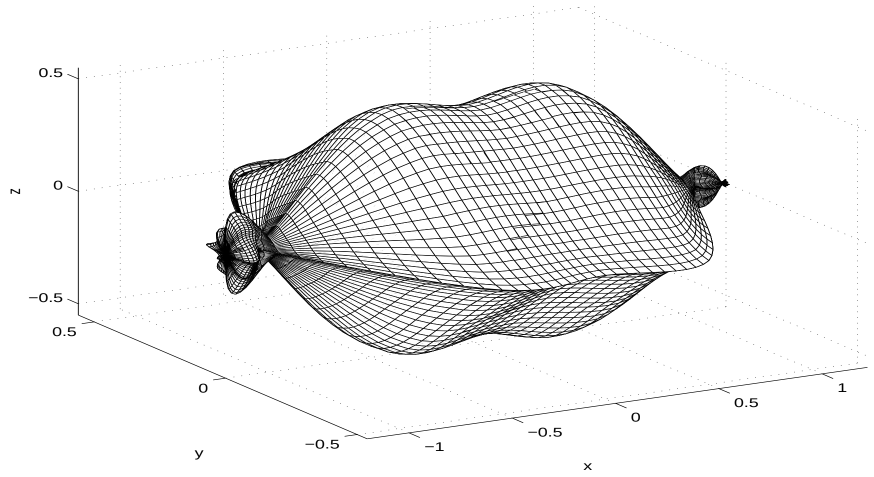

Figure 6.9: The surface with zero tension parameters.

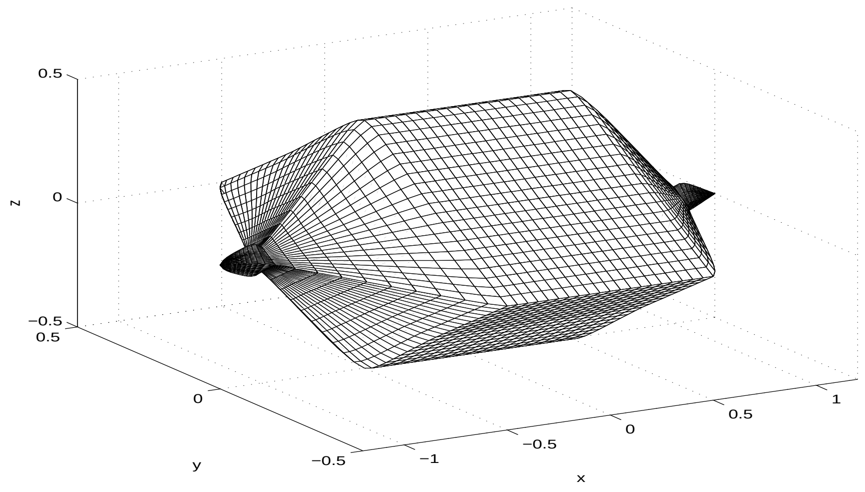

Figure 6.10: The surface with optimal tension parameters.

Example 6.3. We tried to reconstruct the surface by the data of Thailand's topography. The initial data was obtained from Mr. Boonleart A. of King Mongkut's University of Technology Thonburi, Bangkok. Three-dimensional view of the initial data is shown in Figure 6.11. Figure 6.12 was obtained by setting all tension parameters to zero. The surface in Figure 6.12 does not preserve the shape of the data. The new surface which preserves the shape of data was obtained by adjusting the tension parameters and shown in Figure 6.13.

Table 6.3 shows the numerical results obtained by SOR iterative method with optimal value of the relaxation parameter ω and by the method of fractional steps with $\varepsilon = 0.0005$. The rate of convergence and the other characteristics of both methods are again very similar.

Table 6.3: The numerical results in Example 6.3.

Method	Tension Parameters	Iteration Count	
		$\overline{n} = 5$	$\overline{n} = 10$
SOR	0	42	384
Splitting	0	15	149
SOR	Optimal	44	381
Splitting	Optimal	15	134

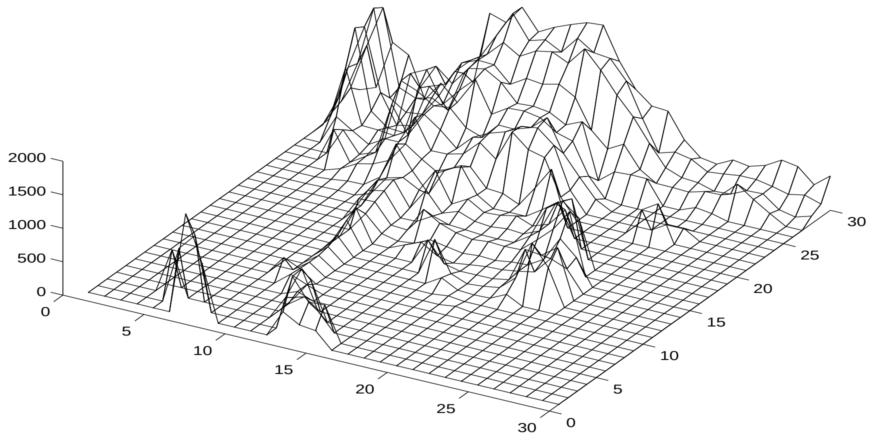

Figure 6.11: The initial data.

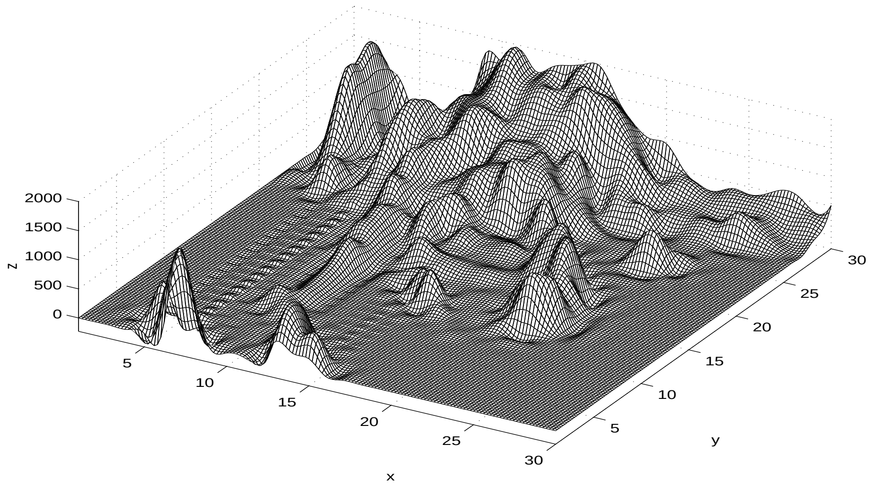

Figure 6.12: The surface with zero tension parameters.

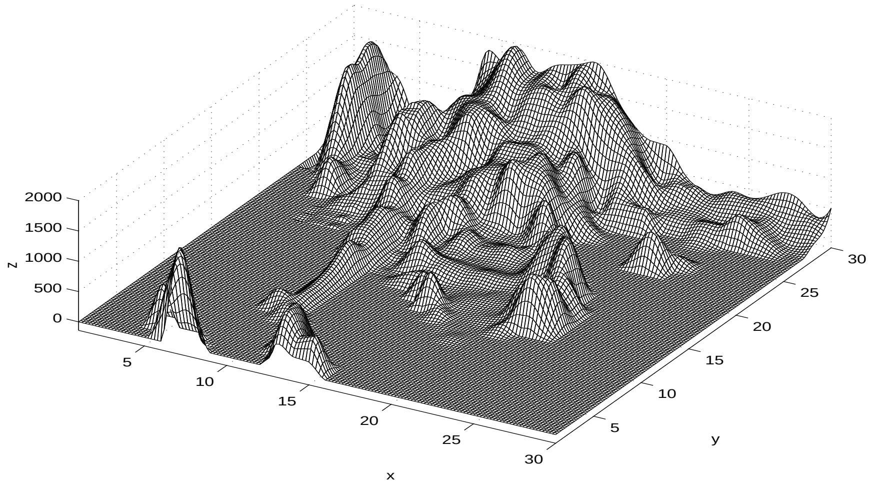

Figure 6.13: The surface with optimal tension parameters.

Example 6.4. We tried to reconstruct the surface of the aircraft data. The initial data was given by Dr. Boris I. Kvasov of Suranaree University of Technology. Three-dimensional views of the initial data are shown in Figures 6.14 (first projection), 6.17 (second projection), and 6.20 (third projection). Figures 6.15, 6.18, and 6.21 were obtained by setting all tension parameters to zero. The surfaces in Figures 6.15, 6.18, and 6.21 do not preserve the shape of the data. The new surfaces which preserve the shape of data were obtained by adjusting the tension parameters and shown in Figures 6.16, 6.19 and 6.22.

Table 6.4 shows the numerical results obtained by SOR iterative method with optimal value of the relaxation parameter ω and by the method of fractional steps with $\varepsilon = 0.0005$. The rate of convergence and the other features of both methods are again very similar.

Table 6.4: The numerical results in Example 6.4.

Method	Tension Parameters	Iteration Count	
		$\overline{n} = 5$	$\overline{n} = 10$
SOR	0	28	161
Splitting	0	9	67
SOR	Optimal	28	184
Splitting	Optimal	9	68

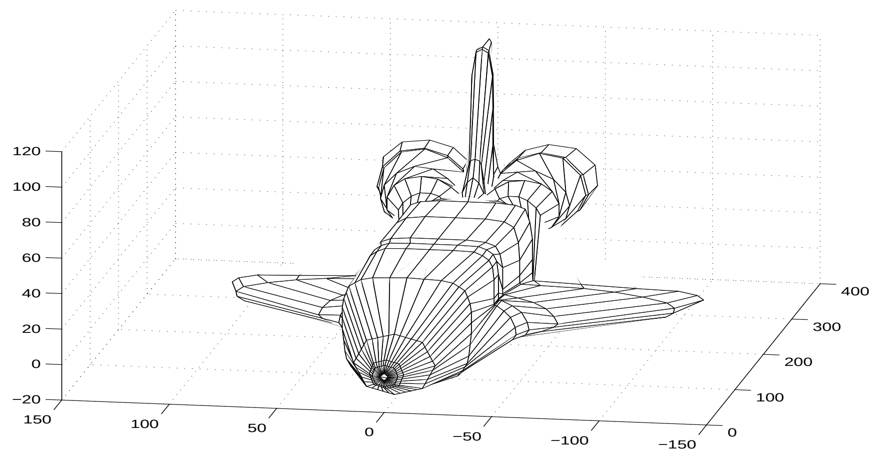

Figure 6.14: The initial data. First projection.

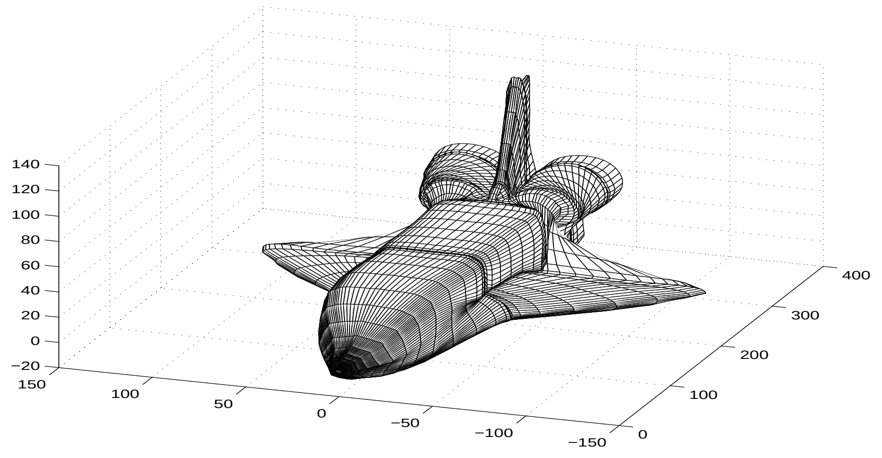

Figure 6.15: The surface with zero tension parameters. First projection.

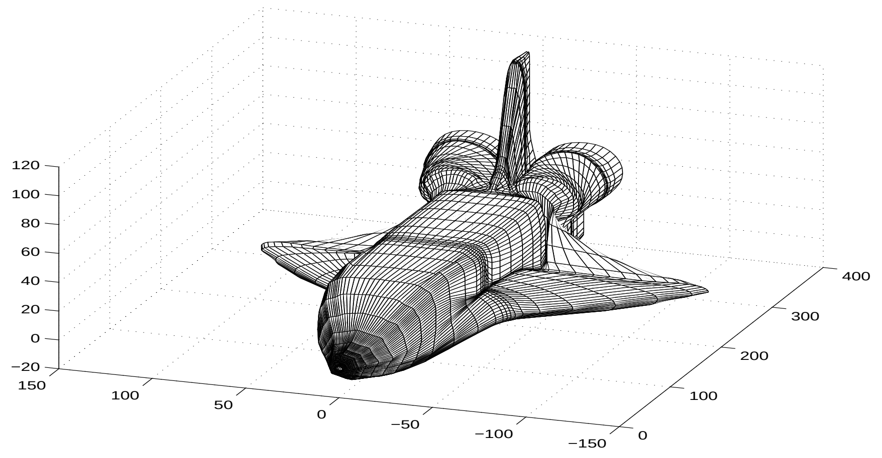

Figure 6.16: The surface with optimal tension parameters. First projection.

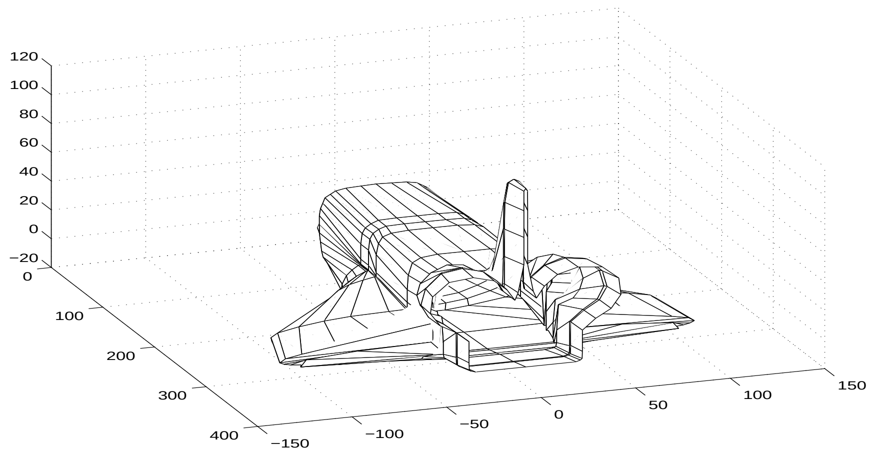

Figure 6.17: The initial data. Second projection.

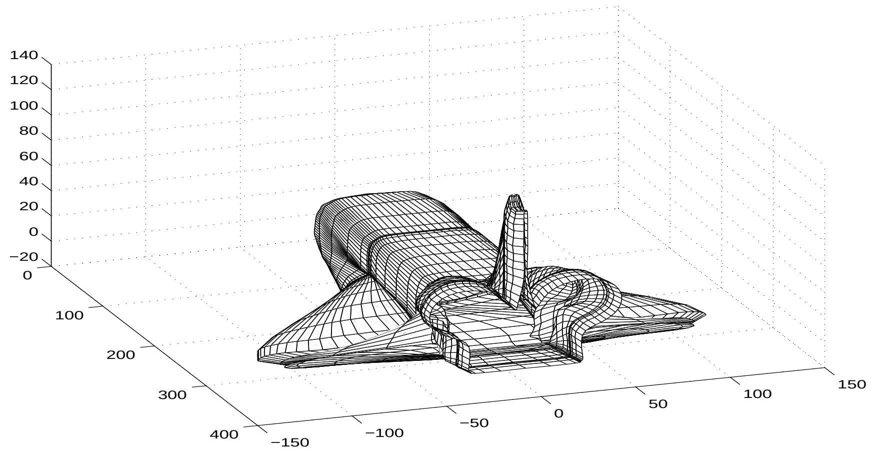

Figure 6.18: The surface with zero tension parameters. Second projection.

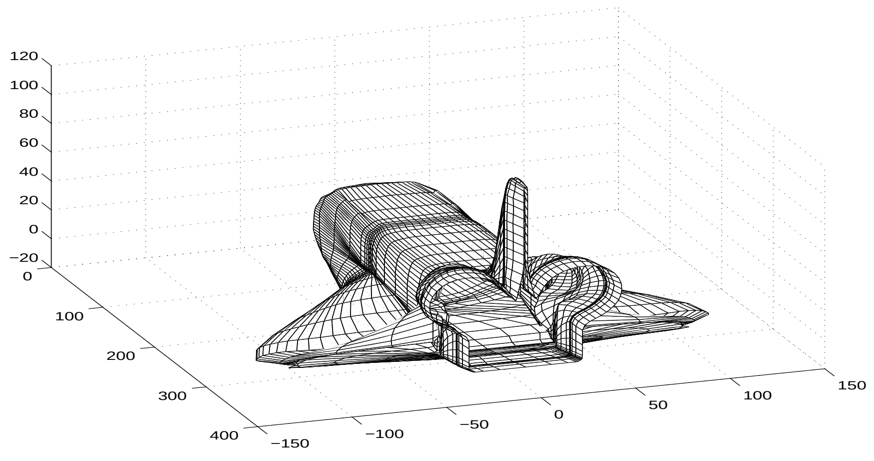

Figure 6.19: The surface with optimal tension parameters. Second projection.

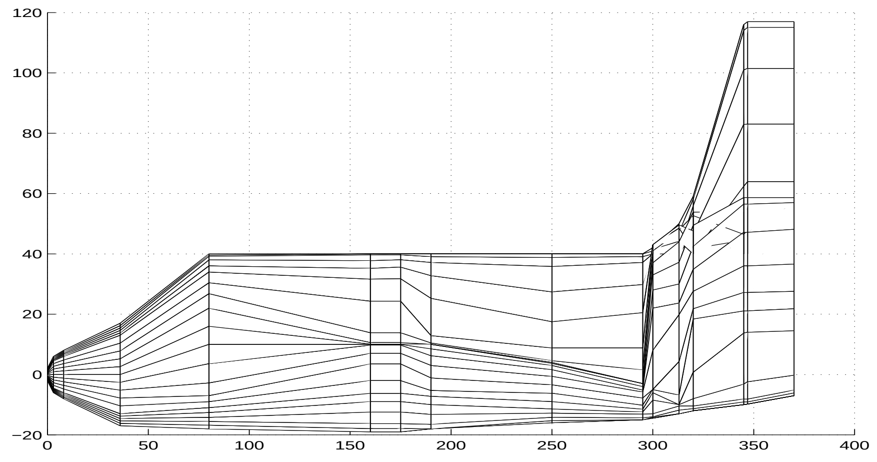

Figure 6.20: The initial data. Third projection.

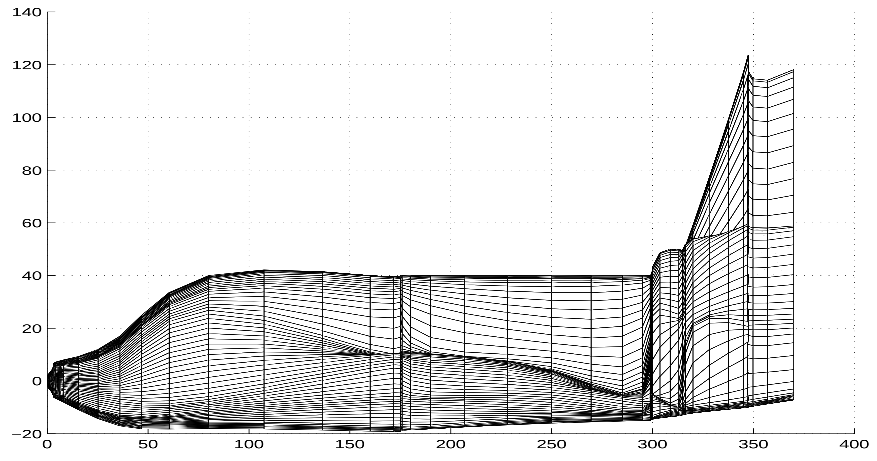

Figure 6.21: The surface with zero tension parameters. Third projection.

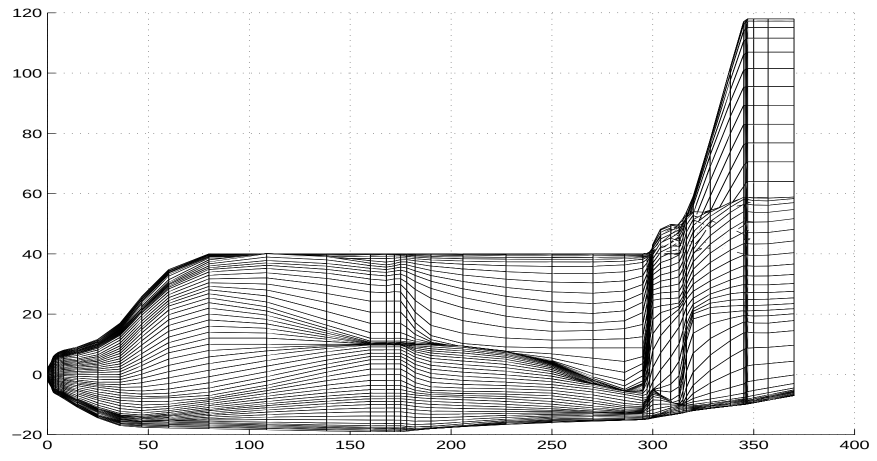

Figure 6.22: The surface with optimal tension parameters. Third projection.

Example 6.5. We tried to reconstruct the surface by the data of the Viking boat. The initial data was given by Dr. Boris I. Kvasov of Suranaree University of Technology. Three-dimensional views of the initial data is shown in Figure 6.23. Figure 6.24 shows the piecewise linear interpolation of the initial data which is not smooth. Figure 6.25 was obtained by setting all tension parameters to zero. The surface in Figure 6.25 does not preserve the shape of the data. The new surface which preserves the shape of the data was obtained by adjusting the tension parameters and shown in Figure 6.26.

Table 6.5 shows the numerical results obtained by SOR iterative method with optimal value of the relaxation parameter ω and by the method of fractional steps with $\varepsilon = 0.0005$. The rate of convergence and the other characteristics of both methods are again very similar.

Table 6.5: The numerical results in Example 6.5.

Method	Tension Parameters	Iteration Count	
		$\overline{n} = 5$	$\overline{n} = 10$
SOR	0	38	349
Splitting	0	14	128
SOR	Optimal	38	349
Splitting	Optimal	14	129

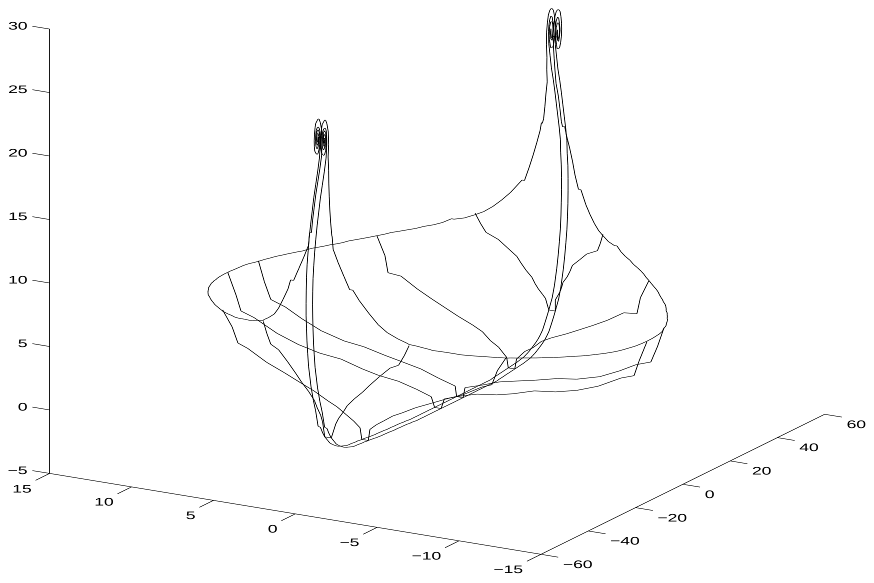

Figure 6.23: The initial data.

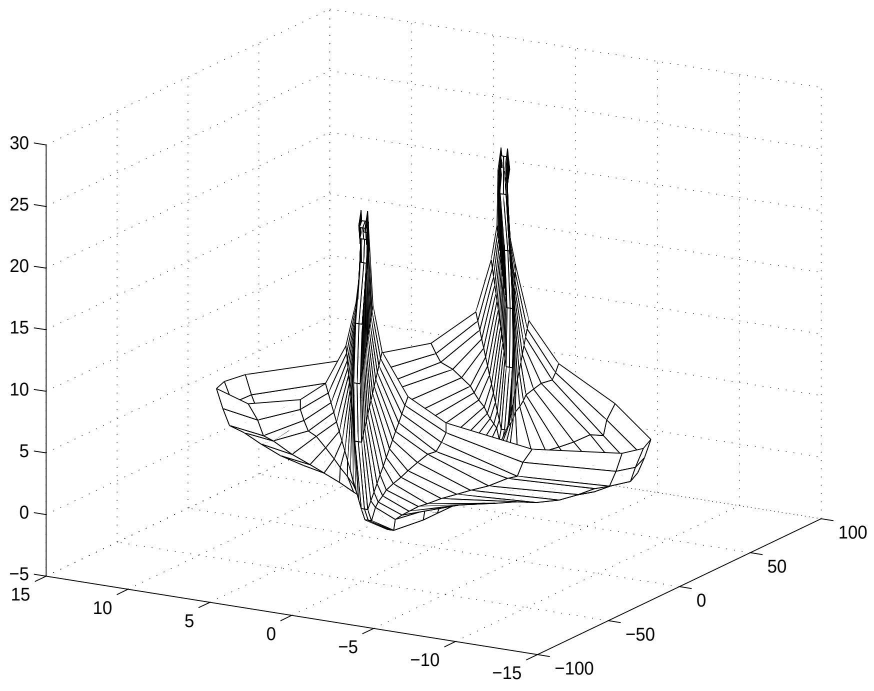

Figure 6.24: The surface with very large tension parameters.

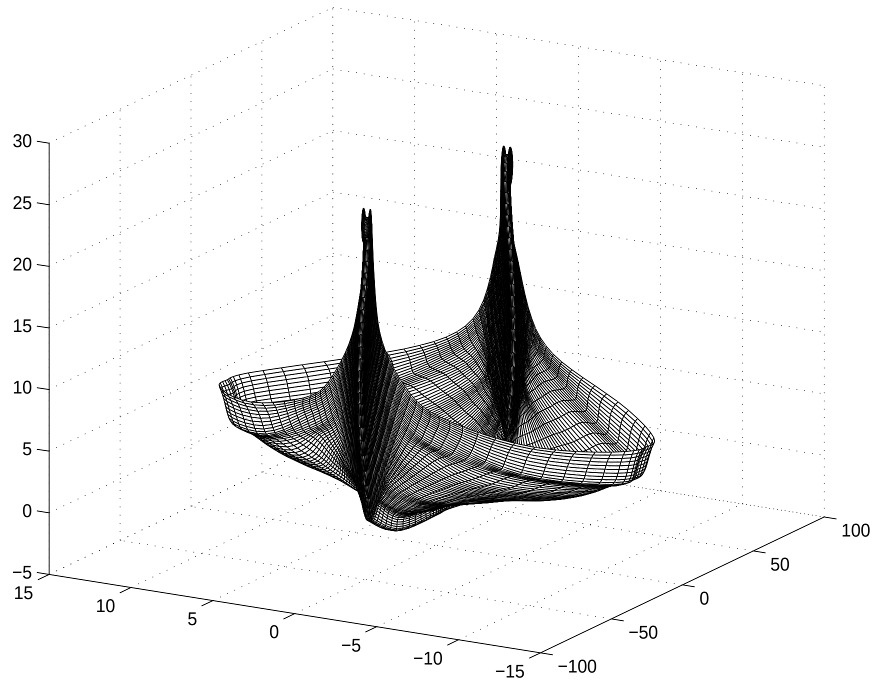

Figure 6.25: The surface with zero tension parameters.

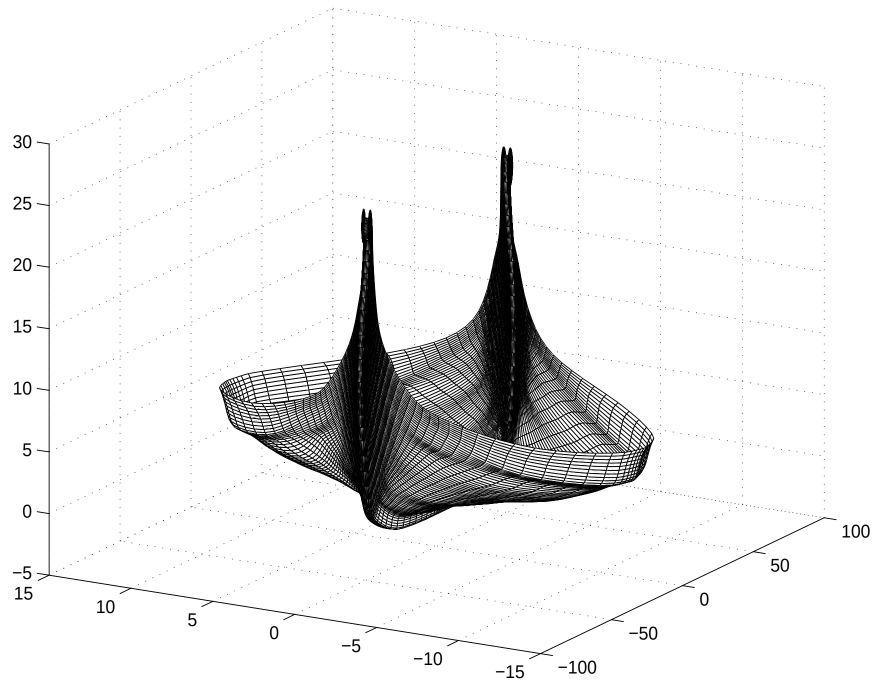

Figure 6.26: The surface with optimal tension parameters.

Example 6.6. We tried to reconstruct the surface of test function with the scattered data,

$$\begin{aligned} f(x,y) &= \frac{3}{4}e^{-\frac{1}{4}[(9x-2)^2+(9y-2)^2]} + \frac{3}{4}e^{-[\frac{1}{49}(9x+1)^2+\frac{1}{10}(9y+1)]} \\ &\quad -\frac{1}{5}e^{-[(9x-4)^2+(9y-7)^2]} + \frac{1}{2}e^{-\frac{1}{4}[(9x-7)^2+(9y-3)^2]}. \end{aligned}$$

This function is well known and usually used for testing each algorithm. Three-dimensional views of the test function was shown in Figure 6.27. Figures 6.28 and 6.29 were the initial grid and data points. Figure 6.30 was obtained by setting all tension parameters to zero. That is considering an approximation of the usual thin plate splines interpolating the data. Figure 6.31 was obtained by setting optimal values of all tension parameters. These two figures are quite similar.

Table 6.6 shows the numerical results obtained by SOR iterative method with optimal value of the relaxation parameter ω and by the method of fractional steps with $\varepsilon = 0.0005$. The rate of convergence and the other characteristics of both methods are again very similar.

Table 6.6: The numerical results of Example 6.6.

Method	Tension Parameters	Iteration Count	
		$\overline{n} = 5$	$\overline{n} = 10$
SOR	0	20	81
Splitting	0	7	37
SOR	Optimal	20	81
Splitting	Optimal	7	38

.

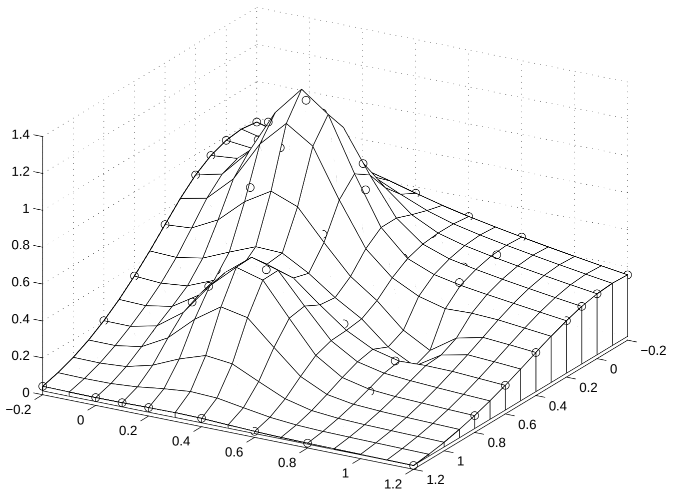

Figure 6.27: 3-D view of the test function.

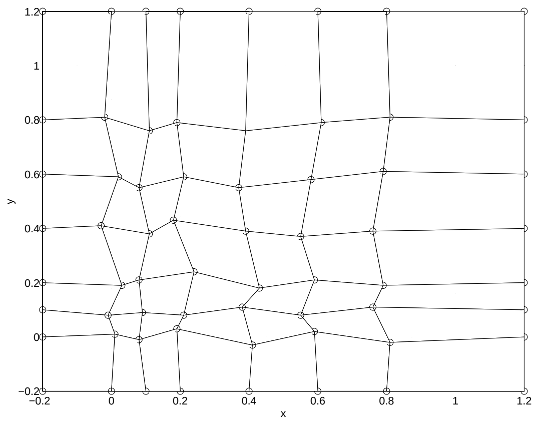

Figure 6.28: The data points.

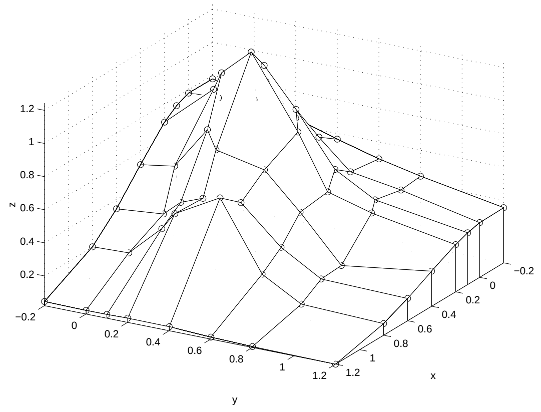

Figure 6.29: The initial data.

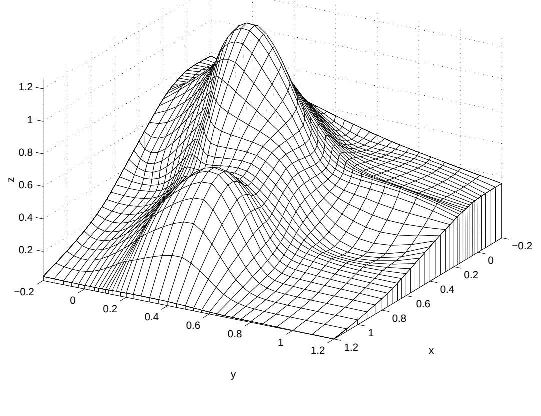

Figure 6.30: The surface with zero tension parameters.

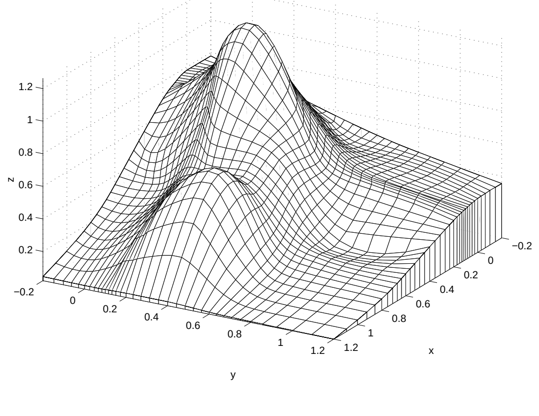

Figure 6.31: The surface with optimal tension parameters.

Chapter VII

Conclusions

This thesis aims to develop new efficient algorithms for solving the problem of shape preserving spline interpolation. Based on formulation of this problem as 1-D and 2-D DMBVP we developed finite-difference methods for approximation and representation of curves and surfaces that arise when these objects have to be processed by a computer. The obtained algorithms have substantial computational advantages, and can work both on single- and multi-processing computers.

The main results of the thesis can be summarized as follows:

1. The problem of shape preserving spline interpolation is formulated as 1-D and 2-D DMBVP for hyperbolic and thin plate tension splines.

2. Difference approximations for the numerical treatment of both 1-D and 2-D DMBVP are constructed.

3. The difference equations are transformed into a matrix form with positive definite matrix of a special structure.

4. Condition number of the matrix is estimated which is independent of the number of data points but substantially depends of the refinement.

5. The existence of the mesh solution is proved by constructing its extension as a discrete hyperbolic spline.

6. An upper bound for the distance between a discrete hyperbolic spline and the corresponding continuous one is established.

7. Explicit formulae and recurrence relations are obtained for discrete GB-splines. Properties of discrete GB-splines and their series are studied. It is shown that the series of discrete GB-splines is a variation diminishing function and the systems of discrete GB-splines are weak Chebyshev systems.

8. Direct algorithms for solving the system of difference equations are developed.

9. SOR iterative method for the numerical solution of 2-D finite-difference system is considered.

10. Finite-difference scheme in fractional steps for the numerical solution of 2-D DMBVP is suggested and investigated. It is shown that this scheme has properties of approximation and absolute stability.
11. Computer codes in FORTRAN 90 and MATLAB are developed to test our algorithms of shape preserving spline interpolation.
12. The features and advantages of the new approach in constructing shape preserving hyperbolic and thin plate tension splines are illustrated by some (famous) examples.

The results of the thesis can be used in many applied problems and first of all in CAGD (design of curves and surfaces in the construction of different product such as car bodies, ship hulls, airplane fuselages, etc.). Other applications include the description of geological, physical and medical phenomena, image analysis, high resolution TV system, cartography, etc.

References

References

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

Bartels, R. H., Beatty, J. C., and Barsky, B. A. (1987). **An Introduction to Splines for Use in Computer Graphics and Geometric Modeling**, Morgan Kaufmann Publishers, Los Altos, CA.

De Boor, C. (1976). Splines as linear combinations of B-splines: A survey, in: **Approximation Theory II**, eds. Lorentz, G.G., Chui, C.K., and Schumaker, L.L., Academic Press, New York, pp. 1–47.

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

De Boor, C. and Pinkus A. (1977). Backward error analysis for totally positive linear systems, **Numer. Math. 27**, pp. 485–490.

Clement, J. C. (1990). Convexity-preserving piecewise rational cubic interpolation, **SIAM J. Numer. Anal. 22**, pp. 386–400.

Cohen, E., Lyche, T., and Riesenfeld, R. (1980). Discrete B-splines and subdivision techniques in computer aided geometric design and computer graphics, **Computer Graphics and Image Processing 14**, pp. 87–111.

Collatz, L. (1964). **Funktional Analysis und Numerische Mathematik**, Springer Verlag, Berlin.

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

Costantini, P., Kvasov, B. I., and Manni, C. (1998). Difference Method for Constructing Hyperbolic Tension Splines, **Rapporto Interno 341/1998**, Università di Siena.

Costantini, P., Kvasov, B. I., and Manni, C. (1999). On discrete hyperbolic tension splines, **Advances in Comp. Math. 11**, pp. 331–354.

Dahmen, W. and Micchelli, C. A. (1989). On multivariate E-splines, **Advances in Mathematics 76**, pp. 33–93.

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

DeVore, R. A. and Lorentz, G. G. (1993). **Constructive Approximation**, Springer Verlag, Berlin.

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

Eisenstat, S. E., Jackson, K. R., and Lewis, J. W. (1985). The order of monotone piecewise cubic interpolation. **SIAM J. Numer. Anal. 22**, pp. 1220–1237.

Farin, G. (1990). **Curves and Surfaces for Computer Aided Geometric Design. A Practical Guide**, 2nd ed., Academic Press, San Diego.

Fiorot, J.C. and Jeannin, P. (1992). **Rational Curves and Surfaces. Application to CAD**, Wiley, Chichester, UK.

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

Forsythe, G.E., Malcolm, M.A., and Moler, C.B. (1977). **Computer Methods for Mathematical Computations**, Prentice Hall, Englewood Cliffs.

Fritsch, F. N. and Butland, J. (1984). A method for constrained local monotone piecewise cubic interpolation, **SIAM J. Sci. Statist. Comput. 5** , pp. 300–304.

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

Golub, G. H. and Van Loan, C. F. (1996). **Matrix Computations**, John Hopkins University Press, Baltimore.

Gregory, J. A. (1984). Shape preserving rational spline interpolation, **Lectures Notes Math. 1105**, pp. 431–441.

Gregory, J. A. and Delbourgo, R. (1982). Piecewise rational quadratic interpolation to monotonic data, **IMAJ. Numer. Anal. 2** , pp. 123–130.

Hegland, M. (1991). On the parallel solution of tridiagonal systems by wrap-around partitioning and incomplete LU factorization, **Numerische Mathematik 59(5)**, pp. 453-472.

Hegland, M., Roberts, S., and Atlas, I. (1997). Finite element thin plate splines for surface fitting, in: **Computational Techniques and applications: CTAC97**, Noye, B.J., Teubner, M.D., and Gill, A.W., eds., World Scientific Publishing Co. Pte., Singapore, pp. 289–296.

Hegland, M., Roberts, S., and Atlas, I. (1998). Finite element thin plate splines for data mining applications, in: **Mathematical Methods for Curves and Surfaces**, Dæhlen, M., Lyche, T. and Schumaker, L.L., eds., Vanderbilt University Press, Nashville, TN, pp. 245–253.

Hoschek, J. and Lasser, D. (1993). **Fundamentals of Computer Aided Geometric Design**, A K Peters Ltd., Wellesley, Massachusetts.

Janenko, N. N. and Kvasov, B. I. (1970). An iterative method for construction of polycubic spline functions, **Soviet Math. Dok. 11**, pp. 1643–1645

Karlin, S. (1968). **Total Positivity, Vol. 1**, Stanford University Press, Stanford, CA.

Kincaid, D. and Cheney, W. (1991). **Numerical Analysis**, Brooks/Cole Publishing Company, Pacific Grove, California.

Koch, P. E. and Lyche, T. (1989). Exponential B-splines in tension, in: **Approximation Theory VI: Proceedings of the Sixth International Symposium on Approximation Theory, Vol. II**, Chui C. K., Schumaker L. L., and Ward J. D., eds., Academic Press, Boston, pp. 361–364.

Koch, P. E. and Lyche, T. (1991). Construction of exponential tension B-splines of arbitrary order, in: **Curves and Surfaces**, eds. Laurent, P. J., Le Méhaute, A. and Schmumaker, L. L., Academic Press, New york, pp. 255–258.

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.

Kvasov, B. I. (1995). Local bases for generalized cubic splines, **Russ. J. Numer. Anal. Math. Modelling 10(1)**, pp. 49–80.

Kvasov, B. I. (1996a). Shape Preserving Spline Approximation via Local Algorithms, in: **Advanced Topics in Multivariate Approximation**, Fontanella, F., Jetter, K., and Laurent, P.J., eds., World Scientific Publishing Co. Pte., Singapore, pp. 181–196.

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

Kvasov, B. I. (1997). On tension spline construction by difference method, in: **Proceedings of the International Conference on Computational Mathematics**, Chulalongkorn University, Bangkok, ISBN 974-637-467-2, pp. 204–211.

Kvasov, B. I. (2000). **Methods of Shape-Preserving Spline Approximation**, World Scientific Publishing Co. Pte., Singapore.

Kvasov, B. I. and Luadsong, A. (2000). Approximation by Lagrange splines, **Proceedings of the Fourth Annual National Symposium on Computational Science and Engineering (ANSCSE'2000)**, Bangkok, pp. 306–315.

Kvasov, B. I. and Luadsong, A. (2001). Difference method for constructing shape-preserving splines, **Proceedings of the Fifth Annual National Symposium on Computational Science and Engineering (ANSCSE'2001)**, Bangkok, pp. 390–397.

Laurent, P. J. (1972). **Approximation et Optimization**, Hermann, Paris.

Lyche, T. (1976). Discrete cubic spline interpolation, **BIT 16**, pp. 281–290.

Malcolm, M. A. (1977). On the computation of nonlinear spline functions, **SIAM J. Numer. Anal. 14**, pp. 254–282.

Mangasarian, O. L. and Schumaker, L. L. (1971). Discrete splines via mathematical programming, **SIAM J. Control 9**, pp. 174–183.

Mangasarian, O. L. and Schumaker, L. L. (1973). Best summation formulae and discrete spline, **SIAM J. Numerical Analysis 10**, pp. 448–459.

Marušić, M. and Rogina, M. (1995). Sharp error bounds for interpolating splines in tension, **J. of Comp. Appl. Math. 61**, pp. 205–223.

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

McCartin, B.J. (1990). Computation of exponential splines, **SIAM J. Sci. Statist. Comput. 11**, pp. 242–262.

Melkman, A. A. (1996). Another proof of the total positivity of the discrete spline collocation matrix, **J. Approx. Theory 84**, pp. 265–273.

Miroshnichenko, V. L. (1997). Optimization of form for rational spline, in: **Computational Systems, No. 159, Spline Functions and Their Applications**, Novosibirsk, pp. 97–109 (in Russian).

Mørken, K. M. (1996). On total positivity of the discrete spline collocation matrix, **J. Approx. Theory 84**, pp. 247–264.

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

Nielson, G. M. (1984). A locally controllable spline with tension for interactive curve design, **Comput. Aided Geom. Design, 1**, pp. 199–205.

Nielson, G. M. and Frank, R. (1984). A method for construction of surfaces under tension, **Rocky Mt. J. Math. 14**, pp. 203–221.

Pruess, S. (1976). Properties of splines in tension, **J. Approx. Th. 17**, pp. 86–96.

Pruess, S. (1979). Alternatives to the exponential spline in tension, **Math. Comput. 33**, pp. 1273–1281.

Pruess, S. (1993). Shape preserving C^2 cubic spline interpolation, **IMA Journal of Numerical Analysis 13**, pp. 493–507.

Rana, S. S. and Dubey, Y. P. (1996). Local behaviour of the deficient discrete cubic spline interpolator, **J. Approx. Theory 86**, pp. 120–127.

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

Rentrop, P. (1980). An algorithm for the computation of exponential splines, **Numer. Math. 35**, pp. 81–93.

Ron, A. (1988). Exponential Box splines, **Constructive Approximation 4**, pp. 357–378.

Saad, Y. (1996). **Iterative Methods for Sparse Linear Systems**, PWS Publishing Company, Boston.

Salkauskas, K. (1984). C^l splines for interpolation of rapidly varying data, **Rocky Mt. J. Math. 14**, pp. 239–250.

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

Sapidis, N. S. and Kaklis, P. D. (1988). An algorithm for constructing convexity and monotonicity-preserving splines in tension, **Computer Aided Geometric Design 5**, pp. 127–137.

Sapidis, N. S., Kaklis, P. D., and Loukakis, T. A. (1988). A Method for computing the tension parameters in convexity-preserving spline-in-tension interpolation, **Numer. Math. 54**, pp. 179–192.

Schaback, R. (1992). Rational curve interpolation, in: **Mathematical Methods in Computer Aided Geometric Design II**, Lyche, T. and Schumaker, L. L., eds., Academic Press, New York, pp. 517–535.

Schmidt, W. J. and Hess, W. (1973). Quadratic and related exponential splines in shape preserving interpolation, **J. Comput. Appl. Math. 18**, pp. 321–330.

Schumaker, L. L. (1973). Constructive aspects of discrete polynomial spline functions, in: **Approximation Theory**, Lorentz, G.G., ed., Academic Press, New York, pp. 469–476.

Schumaker, L. L. (1981). **Spline Functions: Basic Theory**, Academic Press, New York.

Schweikert, D.G. (1966). An interpolating curve using a spline in tension, **J. Math. Phys. 45**, pp. 312–317.

Shashkov, M. (1996). **Conservative Finite-Difference Methods on General Grids**, CRC Press, Boca Raton.

Soanes, R. W. Jr. (1976). VP-splines, an extension of twice differentiable interpolation, in: **Proc. Army Numer. Aal. Comp. Conf.** (Rus. Triangle Park), pp. 141–152.

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.

Yanenko, N. N. (1971). **The Method of Fractional Steps**, Springer Verlag, New York.

Zav'yalov, Yu. S., Kvasov, B. I., and Miroshnichenko, V. L. (1980). **Methods of Spline Functions**, Nauka, Moscow.

Appendix

Appendix A

A Description of the Right Side in the Matrix Equation (6.27)

For writing a computer code it is important to have a detailed description of the right side in the matrix formulation (6.27) of the finite-difference system (6.8)–(6.21),

$$Au = b$$

where

$$b = \{b_{ik;jl} \,|\, k = 1, \ldots, n_i - 1;\; i = 0, \ldots, N;\; l = 1, \ldots, m_j - 1;\; j = 0, \ldots, M\}.$$

Given below expressions for components $b_{ik;jl}$ were obtained by using equation (6.24) and equation for 1-D DMBVP.

for $k = 3, \ldots, n_i - 3;\; i = 0, \ldots, N;\; l = 3, \ldots, m_j - 3;\; j = 0, \ldots, M$,

$$b_{ik;jl} = 0,$$

for $k = 3, \ldots, n_i - 3;\; i = 0, \ldots, N;\; l = 2;\; j = 0, \ldots, M$,

$$b_{ik;jl} = -u_{ik;j,l-2},$$

for $k = 3, \ldots, n_i - 3;\; i = 0, \ldots, N;\; l = m_j - 2;\; j = 0, \ldots, M$,

$$b_{ik;jl} = -u_{ik;j,l+2},$$

for $k = 3, \ldots, n_i - 3;\; i = 0, \ldots, N;\; l = 1;\; j = 1, \ldots, M$,

$$b_{ik;jl} = \gamma_{ij} u_{ik;j,l-1} - 2(u_{i,k-1;j,l-1} + u_{i,k+1;j,l-1}),$$

for $k = 3, \ldots, n_i - 3;\; i = 0, \ldots, N;\; l = 1;\; j = 0$,

$$b_{ik;jl} = \gamma_{ij} u_{ik;j,l-1} - h^2 f^{(0,2)}_{ik;j,l-1} - 2(u_{i,k;j,l-1} + u_{i,k-1;j,l-1} + u_{i,k+1;j,l-1}),$$

for $k = 3, \ldots, n_i - 3;\; i = 0, \ldots, N;\; l = m_j - 1;\; j = 0, \ldots, M - 1$,

$$b_{ik;jl} = \gamma_{ij} u_{ik;j,l+1} - 2(u_{i,k-1;j,l+1} + u_{i,k+1;j,l+1}),$$

for $k = 3, \ldots, n_i - 3;\; i = 0, \ldots, N;\; l = m_j - 1;\; j = M$,

$$b_{ik;jl} = \gamma_{ij} u_{ik;j,l+1} - h^2 f^{(0,2)}_{ik;j,l+1} - 2(u_{i,k;j,l+1} + u_{i,k-1;j,l+1} + u_{i,k+1;j,l+1}),$$

for $k = 2;\ i = 0, \ldots, N;\ l = 3, \ldots, m_j - 3;\ j = 0, \ldots, M,$

$$b_{ik;jl} = -u_{ik-2;jl},$$

for $k = n_i - 2;\ i = 0, \ldots, N;\ l = 3, \ldots, m_j - 3;\ j = 0, \ldots, M,$

$$b_{ik;jl} = -u_{i,k+2;jl},$$

for $k = 1;\ i = 1, \ldots, N;\ l = 3, \ldots, m_j - 3;\ j = 0, \ldots, M,$

$$b_{ik;jl} = \beta_{ij} u_{i,k-1;jl} - 2(u_{i,k-1;j,l-1} + u_{i,k-1;j,l+1}),$$

for $k = 1;\ i = 0;\ l = 3, \ldots, m_j - 3;\ j = 0, \ldots, M,$

$$b_{ik;jl} = \beta_{ij} u_{i,k-1;jl} - h^2 f^{(2,0)}_{i,k-1;jl} - 2(u_{i,k-1;jl} + u_{i,k-1;j,l-1} + u_{i,k-1;j,l+1}),$$

for $k = n_i - 1;\ i = 0, \ldots, N - 1;\ l = 3, \ldots, m_j - 3;\ j = 0, \ldots, M,$

$$b_{ik;jl} = \beta_{ij} u_{i,k+1;jl} - 2(u_{i,k+1;j,l-1} + u_{i,k+1;j,l+1}),$$

for $k = n_i - 1;\ i = N;\ l = 3, \ldots, m_j - 3;\ j = 0, \ldots, M,$

$$b_{ik;jl} = \beta_{ij} u_{i,k+1;jl} - h^2 f^{(2,0)}_{i,k+1;jl} - 2(u_{i,k+1;jl} + u_{i,k+1;j,l-1} + u_{i,k+1;j,l+1}),$$

for $k = 2;\ i = 0, \ldots, N;\ l = 2;\ j = 0, \ldots, M,$

$$b_{ik;jl} = -u_{i,k-2;jl} - u_{ik;j,l-2},$$

for $k = 2;\ i = 0, \ldots, N;\ l = m_j - 2;\ j = 0, \ldots, M,$

$$b_{ik;jl} = -u_{i,k-2;jl} - u_{ik;j,l+2},$$

for $k = n_i - 2;\ i = 0, \ldots, N;\ l = 2;\ j = 0, \ldots, M,$

$$b_{ik;jl} = -u_{i,k+2;jl} - u_{ik;j,l-2},$$

for $k = n_i - 2;\ i = 0, \ldots, N;\ l = m_j - 2;\ j = 0, \ldots, M,$

$$b_{ik;jl} = -u_{i,k+2;jl} - u_{ik;j,l+2},$$

for $k = 1;\ i = 0;\ l = 2;\ j = 0, \ldots, M,$

$$b_{ik;jl} = \beta_{ij} u_{i,k-1;jl} - h^2 f^{(2,0)}_{i,k-1;jl} - 2(u_{i,k-1;j,l-1} + u_{i,k-1;jl} + u_{i,k-1;j,l+1}),$$

for $k = 1;\ i = 1, \ldots, N;\ l = 2;\ j = 0, \ldots, M,$

$$b_{ik;jl} = \beta_{ij} u_{i,k-1;jl} - u_{ik;j,l-2} - 2(u_{i,k-1;j,l-1} + u_{i,k-1;j,l+1}),$$

for $k = n_i - 1;\ i = 0, \ldots, N - 1;\ l = 2;\ j = 0, \ldots, M,$

$$b_{ik;jl} = \beta_{ij} u_{i,k+1;jl} - u_{ik;j,l-2} - 2(u_{i,k+1;j,l-1} + u_{i,k+1;j,l+1}),$$

for $k = n_i - 1;\ i = N;\ l = 2;\ j = 0, \ldots, M,$

$$b_{ik;jl} = \beta_{ij} u_{i,k+1;jl} - h^2 f^{(2,0)}_{i,k+1;jl} - 2(u_{i,k+1;j,l-1} + u_{i,k+1;jl} + u_{i,k+1;j,l+1}),$$

for $k = 2;\ i = 0, \ldots, N;\ l = 1;\ j = 0,$

$$b_{ik;jl} = \gamma_{ij} u_{ik;j,l-1} - u_{i,k-2;jl} - h^2 f^{(0,2)}_{ik;j,l-1} - 2(u_{i,k-1;j,l-1} + u_{ik;j,l-1} + u_{i,k+1;j,l-1}),$$

for $k = 2;\ i = 0, \ldots, N;\ l = 1;\ j = 1, \ldots, M,$
$$b_{ik;jl} = \gamma_{ij} u_{ik;j,l-1} - u_{i,k-2;jl} - 2(u_{i,k-1;j,l-1} + u_{i,k+1;j,l-1}),$$

for $k = 2;\ i = 0, \ldots, N;\ l = m_j - 1;\ j = 0, \ldots, M-1,$
$$b_{ik;jl} = \gamma_{ij} u_{ik;j,l-1} - u_{i,k+2;jl} - 2(u_{i,k-1;j,l-1} + u_{i,k+1;j,l-1}),$$

for $k = 2;\ i = 0, \ldots, N;\ l = m_j - 1;\ j = M,$
$$b_{ik;jl} = \gamma_{ij} u_{ik;j,l-1} - u_{i,k+2;jl} - h^2 f^{(0,2)}_{ik;j,l-1} - 2(u_{i,k-1;j,l-1} + u_{ik;j,l-1} + u_{i,k+1;j,l-1}),$$

for $k = 1;\ i = 1, \ldots, N;\ l = 1;\ j = 1, \ldots, M,$
$$b_{ik;jl} = \beta_{ij} u_{i,k-1;jl} + \gamma_{ij} u_{ik;j,l-1} - 2(u_{i,k-1;j,l+1} + u_{i,k+1;j,l-1} + u_{i,k-1;j,l-1}),$$

for $k = 1;\ i = 1, \ldots, N;\ l = m_j - 1;\ j = 0, \ldots, M-1,$
$$b_{ik;jl} = \beta_{ij} u_{i,k-1;jl} + \gamma_{ij} u_{ik;j,l+1} - 2(u_{i,k-1;j,l+1} + u_{i,k+1;j,l+1} + u_{i,k-1;j,l-1}),$$

for $k = 1;\ i = 1, \ldots, N;\ l = m_j - 1;\ j = M,$
$$b_{ik;jl} = \beta_{ij} u_{i,k-1;jl} - 2(u_{i,k-1;j,l+1} + u_{i,k+1;j,l+1} + u_{i,k-1;j,l-1} + u_{ik;j,l+1}) + \gamma_{ij} u_{ik;j,l+1} - h^2 f^{(0,2)}_{ik;j,l+1},$$

for $k = 1;\ i = 0;\ l = 1;\ j = 1, \ldots, M,$
$$b_{ik;jl} = \beta_{ij} u_{i,k-1;jl} - 2(u_{i,k-1;j,l+1} + u_{i,k+1;j,l-1} + u_{i,k-1;j,l-1} + u_{i,k-1;jl}) + \gamma_{ij} u_{ik;j,l-1} - h^2 f^{(2,0)}_{i,k-1;jl},$$

for $k = 1;\ i = 0;\ l = m_j - 1;\ j = 0, \ldots, M-1,$
$$b_{ik;jl} = \beta_{ij} u_{i,k-1;jl} - 2(u_{i,k-1;j,l+1} + u_{i,k+1;j,l+1} + u_{i,k-1;j,l-1} + u_{i,k-1;jl}) + \gamma_{ij} u_{ik;j,l+1} - h^2 f^{(2,0)}_{i,k-1;jl},$$

for $k = 1;\ i = 0;\ l = m_j - 1;\ j = M,$
$$b_{ik;jl} = \beta_{ij} u_{i,k-1;jl} - 2(u_{i,k-1;j,l+1} + u_{i,k+1;j,l+1} + u_{i,k-1;j,l-1} + u_{ik;j,l+1} + u_{i,k-1;jl}) + \gamma_{ij} u_{ik;j,l+1} - h^2 (f^{(0,2)}_{ik;j,l+1} + f^{(2,0)}_{i,k-1;jl}),$$

for $k = n_i - 1;\ i = 0, \ldots, N-1;\ l = 1;\ j = 1, \ldots, M,$
$$b_{ik;jl} = \beta_{ij} u_{i,k+1;jl} + \gamma_{ij} u_{ik;j,l-1} - 2(u_{i,k+1;j,l+1} + u_{i,k+1;j,l-1} + u_{i,k-1;j,l-1}),$$

for $k = n_i - 1;\ i = 0, \ldots, N-1;\ l = m_j - 1;\ j = 0, \ldots, M-1,$
$$b_{ik;jl} = \beta_{ij} u_{i,k+1;jl} + \gamma_{ij} u_{ik;j,l+1} - 2(u_{i,k+1;j,l+1} + u_{i,k+1;j,l-1} + u_{i,k-1;j,l+1}),$$

for $k = n_i - 1;\ i = 0, \ldots, N-1;\ l = m_j - 1;\ j = M,$
$$b_{ik;jl} = \beta_{ij} u_{i,k+1;jl} - 2(u_{i,k-1;j,l+1} + u_{i,k+1;j,l-1} + u_{i,k+1;j,l+1} + u_{ik;j,l+1}) + \gamma_{ij} u_{ik;j,l+1} - h^2 f^{(0,2)}_{ik;j,l+1},$$

for $k = n_i - 1;\ i = N;\ l = 1;\ j = 1, \ldots, M,$
$$b_{ik;jl} = \beta_{ij} u_{i,k+1;jl} - 2(u_{i,k-1;j,l-1} + u_{i,k+1;j,l+1} + u_{i,k+1;j,l-1} + u_{i,k+1;jl}) + \gamma_{ij} u_{ik;j,l-1} - h^2 f^{(2,0)}_{i,k+1;jl},$$

for $k = n_i - 1;\ i = N;\ l = m_j - 1;\ j = 0, \ldots, M - 1,$

$$\begin{aligned} b_{ik;jl} = {} & \beta_{ij} u_{i,k+1;jl} - 2(u_{i,k-1;j,l+1} + u_{i,k+1;j,l-1} + u_{i,k+1;j,l+1} + u_{i,k+1;jl}) \\ & + \gamma_{ij} u_{ik;j,l+1} - h^2 f^{(2,0)}_{i,k+1;jl}, \end{aligned}$$

for $k = n_i - 1;\ i = N;\ l = m_j - 1;\ j = M,$

$$\begin{aligned} b_{ik;jl} = {} & \beta_{ij} u_{i,k+1;jl} - 2(u_{i,k-1;j,l+1} + u_{i,k+1;j,l-1} + u_{i,k-1;j,l+1} + u_{ik;j,l+1} + u_{i,k+1;jl}) \\ & + \gamma_{ij} u_{ik;j,l+1} - h^2 (f^{(0,2)}_{ik;j,l+1} + f^{(2,0)}_{i,k+1;jl}), \end{aligned}$$

$$\beta_{ij} = 8 + \left(\frac{p_{ij}}{n_i}\right)^2; \qquad \gamma_{ij} = 8 + \left(\frac{q_{ij}}{m_j}\right)^2, \qquad\qquad i = 0, \ldots, N;\ j = 0, \ldots, M.$$

Appendix B

Fortran 90 Programming Code for Solving 1-D DMBVP

```
PROGRAM DMBVP1D ! ! BY MR.ANIRUT LUADSONG ! OCT 2001 ! SOLVING 1-D
DMBVP !
  IMPLICIT NONE
  INTEGER :: i,j,ii,BN,M,MM
  REAL :: TAU,F2P_0,F2P_END,DET_A,TMP1
  INTEGER,DIMENSION (:), ALLOCATABLE :: N_I
  REAL,DIMENSION (:), ALLOCATABLE :: X,F,H,P
  REAL,DIMENSION (:), ALLOCATABLE :: U,B,BA1,BA2,BA3
  REAL,DIMENSION (:), ALLOCATABLE :: A_I,B_I,OMEGA_I

  OPEN (21,  FILE = 'B.TXT')
  OPEN (22,  FILE = 'BA1.TXT')
  OPEN (23,  FILE = 'BA2.TXT')
  OPEN (24,  FILE = 'BA3.TXT')
  OPEN (25,  FILE = 'U.TXT')
  OPEN (26,  FILE = 'INPUT.TXT')
  OPEN (27,  FILE = 'X.TXT')

  READ(26,*) BN,TAU,F2P_0,F2P_END

  ALLOCATE (X(BN+2),F(BN+2),H(BN+1),P(BN+1),N_I(BN+1))
  ALLOCATE (A_I(BN+1),B_I(BN+1),OMEGA_I(BN+1))

  DO i=1,BN+1
    READ(26,*) X(i),F(i),P(i)
  END DO
  READ(26,*) X(BN+2),F(BN+2)

  DO i=1,BN+1
    H(i) = X(i+1)-X(i)
  END DO
  MM = 0
  DO i=1,BN+1
    N_I(i)  = INT(H(i)/TAU+0.5)
    MM      = MM+N_I(i)
  END DO
  M = MM-BN-1
  ALLOCATE (U(M),B(M),BA1(M),BA2(M-1),BA3(M-2))

  DO i=1,BN+1
    OMEGA_I(i) = (P(i)/N_I(i))*(P(i)/N_I(i))
    A_I(i)     = -OMEGA_I(i)-4
    B_I(i)     = 6+2*OMEGA_I(i)
  END DO

  ii    = 0

  DO i=1,BN+1

    ii = ii+1

    IF (i.EQ.1) THEN
      B(ii)   = -TAU*TAU*F2P_0-(2+A_I(i))*F(i)
      BA1(ii) = B_I(i)-1
    ELSE
      B(ii)     = -A_I(i)*F(i)
      BA1(ii)   = B_I(i)
      BA2(ii-1) = 1.0
      BA3(ii-2) = 0.0
    END IF

    ii = ii+1

    B(ii)     = -F(i)
    BA1(ii)   = B_I(i)
    BA2(ii-1) = A_I(i)
    IF (i.GT.1) THEN
```

```
      BA3(ii-2) = 0.0
    END IF
    DO j=3,N_I(i)-3

      ii = ii+1

      B(ii)     = 0.0
      BA1(ii)   = B_I(i)
      BA2(ii-1) = A_I(i)
      BA3(ii-2) = 1.0
    END DO

    ii = ii+1

    B(ii)     = -F(i+1)
    BA1(ii)   = B_I(i)
    BA2(ii-1) = A_I(i)
    BA3(ii-2) = 1.0

    ii = ii+1

    IF (i.EQ.BN+1) THEN
      B(ii)   = -TAU*TAU*F2P_END-(2+A_I(i))*F(i+1)
      BA1(ii) = B_I(i)-1
    ELSE
      B(ii)   = -A_I(i)*F(i+1)
      BA1(ii) = B_I(i)
    END IF
    BA2(ii-1) = A_I(i)
    BA3(ii-2) = 1.0
  END DO

  CALL  SUB5DIAG(M,B,BA1,BA2,BA3,U,DET_A)

  ii = 0

  DO i=1,BN+1
    WRITE(25,*) F(i)
    WRITE(27,*) X(i)
    TMP1 = X(i)
    DO j=1,N_I(i)-1
      ii = ii+1
      WRITE(25,*) U(ii)
      TMP1 = TMP1+TAU
      WRITE(27,*) TMP1
    END DO
  END DO
  WRITE(25,*) F(BN+2)
  WRITE(27,*) X(BN+2)

  DO i=1,M
    WRITE(21,*) B(i)
    WRITE(22,*) BA1(i)
  END DO
  DO i=1,M-1
    WRITE(23,*) BA2(i)
  END DO
  DO i=1,M-2
    WRITE(24,*) BA3(i)
  END DO

  CLOSE(21)
  CLOSE(22)
  CLOSE(23)
  CLOSE(24)
  CLOSE(25)
  CLOSE(26)
  CLOSE(27)

  PRINT*,' det(A) = ', DET_A

END PROGRAM DMBVP1D

SUBROUTINE SUB5DIAG(N,B,D,F,E,X,DET_A)
!
! BY MR.ANIRUT LUADSONG
!
! GIVEN: Ax=b WITH A FIVE-DIAGONAL SYMMETRIC STRONGLY NONSINGULAR MATRIX A
! FIND : SOLUTION x.
!
  IMPLICIT NONE
  INTEGER :: N,i
  REAL,DIMENSION (N) :: B,D,X
  REAL,DIMENSION (N-2) :: E
  REAL,DIMENSION (N-1) :: F
  REAL,DIMENSION (:), ALLOCATABLE :: C,Z
  REAL,DIMENSION (:), ALLOCATABLE :: ALPHA,GAMMA,DELTA
  REAL :: DET_A
  ALLOCATE (C(N),Z(N))
  ALLOCATE (ALPHA(N),GAMMA(N-1),DELTA(N-2))
!
! 1st STEP: FACTOR A=LDL^T
!
```

```
  ALPHA(1) = D(1)
  GAMMA(1) = F(1)/ALPHA(1)
  DELTA(1) = E(1)/ALPHA(1)
  ALPHA(2) = D(2)-F(1)*GAMMA(1)
  GAMMA(2) = (F(2)-E(1)*GAMMA(1))/ALPHA(2)
  DELTA(2) = E(2)/ALPHA(2)
  DO i=3,N-2
    ALPHA(i) = D(i)-E(i-2)*DELTA(i-2)-ALPHA(i-1)*GAMMA(i-1)*GAMMA(i-1)
    GAMMA(i) = (F(i)-E(i-1)*GAMMA(i-1))/ALPHA(i)
    DELTA(i) = E(i)/ALPHA(i)
  END DO
  ALPHA(N-1) = D(N-1)-E(N-3)*DELTA(N-3)-ALPHA(N-2)*GAMMA(N-2)*GAMMA(N-2)
  GAMMA(N-1) = (F(N-1)-E(N-2)*GAMMA(N-2))/ALPHA(N-1)
  ALPHA(N)   = D(N)-E(N-2)*DELTA(N-2)-ALPHA(N-1)*GAMMA(N-1)*GAMMA(N-1)
!
! 2nd STEP: UPDATE Lz=b, Dc=z
!
  Z(1) = B(1)
  Z(2) = B(2)-GAMMA(1)*Z(1)
  DO i=3,N
    Z(i) = B(i)-GAMMA(i-1)*Z(i-1)-DELTA(i-2)*Z(i-2)
  END DO
  DO i=1,N
    C(i) = Z(i)/ALPHA(i)
  END DO
!
! 3rd STEP: BACKSUBSTITUTION L^Tx=c
!
  X(N)   = C(N)
  X(N-1) = C(N-1)-GAMMA(N-1)*X(N)
  DO i=N-2,1,-1
    X(i) = C(i)-GAMMA(i)*X(i+1)-DELTA(i)*X(i+2)
  END DO
!
! FIND DET(A)
!
  DET_A = ALPHA(1)
  DO i=2,N
    DET_A = DET_A*ALPHA(i)
    IF (ABS(DET_A) .GT. 1.0E30) THEN
      DET_A=999
      GOTO 999
    END IF
  END DO
999 CONTINUE END SUBROUTINE SUB5DIAG
```

Appendix C

Fortran 90 Programming Code for Solving 2-D DMBVP

```
PROGRAM SP2DPM
!
! BY MR.ANIRUT LUADSONG
! Feb 2002
! SOLVING 2-D DMBVP WITH PARAMETRIC VARIABLES BY SPLITTING METHOD
!                      AND USING MPI LIBRARY FOR PARALLEL COMPUTER
!
  IMPLICIT NONE
  INCLUDE 'mpif.h'

  INTEGER :: my_id,root_process,ierr,num_procs,status(MPI_STATUS_SIZE)

  INTEGER :: i,j,MX,MY,AUTO_TENSION_X,AUTO_TENSION_Y,AUTO_TENSION_F
  INTEGER :: N,NUM_PT_X,NUM_PT_Y
  REAL    :: EPS,LOWER_QX,LOWER_QY,LOWER_QF,TMP1
  REAL,DIMENSION (:), ALLOCATABLE   :: X,Y
  REAL,DIMENSION (:,:), ALLOCATABLE :: F,OUT_U,XXX,YYY,XX,YY
  REAL,DIMENSION (:,:), ALLOCATABLE :: P,Q,PX,PY,QX,QY
  REAL,DIMENSION (:,:), ALLOCATABLE :: P1,Q1,P1X,P1Y,Q1X,Q1Y

  root_process = 0

  CALL MPI_INIT(ierr)
  CALL MPI_COMM_RANK(MPI_COMM_WORLD, my_id, ierr)
  CALL MPI_COMM_SIZE(MPI_COMM_WORLD, num_procs, ierr)

  IF (my_id .EQ. root_process) THEN

    OPEN (21,  FILE = '2D_U.TXT')
    OPEN (22,  FILE = 'SPLIT_IN.TXT')
    OPEN (23,  FILE = '2D_X.TXT')
    OPEN (24,  FILE = '2D_Y.TXT')
    OPEN (25,  FILE = 'SIZE.TXT')
    OPEN (26,  FILE = '2D_PQX.TXT')
    OPEN (27,  FILE = '2D_PQY.TXT')
    OPEN (28,  FILE = '2D_PQF.TXT')
    OPEN (29,  FILE = '2D_PQX.OUT')
    OPEN (30,  FILE = '2D_PQY.OUT')
    OPEN (31,  FILE = '2D_PQF.OUT')
!
! READ DATA FROM THE HEADER OF FILE SPLIT_IN.TXT
!
    READ(22,*) NUM_PT_X,NUM_PT_Y,N,EPS
    READ(22,*) AUTO_TENSION_X,AUTO_TENSION_Y,AUTO_TENSION_F,  &
               LOWER_QX,LOWER_QY,LOWER_QF
    MX  = (N-1)*(NUM_PT_X-1)
    MY  = (N-1)*(NUM_PT_Y-1)
!
! ASSIGNING THE SIZE OF EACH ARRAY
!
    ALLOCATE(X(NUM_PT_X),Y(NUM_PT_Y))
    ALLOCATE(XX(NUM_PT_X,NUM_PT_Y),YY(NUM_PT_X,NUM_PT_Y))
    ALLOCATE(F(NUM_PT_X,NUM_PT_Y))
    ALLOCATE(P(NUM_PT_X-1,NUM_PT_Y-1),Q(NUM_PT_X-1,NUM_PT_Y-1))
    ALLOCATE(PX(NUM_PT_X-1,NUM_PT_Y-1),QX(NUM_PT_X-1,NUM_PT_Y-1))
    ALLOCATE(PY(NUM_PT_X-1,NUM_PT_Y-1),QY(NUM_PT_X-1,NUM_PT_Y-1))
    ALLOCATE(P1(NUM_PT_X-1,NUM_PT_Y),Q1(NUM_PT_X,NUM_PT_Y-1))
    ALLOCATE(P1X(NUM_PT_X-1,NUM_PT_Y),Q1X(NUM_PT_X,NUM_PT_Y-1))
    ALLOCATE(P1Y(NUM_PT_X-1,NUM_PT_Y),Q1Y(NUM_PT_X,NUM_PT_Y-1))
    ALLOCATE(XXX(MX+NUM_PT_X,MY+NUM_PT_Y),YYY(MX+NUM_PT_X,MY+NUM_PT_Y))
    ALLOCATE(OUT_U(MX+NUM_PT_X,MY+NUM_PT_Y))
    DO i=1,NUM_PT_X
      X(i) = i*1.
    END DO
    DO j=1,NUM_PT_Y
      Y(j) = j*1.
    END DO
```

```
!
! READ DATA FROM FILE SPLIT_IN.TXT
!
    DO i=1,NUM_PT_X
      DO j=1,NUM_PT_Y
        READ(22,*) XX(i,j),YY(i,j),F(i,j)
      END DO
    END DO
    IF (N.LT.5) THEN
      PRINT *, 'N MUST GREATER THAN 4, !!! PRESS ANY KEY TO CONTINUE !!!'
      READ(*,*)
      DO i=1,NUM_PT_X
        DO j=1,NUM_PT_Y
          WRITE(21,*) F(i,j)
          WRITE(23,*) XX(i,j)
      WRITE(24,*) YY(i,j)
        END DO
      END DO
      WRITE(25,*) NUM_PT_X
      WRITE(25,*) NUM_PT_Y
      CLOSE(21)
      CLOSE(22)
      CLOSE(23)
      CLOSE(24)
      CLOSE(25)
      CLOSE(26)
      CLOSE(27)
      CLOSE(28)
      CLOSE(29)
      CLOSE(30)
      CLOSE(31)
      GOTO 8888
    END IF
!
! GENERATE TENSION PARAMETERS FOR XX,YY AND F
!
    PX  = 0.
    QX  = 0.
    P1X = 0.
    Q1X = 0.
    PY  = 0.
    QY  = 0.
    P1Y = 0.
    Q1Y = 0.
    P   = 0.
    Q   = 0.
    P1  = 0.
    Q1  = 0.
    IF (AUTO_TENSION_X .EQ. 0) THEN
      DO i=1,NUM_PT_X-1
        DO j=1,NUM_PT_Y-1
          READ(26,*) PX(i,j),QX(i,j)
        END DO
      END DO
      DO i=1,NUM_PT_X-1
        DO j=1,NUM_PT_Y
          READ(26,*) P1X(i,j)
        END DO
      END DO
      DO i=1,NUM_PT_X
        DO j=1,NUM_PT_Y-1
          READ(26,*) Q1X(i,j)
        END DO
      END DO
    END IF
    IF (AUTO_TENSION_Y .EQ. 0) THEN
      DO i=1,NUM_PT_X-1
        DO j=1,NUM_PT_Y-1
          READ(27,*) PY(i,j),QY(i,j)
        END DO
      END DO
      DO i=1,NUM_PT_X-1
        DO j=1,NUM_PT_Y
          READ(27,*) P1Y(i,j)
        END DO
      END DO
      DO i=1,NUM_PT_X
        DO j=1,NUM_PT_Y-1
          READ(27,*) Q1Y(i,j)
        END DO
      END DO
    END IF
    IF (AUTO_TENSION_F .EQ. 0) THEN
      DO i=1,NUM_PT_X-1
        DO j=1,NUM_PT_Y-1
          READ(28,*) P(i,j),Q(i,j)
        END DO
      END DO
      DO i=1,NUM_PT_X-1
        DO j=1,NUM_PT_Y
          READ(28,*) P1(i,j)
        END DO
      END DO
      DO i=1,NUM_PT_X
        DO j=1,NUM_PT_Y-1
          READ(28,*) Q1(i,j)
        END DO
      END DO
    END IF
    IF (AUTO_TENSION_X .EQ. 1) THEN
      CALL TENSION2D(NUM_PT_X,NUM_PT_Y,X,Y,XX,PX,QX,P1X,Q1X,LOWER_QX)
    END IF
    IF (AUTO_TENSION_Y .EQ. 1) THEN
      CALL TENSION2D(NUM_PT_X,NUM_PT_Y,X,Y,YY,PY,QY,P1Y,Q1Y,LOWER_QY)
    END IF
    IF (AUTO_TENSION_F .EQ. 1) THEN
      CALL TENSION2D(NUM_PT_X,NUM_PT_Y,X,Y,F,P,Q,P1,Q1,LOWER_QF)
    END IF
    IF (AUTO_TENSION_X .EQ. 2) THEN
    PX  = 400.
    QX  = 400.
    P1X = 400.
    Q1X = 400.
    END IF
    IF (AUTO_TENSION_Y .EQ. 2) THEN
    PY  = 400.
    QY  = 400.
    P1Y = 400.
    Q1Y = 400.
    END IF
    IF (AUTO_TENSION_F .EQ. 2) THEN
    P  = 400.
    Q  = 400.
    P1 = 400.
    Q1 = 400.
    END IF

  END IF

8888  CONTINUE

  CALL MPI_BCAST(N,1,MPI_INTEGER,root_process,MPI_COMM_WORLD,ierr)
  IF (N .LT. 5) GOTO 9999
  CALL MPI_BCAST(NUM_PT_X,1,MPI_INTEGER,root_process,MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(NUM_PT_Y,1,MPI_INTEGER,root_process,MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(EPS,1,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(LOWER_QX,1,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(LOWER_QY,1,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(LOWER_QF,1,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(MX,1,MPI_INTEGER,root_process,MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(MY,1,MPI_INTEGER,root_process,MPI_COMM_WORLD,ierr)

  IF (my_id .GT. root_process) THEN
    ALLOCATE(X(NUM_PT_X),Y(NUM_PT_Y))
    ALLOCATE(XX(NUM_PT_X,NUM_PT_Y),YY(NUM_PT_X,NUM_PT_Y))
    ALLOCATE(F(NUM_PT_X,NUM_PT_Y))
    ALLOCATE(P(NUM_PT_X-1,NUM_PT_Y-1),Q(NUM_PT_X-1,NUM_PT_Y-1))
    ALLOCATE(PX(NUM_PT_X-1,NUM_PT_Y-1),QX(NUM_PT_X-1,NUM_PT_Y-1))
    ALLOCATE(PY(NUM_PT_X-1,NUM_PT_Y-1),QY(NUM_PT_X-1,NUM_PT_Y-1))
    ALLOCATE(P1(NUM_PT_X-1,NUM_PT_Y),Q1(NUM_PT_X,NUM_PT_Y-1))
    ALLOCATE(P1X(NUM_PT_X-1,NUM_PT_Y),Q1X(NUM_PT_X,NUM_PT_Y-1))
    ALLOCATE(P1Y(NUM_PT_X-1,NUM_PT_Y),Q1Y(NUM_PT_X,NUM_PT_Y-1))
  END IF

  CALL MPI_BCAST(X(:),NUM_PT_X,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(Y(:),NUM_PT_Y,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)

  DO i=1,NUM_PT_X
    CALL MPI_BCAST(XX(i,:),NUM_PT_Y,MPI_REAL,root_process,  &
                   MPI_COMM_WORLD,ierr)
    CALL MPI_BCAST(YY(i,:),NUM_PT_Y,MPI_REAL,root_process,  &
                   MPI_COMM_WORLD,ierr)
    CALL MPI_BCAST(F(i,:),NUM_PT_Y,MPI_REAL,root_process,   &
                   MPI_COMM_WORLD,ierr)
  END DO
  DO i=1,NUM_PT_X-1
    CALL MPI_BCAST(P(i,:),NUM_PT_Y-1,MPI_REAL,root_process,  &
                   MPI_COMM_WORLD,ierr)
    CALL MPI_BCAST(Q(i,:),NUM_PT_Y-1,MPI_REAL,root_process,  &
                   MPI_COMM_WORLD,ierr)
    CALL MPI_BCAST(PX(i,:),NUM_PT_Y-1,MPI_REAL,root_process, &
                   MPI_COMM_WORLD,ierr)
    CALL MPI_BCAST(QX(i,:),NUM_PT_Y-1,MPI_REAL,root_process, &
                   MPI_COMM_WORLD,ierr)
    CALL MPI_BCAST(PY(i,:),NUM_PT_Y-1,MPI_REAL,root_process, &
                   MPI_COMM_WORLD,ierr)
    CALL MPI_BCAST(QY(i,:),NUM_PT_Y-1,MPI_REAL,root_process, &
                   MPI_COMM_WORLD,ierr)
  END DO
  DO i=1,NUM_PT_X-1
    CALL MPI_BCAST(P1(i,:),NUM_PT_Y,MPI_REAL,root_process,  &
                   MPI_COMM_WORLD,ierr)
    CALL MPI_BCAST(P1X(i,:),NUM_PT_Y,MPI_REAL,root_process, &
                   MPI_COMM_WORLD,ierr)
    CALL MPI_BCAST(P1Y(i,:),NUM_PT_Y,MPI_REAL,root_process, &
                   MPI_COMM_WORLD,ierr)
  END DO
  DO i=1,NUM_PT_X
    CALL MPI_BCAST(Q1(i,:),NUM_PT_Y-1,MPI_REAL,root_process,  &
                   MPI_COMM_WORLD,ierr)
    CALL MPI_BCAST(Q1X(i,:),NUM_PT_Y-1,MPI_REAL,root_process, &
                   MPI_COMM_WORLD,ierr)
    CALL MPI_BCAST(Q1Y(i,:),NUM_PT_Y-1,MPI_REAL,root_process, &
                   MPI_COMM_WORLD,ierr)
  END DO

!
! FIND THE REFINEMENT OF XX,YY,UU NAMELY XXX,YYY,OUT_U
!
  CALL SPLIT2D(NUM_PT_X,NUM_PT_Y,N,EPS,MX,MY,X,PX,P1X,Y,QX,Q1X, &
               XX,XXX,LOWER_QX,my_id,num_procs)

  IF (my_id .EQ. root_process) THEN
    PRINT*, 'PASS X'
  END IF

  CALL SPLIT2D(NUM_PT_X,NUM_PT_Y,N,EPS,MX,MY,X,PY,P1Y,Y,QY,Q1Y, &
               YY,YYY,LOWER_QY,my_id,num_procs)

  IF (my_id .EQ. root_process) THEN
    PRINT*, 'PASS Y'
  END IF

  CALL SPLIT2D(NUM_PT_X,NUM_PT_Y,N,EPS,MX,MY,X,P,P1,Y,Q,Q1,F,   &
               OUT_U,LOWER_QF,my_id,num_procs)

  IF (my_id .EQ. root_process) THEN
    PRINT*, 'PASS U'
!
! PRINT OUT THE OUTPUTS
!
    PRINT*, 'PRINTING THE OUTPUTS ... '
    DO i=1,MX+NUM_PT_X
      DO j=1,MY+NUM_PT_Y
        WRITE(21,*) OUT_U(i,j)
        WRITE(23,*) XXX(i,j)
        WRITE(24,*) YYY(i,j)
      END DO
    END DO
    WRITE(25,*) MX+NUM_PT_X
    WRITE(25,*) MY+NUM_PT_Y
    DO i=1,NUM_PT_X-1
      DO j=1,NUM_PT_Y-1
        WRITE(29,*) PX(i,j),QX(i,j)
        WRITE(30,*) PY(i,j),QY(i,j)
        WRITE(31,*) P(i,j),Q(i,j)
      END DO
    END DO
    DO i=1,NUM_PT_X-1
      DO j=1,NUM_PT_Y
        WRITE(29,*) P1X(i,j)
        WRITE(30,*) P1Y(i,j)
        WRITE(31,*) P1(i,j)
      END DO
    END DO
    DO i=1,NUM_PT_X
      DO j=1,NUM_PT_Y-1
        WRITE(29,*) Q1X(i,j)
        WRITE(30,*) Q1Y(i,j)
        WRITE(31,*) Q1(i,j)
      END DO
    END DO

    CLOSE(21)
    CLOSE(22)
    CLOSE(23)
    CLOSE(24)
    CLOSE(25)
    CLOSE(26)
    CLOSE(27)
    CLOSE(28)
    CLOSE(29)
    CLOSE(30)
    CLOSE(31)

  END IF

9999 CONTINUE

  CALL MPI_FINALIZE(ierr)

END PROGRAM SPLIT2DP

SUBROUTINE SPLIT2D(NUM_PT_X,NUM_PT_Y,N,EPS,MX,MY,X,P,P1,Y,Q,Q1,  &
                   F,OUT_U,LOWER_QF,my_id,num_procs)
!
! BY MR.ANIRUT LUADSONG
! NOV 2001
! SOLVING 2-D DMBVP BY SPLITTING METHOD
```

```
!
  IMPLICIT NONE
  INCLUDE 'mpif.h'
  INTENT(IN)  :: NUM_PT_X,NUM_PT_Y,N,EPS,MX,MY,X,Y,F,P,Q,P1,Q1,LOWER_QF
  INTENT(IN)  :: my_id,num_procs
  INTENT(OUT) :: OUT_U
  INTEGER     :: i,j,k,l,ii,jj,kk,ll,NUM_PT_X,NUM_PT_Y,MX,MY,ITER_NO,N
  INTEGER     :: my_id,num_procs,root_process,ierr
  INTEGER     :: status(MPI_STATUS_SIZE)
  REAL        :: TAU,NORM_U,EPS,LOWER_QF,TMP1
  REAL        :: FXXYY_BL,FXXYY_BR,FXXYY_TL,FXXYY_TR
  REAL,DIMENSION (NUM_PT_Y)                :: Y,FXX_L,FXX_R
  REAL,DIMENSION (NUM_PT_X)                :: X,FYY_B,FYY_T
  REAL,DIMENSION (NUM_PT_Y-1)              :: TMP_Q
  REAL,DIMENSION (NUM_PT_X-1)              :: TMP_P
  REAL,DIMENSION (MY)                      :: UXX_L,UXX_R
  REAL,DIMENSION (MX)                      :: UYY_B,UYY_T
  REAL,DIMENSION (NUM_PT_X,NUM_PT_Y)       :: F
  REAL,DIMENSION (MX,NUM_PT_Y)             :: U_H
  REAL,DIMENSION (NUM_PT_X,MY)             :: U_V
  REAL,DIMENSION (MX,MY)                   :: U0,U1,U2
  REAL,DIMENSION (NUM_PT_X-1,NUM_PT_Y-1)   :: P,Q
  REAL,DIMENSION (NUM_PT_X-1,NUM_PT_Y)     :: P1
  REAL,DIMENSION (NUM_PT_X,NUM_PT_Y-1)     :: Q1
  REAL,DIMENSION (MX+NUM_PT_X,MY+NUM_PT_Y) :: OUT_U
!
! CALCULATE THE NECESSARY NOTATIONS
!
  root_process = 0
!  IF (my_id .EQ. root_process) THEN
  TAU  = 1./(N*1.)
!
! FIND BOUNDARY VALUES
!
  DO i=my_id+1,NUM_PT_X,num_procs
    FYY_B(i) = -F(i,4)+4.*F(i,3)-5.*F(i,2)+2.*F(i,1)
    FYY_T(i) = -F(i,NUM_PT_Y-3)+4.*F(i,NUM_PT_Y-2)              &
               -5.*F(i,NUM_PT_Y-1)+2.*F(i,NUM_PT_Y)
    IF (my_id .EQ. root_process) THEN
      DO j=1,num_procs-1
        IF (i+j .LE. NUM_PT_X) THEN
          CALL MPI_RECV(FYY_B(i+j),1,MPI_REAL,j,                &
                        MPI_ANY_TAG,MPI_COMM_WORLD,status,ierr)
          CALL MPI_RECV(FYY_T(i+j),1,MPI_REAL,j,                &
                        MPI_ANY_TAG,MPI_COMM_WORLD,status,ierr)
        END IF
      END DO
    ELSE
      CALL MPI_SEND(FYY_B(i),1,MPI_REAL,root_process,99,       &
                    MPI_COMM_WORLD,ierr)
      CALL MPI_SEND(FYY_T(i),1,MPI_REAL,root_process,99,       &
                    MPI_COMM_WORLD,ierr)
    END IF
  END DO
  DO j=my_id+1,NUM_PT_Y,num_procs
    FXX_L(j) = -F(4,j)+4.*F(3,j)-5.*F(2,j)+2.*F(1,j)
    FXX_R(j) = -F(NUM_PT_X-3,j)+4.*F(NUM_PT_X-2,j)              &
               -5.*F(NUM_PT_X-1,j)+2.*F(NUM_PT_X,j)
    IF (my_id .EQ. root_process) THEN
      DO i=1,num_procs-1
        IF (i+j .LE. NUM_PT_Y) THEN
          CALL MPI_RECV(FXX_L(i+j),1,MPI_REAL,i,                &
                        MPI_ANY_TAG,MPI_COMM_WORLD,status,ierr)
          CALL MPI_RECV(FXX_R(i+j),1,MPI_REAL,i,                &
                        MPI_ANY_TAG,MPI_COMM_WORLD,status,ierr)
        END IF
      END DO
    ELSE
      CALL MPI_SEND(FXX_L(j),1,MPI_REAL,root_process,99,       &
                    MPI_COMM_WORLD,ierr)
      CALL MPI_SEND(FXX_R(j),1,MPI_REAL,root_process,99,       &
                    MPI_COMM_WORLD,ierr)
    END IF
  END DO
  IF (my_id .EQ. root_process) THEN
    FXXYY_BL = -FXX_L(4)+4.*FXX_L(3)-5.*FXX_L(2)+2.*FXX_L(1)
    FXXYY_BR = -FXX_R(4)+4.*FXX_R(3)-5.*FXX_R(2)+2.*FXX_R(1)
    FXXYY_TL = -FXX_L(NUM_PT_Y-3)+4.*FXX_L(NUM_PT_Y-2)       &
               -5.*FXX_L(NUM_PT_Y-1)+2.*FXX_L(NUM_PT_Y)
    FXXYY_TR = -FXX_R(NUM_PT_Y-3)+4.*FXX_R(NUM_PT_Y-2)       &
               -5.*FXX_R(NUM_PT_Y-1)+2.*FXX_R(NUM_PT_Y)
  END IF
  CALL MPI_BCAST(FYY_B(:),NUM_PT_X,MPI_REAL,root_process,    &
                 MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(FYY_T(:),NUM_PT_X,MPI_REAL,root_process,    &
                 MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(FXX_L(:),NUM_PT_Y,MPI_REAL,root_process,    &
                 MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(FXX_R(:),NUM_PT_Y,MPI_REAL,root_process,    &
                 MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(FXXYY_BL,1,MPI_REAL,root_process,           &
                 MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(FXXYY_BR,1,MPI_REAL,root_process,           &
               MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(FXXYY_TL,1,MPI_REAL,root_process,           &
               MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(FXXYY_TR,1,MPI_REAL,root_process,           &
               MPI_COMM_WORLD,ierr)
!
! FIND U_ik;jl ALONG TH MAIN MESH
!
  DO j=my_id+1,NUM_PT_Y,num_procs
    CALL DMBVP1D(NUM_PT_X,TAU,FXX_L(j),FXX_R(j),F(:,j),        &
               U_H(:,j),P1(:,j),N)
    IF (my_id .EQ. root_process) THEN
      DO i=1,num_procs-1
        IF (i+j .LE. NUM_PT_Y) THEN
          CALL MPI_RECV(U_H(:,j+i),MX,MPI_REAL,i,              &
                      MPI_ANY_TAG,MPI_COMM_WORLD,status,ierr)
        END IF
      END DO
    ELSE
      CALL MPI_SEND(U_H(:,j),MX,MPI_REAL,root_process,99,      &
                  MPI_COMM_WORLD,ierr)
    END IF
  END DO
  DO i=my_id+1,NUM_PT_X,num_procs
    CALL DMBVP1D(NUM_PT_Y,TAU,FYY_B(i),FYY_T(i),F(i,:),        &
               U_V(i,:),Q1(i,:),N)
    IF (my_id .EQ. root_process) THEN
      DO j=1,num_procs-1
        IF (i+j .LE. NUM_PT_X) THEN
          CALL MPI_RECV(U_V(i+j,:),MY,MPI_REAL,j,              &
                      MPI_ANY_TAG,MPI_COMM_WORLD,status,ierr)
        END IF
      END DO
    ELSE
      CALL MPI_SEND(U_V(i,:),MY,MPI_REAL,root_process,99,      &
                  MPI_COMM_WORLD,ierr)
    END IF
  END DO
!
! FIND Uxx AND Uyy ON THE BORDER OF THE MAIN DOMAIN
!
  IF (my_id .EQ. root_process) THEN

    CALL TENSION(NUM_PT_X,X,FYY_B,TMP_P,LOWER_QF)
    CALL DMBVP1D (NUM_PT_X,TAU,FXXYY_BL,FXXYY_BR,FYY_B,UYY_B,TMP_P,N)
    CALL TENSION(NUM_PT_X,X,FYY_T,TMP_P,LOWER_QF)
    CALL DMBVP1D (NUM_PT_X,TAU,FXXYY_TL,FXXYY_TR,FYY_T,UYY_T,TMP_P,N)
    CALL TENSION(NUM_PT_Y,Y,FXX_L,TMP_Q,LOWER_QF)
    CALL DMBVP1D (NUM_PT_Y,TAU,FXXYY_BL,FXXYY_TL,FXX_L,UXX_L,TMP_Q,N)
    CALL TENSION(NUM_PT_Y,Y,FXX_R,TMP_Q,LOWER_QF)
    CALL DMBVP1D (NUM_PT_Y,TAU,FXXYY_BR,FXXYY_TR,FXX_R,UXX_R,TMP_Q,N)
!
! TO FIND THE INITIAL VALUES BY LINEAR INTERPOLATION
!
    DO ll=1,MY
      kk = 0
      DO i=1,NUM_PT_X-1
        TMP1 = X(i)
        DO k=1,N-1
          kk        = kk+1
          TMP1      = TMP1+TAU
          U0(kk,ll) = U_V(i,ll)*(X(i+1)-TMP1)+U_V(i+1,ll)*(TMP1-X(i))
        END DO
      END DO
    END DO
    DO kk=1,MX
      ll = 0
      DO j=1,NUM_PT_Y-1
        TMP1 = Y(j)
        DO l=1,N-1
          ll        = ll+1
          TMP1      = TMP1+TAU
          U1(kk,ll) = U_H(kk,j)*(Y(j+1)-TMP1)+U_H(kk,j+1)*(TMP1-Y(j))
        END DO
      END DO
    END DO
    U0 = (U0+U1)/2.
!
! STARTING THE ITERATION
!
    ITER_NO = 0

  END IF

  CALL MPI_BCAST(UXX_L(:),MY,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(UXX_R(:),MY,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(UYY_B(:),MX,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  CALL MPI_BCAST(UYY_T(:),MX,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  DO i=1,NUM_PT_X
    CALL MPI_BCAST(U_V(i,:),MY,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  END DO
  DO j=1,NUM_PT_Y
    CALL MPI_BCAST(U_H(:,j),MX,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  END DO

5555 CONTINUE
  DO j=1,MY
    CALL MPI_BCAST(U0(:,j),MX,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  END DO
!
! THE FIRST HALF STEP, ALONG X DIRECTION
!
    DO ll=my_id+1, MY, num_procs
      j = ((ll-0.1)/(N-1))+1
      l = ((ll*1.0)/(N-1))
      IF (ll .EQ. (j-1)*(N-1)+1) THEN
        CALL SPLIT1(NUM_PT_X,N,UXX_L(ll),UXX_R(ll),P(:,j)       &
                   ,U_V(:,ll),F(:,j),U_V(:,ll+1)                &
                   ,U0(:,ll),U_H(:,j),U0(:,ll+1),U1(:,ll))
      ELSE IF (l .EQ. j) THEN
        CALL SPLIT1(NUM_PT_X,N,UXX_L(ll),UXX_R(ll),P(:,j)       &
                   ,U_V(:,ll),U_V(:,ll-1),F(:,j+1)              &
                   ,U0(:,ll),U0(:,ll-1),U_H(:,j+1),U1(:,ll))
      ELSE
        CALL SPLIT1(NUM_PT_X,N,UXX_L(ll),UXX_R(ll),P(:,j)       &
                   ,U_V(:,ll),U_V(:,ll-1),U_V(:,ll+1)           &
                   ,U0(:,ll),U0(:,ll-1),U0(:,ll+1),U1(:,ll))
      END IF
      IF (my_id .EQ. root_process) THEN
        DO i=1,num_procs-1
          IF (ll+i .LE. MY) THEN
            CALL MPI_RECV(U1(:,ll+i),MX,MPI_REAL,i,              &
                          MPI_ANY_TAG,MPI_COMM_WORLD,status,ierr)
          END IF
        END DO
      ELSE
        CALL MPI_SEND(U1(:,ll),MX,MPI_REAL,root_process,99, &
                      MPI_COMM_WORLD,ierr)
      END IF
    END DO
    DO j=1,MY
      CALL MPI_BCAST(U1(:,j),MX,MPI_REAL,root_process,      &
                     MPI_COMM_WORLD,ierr)
    END DO
!
! THE SECOND HALF STEP, ALONG Y DIRECTION
!
    DO kk=my_id+1, MX, num_procs
      i = ((kk-0.1)/(N-1))+1
      k = ((kk*1.0)/(N-1))
      IF (kk .EQ. (i-1)*(N-1)+1) THEN
        CALL SPLIT1(NUM_PT_Y,N,UYY_B(kk),UYY_T(kk),Q(i,:)      &
                   ,U_H(kk,:),F(i,:),U_H(kk+1,:)               &
                   ,U1(kk,:),U_V(i,:),U1(kk+1,:),U2(kk,:))
      ELSE IF (k .EQ. i) THEN
        CALL SPLIT1(NUM_PT_Y,N,UYY_B(kk),UYY_T(kk),Q(i,:)      &
                   ,U_H(kk,:),U_H(kk-1,:),F(i+1,:)             &
                   ,U1(kk,:),U1(kk-1,:),U_V(i+1,:),U2(kk,:))
      ELSE
        CALL SPLIT1(NUM_PT_Y,N,UYY_B(kk),UYY_T(kk),Q(i,:)      &
                   ,U_H(kk,:),U_H(kk-1,:),U_H(kk+1,:)          &
                   ,U1(kk,:),U1(kk-1,:),U1(kk+1,:),U2(kk,:))
      END IF
      IF (my_id .EQ. root_process) THEN
        DO j=1,num_procs-1
          IF (kk+j .LE. MX) THEN
            CALL MPI_RECV(U2(kk+j,:),MY,MPI_REAL,j,             &
                          MPI_ANY_TAG,MPI_COMM_WORLD,status,ierr)
          END IF
        END DO
      ELSE
        CALL MPI_SEND(U2(kk,:),MY,MPI_REAL,root_process,99,      &
                      MPI_COMM_WORLD,ierr)
      END IF
    END DO
!
! FIND NORM U AND UPDATE THE INITIAL DATA
!
  IF (my_id .EQ. root_process) THEN

    NORM_U  = MAXVAL(ABS(U2-U0))
    U0      = U2
    ITER_NO = ITER_NO+1
    PRINT *, 'ITERATION NUMBER = ',ITER_NO, '  NORM U = ',NORM_U

  END IF

  CALL MPI_BCAST(NORM_U,1,MPI_REAL,root_process,MPI_COMM_WORLD,ierr)
  IF (NORM_U .GE. EPS) GOTO 5555

  IF (my_id .EQ. root_process) THEN
!
! REARRANGE THE OUTPUT OUT_U
!
    ll = 0
    jj = 0
    DO j=1,NUM_PT_Y-1
      ll = ll+1
      kk = 1
      ii = 0
      DO i=1,NUM_PT_X-1
        OUT_U(kk,ll) =  F(i,j)
        kk           = kk+1
        DO k=1,N-1
          ii           = ii+1
          OUT_U(kk,ll) =  U_H(ii,j)
      kk = kk+1
        END DO
      END DO
      OUT_U(kk,ll) = F(NUM_PT_X,j)
      DO l=1,N-1
        ll = ll+1
        jj = jj+1
        kk = 1
        ii = 0
        DO i=1,NUM_PT_X-1
          OUT_U(kk,ll) =  U_V(i,jj)
      kk           = kk+1
      DO k=1,N-1
        ii           = ii+1
            OUT_U(kk,ll) = U2(ii,jj)
        kk           = kk+1
      END DO
        END DO
        OUT_U(kk,ll) = U_V(NUM_PT_X,jj)
      END DO
    END DO
    kk = 1
    ii = 0
    ll = ll+1
    DO i=1,NUM_PT_X-1
      OUT_U(kk,ll) = F(i,NUM_PT_Y)
      kk           = kk+1
      DO k=1,N-1
        ii           = ii+1
        OUT_U(kk,ll) = U_H(ii,NUM_PT_Y)
        kk           = kk+1
      END DO
    END DO
    OUT_U(kk,ll) = F(NUM_PT_X,NUM_PT_Y)

  END IF

END SUBROUTINE  SPLIT2D

SUBROUTINE SPLIT1(NUM_PT,N,F2P_0,F2P_N,P,F,F_B,F_T,U0,U0_B,U0_T,U1)
!
! BY MR.ANIRUT LUADSONG
! NOV 2001
! SOLVING EACH HALF STEP OF 2D-DMBVP
!
  IMPLICIT NONE
  INTENT(IN)  :: NUM_PT,N,F2P_0,F2P_N,P,F,F_B,F_T,U0,U0_B,U0_T
  INTENT(OUT) :: U1
  INTEGER     :: i,k,kk,NUM_PT,N
  REAL        :: T,T1,TAU,TAU2,F2P_0,F2P_N,TMP1
  REAL,DIMENSION (NUM_PT)             :: F,F_B,F_T
  REAL,DIMENSION (NUM_PT-1)           :: P,A_I,B_I,OMEGA_I
  REAL,DIMENSION ((NUM_PT-1)*(N-1))   :: U0,U0_B,U0_T,U1,B,BA1
  REAL,DIMENSION ((NUM_PT-1)*(N-1)-1) :: BA2
  REAL,DIMENSION ((NUM_PT-1)*(N-1)-2) :: BA3

  TAU  = 1./(N*1.)
  TAU2 = TAU*TAU
  T    = TAU2*TAU2
  T1   = TAU2*TAU2/T-4
  DO i=1,NUM_PT-1
    TMP1       = P(i)/(1.*N)
    OMEGA_I(i) = TMP1*TMP1
    A_I(i)     = -OMEGA_I(i)-4
    B_I(i)     = 6+2.*OMEGA_I(i)+TAU2*TAU2/T
  END DO
  kk = 0
  DO i=1,NUM_PT-1
    kk = kk+1
    IF (i.EQ.1) THEN
      B(kk)   = -TAU2*F2P_0-(2+A_I(i))*F(i)                      &
            -(F_B(i)+F_T(i)+U0_B(kk+1)+U0_T(kk+1))           &
        +2.*(U0_B(kk)+U0_T(kk)+F(i)+U0(kk+1))+T1*U0(kk)
      BA1(kk) = B_I(i)-1
    ELSE
      B(kk)   = -A_I(i)*F(i)                                     &
                 -(F_B(i)+F_T(i)+U0_B(kk+1)+U0_T(kk+1))          &
                 +2.*(U0_B(kk)+U0_T(kk)+F(i)+U0(kk+1))+T1*U0(kk)
      BA1(kk)   = B_I(i)
      BA2(kk-1) = 1.0
      BA3(kk-2) = 0.0
    END IF
    kk      = kk+1
    B(kk)   = -F(i)                                                 &
          -(U0_B(kk-1)+U0_T(kk-1)+U0_B(kk+1)+U0_T(kk+1))       &
          +2.*(U0_B(kk)+U0_T(kk)+U0(kk-1)+U0(kk+1))+T1*U0(kk)
    BA1(kk)   = B_I(i)
    BA2(kk-1) = A_I(i)
    IF (i.GT.1) THEN
      BA3(kk-2) = 0.0
    END IF
    DO k=3,N-3
      kk      = kk+1
      B(kk)   = -(U0_B(kk-1)+U0_T(kk-1)+U0_B(kk+1)+U0_T(kk+1))       &
            +2.*(U0_B(kk)+U0_T(kk)+U0(kk-1)+U0(kk+1))+T1*U0(kk)
      BA1(kk)   = B_I(i)
      BA2(kk-1) = A_I(i)
      BA3(kk-2) = 1.0
    END DO
    kk    = kk+1
    B(kk) = -F(i+1)                                                &
        -(U0_B(kk-1)+U0_T(kk-1)+U0_B(kk+1)+U0_T(kk+1))       &
        +2.*(U0_B(kk)+U0_T(kk)+U0(kk-1)+U0(kk+1))+T1*U0(kk)
    BA1(kk)   = B_I(i)
    BA2(kk-1) = A_I(i)
    BA3(kk-2) = 1.0
    kk        = kk+1
    IF (i.EQ.NUM_PT-1) THEN
      B(kk)   = -TAU2*F2P_N-(2+A_I(i))*F(i+1)                         &
                -(U0_B(kk-1)+U0_T(kk-1)+F_B(i+1)+F_T(i+1))          &
                +2.*(U0_B(kk)+U0_T(kk)+U0(kk-1)+F(i+1))+T1*U0(kk)
      BA1(kk) = B_I(i)-1
    ELSE
      B(kk)   = -A_I(i)*F(i+1)                                        &
                -(U0_B(kk-1)+U0_T(kk-1)+F_B(i+1)+F_T(i+1))          &
            +2.*(U0_B(kk)+U0_T(kk)+U0(kk-1)+F(i+1))+T1*U0(kk)
      BA1(kk) = B_I(i)
    END IF
    BA2(kk-1) = A_I(i)
    BA3(kk-2) = 1.0
  END DO
  CALL  SUB5DIAG((N-1)*(NUM_PT-1),B,BA1,BA2,BA3,U1)

END SUBROUTINE SPLIT1

SUBROUTINE DMBVP1D(NUM_PT,TAU,F2P_0,F2P_END,F,U,P,N)
!
! BY MR.ANIRUT LUADSONG
! OCT 2001
! SOLVING 1D-DMBVP
!
  IMPLICIT NONE
  INTENT(IN)  :: NUM_PT,TAU,F2P_0,F2P_END,F,P,N
  INTENT(OUT) :: U
  INTEGER     :: i,j,ii,NUM_PT,N
  REAL        :: TAU,TAU2,F2P_0,F2P_END,TMP1
  REAL,DIMENSION (NUM_PT)            :: F
  REAL,DIMENSION (NUM_PT-1)          :: P,A_I,B_I,OMEGA_I
  REAL,DIMENSION ((N-1)*(NUM_PT-1))  :: U,B,BA1
  REAL,DIMENSION ((N-1)*(NUM_PT-1)-1):: BA2
  REAL,DIMENSION ((N-1)*(NUM_PT-1)-2):: BA3

  TAU2 = TAU*TAU
  DO i=1,NUM_PT-1
    TMP1       = P(i)/N
    OMEGA_I(i) = TMP1*TMP1
    A_I(i)     = -OMEGA_I(i)-4
    B_I(i)     = 6+2*OMEGA_I(i)
  END DO
  ii    = 0
  DO i=1,NUM_PT-1
    ii = ii+1
    IF (i.EQ.1) THEN
      B(ii)   = -TAU2*F2P_0-(2+A_I(i))*F(i)
      BA1(ii) = B_I(i)-1
    ELSE
      B(ii)     = -A_I(i)*F(i)
      BA1(ii)   = B_I(i)
      BA2(ii-1) = 1.0
      BA3(ii-2) = 0.0
    END IF
    ii        = ii+1
    B(ii)     = -F(i)
    BA1(ii)   = B_I(i)
    BA2(ii-1) = A_I(i)
    IF (i.GT.1) THEN
      BA3(ii-2) = 0.0
    END IF
    DO j=3,N-3
      ii        = ii+1
      B(ii)     = 0.0
      BA1(ii)   = B_I(i)
      BA2(ii-1) = A_I(i)
      BA3(ii-2) = 1.0
    END DO
    ii = ii+1
    B(ii)     = -F(i+1)
    BA1(ii)   = B_I(i)
    BA2(ii-1) = A_I(i)
    BA3(ii-2) = 1.0
    ii = ii+1
    IF (i.EQ.NUM_PT-1) THEN
      B(ii)   = -TAU2*F2P_END-(2+A_I(i))*F(i+1)
      BA1(ii) = B_I(i)-1
    ELSE
      B(ii)   = -A_I(i)*F(i+1)
      BA1(ii) = B_I(i)
    END IF
    BA2(ii-1) = A_I(i)
    BA3(ii-2) = 1.0
  END DO
  CALL  SUB5DIAG((N-1)*(NUM_PT-1),B,BA1,BA2,BA3,U)

END SUBROUTINE DMBVP1D

SUBROUTINE SUB5DIAG(N,B,D,F,E,X)
!
! BY MR.ANIRUT LUADSONG
! GIVEN: Ax=b WITH A FIVE-DIAGONAL SYMMETRIC STRONGLY
!        NONSINGULAR MATRIX A
! FIND : SOLUTION x.
!
  IMPLICIT NONE
  INTENT(IN)  :: N,B,D,F,E
  INTENT(OUT) :: X
  INTEGER     :: N,i
  REAL,DIMENSION (N)   :: B,D,X,C,Z,ALPHA
  REAL,DIMENSION (N-2) :: E,DELTA
  REAL,DIMENSION (N-1) :: F,GAMMA
!
! 1st STEP: FACTOR A=LDL^T
!
  ALPHA(1) = D(1)
  GAMMA(1) = F(1)/ALPHA(1)
  DELTA(1) = E(1)/ALPHA(1)
  ALPHA(2) = D(2)-F(1)*GAMMA(1)
  GAMMA(2) = (F(2)-E(1)*GAMMA(1))/ALPHA(2)
  DELTA(2) = E(2)/ALPHA(2)
  DO i=3,N-2
    ALPHA(i) = D(i)-E(i-2)*DELTA(i-2)-ALPHA(i-1)             &
               *GAMMA(i-1)*GAMMA(i-1)
    GAMMA(i) = (F(i)-E(i-1)*GAMMA(i-1))/ALPHA(i)
    DELTA(i) = E(i)/ALPHA(i)
  END DO
  ALPHA(N-1) = D(N-1)-E(N-3)*DELTA(N-3)-ALPHA(N-2)           &
               *GAMMA(N-2)*GAMMA(N-2)
  GAMMA(N-1) = (F(N-1)-E(N-2)*GAMMA(N-2))/ALPHA(N-1)
  ALPHA(N)   = D(N)-E(N-2)*DELTA(N-2)-ALPHA(N-1)             &
               *GAMMA(N-1)*GAMMA(N-1)
!
! 2nd STEP: UPDATE Lz=b, Dc=z
!
  Z(1) = B(1)
  Z(2) = B(2)-GAMMA(1)*Z(1)
  DO i=3,N
    Z(i) = B(i)-GAMMA(i-1)*Z(i-1)-DELTA(i-2)*Z(i-2)
  END DO
  DO i=1,N
    C(i) = Z(i)/ALPHA(i)
  END DO
!
! 3rd STEP: BACKSUBSTITUTION L^Tx=c
!
  X(N)   = C(N)
  X(N-1) = C(N-1)-GAMMA(N-1)*X(N)
  DO i=N-2,1,-1
    X(i) = C(i)-GAMMA(i)*X(i+1)-DELTA(i)*X(i+2)
  END DO
END SUBROUTINE SUB5DIAG

SUBROUTINE TENSION2D(NUM_PT_X,NUM_PT_Y,X,Y,F,P,Q,P1,Q1,LOWER_Q)
!
! BY MR.ANIRUT LUADSONG
! NOV 2001
! GENERATE 2D TENSION PARAMETERS
!
  IMPLICIT NONE
  INTENT(IN)  :: NUM_PT_X,NUM_PT_Y,X,Y,F,LOWER_Q
  INTENT(OUT) :: P,Q,P1,Q1
  INTEGER     :: i,j,NUM_PT_X,NUM_PT_Y
  REAL        :: LOWER_Q
  REAL,DIMENSION (NUM_PT_X-1,NUM_PT_Y-1) :: P,Q
  REAL,DIMENSION (NUM_PT_X-1,NUM_PT_Y)   :: P1
  REAL,DIMENSION (NUM_PT_X,NUM_PT_Y-1)   :: Q1
  REAL,DIMENSION (NUM_PT_X,NUM_PT_Y)     :: F
  REAL,DIMENSION (NUM_PT_X)              :: X,TMP_FX
  REAL,DIMENSION (NUM_PT_Y)              :: Y,TMP_FY
!
! FIND TENSION PARAMETERS ALONG THE MAIN MESH
```

```
!
  DO j=1,NUM_PT_Y
    CALL TENSION(NUM_PT_X,X,F(:,j),P1(:,j),LOWER_Q)
  END DO
  DO i=1,NUM_PT_X
    CALL TENSION(NUM_PT_Y,Y,F(i,:),Q1(i,:),LOWER_Q)
  END DO
!
! FIND TENSION PARAMETERS INSIDE SUBDOMAIN
!
  DO j=1,NUM_PT_Y-1
    TMP_FX(:) = (F(:,j)+F(:,j+1))/2.
    CALL TENSION(NUM_PT_X,X,TMP_FX,P(:,j),LOWER_Q)
  END DO
  DO i=1,NUM_PT_X-1
    TMP_FY(:) = (F(i,:)+F(i+1,:))/2.
    CALL TENSION(NUM_PT_Y,Y,TMP_FY,Q(i,:),LOWER_Q)
  END DO

END SUBROUTINE TENSION2D

SUBROUTINE TENSION(END_PT,X,F,Q,LOWER_Q)
!
! BY MR.ANIRUT LUADSONG
! NOV 2001
! GENERATE 1D TENSION PARAMETERS
!
  IMPLICIT NONE
  INTENT(IN)  :: END_PT,X,F,LOWER_Q
  INTENT(OUT) :: Q
  INTEGER     :: i,END_PT
  REAL        :: SLOPE_F_STPT,SLOPE_F_EDPT,TMP_L,TMP_R,LOWER_Q
  REAL,DIMENSION (END_PT)     :: X,F
  REAL,DIMENSION (END_PT-1)   :: Q,H,DIVID
  REAL,DIMENSION (2*(END_PT)) :: L_EQ_Q,R_EQ_Q

  DO i=1,END_PT-1
    H(i)     = X(i+1)-X(i)
    DIVID(i) = (F(i+1)-F(i))/(1.*H(i))
  END DO
  SLOPE_F_STPT = (-F(3)+4.*F(2)-3.*F(1))/(1.*H(1)+1.*H(2))
  SLOPE_F_EDPT = (F(END_PT-2)-4*F(END_PT-1)+3*F(END_PT))/         &
                 (1.*H(END_PT-1)+1.*H(END_PT-2))
  IF (LOWER_Q .EQ. 0) THEN
    Q = 0.00000001
  ELSE
    DO i=1,END_PT-1
      Q = LOWER_Q
    END DO
  END IF
  i         = 1
  R_EQ_Q(i) = 1.
  L_EQ_Q(i) = 0.
  DO WHILE ((L_EQ_Q(i).LT.R_EQ_Q(i)).AND.(Q(i).LE.85))
    IF ((SLOPE_F_STPT .EQ. 0) .AND. (DIVID(i) .EQ. 0)) THEN
      R_EQ_Q(i) = 200
    ELSE IF (SLOPE_F_STPT .EQ. 0) THEN
      R_EQ_Q(i) = 0.005
    ELSE IF (DIVID(i) .EQ. 0) THEN
      R_EQ_Q(i) = 200
    ELSE
      R_EQ_Q(i) = SLOPE_F_STPT/DIVID(i)
    END IF
!
!  Added in 28/01/02 at ANU, Canberra,
!  for supporting UNIX(SUN) system
!
    DO WHILE (ABS(SINH(Q(i))-Q(i)).EQ.0.)
      Q(i) = Q(i)+0.0001
    END DO
!  End of added
    L_EQ_Q(i) = -Q(i)*(1-COSH(Q(i)))/(SINH(Q(i))-Q(i))
    Q(i)      = Q(i)+1
  END DO
  Q(i) = Q(i)-1
  DO i=2,END_PT-2
    TMP_R = 1
    TMP_L = 0
    DO WHILE ((TMP_L .LT. TMP_R).AND.(Q(i).LE.85))
      IF ((DIVID(i) .EQ. 0) .AND. (DIVID(i-1) .EQ. 0)) THEN
        R_EQ_Q(i) = 200
      ELSE IF (DIVID(i-1) .EQ. 0) THEN
        R_EQ_Q(i) = 0.005
      ELSE IF (DIVID(i) .EQ. 0) THEN
        R_EQ_Q(i) = 200
      ELSE
        R_EQ_Q(i) = DIVID(i-1)/DIVID(i)
      END IF
!
!  Added in 28/01/02 at ANU, Canberra,
!  for supporting UNIX(SUN) system
!
      DO WHILE (ABS(SINH(Q(i))-Q(i)).EQ.0.)
```

```
        Q(i) = Q(i)+0.0001
      END DO
!  End of added
      L_EQ_Q(i) = -(2./3.)*Q(i)*(1-COSH(Q(i)))/(SINH(Q(i))-Q(i))
      IF ((DIVID(i) .EQ. 0) .AND. (DIVID(i+1) .EQ. 0)) THEN
        R_EQ_Q(i+END_PT) = 200
      ELSE IF (DIVID(i+1) .EQ. 0) THEN
        R_EQ_Q(i+END_PT) = 0.005
      ELSE IF (DIVID(i) .EQ. 0) THEN
        R_EQ_Q(i+END_PT) = 200
      ELSE
        R_EQ_Q(i+END_PT) = DIVID(i+1)/DIVID(i)
      END IF
!
!  Added in 28/01/02 at ANU, Canberra,
!  for supporting UNIX(SUN) system
!
      DO WHILE (ABS(SINH(Q(i))-Q(i)).EQ.0.)
        Q(i) = Q(i)+0.0001
      END DO
!  End of added
      L_EQ_Q(i+END_PT) = (2./3.)*Q(i)*(COSH(Q(i))-1)/(SINH(Q(i))-Q(i))
      TMP_L = L_EQ_Q(i)+L_EQ_Q(i+END_PT)
      TMP_R = R_EQ_Q(i)+R_EQ_Q(i+END_PT)
      Q(i)  = Q(i)+1
    END DO
    Q(i) = Q(i)-1
  END DO
  i     = END_PT-1
  TMP_R = 1
  TMP_L = 0
  DO WHILE ((TMP_L .LT. TMP_R).AND.(Q(i).LE.85))
    IF ((DIVID(i) .EQ. 0) .AND. (DIVID(i-1) .EQ. 0)) THEN
      R_EQ_Q(i) = 200
    ELSE IF (DIVID(i-1) .EQ. 0) THEN
      R_EQ_Q(i) = 0.005
    ELSE IF (DIVID(i) .EQ. 0) THEN
      R_EQ_Q(i) = 200
    ELSE
      R_EQ_Q(i) = DIVID(i-1)/DIVID(i)
    END IF
!
!  Added in 28/01/02 at ANU, Canberra,
!  for supporting UNIX(SUN) system
!
    DO WHILE (ABS(SINH(Q(i))-Q(i)).EQ.0.)
      Q(i) = Q(i)+0.0001
    END DO
!  End of added
    L_EQ_Q(i) = -(2./3.)*Q(i)*(1-COSH(Q(i)))/(SINH(Q(i))-Q(i))
    IF ((SLOPE_F_EDPT.EQ.0).AND.(DIVID(i).EQ.0)) THEN
      R_EQ_Q(i+1) = 200
    ELSE IF (SLOPE_F_EDPT .EQ. 0) THEN
      R_EQ_Q(i+1) = 0.005
    ELSE IF (DIVID(i) .EQ. 0) THEN
      R_EQ_Q(i+1) = 200
    ELSE
      R_EQ_Q(i+1) = SLOPE_F_EDPT/DIVID(i)
    END IF
!
!  Added in 28/01/02 at ANU, Canberra,
!  for supporting UNIX(SUN) system
!
    DO WHILE (ABS(SINH(Q(i))-Q(i)).EQ.0.)
      Q(i) = Q(i)+0.0001
    END DO
!  End of added
      L_EQ_Q(i+1) = Q(i)*(COSH(Q(i))-1)/(SINH(Q(i))-Q(i))
      TMP_L = L_EQ_Q(i)+L_EQ_Q(i+1)
      TMP_R = R_EQ_Q(i)+R_EQ_Q(i+1)
      Q(i)  = Q(i)+1
    END DO
    Q(i) = Q(i)-1
    DO i=2,END_PT-2
      IF ((DIVID(i).EQ.DIVID(i+1)).AND.(DIVID(i).EQ.DIVID(i-1))) THEN
        Q(i) = 400.
      END IF
    END DO
    IF ((DIVID(1).EQ.DIVID(2)).AND.(DIVID(1).EQ.SLOPE_F_STPT)) THEN
      Q(1) = 400.
    END IF
    IF ((DIVID(END_PT-1).EQ.DIVID(END_PT-2)).AND.(DIVID(END_PT-1).EQ. &
         SLOPE_F_EDPT)) THEN
      Q(END_PT-1) = 400.
    END IF

END SUBROUTINE TENSION
```

Appendix D

MatLab Code for Creating Graphics

```
===========================================================================
To Produce a Graphic for 1-D DMBVP by MatLab
===========================================================================
fx=fopen('X.TXT','r');
fu=fopen('U.TXT','r');
x=fscanf(fx,'%g');
u=fscanf(fu,'%g');
plot(x,u);
fclose(fx);
fclose(fu);
```

```
===========================================================================
To Produce a Graphic for 2-D DMBVP by MatLab
===========================================================================
fid1=fopen('2D_X.TXT','r');
fid2=fopen('2D_Y.TXT','r');
fid3=fopen('2D_U.TXT','r');
fid4=fopen('SIZE.TXT','r');
x=fscanf(fid1,'%g');
y=fscanf(fid2,'%g');
u=fscanf(fid3,'%g');
s=fscanf(fid4,'%g');
x=reshape(x,s(2,1),s(1,1));
y=reshape(y,s(2,1),s(1,1));
u=reshape(u,s(2,1),s(1,1));
xx=transpose(x);yy=transpose(y);
uu=transpose(u);fclose(fid1);
fclose(fid2);
fclose(fid3);
fclose(fid4);
mesh(xx,yy,uu)
xlabel('x')ylabel('y')zlabel('z')
axis([min(min(xx)),max(max(xx)),min(min(yy)),
      max(max(yy)),min(min(uu)),max(max(uu))])
```

Curriculum Vitae

FIRST NAME: Anirut

LAST NAME: Luadsong

GENDER: Male

NATIONALITY: Thai

DATE OF BIRTH: December 28, 1971

EDUCATION BACKGROUND:

- B.Sc. 2nd Class Honors in Mathematics, May 1993, Ramkhamhaeng University, Bangkok, Thailand.
- M.Sc. in Applied Mathematics, February 1997, King Mongkut's University of Technology Thonburi, Bangkok, Thailand.

WORK EXPERIENCE:

- Customer Support, Professional Computer Company Limited (A joint venture company of Loxley & IBM Thailand), 1994-1995.
- Lecturer in Mathematics at King Mongkut's University of Technology Thonburi, Bangkok, Thailand, Since 1997.
- Long Term Visitor, Centre for Mathematics and its Applications, The Australian National University, Canberra, Australia, January 2002 - May 2002.

PUBLICATIONS:

- Boris Kvasov and Anirut Luadsong, "Approximation by Lagrange splines", in: Proceedings of the Fourth Annual National Symposium on Computational Science and Engineering (ANSCSE'2000), Kasetsart University, March 27-29, 2000, pp. 306–315.
- Boris Kvasov and Anirut Luadsong, "Difference method for constructing shape-preserving splines", in: Proceedings of the Fifth Annual National Symposium on Computational Science and Engineering (ANSCSE'2001), Bangkok Conventional Center (BCC), Central Plaza, June 18-20, 2001, pp. 390–397.