NASA Contractor Report 177968

ALGEBRAIC GRID GENERATION USING
TENSOR PRODUCT B-SPLINES

ALGEBRAIC GRID GENERATION USING TENSOR PRODUCT B-SPLINES Ph.D. Thesis (~~Oklahoma~~ Old Dominion Univ.) 152 p HC A08/MF A01 CSCL 12A
N86-11908
Unclas
G3/64 27608

Bonita V. Saunders

OLD DOMINION UNIVERSITY
Norfolk, Virginia

Grant NGT 47-003-802
September 1985

NASA

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

ABSTRACT

ALGEBRAIC GRID GENERATION USING TENSOR PRODUCT B-SPLINES

Bonita Valerie Saunders
Old Dominion University, 1985
Director: Dr. Philip W. Smith

In general, finite difference methods are more successful if the accompanying grid has lines which are smooth and nearly orthogonal. This thesis discusses the development of an algorithm which produces such a grid when given the boundary description.

Topological considerations in structuring the grid generation mapping are discussed. In particular, this thesis examines the concept of the degree of a mapping and how it can be used to determine what requirements are necessary if a mapping is to produce a suitable grid.

The grid generation algorithm uses a mapping composed of bicubic B-splines. Boundary coefficients are chosen so that the splines produce Schoenberg's variation diminishing spline approximation to the boundary. Interior coefficients are initially chosen to give a variation diminishing approximation to the transfinite bilinear interpolant of the function mapping the boundary of the unit square onto

the boundary of the grid.

The practicality of optimizing the grid by minimizing a functional involving the Jacobian of the grid generation mapping at each interior grid point and the dot product of vectors tangent to the grid lines is investigated.

Grids generated by using the algorithm are presented.

ACKNOWLEDGEMENTS

I would like to thank my dissertation director, Dr. Philip W. Smith, for his encouragement, patience and assistance during all stages of the development of this dissertation. I also thank Dr. Robert Smith of NASA Langley Research Center for his guidance and instruction during my past three years at NASA. Dr. Smith offered invaluable suggestions and advice to me during my research. I would also like to thank others who helped in the reviewing of this thesis. The comments and suggestions of Dr. John Tweed, Dr. James Schwing and Dr. John Swetits are greatly appreciated.

I must also express my gratitude to the entire Computer Science and Applications Branch in the Analysis and Computation Division at Langley Research Center. Everyone always willingly answered my questions or found others who could assist me.

Finally, my sincere thanks to Mrs. Harriet Sims whose patience and perseverance made the typing of this dissertation possible.

TABLE OF CONTENTS

Page

LIST OF FIGURES. iv

Chapter

1. INTRODUCTION 1

2. APPLICATIONS OF DEGREE THEORY. 6

   2.1 Defining the Degree of a Mapping. 7

   2.2 Properties of the Degree. 8

   2.3 A Topological Definition of the Degree. . . 11

   2.4 Applications to Grid Generation 19

   2.5 Additional Topological Questions. 22

3. AN ALGEBRAIC GRID GENERATION MAPPING 27

   3.1 A First Order Example 27

   3.2 B-splines 30

      3.2.1 Defining B-splines 30

      3.2.2 Properties of B-splines. 31

      3.2.3 Spline Functions 34

      3.2.4 Variation Diminishing Splines. . . . 36

      3.2.5 Tensor Product B-splines 38

   3.3 A Smoothing Functional. 40

      3.3.1 Characteristics of the Functional. . 42

      3.3.2 Convergence of the Smoothing Functional. 46

Page

4. PROGRAM TENTEST. 50

4.1 The Algorithm 50

4.2 Computing the Tensor Product B-splines. . . 51

4.3 Choosing the Initial Coefficients 53

4.4 Minimizing the Smoothing Functional 56

4.5 Distribution Functions. 60

5. RESULTS AND DISCUSSION 66

5.1 Convex Domains. 68

5.1.1 Trapezoid. 68

5.1.2 Quadrilateral with Unequal Sides . . 70

5.1.3 Triangle 74

5.1.4 Circle 74

5.2 Nonconvex Domains 77

5.2.1 Nonconvex Quadrilateral. 77

5.2.2 Puzzle Pieces. 81

5.3 Grids for Specific Objects. 85

5.3.1 Airfoil. 91

5.3.2 Spike-Nosed Body 95

5.3.3 Shuttle. 95

6. CONCLUSIONS. 102

APPENDIX 106

LIST OF FIGURES

FIGURE		Page
1.	Invariance of the degree when points connected by a continuous path avoiding $F(\partial C)$.	10
2.	Mapping the unit square onto a nonconvex domain	24
3.	Initial grid on nonconvex domain.	25
4.	Optimized grid on nonconvex domain.	26
5.	Support of tensor product B-spline B_{ij}. . . .	41
6.	Mapping from computational domain to physical domain	55
7.	Trapezoid Grid.	57
8.	Support for tensor product B-spline $B_{6,5}$. . .	59
9.	Obtaining a concentration of grid points. . .	62
10.	Concentrating grid points on trapezoid domain	64
11.	Grids on trapezoid domain	69
12.	Effect of weights, wj and wd.	71
13.	Grids on quadrilateral with unequal sides . .	72
14.	Concentrating grid points on quadrilateral. .	73
15.	Division of triangular boundary into four sections	75
16.	Grids on triangular domain.	76
17.	Grids on circular domain.	78
18.	Grids on circular domain after optimization .	79

FIGURE		Page

19. Nonconvex quadrilateral. 80

20. Grids on nonconvex quadrilateral domain. . . . 82

21. Optimized grids on nonconvex quadrilateral domain. 83

22. Enlarged corner of optimized grid. 84

23. Puzzle shaped domains. 86

24. Grids on first puzzle shaped domain. 87

25. Grids obtained after various iterations. . . . 88

26. Grids on second puzzle shaped domain 89

27. Exponential distributions on puzzle shaped grids 90

28. Domain around Kármán - Trefftz airfoil 92

29. Grids for Kármán - Trefftz airfoil 93

30. Optimized grids for Kármán - Trefftz airfoil . 94

31. Grids for spike-nosed body 96

32. Optimized grids for spike-nosed body 97

33. Original grid for shuttle. 99

34. Optimized grid for shuttle 100

35. Optimized grid concentrated near shuttle boundary. 101

1. INTRODUCTION

Grid generation is the numerical development of curvilinear coordinate systems. In recent years grid generation has been the key to solving partial differential equations on arbitrarily shaped regions by finite difference methods. Although much of the motivation for grid generation has come from fluid dynamics, the techniques apply to any area, such as electromagnetics and heat transfer, which involves the solving of partial differential equations on a physical domain.

Inherent in grid generation techniques is a mapping T from some canonical domain such as a square or rectangle in two dimensions, or cube in three dimensions, onto the physical domain on which the partial differential equations are to be solved. The image of a mesh on the canonical, or computational, domain will be a grid on the physical domain. When the grid boundary coincides with the boundary of the physical domain, the system generated is called a boundary fitted coordinate system.

A boundary fitted coordinate system allows one to apply boundary conditions exactly, thus avoiding interpolation errors. However, such a system may make the equations to be solved more complex [Sm].

The distribution of the coordinate lines, or grid lines, should be smooth, but concentrated in areas where a large gradient occurs in the physical solution. As stated by Thompson, Warsi and Mastin [TWM], "the grid points may be thought of as a finite set of observers of the physical solution, stationed to be most effective in covering all of the action on the field." Ideally, the grid should be adaptive, that is, coupled with the physical solution so that it automatically redistributes its grid lines to obtain the desired regions of concentration as the solution evolves. However, the interior lines should not cross the physical boundary and should be nearly orthogonal at the intersection points to avoid large truncation errors in the finite difference approximations.

Grid generation is based on the observation that finite difference computations are much easier to make on a uniform mesh over a canonical domain such as a square or cube than on a grid over an irregularly shaped region. Therefore, the partial differential equations to be solved must first be transformed so that the computational coordinates become the independent coordinates. The resulting equations may then be expressed as finite difference equations on the computational domain.

Grid generation techniques may be divided into two general types: partial differential equation methods and algebraic methods. P.d.e. methods include elliptic, hyperbolic and conformal mapping techniques. All of these methods involve the solving of partial differential equations to obtain the grid coordinates. The simplest elliptic method for grid generation uses the Laplace equations

$$\Delta^2 \xi = \frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2} = 0$$

$$\Delta^2 \eta = \frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2} = 0$$

where ξ and η are the computational coordinates and x and y are the physical coordinates in two dimensions. The equations are first transformed so that the independent and dependent variables are interchanged. Then the new equations are solved for x and y in terms of ξ and η. Some control over the grid cell spacing can be accomplished by introducing control functions $P(\xi,\eta)$, $Q(\xi,\eta)$ and solving the Poisson equations [TWM, p. 39]

$$\Delta^2 \xi = P(\xi,\eta)$$

$$\Delta^2 \eta = Q(\xi,\eta).$$

Solving the Laplace equations

$$\Delta^2 \xi = 0$$

$$\Delta^2 \eta = 0$$

with boundary conditions

$$\xi_x = \eta_y$$

$$\xi_y = -\eta_x$$

produces a conformal transformation [TWM, p. 11]

Starius [St, p. 27] shows that solving an initial value problem satisfying

$$x_\eta = -y_\xi F$$

$$y_\eta = x_\xi F$$

where F is chosen so that the system is hyperbolic produces a hyperbolic grid generating system. Grids generated from elliptic equations are generally smooth regardless of the type of boundary, but slope discontinuities propagate through hyperbolically generated grids [St]. Generating a grid using conformal mapping techniques requires careful selection of the boundary data, making it difficult to structure the grid to obtain a high concentration of grid points in areas of large gradients in the physical solution. More grid points may have to be added in order to capture regions of rapid change such as shocks and boundary layers. Also in p.d.e. generated systems the Jacobian information needed for the transformation of the equations being solved must be computed numerically.

In algebraic methods an explicit functional relationship between the computational and physical domains is defined. Therefore, no p.d.e. need be solved to obtain the grid coordinates and the Jacobian matrix can be computed analytically. Such methods allow more precise controls of the grid structure making it easier to concentrate grid points in large gradient areas. However, algebraically generated grids are more sensitive to point distributions on the boundary and, in general, may not be as smooth as those generated by elliptic techniques [Sm]. Slope discontinuities on the boundary may propagate into the field. Nevertheless, a variety of techniques have been used to produce acceptable smoothness in algebraically generated grids.

This thesis discusses an algebraic grid generation technique for creating boundary fitted coordinate systems. This technique uses a mapping which is a sum of tensor product B-splines. Chapter 2 discusses degree theory, explaining how the degree of a mapping can be used to determine what conditions must be met if an algebraic transformation is to produce a suitable grid. Chapter 3 presents the tensor product grid generation mapping and discusses the properties of B-splines to show their suitability for use in such a mapping. Chapter 3 also introduces a functional which can be used to change the coefficients in the mapping in order to enhance the smoothness and orthogonality in the generated grid.

Chapter 4 discusses the computer program TENTEST which uses the techniques presented in Chapter 3 to generate grids on arbitrarily shaped two-dimensional domains. Some of the grids created using TENTEST are illustrated and discussed in Chapter 5. Conclusions and suggestions for further study are presented in Chapter 6.

2. APPLICATIONS OF DEGREE THEORY

This chapter discusses degree thoery and shows how the degree of a mapping can be used to help determine what requirements are necessary if a transformation T is to produce a suitable grid.

Since the distribution of grid lines should be smooth with concentration in areas of large gradients in the physical solution, the image of T should cover the entire physical domain, that is, T should be onto. Also, the transformation should be one to one. In terms of the grid, this means that the grid lines should not overlap the physical boundary and should intersect only at points corresponding to intersection points on the mesh in the computation domain.

Requiring T to be one to one and onto is equivalent to saying that the system $T(s)=p$ must have one and only one solution in the computational domain for each point p in the physical domain. This provides the motivation for looking at the following general problem:

Pick an open set $D \subset R^n$, where R^n is euclidean n-space, and let C be an open bounded set such that $\bar{C} \subset D$. If $F: D \subset R^n \to R^n$ is a continuous mapping and $y \in R^n$ is given, how many solutions of $F(x)=y$ exist in C?

The difficulty in solving this problem lies in the fact that in general the solutions do not vary continuously with F or y. This difficulty may be resolved by looking instead at the difference between the number of solutions for which the Jacobian of F is positive and the number of solutions for which the Jacobian of F is negative. Loosely, this is what is called the degree of F at y with respect to C.

2.1 Defining the Degree of a Mapping

A more precise definition of the degree of a mapping F takes on different forms depending on what restrictions are placed on F. What follows are essentially the definitions presented in references [S] and [O].

2.1-1 Definition. Let $C \subset R^n$ be an open bounded set and let $F:\bar{C} \subset R^n \to R^n$ be continuously differentiable on C. Pick $y \notin F(\partial C)$ and let $\Gamma = \{x \in C \mid F(x) = y\}$. If $F'(x)$ is nonsingular for all $x \in \Gamma$ then one defines the degree of F at y with respect to C by

$$\deg(F,C,y) = \sum_{x \in \Gamma} \operatorname{sign} \det F'(x).$$

In [O], Ortega and Rheinboldt actually define the degree in terms of an integral and then show that it has the equivalent form given above.

On removing the restriction that $\det F'(x) \neq 0$ for $x \in \Gamma$ the definition becomes

$$\deg(F,C,y) = \lim_{k \to \infty} \deg(F,C,y_k)$$

where $\lim_{k \to \infty} y_k = y$ and each element of $\{y_k\}$ satisfies $y_k \notin F(\partial C)$ and $\det F'(x) \neq 0$ whenever $F(x) = y_k$.

Actually, one can make the stronger statement that for any such sequence $\{y_k\}$ there is a k_0 such that $\deg(F,C,y) = \deg(F,C,y_k)$ for $k \geq k_0$ [0, p. 159].

The Weierstrass approximation theorem makes it possible to extend the definition of the degree of a mapping to a continuous function.

2.1-2 Definition. Let $F:\bar{C}\subset R^n \to R^n$ be continuous on the bounded open set C. Define $||F||_{\bar{C}} \equiv \sup_{x \notin C} |F(x)|$ where $|\cdot|$ is the Euclidean norm. Then for $y \notin F(\partial C)$ one defines the degree of F at y with respect to C by

$$\deg(F,C,y) = \lim_{j\to\infty} \deg(F_j,C,y)$$

where $\{F_j\}$ is a sequence of maps which are continuously differentiable on an open set $D \supset \bar{C}$ and which satisfy $\lim_{j\to\infty} ||F_j - F||_C = 0$

2.2 Properties of the Degree

The principal properties of the degree are given below. Excellent proofs may be found in [S], [O] and [H].

2.2-1 Theorem. Let $F:\bar{C}\subset R^n \to R^n$ be continuous on the open bounded set C and let $\Gamma = \{x \in C | F(x)=y\}$. For any $y \notin F(\partial C)$ there exists a quantity, $\deg(F,C,y)$, which has the properties listed below. It is:

1. Integer valued
2. Invariant under homotopy

If $W:\bar{C}\times[0,1]\subset R^{n+1}\to R^n$ is continuous, then for any $z\in R^n$ satisfying $W(x,t)\neq z$ whenever $(x,t)\in \partial C\times[0,1]$, $\deg(W(\cdot,t),C,z)$ is constant for all $t\in[0,1]$.

3. <u>Dependent only on boundary values</u>

   If $G:\bar{C}\subset R^n\to R^n$ is continuous and $G|_{\partial C} = F|_{\partial C}$, then $\deg(F,C,y)=\deg(G,C,y)$.

4. <u>Invariant under translation</u>

   For any $z\in R^n$.
   $\deg(F-z,C,y-z)=\deg(F,C,y)$.

5. <u>Invariant for points which can be connected by a continuous path avoiding $F(\partial C)$</u>

   See Figure 1.

6. <u>Invariant under the excision from C of any closed set Q satisfying $Q\cap\Gamma = \emptyset$</u>

   In other words, if $Q\cap\Gamma=\emptyset$, then $\deg(F,C,y) = \deg(F,C-Q,y)$. In particular, if $Q=\bar{C}$, $\deg(F,C-Q,y)=0$. This property will be called the Excision Property.

The Excision Property can be used to prove a very important result which is called the Kronecker Theorem in [0, p. 161].

<u>2.2-2</u> <u>Theorem</u> (Kronecker). If $F:\bar{C}\subset R^n\to R^n$ is continuous on the bounded open set C, $y\notin F(\partial C)$ and $\deg(F,C,y)\neq 0$, then the equation $F(x)=y$ has a solution in C.

<u>Proof</u>: Suppose F has no solutions in C. Let $Q=\bar{C}$. Since $y\notin F(Q)$, the Excision Property implies $\deg(F,C,y)=0$.

Q.E.D.

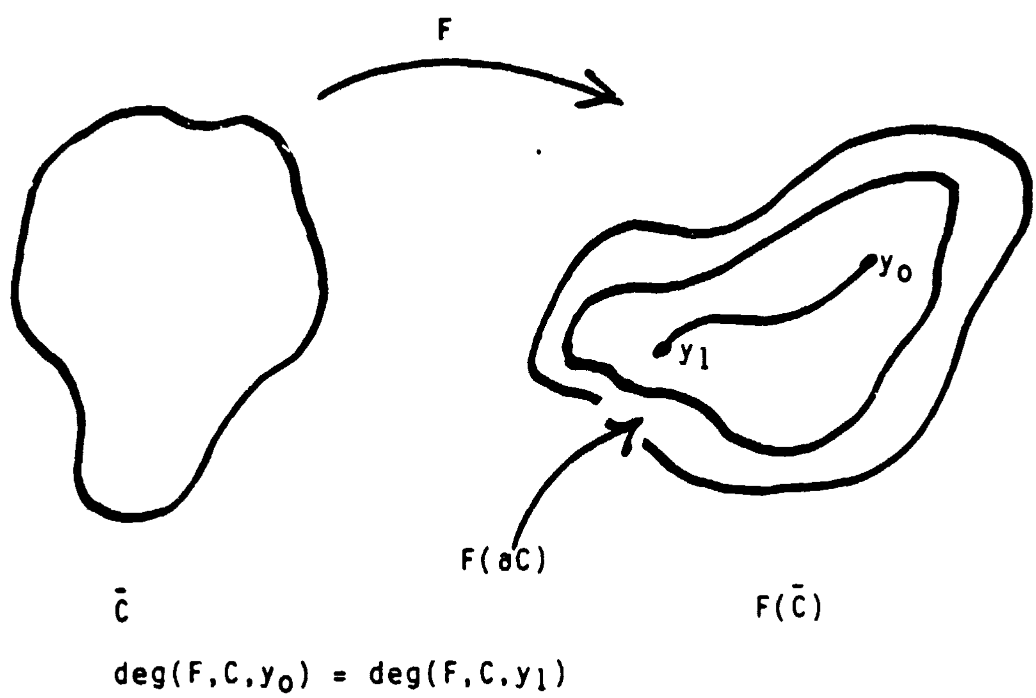

Figure 1. Invariance of the degree when points connected by a continuous path avoiding $F(\partial C)$.

2.3 A Topological Definition of the Degree

Dugundji [D] presents an alternate formulation for the degree of a mapping. He defines the degree of a mapping $f:S \to S$ where S is the unit n-sphere in R^n, that is,

$$S = \{x \in R^n \mid |x| = 1\}.$$

This degree can be shown to be equivalent to the analytically defined degree in the previous sections.

Before defining this degree, several terms must be discussed.

2.3-1 Definition. A set $E \subset R^n$ is called a linear variety if $x_1, x_2 \in E$ implies $\lambda x_1 + (1-\lambda)x_2 \in E$ for all real λ.

2.3-2 Definition. A hyperplane in R^n is an (n-1) dimensional linear variety. If $n=1$ then a hyperplane will be a point. For $n=2$ it will be a line, and for $n=3$ it is a plane.

2.3-3 Definition. If $\{x_0, x_1, \ldots, x_n\}$ is a set of $n+1$ points in R^n, then the convex hull is called an n-simplex. It will be denoted by $\sigma = (x_0, x_1, \ldots, x_n)$.

The points $x_0, x_1, \ldots, x_n$ are called the vertices of the n-simplex. If the vertices lie on a hyperplane in R^n, then the n-simplex is said to be degenerate. Now if $(x_i^1, \ldots, x_i^n)$ are the coordinates of point x_i, then the volume of an n-simplex [F, p. 208] is given by

$$\frac{1}{n!}\left|\det(x_1-x_0,x_2-x_0,\ldots,x_n-x_0)\right|$$

$$= \frac{1}{n!}\left|\det\begin{bmatrix} x_1^1-x_0^1 & x_2^1-x_0^1 & \cdots & x_n^1-x_0^1 \\ \vdots & \vdots & & \vdots \\ x_1^n-x_0^n & x_2^n-x_0^n & \cdots & x_n^n-x_0^n \end{bmatrix}\right|$$

An n-simplex is degenerate if and only if $\det(x_1-x_0,x_2-x_0,\ldots,x_n-x_0)=0$.

The next three definitions will be used to explain the term "ordered n-simplex."

<u>2.3-4</u> <u>Definition</u>. A binary relation Λ in a set A is a subset $\Lambda \subset A\times A$.

<u>2.3-5</u> <u>Definition</u>. If Λ is a binary relation in a set A, then Λ is trichotomous if exactly one of the following is true for each $x,y\varepsilon A$:

$$x\Lambda y, \quad x=y, \quad y\Lambda x.$$

<u>2.3-6</u> <u>Definition</u>. Let Λ be a binary relation in a set A. Then Λ is a total order if it is transitive and trichotomous [G, p. 2].

<u>2.3-7</u> <u>Definition</u>. An <u>ordered</u> <u>n-simplex</u> [D, p. 336] is an n-simplex together with a total ordering on its vertices.

Therefore, if the vertices $x_0,x_1,\ldots,x_n$ of an n-simplex satisfy $x_0<x_1<\ldots<x_n$, then "<" totally orders the set $\{x_0,x_1,\ldots,x_n\}$. Therefore, the n-simplex $\delta = (x_0,x_1,\ldots,x_n)$

is an ordered n-simplex. Such a simplex will be denoted $[\sigma] = [x_0, x_1, \ldots, x_n]$. The sign of the ordered simplex is the sign of $\det(x_1 - x_0, x_2 - x_0, \ldots, x_n - x_0)$.

Now suppose $x_0, x_1, \ldots, x_{n-1}$ is a set of n points on S having a diameter less than 1 so that the convex hull of the set does not contain the origin. Then the convex hull can be projected onto S by choosing the points on S lying on the directed rays which start at the origin and pass through the convex hull. The points on S form what will be called the <u>spherical</u> <u>(n-1)</u> - <u>simplex</u> $\sigma = (x_0, \ldots, x_{n-1})$. The spherical simplex σ is degenerate if and only if $x_0, x_1, \ldots, x_{n-1}$, lie on a hyperplane in R^n passing through the origin, that is, if and only if $(x_0, x_1, \ldots, x_{n-1}, 0)$ is a degenerate n-simplex in R^n. An <u>ordered</u> <u>spherical</u> <u>(n-1)</u> - <u>simplex</u> is a spherical (n-1)-simplex with a total order on its vertices. The <u>sign</u> of an ordered spherical (n-1)-simplex $[\sigma] = [x_0, \ldots, x_{n-1}]$ is defined to be the sign of the n-simplex $[x_0, \ldots, x_{n-1}, 0]$ in R^n [D, p. 337].

The next two definitions, which can be found in [D, p. 337], complete the terminology needed to define the Dugundji degree.

<u>2.3-8</u> <u>Definition</u>. A <u>triangulation</u> Δ of S is a decomposition of S into a finite number of nonoverlapping, nondegenerate spherical (n-1)-simplexes such that each face of an (n-1)-simplex is the common face of exactly two (n-1)-simplexes.

<u>2.3-9</u> <u>Definition</u>. Suppose S and Σ are unit n-spheres in R^n. (Different symbols are used to make the concepts more clear.) Let Δ be a triangulation of S. A <u>proper</u> <u>vertex</u> <u>map</u> $\varphi:\Delta \to \Sigma$ is a map defined only on the vertices of the spherical (n-1)-simplexes in Δ and is such that whenever $x_0, x_1, \ldots, x_{n-1}$ are vertices of a simplex in Δ, the set $\{\varphi(x_0), \varphi(x_1), \ldots, \varphi(x_{n-1})\} \subset \Sigma$ has diameter less than 1.

Under the proper vertex map $\varphi:\Delta \to \Sigma$ there will be a unique simplex $\varphi(\sigma)$ lying on Σ corresponding to each simplex $\sigma \in \Delta$. There will be a unique ordered (n-1)-simplex $\varphi[\sigma] = [\varphi(x_0), \varphi(x_1), \ldots, \varphi(x_{n-1})]$ on Σ corresponding to each ordered (n-1)-spherical simplex $[\sigma]$. The sign of $[\sigma]$ may differ from that of $\varphi[\sigma]$, and the family of sets $\{\varphi(\sigma) | \sigma \in \Delta\}$ may not form a triangulation of Σ since it may contain overlapping simplexes and degenerate simplexes. However, the family does have the fundamental property presented in the following theorem which Dugundji proves [D, p. 237].

<u>2.3-10</u> <u>Theorem</u>. Suppose Δ is a triangulation of S and $\varphi:\Delta \to \Sigma$ a proper vertex map. Let y be any point not on the boundary of any set $\varphi(\sigma)$. If $p(y,\Delta,\varphi)$ is the number of positive $\varphi[\sigma]$ containing y and $n(y,\Delta,\varphi)$ is the number of negative, then the number $D(y,\Delta,\varphi) = p(y,\Delta,\varphi) - n(y,\Delta,\varphi)$ is the same for all $y \in \Sigma$ not on the boundary of any $\varphi(\sigma)$.

Since $D(y,\Delta,\varphi)$ is independent of y it can be denoted $D(\Delta,\varphi)$.

Now if $F:S\to\Sigma$ is continuous then the compactness of S makes it possible to find a triangulation Δ of S such that the diameter of $F(\sigma)$ is less than 1 for each $\sigma\epsilon\Delta$. Then if $\varphi_F:\Delta\to\Sigma$ is the proper vertex map defined by $\varphi_F(x) = F(x)$ for each vertex x of Δ, Dugundji [D, p. 339] shows that the number $D(\Delta,\varphi_F)$, where φ_F is the proper vertex map associated with Δ, is independent of the triangulation of S. He calls the quantity $D(\Delta,\varphi_F)$ the degree of F. Since Δ and φ_F actually depend only on F, $D(\Delta,\varphi_F)$ can be denoted D(F).

Like the analytically defined degree, this degree is invariant under homotopy [D, p. 239].

__2.3-11__ __Theorem__. If $F:S\to\Sigma$ is homotopic to $\tilde{F}:S\to\Sigma$, then $D(F) = D(\tilde{F})$.

Now let V be the unit n-ball in R^n, that is, $V = \{x\epsilon R^n \mid |x|\leq 1\}$. Dugundji's degree can be extended to a continuous map $H:V\to V$ provided $H|_S$ maps S into S. S is clearly the boundary of V. Dugundji calls such maps __regular__. The technique for determining the degree of H is analogous to what is done to obtain D(F) for $F:S\to S$. V is triangulated into n-simplexes such that each face not on S is the face of exactly two n-simplexes. Then a __regular vertex map__ is defined on the triangulation. φ is a regular vertex map of a triangulation Δ of V if φ maps each vertex on S to a point on S and $\varphi|_S$ is a proper vertex map. To calculate the degree of H, which will be denoted $D_{reg}(H)$, one chooses the regular vertex map $\varphi_H:\Delta \to V$ defined by $\varphi_H(x) = H(x)$ for any vertex $x \in \Delta$. Then choosing $y \in V-S$ such that y is not on the boundary of any $\varphi_H(\sigma)$ one computes

$$D_{reg}(H) = (\text{number of positive } \varphi_H[\sigma] \text{ containing } y) - (\text{number of negative } \varphi_H[\sigma] \text{ containing } y).$$

$D_{reg}(H)$ depends only on H, and if H and $\tilde{H}$ are homotopic in such a way that the image of S remains on S throughout the entire deformation, then $D_{reg}(H) = D_{reg}(\tilde{H})$. Furthermore, Dugundji proves the following very useful result.

<u>2.3-12</u> <u>Theorem</u>. Suppose $H:V \to V$ is a regular map. Let $F=H|_S:S \to S$. Then $D(F) = D_{reg}(H)$.

This theorem provides the information needed to show that Dugundji's degree is equivalent to the analytically defined degree. The following lemma will be used in the proof.

<u>2.3-13</u> <u>Lemma</u>. Suppose the following hypotheses are given:

1. $H:V \to V$ is a continuously differentiable regular map
2. $y \in V-S$ and $\Gamma = \{x \in V | H(x)=y\}$
3. $H'(x)$ is nonsingular for all $x \in \Gamma$

Then there exists a triangulation Δ of V with associated regular vertex map φ_H such that whenever $\sigma \in \Delta$ contains $x \in \Gamma$ and $\varphi_H[\sigma]$ is nondegenerate,

$$\text{sign } \varphi_H[\sigma] = \text{sign det } H'(x).$$

Proof: For each $x \in \Gamma$ choose a neighborhood N_x of x so that either sign det $H'(p)>0$ for all points $p \in N_x$ or sign det $H'(p)<0$ for all points $p \in N_x$. Choose the neighborhoods small enough so that the family of sets $\{N_x | x \in \Gamma\}$ is disjoint. Then triangulate V so that each N_x contains a nondegenerate equilateral simplex σ_x in which x lies, that is, a nondegenerate simplex in which the distance between any two vertices is the same. Call this triangulation Δ. Now suppose $\sigma_x \in \Delta$ and $\sigma_x = (x_0, x_1, \ldots, x_n)$ with the vertices labeled so that $[\sigma_x] = [x_0, x_1, \ldots, x_n]$ is positive. Since $\varphi_H(x_i) = H(x_i)$, $i = 0, 1, \ldots, n$, the sign of $\varphi_H [\sigma_x]$

$$= \text{sign det } [H(x_1) - H(x_0), \ldots, H(x_n) - H(x_0)].$$

However, $H(x_i) - H(x_0) = H'(x_0)(x_i - x_0) + o(|x_i - x_0|)$ for $i = 1, \ldots, n$. Therefore, the sign of $\varphi_H [\sigma_x]$ = sign det $[H'(x_0)(x_1-x_0) + o(|x_1-x_0|), \ldots, H'(x_0)(x_n-x_0) + o(x_n-x_0)]$. Now if $|x_i - x_0| = \varepsilon$ for $i = o, 1, \ldots, n$ then expanding the determinant above yields det $[H'(x_0)(x_1-x_0, \ldots, x_n-x_0)]$ plus terms of order $o(\varepsilon^n)$. Therefore, det $[H'(x_0)(x_1-x_0, \ldots, x_n-x_0)]$, which has order $O(\varepsilon^n)$, is the dominant term, and the other terms can be neglected. Consequently, the sign of

$$\varphi_H[\sigma_x] = \text{sign det } [H'(x_0)(x_1-x_0, \ldots, x_n-x_0)]$$
$$= \text{sign } [(\det H'(x_0)) \cdot (\det (x_1-x_0, \ldots, x_n-x_0)]$$
$$= \text{sign det } H'(x_0)$$ since $[\sigma_x]$ is positive. However,

since $x_0 \in N_x$, sign det $H'(x_0)$ = sign det $H'(x)$.

Q.E.D.

<u>2.3-14</u> <u>Theorem</u>. Suppose $H:V \to V$ is a continuously differentiable regular map. Pick $y \in V-S$ and let $\Gamma = \{x \in V | H(x) = y\}$. Let $F = H|_S : S \to S$. If $H'(x)$ is nonsingular for all $x \in \Gamma$ then

$$D(F) = \deg(H, V, y).$$

<u>Proof</u>: Since 2.3-12 says $D(F) = D_{reg}(H)$, it suffices to show that $D_{reg}(H) = \deg(H, V, y)$.

Choose a triangulation Δ of V as specified in 2.3-13 and let φ_H be the regular vertex map associated with Δ. Without loss of generality, one can assume that $\varphi_H(\sigma)$ is nondegenerate for all $\sigma \in \Delta$ because any degenerate $\varphi_H(\sigma)$ can be approximated by a nondegenerate simplex $\varphi(\sigma)$ where φ is defined on all vertices p in Δ so that $|\varphi(p) - \varphi_H(p)| < \varepsilon$ for a given ε. According to Dugundji [D, p. 338], $D(\Delta, \varphi) = D(\Delta, \varphi_H) = D(H)$ if ε is sufficiently small.

Furthermore, one can also assume that y does not lie on the boundary of any $\varphi_H(\sigma)$ for $\sigma \in \Delta$ since property number 5 of Section 2.2 implies $\deg(H, V, y) = \deg(H, V, \rho)$ for all $\rho \in V-S$.

Therefore, it follows from 2.3-13 that $D_{reg}(H)$ = number of positive $\varphi_H(\sigma)$ containing y

- number of negative $\varphi_H(\sigma)$ containing y

$$= \sum_{x \in \Gamma} \operatorname{sign} \det H'(x).$$

Q.E.D.

The following corollary shows that the restriction that $H'(x)$ be nonsingular for all $x \in \Gamma$ can be removed.

<u>2.3-15</u> <u>Corollary</u>. Suppose $H:V \to V$ is a continuously differentiable regular map. Pick $y \in V-S$ and let $\Gamma = \{x \in V \mid H(x)=y\}$. Let $F=H|_S : S \to S$. Then $D(F)=\deg(H,V,y)$.

<u>Proof</u>: By Ortega and Rheinboldt [0, p. 159], there exists a sequence $\{y_k\}$ which converges to y and has the following properties:

1. Each $y_k \notin H(S)$
2. For each y_k, $\det H'(x) \neq 0$ for all x such that $H(x)=y_k$
3. For some k_0, $\deg(H,V,y)=\deg(H,V,y_k)$ for all $k \geq k_0$

So pick k^* so that $k^* \geq k_0$ and whenever $k \geq k^*$, $y_k \in V-S$. Then by 2.3-14 $D(F)=\deg(H,V,y_k{}^*)=\deg(H,V,y)$.

Q.E.D.

It should be noted that this corollary still holds if H maps some of the interior points outside of V. The points outside of V can be projected onto S so that one obtains a mapping from V into V.

2.4 Applications to Grid Generation

The usefulness of degree theory in grid generation surfaces when one studies a grid generating transformation T. One might immediately note from the Kronecker theorem that determining the degree at every point in the physical domain would show whether or not T were onto. Unfortunately, the degree is not always easy to compute in practice.

One therefore looks instead at how the degree can be used to prove some things about those quantities, such as the Jacobian of T, which can be easily computed.

Recall that if $A \subset R^n$, $B \subset R^n$, then A homeomorphic to B means there is a continuous one to one, onto mapping from A to B whose inverse is also continuous. It is clear that T should be a homeomorphism from the computational domain onto the physical domain.

The following result shows that if T is a homeomorphism, its Jacobian does not change sign.

In all of the theorems which follow $I_n = [0,1]^n$, JT = Jacobian of T, C^0=interior of C, C^c=complement of C, ∂C = boundary of C and $\Omega \subset R^n$ is homeomorphic to I_n.

2.4-1 Theorem. If T is a homeomorphism from I_n to Ω and T is continuously differentiable, then the Jacobian, JT, of T has one sign in I_n^0, i.e., either $JT(x) \geq 0$ for all $x \in I_n^0$ or $JT(x) \leq 0$ for all $x \in I_n^0$.

Proof: Suppose by way of contradiction that $JT(x_0) > 0$ while $JT(x_1) < 0$ for some $x_0, x_1 \in I_n^0$. Let $y_0 = T(x_0)$ and $y_1 = T(x_1)$. Define $p:[0,1] \to R^n$ by

$$p(t) = T\,(\,(1-t)x_0 + tx_1).$$

Then $p(0)=y_0$, $p(1)=y_1$, and $p(t) \notin T(\partial I_n)$ for $t \in [0,1]$. Hence, by property 5, $1 = \deg(T, I_n^0, y_0) = \deg(T, I_n^0, y_1) = -1$. Therefore, either $JT(x) \geq 0$ or $JT(x) \leq 0$ for all $x \in I_n^0$.

Q.E.D.

In an algebraic grid generation algorithm, the construction of T will be based on boundary information. The next theorem shows that requiring T to be a homemorphism from the boundary of the computational domain to the boundary of the physical domain will insure that the image of T covers all of the physical domain.

2.4-2 Theorem. If $T: I_n \to R^n$ is continuously differentiable and T maps ∂I_n homeomorphically onto $\partial\Omega$, then $T(I_n) \supset \Omega$.

Proof: Let S be the unit n-sphere in R^n. By Dugundji [D, p. 353], the Dugundji degree D of a map which is a homeomorphism from S to S is +1 or -1. But ∂I_n and $\partial\Omega$ are homeomorphic to S. Therefore, from 2.3-15 it follows that for any $y \in \Omega^o$, $\deg(T, i_n^c, y) = \pm 1$. Therefore, by the Kronecker Theorem (2.2-2) Ω lies in the image of T.

Q.E.D.

Smith and Sritharan [SS] show that if an additional hypothesis is added, one can obtain a much stronger conclusion:

2.4-3 Theorem. If $T: I_n \to R^n$ is continuously differentiable, T maps ∂I_n homeomorphically onto $\partial\Omega$ and $JT(x) \neq 0$ for all $x \in I_n^o$, then T is a homeomorphism from I_n to Ω.

The next theorem shows that the Jacobian changes sign when the image of T overlaps the physical boundary. Theorem 2.4-3 and Theorem 2.4-4 show that it is important that T be constructed so that its Jacobian does not change sign.

<u>2.4-4 Theorem</u>. Suppose $T: I_n \to R^n$ has the following properties:

1. T is continuously differentiable
2. T maps ∂I_n homeomorphically onto $\partial\Omega$
3. $m(T(I_n) - \Omega) > 0$

Then JT has a sign change.

<u>Proof</u>: Let $C(I_n) = \{x \in I_n \mid JT(x) = 0\}$ Sard's Theorem [O, p. 130] says that $m(T(C(I_n))) = 0$. Since $m(T(I_n) - \Omega) > 0$, there exists $z^* \in T(I_n) - \Omega$ such that $T(x) = z^*$ implies $JT(x) \neq 0$. Now, choose $w \in [T(I_n)]^c$. By property 5, $\deg(T, I_n^o, z^*) = \deg(T, I_n^o, w) = 0$. Since $\deg(T, I_n^o, z^*) = \sum_{\{x \mid T(x) = z^*\}} \operatorname{sign} JT(x)$, the Jacobian values at all x satisfying $T(x) = z^*$ must cancel each other.

Q.E.D.

2.5 <u>Additional Topological Questions</u>

Section 2.4 suggests other questions which should be asked. Can a continuously differentiable homeomorphism from ∂I_n to $\partial\Omega$ always be extended to a continuously differentiable homeomorphism from I_n to Ω? If not, under what conditions is such an extension possible? How can one guarantee that a mapping from I_n to R^n will be a diffeomorphism?

The answers to these questions will provide valuable information for creating an algebraic grid generation mapping. Although this paper does not answer all of these questions, partial answers were presented in the previous section. Also, the following example shows that continuously differentiable boundary homeomorphisms cannot always be extended.

2.5-1 Example. Suppose $T: I_2 \to R^2$ is continuously differentiable and maps the boundary of the square homeomorphically onto the boundary of the nonconvex region Ω shown in figure 2. Let p be the point indicated and $\Delta p = \begin{pmatrix} \Delta\xi \\ -\Delta\eta \end{pmatrix}$. If $T_1(p) = \frac{\partial T}{\partial \xi}(p)$ and $T_2(p) = \frac{\partial T}{\partial \eta}(p)$ then

$$T(p+\Delta p) = T(p) + T'(p)(\Delta p) + o(|\Delta p|)$$
$$= T(p) + \Delta\xi T_1(p) - \Delta\eta T_2(p) + o(|\Delta p|)$$

When $|\Delta p|$ is small, the terms of order $o(|\Delta p|)$ are negligible in size when compared to $\Delta\xi T_1(p)$ and $\Delta\eta T_2(p)$. Therefore, those terms may be neglected from the equation above. However, then it is clear that $T(p+\Delta p)$ must lie outside the boundary of Ω. Consequently, T cannot be a homeomorphism from I_2 to Ω. This is illustrated in figures 3 and 4 which show the result of attempts to construct a tensor product spline transformation that maps the square onto Ω. In each case points overlap the boundary near the "V" shaped corner.

The first grid was obtained by choosing the B-spline coefficients so that the transformation approximated a transfinite bilinear interpolation mapping. This is discussed in Chapter 4. The second grid was obtained by changing some of the coefficients in order to minimize a functional which is described in the next chapter.

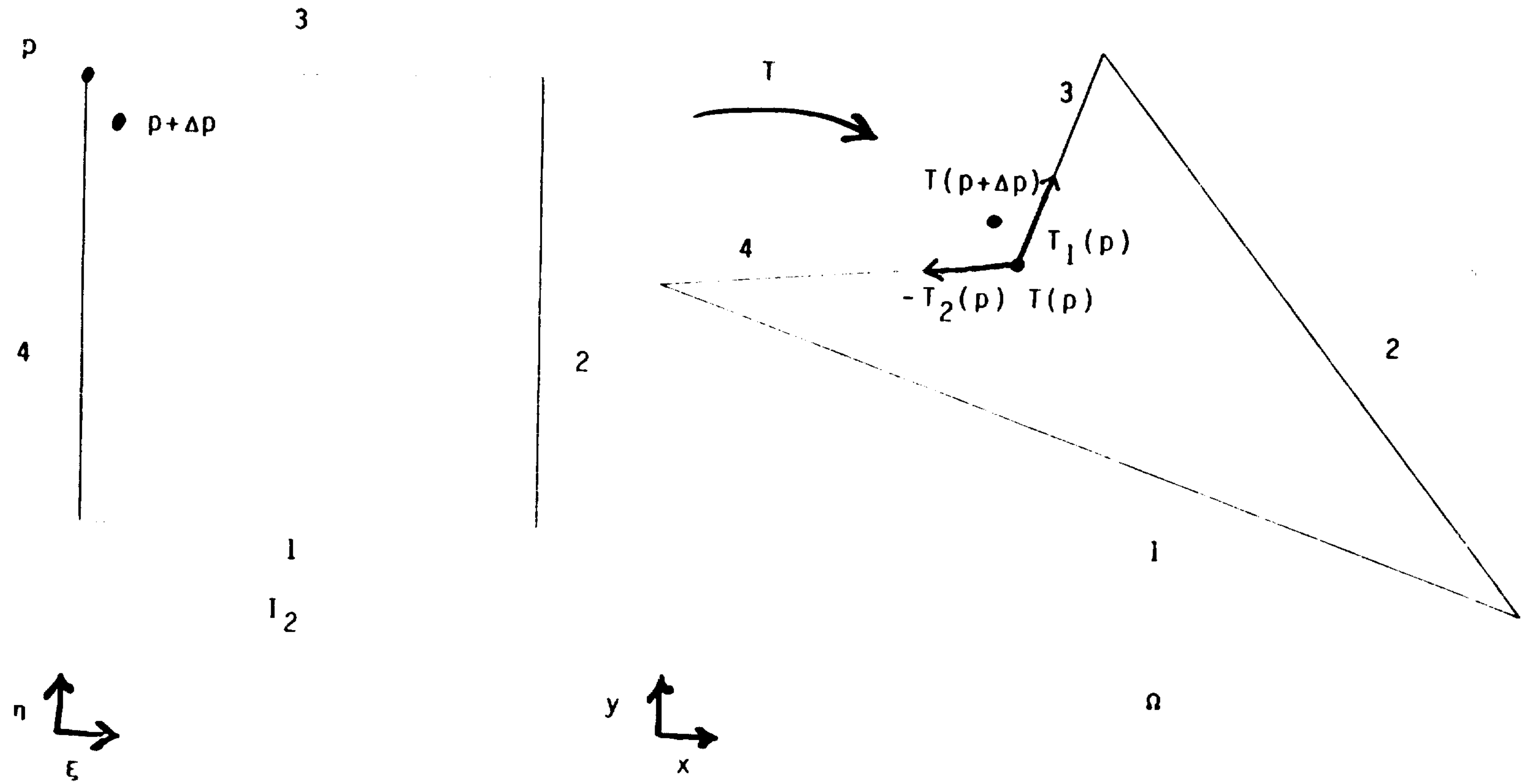

Figure 2. Mapping the square onto a nonconvex region.

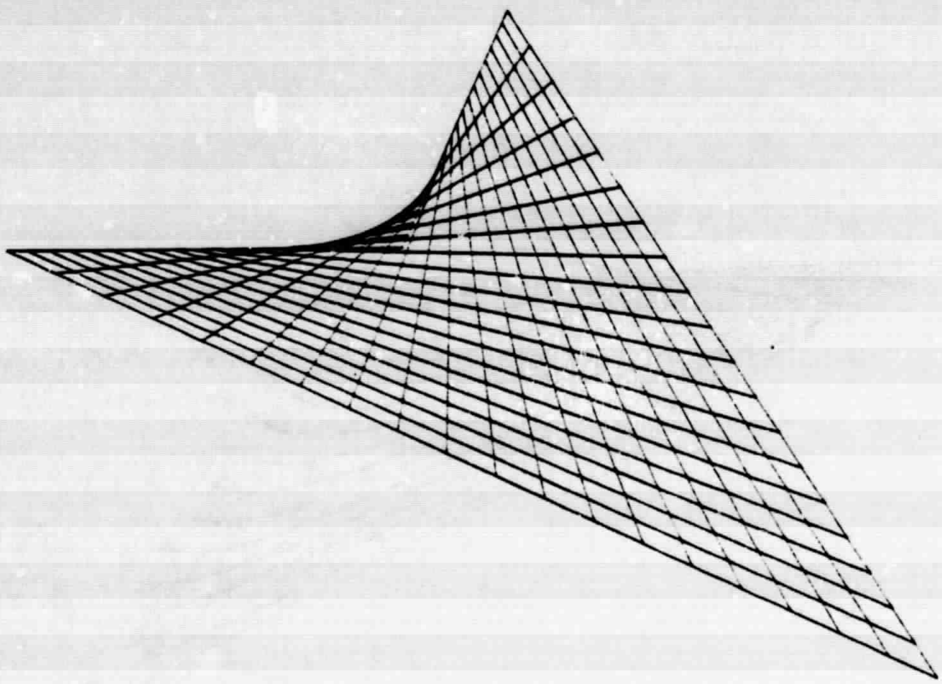

Figure 3. Initial grid on nonconvex domain.

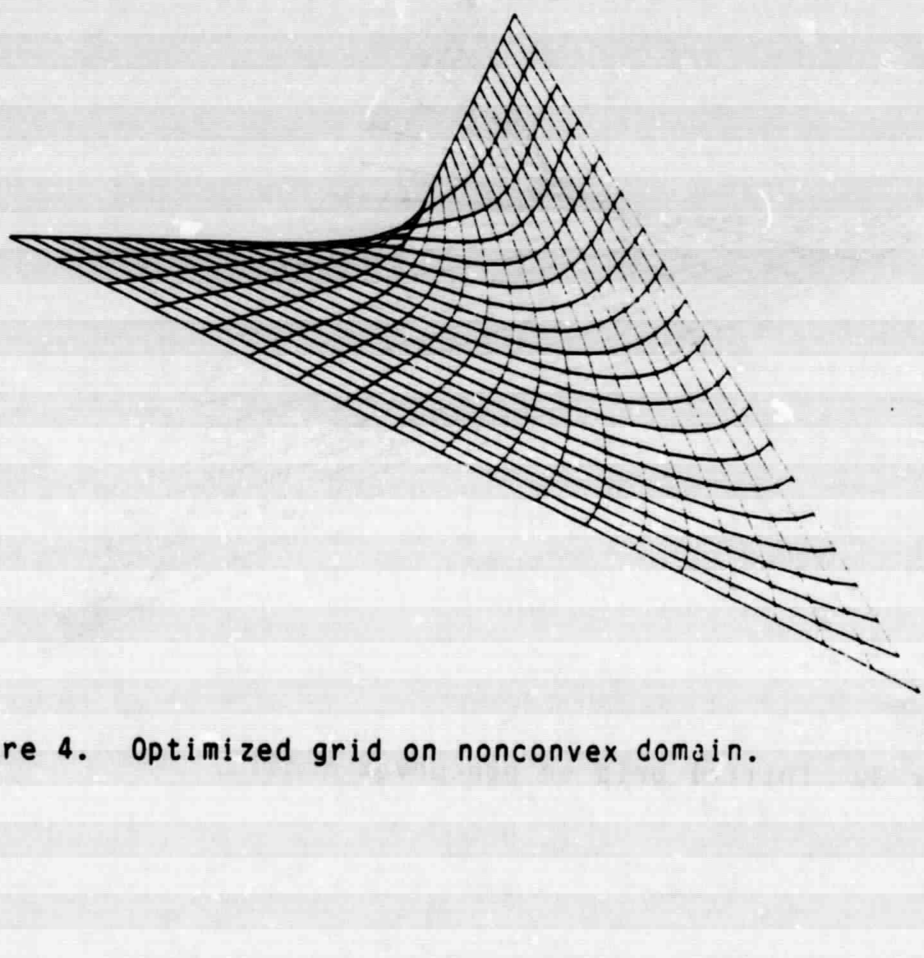

Figure 4. Optimized grid on nonconvex domain.

3. AN ALGEBRAIC GRID GENERATION MAPPING

In this chapter an algebraic grid generation technique which uses a transformation consisting of tensor product B-splines is discussed. In the first section, finite difference approximations to the transformed derivatives of a first order partial differential equation are examined. The effect of the size of the Jacobian on smoothness and orthogonality is discussed, and its influence on local truncation error is examined. The next section defines the particular transformation of interest in this paper and discusses the properties of the building blocks for this transformation: kth order B-splines. The final section discusses a functional which can be used to modify the transformation so that the grid lines are distributed more smoothly and are nearly orthogonal at points of intersection.

3.1 A First Order Example

If ξ and η are the computational coordinates, satisfying $0\leq\xi\leq 1$ and $0\leq\eta\leq 1$, and x and y are the physical coordinates, then the grid on the physical domain will consist of coordinate lines produced by a mapping

$$T(\xi,\eta) = \begin{pmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{pmatrix} .$$

If $u_t = F(x,y,u,u_x,u_y)$ is a first order partial differential equation defined on the physical domain, then the chain rule yields $(u_\xi \ u_\eta) = (u_x \ u_y) \times J$ where $J = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}$, the Jacobian matrix for the transformation T. Hence

$$(u_x \ u_y) = (u_\xi \ u_\eta) \times J^{-1}$$

$$= (u_\xi \ u_\eta) \times \begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix} \Big/ JT$$

where $JT = |J| = x_\xi y_\eta - x_\eta y_\xi$. It is clear that the partial differential equation can be transformed once the elements of J are computed. These elements may be approximated by differences when explicit formulas are not available. The transformed expressions for u_x and u_y show immediately that the grid must be structured so that $JT \neq 0$ at all mesh points (ξ,η).

Once the partial differential equations are transformed, difference approximations can be written for u_ξ and u_η. Large truncation errors in the approximations will affect the solution of the partial differential equations. One can obtain an expression for the truncation error at mesh point (ξ_i,η_j) by doing a Taylor series expansion at (ξ_i,η_j). If $u_{ij} = u(\xi_i,\eta_j)$, then

$$u_{i+1,j} = u_{ij} + u_\xi \Delta\xi + u_{\xi\xi} \frac{(\Delta\xi)^2}{2!} + u_{\xi\xi\xi} \frac{(\Delta\xi)^3}{3!} + \text{HOT}$$

$$u_{i-1,j} = u_{ij} - u_\xi \Delta\xi + u_{\xi\xi} \frac{(\Delta\xi)^2}{2!} - u_{\xi\xi\xi} \frac{(\Delta\xi)^3}{3!} + \text{HOT}$$

where HOT = higher order terms. Subtracting these two equations and solving for u_ξ yields

$$u_\xi = \frac{u_{i+1,j} - u_{i-1,j}}{2\,\Delta\xi} - u_{\xi\xi\xi}\frac{(\Delta\xi)^2}{6} + \text{HOT}$$

Similarly,

$$u_\eta = \frac{u_{i,j+1} - u_{i,j-1}}{2\,\Delta\eta} - u_{\eta\eta\eta}\frac{(\Delta\eta)^2}{6} + \text{HOT}$$

Therefore

$$u_x = \frac{1}{JT}\,(y_\eta\delta_\xi u - y_\xi\delta_\eta u) - \frac{1}{6JT}\,(y_\eta u_{\xi\xi\xi}(\Delta\xi)^2 - y_\xi u_{\eta\eta\eta}(\Delta\eta)^2)$$

$$+ \dots$$

where $\delta_\xi u$ and $\delta_\eta u$ are the central difference approximations for u_ξ and u , respectively. The truncation error is

$$\frac{-1}{6JT}\,(y_\eta u_{\xi\xi\xi}(\Delta_\xi)^2 - y_\xi u_{\eta\eta\eta}(\Delta_\eta)^2) + \dots$$

Now if $r = \binom{x}{y}$, then

$$JT = x_\xi y_\eta - x_\eta y_\xi$$

$$= (r_\xi x r_\eta) \cdot (0,0,1)^T$$

$$= |r_\xi|\ |r_\eta| \sin\theta$$

where θ is the angle of intersection of the grid lines at (ξ,η). Again, the importance of $JT \neq 0$ is evident, but one can also see why the grid lines should be as orthogonal as possible. The expression for JT implies that the truncation error is inversely proportional to $\sin\theta$. However, according to Thompson, Warsi and Mastin [TWM, p. 82] a departure from orthogonality of up to 45° is usually tolerable.

3.2 B-splines

The mapping T discussed in this paper has the form

$$T(\xi,\eta) = \begin{bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{ij} B_{ij}(\xi,\eta) \\ \\ \sum_{i=1}^{m} \sum_{j=1}^{n} \beta_{ij} B_{ij}(\xi,\eta) \end{bmatrix} \begin{matrix} 0 \le \xi \le 1 \\ \\ 0 \le \eta \le 1 \end{matrix}$$

where the B_{ij}, $i=1,\dots m$; $j=1,\dots,n$ are tensor products of B-splines and the coefficients α_{ij}, β_{ij}, $i=1,\dots,m$; $j=1,\dots,n$ are real numbers. In this section, the terms B-spline, spline function and tensor product B-spline are defined, and some of the important properties of these functions are discussed.

3.2-1 Defining B-splines

The following definition is from A Practical Guide to Splines by Carl de Boor [de B, p. 108].

3.2-1-1 Definition. If $t = \{t_i\}$ is a nondecreasing sequence, then the i-th normalized B-spline of order k for knot sequence t is defined by

$$B_{i,k,t}(x) = (t_{i+k}-t_i)[t_i,\dots,t_{i+k}](\cdot-x)_+^{k-1} \text{ where } x\in R.$$

The sequence t may be finite, infinite or biinfinite. The expression $[t_i,\dots,t_{i+k}](\cdot-x)_+^{k-1}$ denotes the kth divided difference of $(\cdot-x)_+^{k-1}$, or the leading coefficient of the polynomial of degree k which interpolates $(\cdot-x)_+^{k-1}$

at $t_i, \ldots, t_{i+k}$. The notation $(\cdot - x)_+^{k-1}$ represents the truncated power function $(\tau - x)_+^{k-1}$ which is defined by

$$(\tau - x)_+^{k-1} = \begin{cases} (\tau - x)^{k-1} & \text{for } \tau > x \\ 0 & \text{for } \tau \le x \end{cases} .$$

The $\cdot$ indicates that the kth divided difference above should be evaluated by holding x fixed and considering $(\tau - x)_+^{k-1}$ as a function of τ only. Nevertheless, since $B_{i,k,t}(x)$ changes as one chooses different values for x, it is clearly a function of x.

The definition above differs slightly from the original definition given by Curry and Schoenberg. Their B-spline $M_{i,k,t}$ is related to $B_{i,k,t}$ by the equation

$$M_{i,k,t} = [k/(t_{i+k} - t_i)]\ B_{i,k,t} \text{ [de B, p. 109].}$$

3.2.2 Properties of B-splines

A kth order B-spline $B_{i,k,t}$ is a piecewise polynomial of degree k-1 with breakpoints at $t_i, \ldots, t_{i+k}$. On each interval (t_j, t_{j+1}), $B_{i,k,t}$ is a polynomial of degree k-1 or less. For convenience it will be assumed that $B_{i,k,t}$ is continuous from the right at breakpoints.

B-splines have many properties which make them convenient for applications involving computers. One important property is their small support. If $x \notin [t_i, t_{i+k}]$, then $(\tau - x)_+^{k-1}$ will be a polynomial of degree k-1 or less on $[t_i, t_{i+k}]$. Hence $[t_i, \ldots, t_{i+k}]\,(\tau - x)_+^{k-1} = 0$. Therefore, $B_{i,k,t}(x) = 0$ for $x \notin [t_i, t_{i+k}]$.

This implies that the support of $B_{i,k,t}$ can lie in at most k intervals of the form $[t_j, t_{j+1}]$. Therefore, if $\{B_i\}$ represents the sequence of B-splines of order k for the knot sequence $t = \{t_i\}$, it follows that only the k B-splines $B_{j-k+1}, B_{j-k+2}, \ldots, B_j$ can have support in any given interval $[t_j, t_{j+1}]$.

The next two results, which are proved in [de B, p. 110] and [de B, p. 130], respectively, show that B-splines form a partition of unity, i.e., the sequence $\{B_i\}$ consists of nonnegative functions which sum up to 1.

3.2.2-1 Theorem. If $\{B_i\}$ is the sequence of B-splines of order k for a nondecreasing sequence $t = \{t_i\}$, then

$$\sum_i B_i(x) = \sum_{i=p-k+1}^{q-1} B_i(x) = 1$$

for any $x \in (t_p, t_q)$ where p and q are such that p-k+1 and q+k-1 lie in the index set for t.

3.2.2-2 Theorem. If B_i is the ith element of the sequence of B-splines of order k for a nondecreasing sequence $t = \{t_i\}$, then $B_i(x) > 0$ for $t_i < x < t_{i+k}$.

One can think of the "B" in B-splines as representing the word "basis," for when the knot sequence t is chosen appropriately, the kth order B-splines for t form a basis for the piecewise polynomial space $P_{k,\xi,\nu}$. $P_{k,\xi,\nu}$ is the notation used by de Boor [de B, p. 100] to represent the space of piecewise polynomials of degree k-1 which have breakpoint sequence ξ and which satisfy smoothness conditions specified by ν. If $\xi = (\xi_i)_1^{m+1}$, then the nonnegative sequence $\nu = (\nu_i)_2^m$ gives the number of smoothness conditions at each ξ_i, $i = 2,\ldots,m$. For example, if $\nu_\ell = 3$ then any $f \in P_{k,\xi,\nu}$ must have at least 3 smoothness conditions at ξ_ℓ, that is, the function, its derivative and second derivative must be continuous at ξ_ℓ. The dimension of $P_{k,\xi,\nu}$ is $km - \sum_{i=2}^{m} \nu_i$.

The following theorem of Curry and Schoenberg [de B,C] shows how the knot sequence t should be chosen so that the corresponding B-spline sequence forms a basis for $P_{k,\xi,\nu}$.

<u>3.2.2-3</u> <u>Theorem</u> (Curry and Schoenberg).

Let $\xi = (\xi_i)_1^{m+1}$ be a strictly increasing sequence and $\nu = (\nu_i)_2^m$ be a nonnegative integer sequence such that $\nu_i \le k$ for all i. Set $n = k + \sum_{i=2}^{m} (k - \nu_i) = km - \sum_{i=2}^{m} \nu_i$ and let $t = (t_i)_1^{n+k}$ be a nondecreasing sequence such that

(i) $t_1 \le t_2 \le \cdots \le t_k \le \xi_1$ and $\xi_{m+1} \le t_{n+1} \le \cdots \le t_{n+k}$

(ii) for $i=2,\ldots,m$, the number ξ_i occurs exactly $k-\nu_i$ times in t.

Then the sequence $B_1,\ldots,B_n$ of B-splines of order k for the knot sequence t is a basis for $P_{k,\xi,\nu,}$ viewed as functions on $[t_k,t_{n+1}]$.

This theorem shows how the number of knots at a breakpoint translates into the amount of smoothness there. Since the number ξ_i occurs exactly $k-\nu_i$ times in t and ν_i represents the number of smoothness conditions at ξ_i, the number of smoothness conditions at ξ_i equals k minus the number of knots at ξ_i. Hence if $k=4$ and ξ_j, $2\leq j\leq m$, occurs exactly once in t then the piecewise polynomials generated by $B_1,\ldots,B_n$ will satisfy three smoothness conditions at ξ_j, i.e., the piecewise polynomials, their first derivative and their second derivative will be continuous at ξ_j.

3.2.3 Spline Functions

In early studies of splines, a spline function of order k was defined to be a piecewise polynomial of degree k-1 with k-2 continuous derivatives. However, in this paper the more general definition in [de B] is used.

3.2.3-1 Definition. If $t = \{t_i\}$ is a nondecreasing sequence, then a spline function of order k with knot sequence t is any linear combination of the B-splines of order k for the knot sequence t. If one denotes the collection of all such functions by $S_{k,t}$ then

$$S_{k,t} = \{\sum_i \alpha_i \; B_{i,k,t} : \alpha_i \text{ real for all } i\}.$$

It is clear that when t has the form described in the Curry and Schoenberg theorem 3.2.2-3, $S_{k,t}=P_{k,\xi,\nu}$ on $[t_k,t_{n+1}]$.

The first derivative of a spline function $\sum_i \alpha_i \; B_{i,k,t}$ can be found by using the differences between successive coefficients. The following result, proved in [de B, p. 138], shows that the derivative of a spline function of order k will be a spline function of order k-1.

<u>3.2.3-2</u> <u>Theorem</u>. Let $\sum_i \alpha_i \; B_{i,k,t}$ be a kth order spline function constructed with B-splines $B_{i,k,t}$ corresponding to a nondecreasing sequence $t = \{t_i\}$. Then the first derivative of $\sum_i \alpha_i \; B_{i,k,t}$ is given by

$$\frac{d}{dx}\left(\sum_i \alpha_i \; B_{i,k,t}\right) = \sum_i (k-1) \; \frac{\alpha_i - \alpha_{i-1}}{t_{i+k-1} - t_i} \; B_{i,k-1,t}$$

The value of a spline function $f = \sum_j \alpha_j \; B_{j,k,t}$ at a point x satisfying $t_i<x<t_{i+1}$ is a convex combination of the k coefficients $\alpha_{i+1-k},\ldots,\alpha_i$. For if $t_i<x<t_{i+1}$, then $f(x) = \sum_j \alpha_j \; B_{j,k,t}(x) = \sum_{j=i-k+1}^{i} \alpha_j \; B_{j,k,t}(x)$ with the $B_{j,k,t}$ satisfying $\sum_j B_{j,k,t}(x)= 1$ and $B_k(x)\geq 0$ for all j.

B-spline coefficients model the functions that they represent. In other words, the coefficients are approximately equal to the value of the function at certain points. This is illustrated in the next section.

Carl de Boor [de B] proves the following result concerning the relationship between a spline function and its B-spline coefficients. The notation $||f||_{[a,b]}$ denotes $\max_{x\epsilon[a,b]} |f(x)|$.

<u>3.2.3-3</u> <u>Theorem</u>. Let $\sum_i \alpha_i B_{i,k,t}$ be a kth order spline function constructed with B-splines $B_{i,k,t}$ corresponding to a nondecreasing sequence $t = \{t_i\}$. Then there exists a positive constant D_k, depending only on k, so that for all i,

$$|\alpha_i| \leq D_k || \sum_j \alpha_j B_{j,k,t} ||_{[t_{i+1}, t_{i+k-1}]}$$

3.2.4 Variation Diminishing Splines

Given an f known to lie in $P_{k,\xi,\nu}$ one can write it in the form $f = \sum_{i=1}^{n} \alpha_i B_i$. The Curry and Schoenberg Theorem (3.2.2-3) shows how one obtains the B-spline basis and the following lemma suggests how one might obtain the coefficients. Its proof may be found in [de B, p. 116].

<u>3.2.4-1</u> <u>Lemma</u> (de Boor and Fix). Let B_i be the sequence of B-splines of order k for a nondecreasing sequence $t = \{t_i\}$. Let λ_i be the linear functional defined for all f by $\lambda_i f = \sum_{r=0}^{k-1} (-)^{k-1-r} \phi^{(k-1-r)}(\tau_i) f^{(r)}(\tau_i)$ where $\phi(t) = (t_{i+1}-t) \ldots (t_{i+k}-t)/(k-1)!$ and τ_i is some arbitrary point in the open interval (t_i, t_{i+k}). Then

$\lambda_i B_j = \sigma_{ij}$ for all j.

Hence, if $f = \sum_{i=1}^{n} \alpha_i B_i$ it follows that α_k, $1 \leq k \leq n$ may be found by computing $\lambda_k f = \lambda_k(\Sigma \alpha_i B_i) = \alpha_k$. By explicitly writing out the expression for $\lambda_k f$ one can easily show [de B, p. 159] that $\alpha_i = f(\tau_i) + O(|t|)$ if τ_i is any point in (t_i, t_{i+k}) and $|t| = \max_i \{t_{i+1}-t_i\}$. However, if $\tau_i = t_i^*$, $1 \leq i \leq n$ where $t_i^* = (t_{i+1} + \ldots + t_{i+k-1}) / (k-1)$ then $\alpha_i = f(t_i^*) + O(|t|^2)$. Choosing $\alpha_i = f(t_i)$ for $1 \leq i \leq n$ yields a shape preserving approximation called Schoenberg's <u>variation diminishing spline approximation</u> [de B, p. 159]. So if $t^* = \{t_i^*\}_1^n$ the variation diminishing spline approximation to f, vf, is defined by

$$vf = \sum_{i=1}^{n} f(t_i^*) B_i .$$

This spline reproduces polynomials of degree one, i.e., if f is a straight line then vf = f. For any f the number of times the spline approximation crosses a given line will be less than or equal to the number of times f crosses the line. From this it follows that if f is nonnegative, then vf is nonnegative and if f is convex then vf is convex. However, since v has these shape preserving properties, it is not a very high order approximation. In fact, if g is a function defined on [a,b] and g has m continuous derivatives for some $m \geq 2$, then de Boor [de B, p. 161] states that $||g-vg||_{[a,b]} \leq c_{g,k} |t|^2$, where $c_{g,k}$ is a constant depending on the order of the spline function k and the function g. No matter how large m is, no exponent larger than 2 can be put in the inequality. De Boor shows that it is possible to obtain other spline approximations which are more accurate, but variation diminishing splines are convenient for applications such as computer-aided design and grid generation where shape preservation is important.

3.2.5 Tensor Product B-splines

3.2.5-1 Definition. Let R be the set of real numbers. If V is a linear space of functions mapping some set X into R and W is a linear space of functions mapping some set Y into R, then for each $v \in V$ and $w \in W$ the tensor product, $v \otimes w$ of v and w is defined by

$$v \otimes w(x,y) = v(x)w(y) \text{ for } (x,y) \in X \times Y.$$

Furthermore, the set of all finite linear combinations of the form $v \otimes w$ for some $v \in V$ and $w \in W$ is called the tensor product, $V \otimes W$ of V with W.

A typical element u of $V \otimes W$ has the form

$$u = \sum_{j=1}^{n} \alpha_j (v_j \otimes w_j)$$

where $\alpha_j \in R$, $v_j \in V$, $w_j \in W$ for $j=1,\dots,n$.

If V and W are the linear spaces of spline functions $S_{h,s}$ and $S_{k,t}$, respectively, then the elements of $V \otimes W$ are linear combinations of tensor product B-splines. A tensor product B-spline B_{ij} is defined by $B_{ij}(x,y)=B_{i,h,s}(x)B_{j,k,t}(y)$ where $B_{i,h,s}$ is the ith B-spline of order h for the knot sequence $s = \{s_i\}$ and $B_{j,k,t}$ is the jth B-spline of order k for the knot sequence $t = \{t_j\}$. An element u of $V \otimes W$ will be called a tensor product spline and will have the form

$$u = \sum_i \sum_j \alpha_{ij} \; B_{ij}$$

where $\alpha_{ij} \in R$ for all i,j. When h=k=4 u may also be called a bicubic spline.

Many of the properties of tensor product B-splines follow trivially from B-spline properties. For example, the tensor product B-spline B_{ij} will be positive on its support since both $B_{i,h,s}$ and $B_{j,k,t}$ are positive on their support. Furthermore, the support of B_{ij} is small. Since $B_{i,h,s}(x) = 0$ for $x \notin [s_i, s_{i+h}]$ and $B_{j,k,t}(y) = 0$ for $y \notin [t_j, t_{j+k}]$ it is clear that $B_{ij}(x,y) = 0$ if either $x \notin [s_i, s_{i+h}]$ or $y \notin [t_j, t_{j+k}]$. Hence the support of B_{ij} lies in the shaded area shown in figure 5.

Tensor product B-splines also form a partition of unity. It follows from 3.2.2-1 that $\sum_i\sum_j B_{ij}(x,y) = \sum_i B_i(x)\sum_j B_j(y) = 1$ for any $(x,y) \in (s_p,s_q)\times(t_r,t_m)$ where p and q are such that p-h+1 and q+h-1 lie in the index set for sequence s and r-k+1 and m+k-1 lie in the index set for sequence t.

Partial derivatives of tensor product splines are easy to compute since they reduce to derivatives of spline functions.

$$\frac{\partial}{\partial x}\left(\sum_{i}\sum_{j} \alpha_{ij} B_{ij}(x,y)\right) = \frac{\partial}{\partial x}\sum_{i}\sum_{j}\alpha_{ij} B_i(x)B_j(y)$$

$$= \sum_j B_j(y)\,\frac{d}{dx}\left(\sum_i \alpha_{ij} B_i(x)\right)$$

$$\frac{\partial}{\partial y}\left(\sum_{ij} \alpha_{ij} B_{ij}(x,y)\right) = \frac{\partial}{\partial y}\sum_{i}\sum_{j}\alpha_{ij} B_i(x)B_j(y)$$

$$= \sum_i B_i(x)\,\frac{d}{dy}\left(\sum_j \alpha_{ij} B_j(y)\right)$$

3.3 A Smoothing Functional

The mapping T described in this paper uses tensor product B-splines to map the unit square onto a physical domain of arbitrary shape. This section shows that choosing the coefficients of the tensor product B-splines so that they minimize a certain functional can improve the quality of the physical grid produced by T. This functional is described and conditions under which it will

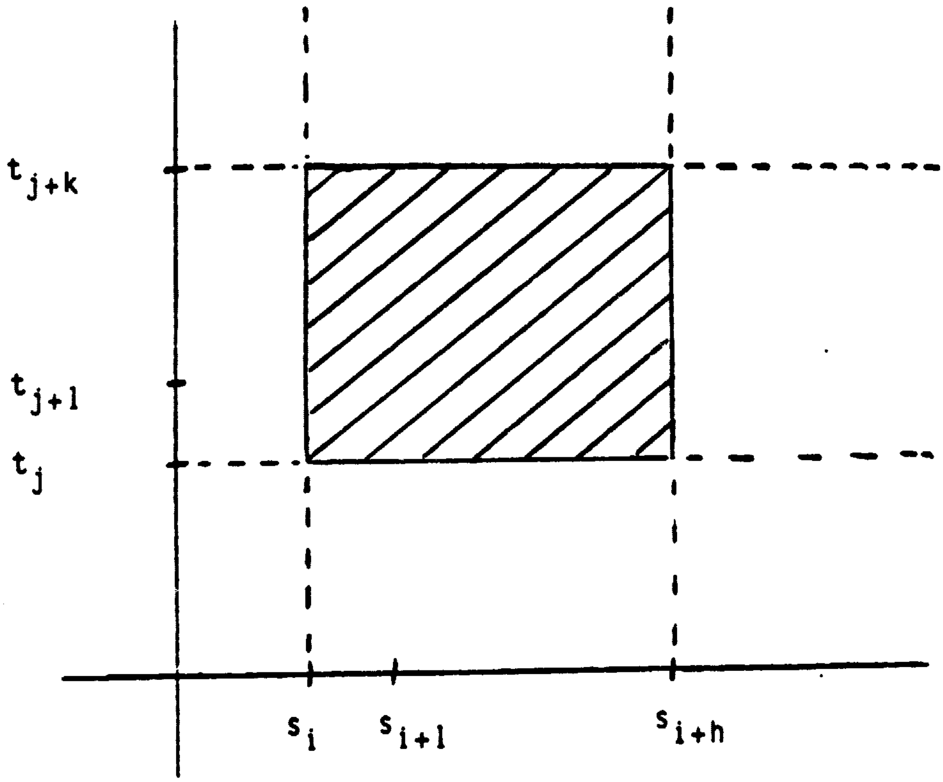

Figure 5 Support of tensor product B-spline B_{ij}.

have a minimum are examined.

3.3.1 Characteristics of the Functional

The coefficients of the mapping defined by

$$T(\xi,\eta) = \begin{bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{bmatrix} = \begin{bmatrix} \sum_{i=j}^{m} \sum_{j=1}^{n} \alpha_{ij} B_{ij}(\xi,\eta) & 0 \le \xi \le 1 \\ \\ \sum_{i=1}^{m} \sum_{j=1}^{n} \beta_{ij} B_{ij}(\xi,\eta) & 0 \le \eta \le 1 \end{bmatrix}$$

can be divided into two groups: boundary coefficients and interior coefficients. T uses the boundary coefficients, α_{ij}, β_{ij}, $j=1,\ldots,m$ and α_{im}, β_{im}, $i=1,\ldots,n$ to map the boundary of the square onto the boundary of the physical domain. Hence, the flexibility of their values is limited. The rest of the coefficients, the interior coefficients, can be moved around in order to change the characteristics of the physical grid. To produce orthogonality in the grid lines and maximize the smoothness of the distribution of grid lines one can choose the interior coefficients to minimize the functional

$$F = \int_{I_2} w_1\left(\left(\frac{\partial JT}{\partial \xi}\right)^2 + \left(\frac{\partial JT}{\partial \eta}\right)^2\right) dA + \int_{I_2} w_2 (\mathrm{Dot})^2 dA$$

where

$$JT(\xi,\eta) = \text{Jacobian of } T \text{ at } (\xi,\eta)$$

$$= \begin{vmatrix} \frac{\partial x}{\partial \xi}(\xi,\eta) & \frac{\partial x}{\partial \eta}(\xi,\eta) \\ \frac{\partial y}{\partial \xi}(\xi,\eta) & \frac{\partial y}{\partial \eta}(\xi,\eta) \end{vmatrix}$$

$$= \frac{\partial x}{\partial \xi}(\xi,\eta)\,\frac{\partial y}{\partial \eta}(\xi,\eta) - \frac{\partial y}{\partial \xi}(\xi,\eta)\,\frac{\partial x}{\partial \eta}(\xi,\eta),$$

$$\text{Dot}(\xi,\eta) = \frac{\partial T}{\partial \xi}(\xi,\eta) \cdot \frac{\partial T}{\partial \eta}(\xi,\eta)$$

$$= \begin{bmatrix} \frac{\partial x}{\partial \xi}(\xi,\eta) \\ \\ \frac{\partial y}{\partial \xi}(\xi,\eta) \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial x}{\partial \eta}(\xi,\eta) \\ \\ \frac{\partial y}{\partial \eta}(\xi,\eta) \end{bmatrix}$$

$$= \frac{\partial x}{\partial \xi}(\xi,\eta)\,\frac{\partial x}{\partial \eta}(\xi,\eta) + \frac{\partial y}{\partial \xi}(\xi,\eta)\,\frac{\partial y}{\partial \eta}(\xi,\eta)$$

and $w_1(\xi,\eta)$, $w_2(\xi,\eta)$ = weight functions evaluated at (ξ,η). After the minimization of F is completed, where w_1 is large the variation of the Jacobian values at nearby points will be small. Hence, w_1 can be used to decrease skewness in a grid. Where w_2 is large, Dot will be small causing the grid lines to approach orthogonality.

To avoid the tedious differentiation and integration of tensor product B-splines, the following discrete approximation to F can be implemented in computer algorithms:

$$G = \sum_{i=1}^{p} \sum_{j=1}^{q} w_1 \left(\frac{(JT_{i+1,j} - JT_{ij})^2}{(\Delta\xi)^2} + \frac{(JT_{i,j+1} - JT_{ij})^2}{(\Delta\eta)^2} \right) \Delta\xi\Delta\eta$$

$$+ \sum_{i=1}^{p} \sum_{j=1}^{q} w_2 (\text{Dot}_{ij})^2 \, \Delta\xi\Delta\eta$$

where

$$0 = \xi_1 < \xi_2 < \ldots < \xi_p = 1,$$

$$0 = \eta_1 < \eta_2 < \dots < \eta_q = 1,$$

$$JT_{ij} = JT(\xi_i \eta_j), \quad Dot_{ij} = Dot(\xi_i, \eta_j),$$

$$\Delta\xi = 1/(p-1), \quad \Delta\eta = 1/(q-1), \text{ and}$$

the parameters w_1 and w_2 are weight functions. Both F and G depend only on the coefficients of the tensor product B-splines which compose T.

Now

$$\frac{\partial x}{\partial \xi}(\xi,\eta) = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{ij} \frac{\partial}{\partial \xi}(B_{ij}(\xi,\eta))$$

$$\frac{\partial x}{\partial \eta}(\xi,\eta) = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{ij} \frac{\partial}{\partial \eta}(B_{ij}(\xi,\eta))$$

$$\frac{\partial y}{\partial \xi}(\xi,\eta) = \sum_{i=1}^{m} \sum_{j=1}^{n} \beta_{ij} \frac{\partial}{\partial \xi}(B_{ij}(\xi,\eta))$$

$$\frac{\partial y}{\partial \eta}(\xi,\eta) = \sum_{i=1}^{m} \sum_{j=1}^{n} \beta_{ij} \frac{\partial}{\partial \eta}(B_{ij}(\xi,\eta)).$$

Thus, for ξ,η fixed, $JT(\xi,\eta)$ is a linear function in each coefficient α_{ij}, β_{ij}, $i=1,\dots,m$, $j=1,\dots,n$ and $Dot(\xi,\eta)$ is a quadratic polynomial in each coefficient. Since the terms involving $Dot(\xi,\eta)$ and $JT(\xi,\eta)$ are squared in G, one can see that G is actually a quartic polynomial in each coefficient. This suggests an elementary iteration method for finding the minimum of G: the cyclic coordinate method [B, p. 271].

The cyclic coordinate method is a multidimensional search technique for minimizing a function of several variables without using derivatives. It searches for a minimum along each coordinate direction. This method, when applied to a differentiable function, converges to a point where the gradient is zero [8, p. 273]. It can be applied to G if one treats each coefficient α_{ij}, β_{ij}, $i=1,\ldots,m$, $j=1,\ldots,n$ as a variable representing a particular coordinate direction. This technique is discussed further in the next chapter.

The importance of requiring that the Jacobian of T be of one sign was illustrated in Chapter 2. For this reason, if possible, the feasible region for the minimization problem is chosen to be a region where the Jacobian of T is nonnegative. Now since B-splines have small support, any given coefficient α_{rs} or β_{rs} only affects the Jacobian of T at a small number of points on the unit square mesh. By solving the inequality $JT \geq 0$ for α_{rs} at each of these points one can determine on what interval α_{rs} must lie so that the Jacobian values at the points it affects are nonnegative. This inequality is easy to solve since JT is linear in α_{rs}. Repeating this procedure for each coefficient will, in most cases, produce a satisfactory approximation to the desired feasible region. However, since the boundary coefficients are fixed, there may sometimes be problems near the boundary. This is the case with the nonconvex region examined in Section 2.5. The Jacobian will remain negative at one of its corner points even after the domain for the coefficients is restricted by using the procedure above. This is because the boundary points are fixed and not affected by the procedure. The Jacobian will also remain negative near this corner because of continuity.

3.3.2 Convergence of the Smoothing Functional

Under what conditions will the discrete smoothing functional G converge to a minimum value? Is it important that G be restricted to a region where the Jacobian of T is nonnegative? What happens if one of the tensor product coefficients becomes large?

These are some of the questions which might be asked about G. The notation defined below will be used to discuss these problems:

$\{A_r\}$ is a sequence in which each term represents a set of coefficients for the mapping $T:I_2 \quad R^2$ defined by

$$T(\xi,\eta) = \begin{bmatrix} \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{ij} B_{ij}(\xi,\eta) \\ \\ \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{ij} B_{ij}(\xi,\eta) \end{bmatrix}, \quad 0 \le \xi \le 1, \quad 0 \le \eta \le 1.$$

Each A_r can be considered a discrete function defined by

$$A_r(s,i,j) = \begin{bmatrix} \alpha^r_{ij}, \text{ if } s = 1 \\ \\ \beta^r_{ij} \text{ if } s = 2 \end{bmatrix}, \quad i=1,\dots,m;\ j=1,\dots,n.$$

T_r denotes the mapping obtained when the coefficients given by A_r are used for T.

JT_r denotes the Jacobian of T_r.

$|A_r|_{max} = \max\limits_{s,i,j} |A_r(s,i,j)|$.

It follows that if the sequence $\{A_r\}$ of coefficients converges to a single point then the corresponding values of G also converge. Hence, it is important to determine conditions which guarantee the convergence of the coefficient sequence. Well, since the elements of $\{A_r\}$ can be viewed as points in R^{2mn}, the sequence converges if and only if it is a Cauchy sequence; however, a necessary condition for the convergence of $\{A_r\}$ is that the sequence be bounded. The following theorem and corollary show how the Jacobian affects the boundedness of the sequence.

<u>3.3.2-1</u> <u>Theorem</u>. Suppose for all r T_r maps ∂I_2 homeomorphically onto $\partial\Omega$. If $JT_r(\xi,\eta) \geq 0$ for all r and all points $(\xi,\eta) \in I_2^0$, then either $\{A_r\}$ is bounded or $JT_r(\xi_0,\eta_0) = 0$ for some point $(\xi_0,\eta_0) \in I_2^0$.

Proof: By way of contradiction, suppose $\{A_r\}$ is not bounded and $JT_r(\xi,\eta)>0$ for all r and all points $(\xi,\eta) \in I_2^o$. For any integer N there exists an $A_{r_N} \in \{A_r\}$ such that $|A_{r_N}|_{max} > N$. But this implies that either $|\alpha_{ij}^{r_N}|>N$ or $|\beta_{ij}^{r_N}|>N$ for some i,j. Now since Ω is bounded there exists $M>o$ such that $|\vec{p}|\leq M$ for all $\vec{p}\in\Omega$. From3.2.3-3 it follows that for large enough N, $\max\limits_{\substack{0\leq\xi\leq 1\\ 0\leq\eta\leq 1}} |T_{r_N}(\xi,\eta)| > 2M$.

Hence T_{r_N} maps some point $(\xi_1,\eta_1) \in I_2^o$ outside $\partial\Omega$. Now since $JT_{r_N}>0$ on I_2^o, $m(T_{r_N}(I_2)-\Omega)>0$. But then 2.4.4 says that JT_{r_N} has a sign change.

Q.E.D.

The corollary below follows immediately.

3.3.2-2 Corollary. Suppose for all r T_r maps I_2 homeomorphically onto $\partial\Omega$. If $JT_r(\xi,\eta)>0$ for all r and all points $(\xi,\eta) \in I_2^o$, then $\{A_r\}$ is bounded.

One would like to show that the requirement $JT_r(\xi,\eta)\geq 0$ for all r and all points $(\xi,\eta) \in I_2^o$ is sufficient to guarantee the boundedness of $\{A_r\}$. As indicated in 3.3.2-1, it is clear that if the magnitude of a coefficient is large enough, then the mapping T_r associated with the coefficient will map some point in I_2^o outside $\partial\Omega$. However, it is no longer clear that $m(T_r(I_2)-\Omega)>0$ since $JT_r(\xi,\eta)$ may be 0 outside $\partial\Omega$. Thus 2.4.4 cannot be used to obtain the contradiction that JT_r must have a sign change as was done in Theorem 3.3.2-1. Although the writer has been unable to devise an acceptable proof to date, further study may show that the inequality, $m(T_r(I_2)-\Omega)>0$, is actually true.

4. PROGRAM TENTEST

This chapter discusses the computer program TENTEST which algebraically generates grids using tensor product cubic B-splines. A listing of TENTEST is given in the appendix at the end of this paper.

The first section of this chapter presents the major steps involved in the computer algorithm. Sections 2 through 5 examine the important features of the program, briefly discussing the subroutines involved.

4.1 The Algorithm

Although TENTEST contains almost a thousand lines of code, it is based on the following eight step algorithm:

i. Input knot sequences $\{s_i\}$ and $\{t_j\}$ consisting of values from [0,1].

ii. Compute the tensor product cubic B-splines corresponding to the knot sequences.

iii. Choose initial coefficients to form a bicubic spline mapping from the square to a physical domain.

iv. Use the mapping to plot a grid on the physical domain.

v. If grid satisfactory, stop. If grid unsatisfactory, continue.

vi. Input weights for smoothing functional.

vii. Complete one iteration of minimization routine to obtain new coefficients.

viii. Go to step iv.

There also exists a batch version of TENTEST which allows the user to request several iterations of the minimization routine at a time. All the information needed to plot the initial and final grids is stored in files which can be interactively accessed after the execution of the program is completed.

The programs were run on a PRIME 750 computer. The PRIMOS operating system, coupled with a PLOT 10 graphics package, was used to interactively draw the grids on a Tektronics 4014 terminal. The PRIME 750 can communicate at a baud rate of up to 9600 thus making it satisfactory for interactive graphics.

4.2 Computing the Tensor Product B-splines

Since B-splines are determined by the knots with which they are associated, the first concern of the user is to choose appropriate knot sequences. The user must pick two sequences $s=\{s_i\}$ and $t=\{t_j\}$, placing them in file TENSORDAT. The user actually picks only the "interior" knots for each sequence. In other words, he constructs two increasing sequences of numbers between 0 and 1. After reading the numbers from file TENSORDAT, TENTEST places four 0's at the beginning of each sequence and four 1's at the end of each sequence. By 3.2.2-3 (Curry and Schoenberg) and 3.2.3-1, the cubic B-splines associated with s and t form bases for spline spaces $S_{4,s}$ and $S_{4,t}$. The functions in each of these spaces will have three continuity conditions at each interior knot. The products of the B-splines will form a basis for the tensor product of $S_{4,s}$ and $S_{4,t}$. The tensor product B-splines can be used to construct a transformation T on the square which maps the boundary of the square onto the boundary of a physical domain as described in Chapter 3. The user may obtain a better approximation to the boundary of the physical domain by increasing the number of interior knots in s and t or by redistributing the knots. This is discussed in more detail in Section 4.5.

On a given pxq mesh on the square with mesh points (ξ_u,η_v), $u=1,\ldots,p$, $v=1,\ldots,q$, the values of the tensor product B-splines which compose T are fixed. Since these tensor product B-splines are the products of B-splines $B_i, i=1,\ldots,m$ and $B_j, j=1,\ldots,n$ for some m and n, it is convenient to store the function values and first derivatives of these B-splines at each ξ_u and η_v. Subroutine COMSPLINE uses the de Boor routine BSPLVD [de B, p. 288] to compute these values. BSPLVD calculates the function value and derivatives of all the nonvanishing B-splines at a given point. COMSPLINE stores the function values and first derivatives in two arrays: XSPLINE and YSPLINE. Therefore, after a call to COMSPLINE is completed, XSPLINE will contain the function value and first derivative of each B-spline in $\{B_i\}_{i=1}^{m}$ at ξ_u, $u=1,\ldots,p$ and YSPLINE contains the function value and first derivative of each B-spline in $\{B_j\}_{j=1}^{n}$ at $\eta_v, v=1,\ldots,q$. Computing T or its partial derivatives at a mesh point becomes a matter of calculating the sum of the products of the tensor product coefficients with the appropriate elements of XSPLINE and YSPLINE. This computation is done in subroutine TENVALF.

The next section explains how the coefficients are chosen initially.

4.3 Choosing the Initial Coefficients

Many different methods can be used to choose the coefficients initially. Since B-spline coefficients model the function they represent, one might simply choose the boundary coefficients to equal points along the boundary of the physical domain, and choose the interior coefficients to equal points known to lie in the interior of the physical domain. However, this creates the problem of deciding which interior points should be chosen as coefficients. Ideally, the original coefficients should produce a grid which is somewhat smooth so that only a few iterations are needed to obtain an acceptable degree of smoothness and orthogonality.

For this reason, the computer program described in this paper initially selects coefficients which produce an approximation to the transfinite bilinear interpolant of a mapping $V: I_2 \rightarrow R^2$ satisfying $V: \partial I_2 \rightarrow \partial\Omega$. In reality one need only define V on ∂I_2. The user may provide parametric equations which map the boundary of the square onto the boundary of the physical domain, or simply input a set of boundary points for the physical domain. In the first instance V is defined by using the parametric equations. In the latter case V is obtained by linearly interpolating between successive boundary points. The parametric equations below map the four sides of the unit square onto the four sides of the trapezoid as shown in figure 6.

$$V(\xi,0) = g_1(\xi) = \begin{pmatrix} 2\xi+1 \\ 0 \end{pmatrix}$$

$$V(1,\eta) = g_2(\eta) = \begin{pmatrix} 3+\eta \\ 2\eta \end{pmatrix}$$

$$V(\xi,1) = g_3(\xi) = \begin{pmatrix} 4\xi \\ 2 \end{pmatrix}$$

$$V(0,\eta) = g_4(\eta) = \begin{pmatrix} 1-\eta \\ 2\eta \end{pmatrix}$$

The tra..sfinite bilinear interpolant U of V is defined by

$$\begin{aligned} U(\xi,\eta) = {} & (1-\eta)V(\xi,0) + \eta V(\xi,1) \\ & + \xi V(1,\eta) + (1-\xi)V(0,\eta) \\ & - (1-\xi)(1-\eta)V(0,0) - \xi(1-\eta)V(1,0) \\ & - (1-\xi)\eta V(0,1) - \xi\eta V(1,1). \end{aligned}$$

U agrees with V on the boundary of the square and hence interpolates V at an infinite number of points. Transfinite interpolants are discussed by William J. Gordon and Charles A. Hall in [G].

The program selects initial coefficients which produce a variation diminishing spline approximation to U.

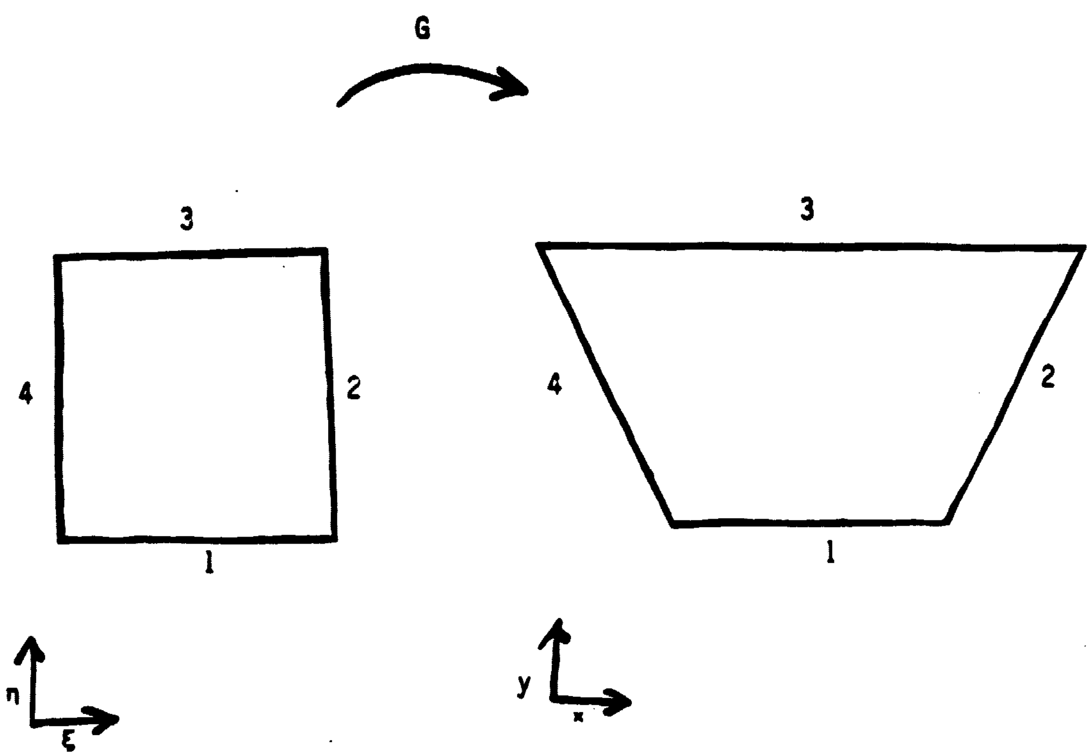

Figure 6. Mapping from computational domain to physical domain.

Hence, if T is constructed from tensor products of B-splines $B_i=B_{i,4,s}$, $i=1,\ldots,m$ and $B_j=B_{j,4,t}$, $j=1,\ldots,n$, which correspond to knot sequences $s=\{s_i\}_{i=1}^{m+4}$ and $t=\{t_j\}_{j=1}^{n+4}$, respectively, then the initial coefficients of the tensor product splines are $\begin{Bmatrix}\alpha_{ij}\\ \beta_{ij}\end{Bmatrix} = U(s_i^*,t_j^*)$, $i=1,\ldots,m$; $j=1,\ldots,n$ where $s_i^*=(s_{i+1}+\ldots+s_{i+3})/3$, $i=1,\ldots,m$ and $t_j^*=(t_{j+1}+\ldots+t_{j+3})/3$, $j=1,\ldots,n$. Since variation diminishing splines yield exact approximations to linear polynomials, T will reproduce the boundary of any physical domain which can be divided into four line segments. Arbitrarily shaped boundaries can be approximated as accurately as desired by increasing the number of knots used to define the tensor product splines or by changing the placement of knots to increase the concentration in complex shaped areas of the boundary.

The initial tensor product coefficients are constructed in subroutines BOUNCOEF and INNERCOEF. Figure 7 shows a grid on a trapezoid domain constructed with a mapping T having coefficients as described above. The grid is the image of T over an equally spaced mesh on the square.

4.4 Minimizing the Smoothing Functional

In TENTEST, the cyclic coordinate method is used to find the minimum of the smoothing functional G described in Section 3.3. As the name suggests, this method attempts to find the minimum of a multivariable function by cyclicly searching in the direction of each coordinate axis. For

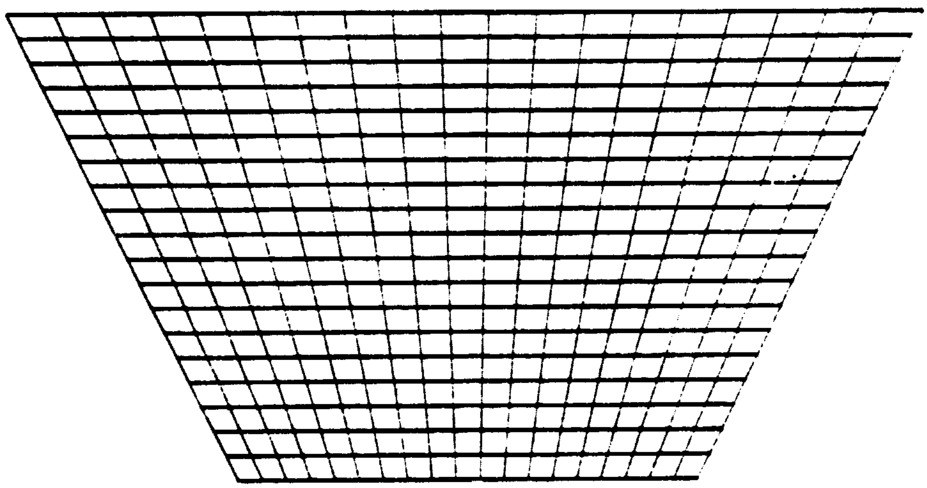

Figure 7. Trapezoid Grid

G, the coordinate directions are represented by the tensor product coefficients α_{ij}, β_{ij}, $i=1,\ldots,m$; $j=1,\ldots,n$.

The user must first decide what size mesh should be used to obtain a grid with acceptable smoothness and orthogonality. G is a function of 2mn coefficients, however, since the boundary coefficients are fixed only 2(m-2)(n-2) coefficients are free. Therefore, in general, the mesh used for the minimization technique should contain at least 2(m-2)(n-2)points.

The user must also decide on the size of the weights w_1, w_2 for G. One can choose constant weights for both JT and Dot, or choose a weight function for Dot which produces more orthogonality near the boundary of the grid than in the interior. Small constant weights of values between 1 and 10 can be used initially to determine how they affect the smoothness and orthogonality of the grid.

Changing coefficient α_{ij} (or β_{ij}) changes the value of the mapping T only on the support of the tensor product B-spline B_{ij}. Therefore, in order to locate the minimum of G in the direction represented by α_{ij} one need only consider the sum over those terms in G which contain the value of JT or Dot at mesh points (ξ, η) lying on the support of B_{ij}. Subroutine CORANGE determines the range of summation associated with each tensor product coefficient for a given mesh on the square, and function GF computes the sum over the range indicated by CORANGE. Figure 8 shows the support of a tensor product B-spline associated with

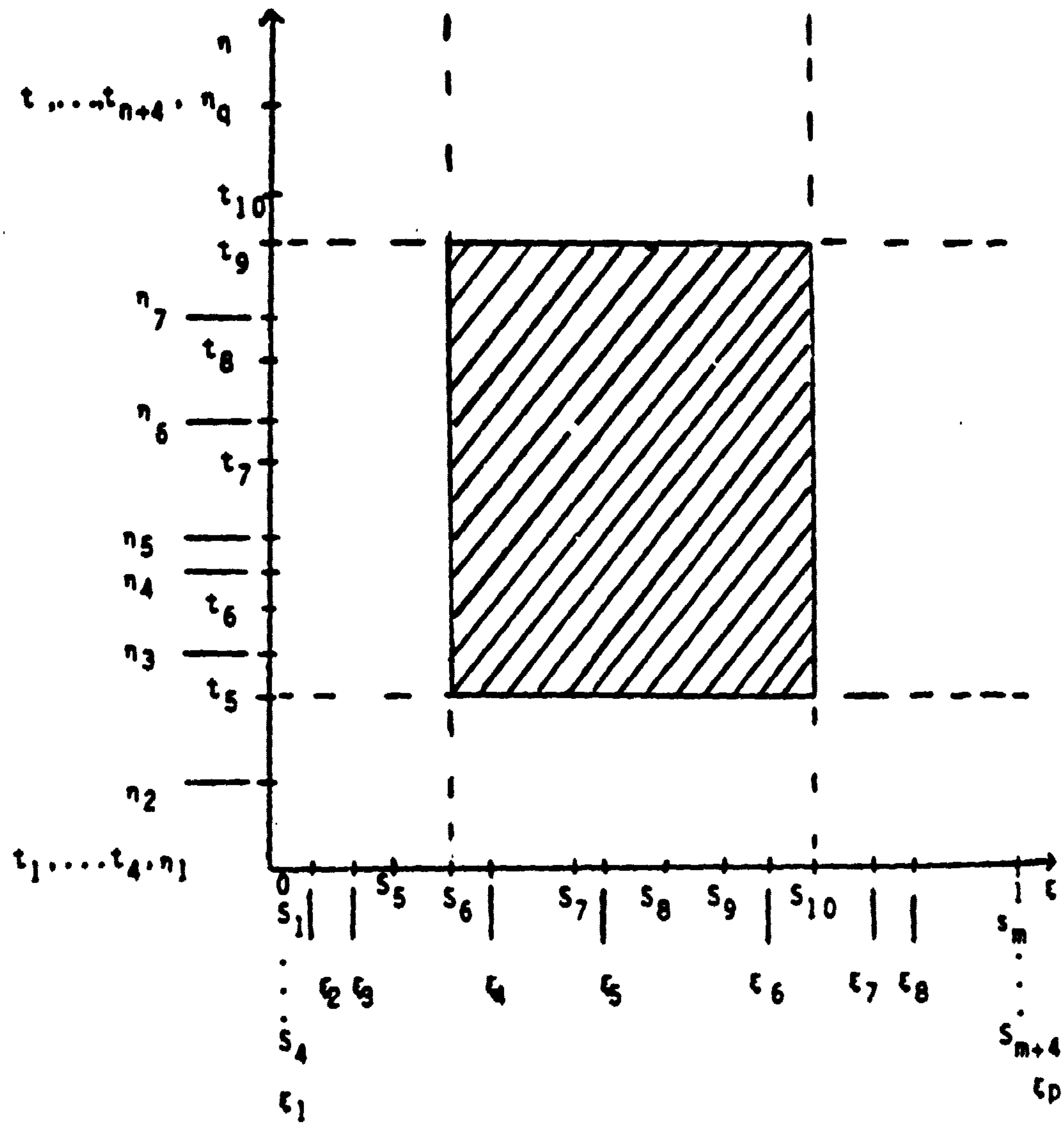

Figure 8. Support for tensor Product B-spline $B_{6,5}$

knot sequences $s = \{s_i\}_{i=1}^{m+4}$ and $t = \{t_j\}_{j=1}^{n+4}$. The shaded section represents the support of tensor product B-spline $B_{6,5}$. In order to minimize in the direction of coefficient $\alpha_{6,5}$ it would be sufficient to look at the sum

$$GF = \sum_{i=3}^{6} \sum_{j=3}^{7} w_1(JT_{i+1,j} - JT_{ij})^2 \frac{\Delta\eta}{\Delta\xi} + \sum_{i=4}^{6} \sum_{j=2}^{7} w_1(JT_{i,j+1} - JT_{ij})^2 \frac{\Delta\xi}{\Delta\eta}$$

$$+ \sum_{i=4}^{6} \sum_{j=3}^{7} w_2(Dot_{ij})^2 \Delta\xi\Delta\eta.$$

Like G, the partial sum, GF, will be a quartic polynomial in each coefficient.

All of this information is used by the minimization routine FFMIN. Each call to FFMIN produces one complete iteration of the cyclic coordinate method. For each coefficient, the routine first determines the interval on which the coefficient must lie if JT is to be nonnegative at most of the mesh points affected by the coefficient. Then it calls either TESTMINØ, TESTMINL, TESTMINR, or TESTMINB depending on whether the interval is biinfinite, has a left endpoint, a right endpoint, or two endpoints. The chosen subroutine finds the location of the minimum of GF on the interval and changes the value of the appropriate coefficient accordingly.

4.5 Distribution Functions

If solutions of partial differential equations on a domain are to be accurate, the grid on the domain must be concentrated in areas of rapid change such as boundary layers and shocks. In most cases concentration near the boundary of the domain can be easily accomplished through the use of distribution functions.

Rearranging the points on the square mesh changes the distribution of grid points on the physical domain. A nonuniform distribution of points on the square mesh can be viewed as the image of functions $\phi_1 : I_1 \to I_1$, and $\phi_2 : I_1 \to I_1$ defined on ξ and η, respectively. The grid is then generated by the mapping T defined by

$$\tilde{T}(\xi,\eta) = T \circ \phi(\xi,\eta)$$

where $\phi : I_2 \to I_2$ satisfies

$$\phi(\xi,\eta) = \begin{bmatrix} \phi_1(\xi) \\ \phi_2(\eta) \end{bmatrix}.$$

This is graphically illustrated in figure 9. The grid on the physical domain is the image under $\tilde{T}$ of an equally spaced mesh on the square.

In the current version of TENTEST, the user may request one of three distributions for ξ and η: uniform, exponential, or arctangent. Selecting the uniform option produces an equally spaced distribution. The distribution function is simply the identity function on I_1. If the exponential option is selected, TENTEST calls routine EXPONENTIAL which maps $\zeta \epsilon I_1$ into $\psi(\zeta) = \dfrac{e^{c\zeta}-1}{e^{c}-1}$

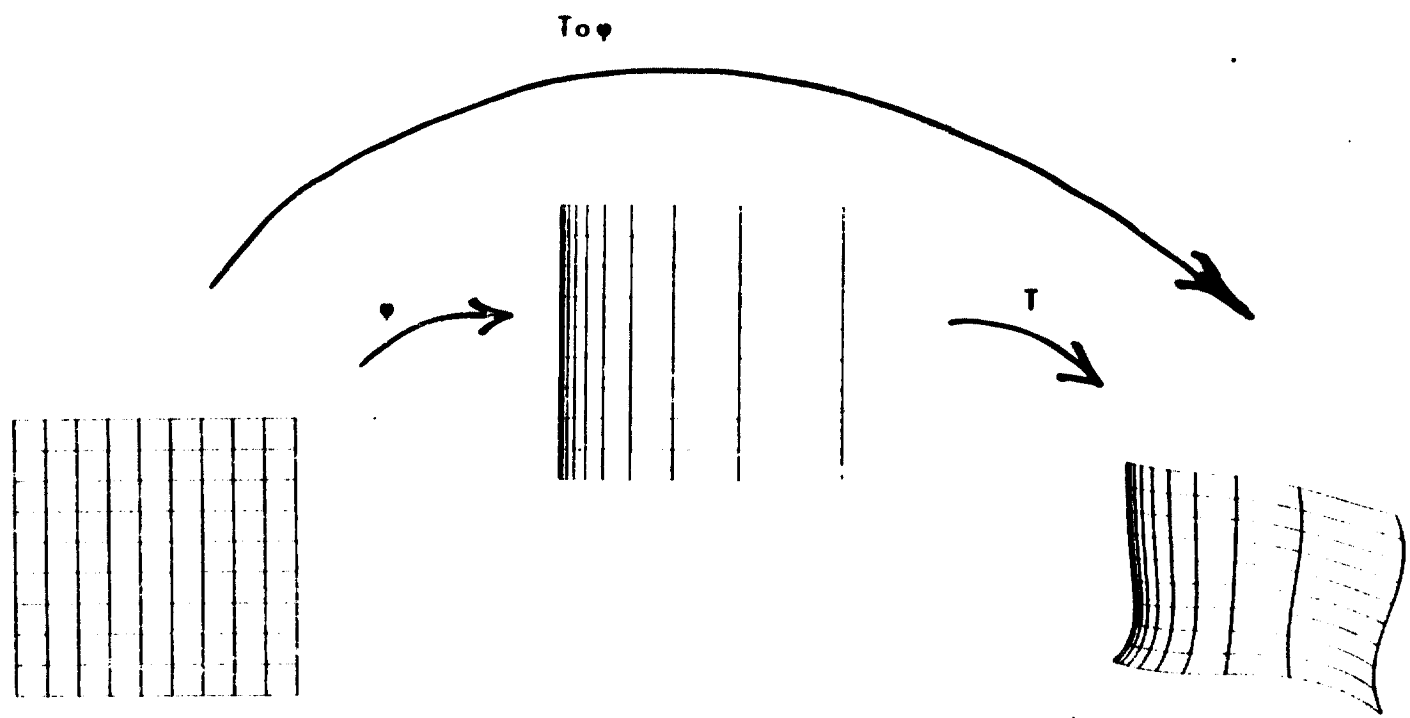

Figure 9. Obtaining a Concentration of Gridpoints

where c is a nonzero constant. If $c>0$, ϕ concentrates the grid lines near the line corresponding to $\zeta=0$. If $c<0$, ϕ concentrates the grid lines closer to the line corresponding to $\zeta=1$. The grid in figure 10a was produced with $\varphi_1(\xi) = \xi$ and $\varphi_2(\eta) = \phi(\eta)$. The constant c is 4. In figure 10b, $\varphi_1(\xi) = \phi(\xi)$ with $c=5$ and $\varphi_2(\eta) = \eta$. The degree of concentration increases or decreases as $|c|$ is increased or decreased. In figure 10c, $\varphi_1(\xi) = \phi(\xi)$ with $c=2$ and $\varphi_2(\eta) = \eta$.

TENTEST calls ARCTANGENT when the user selects the arctangent distribution option. ARCTANGENT maps $\zeta \in I_1$ into

$$\gamma(\zeta) = \frac{\text{arctangent }(2c\zeta - c) - \text{arctangent }(-c)}{\text{arctangent }(c) - \text{arctangent }(-c)}$$

where c is a positive constant. This function concentrates grid lines near points corresponding to $\zeta=0$ and $\zeta=1$ simultaneously. This is shown in figure 10d with $\varphi_1(\xi) = \xi$, $\varphi_2(\eta) = \gamma(\eta)$ and $c=5$.

Future improvements to TENTEST might include the addition of more distribution functions and the creation of a routine which allows the user to create his own distribution function by interactively digitizing points on the unit square. The routine would then create a variation diminishing spline approximation to the points to form the distribution function.

Since the distribution functions described in this section are defined on I_1, they can also be used to

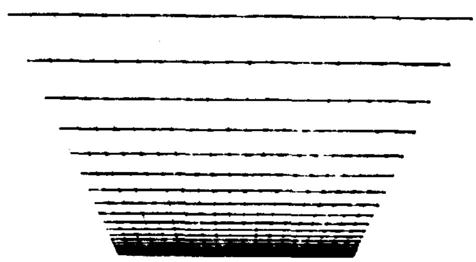

Figure 10a. Exponential distribution on η with c=4.

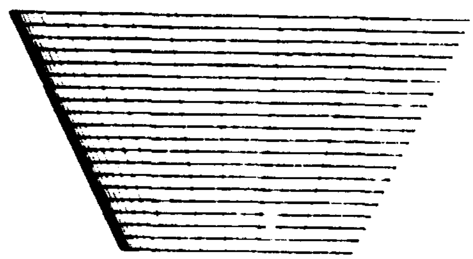

Figure 10b. Exponential distribution on ξ with c=5.

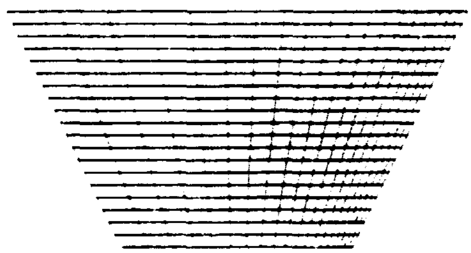

Figure 10c. Exponential distribution on ξ with c= -2

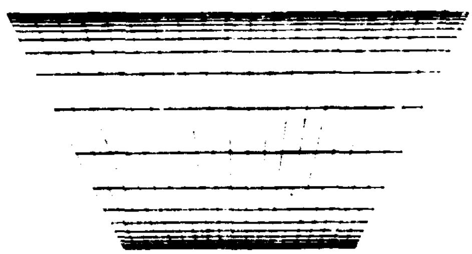

Figure 10d. Arctangent distribution on η with c=5.

Figure 10. Concentrating grid points on trapezoid domain.

redistribute the knots which define the tensor product B-splines that form T. This will permit the user to concentrate more knots in areas mapped to complex portions of the physical boundary so that T produces a better boundary approximation. Presently the user can choose to keep the original distribution on the knots or choose to redistribute the knots to obtain an exponential or arctangent distribution.

5. RESULTS AND DISCUSSION

This chapter examines some of the grids produced by TENTEST. Physical domains of various shapes are illustrated. Some of the grids are for actual objects, such as an airfoil or part of the space shuttle, but most are simply grids on domains of various shapes and sizes chosen to illustrate the range of the program.

The user's chief concern is the creation of an acceptable grid on a given physical domain in the shortest amount of time possible. Since the grid will be the image of a continuous mapping on the square, the best technique is to minimize the smoothing functional by using a grid generated from a coarse mesh. Then, once the new coefficients are obtained, the user can request that the grid be plotted using a much finer mesh. This technique is illustrated in the examples which follow. Most of the examples contain at least four grids: The image under T, with its initial coefficients, of a coarse square mesh; the image of a finer mesh; the image of the coarse mesh after several iterations of the minimization procedure; and the image of a finer mesh after application of the minimization procedure. Any other grids shown are chosen to illustrate grid concentration or other points of interest. In all the examples shown, only constant weight functions were used in the smoothing functional.

The first four examples show grids on domains with common geometric shapes: a trapezoid, a quadrilateral with unequal, nonparallel sides, a triangle and a circle. Since the domains are simply connected and convex, only a few interior points are needed for the sequences s and t which determine the tensor product B-splines that compose T.

The next three examples show grids on domains which are not convex. The major concern with such grids is the overlapping of grid lines near the boundary.

The last examples deal with grids around concrete objects such as an airfoil or part of the space shuttle. The irregular boundaries of some of these grids make it necessary to use more knots to define T.

For convenience, the following notation is used in this chapter.

N_ξ = number of B-splines B_i in the sequence corresponding to knot sequence s, or 4 + number of interior knots in s.

N_η = number of B-splines B_j in the sequence corresponding to knot sequence t, or 4 + number of interior knots in t.

wj = constant weight multiplied times the terms in the smoothing functional involving the Jacobian, JT, of T.

wd = constant weight multiplied times the terms in the smoothing functional containing Dot.

$N_\xi \times N_\eta$ will be the dimension of the tensor product spline space generated by $B_{ij} = B_i \times B_j$, $i=1,\ldots,N_\xi$; $j = 1,\ldots N_\xi$. cpu = central processing unit - main control section of a computer.

5.1 Convex Domains

The first three examples, which have linear boundaries, require only one interior knot for each of the knot sequences s and t. The simplicity of the domains also means that a very coarse grid can be used to minimize the smoothing functional. Four or five iterations produce good results. The circular grids in the fourth example require more interior knots.

5.1.1 Trapezoid

In this example $N_\xi = N_\eta = 5$ and wj = wd = 1. The first picture in figure 11 is the grid obtained using the initial coefficients in Section 4.3. It is the image under T of an equally spaced 5x5 mesh on the square. This is the grid on which the minimization procedure was applied. Note that the number of grid points is 25, while the number of free coefficients is given by $2(N_\xi-2)(N_\eta-2) = 18$. Figure 11b is a finer grid constructed using the same coefficients. Figure 11c shows how the initial 5x5 grid changes after five iterations of the minimization procedure. The new coefficients produce grid lines that appear to be nearly orthogonal at most grid points. The image under

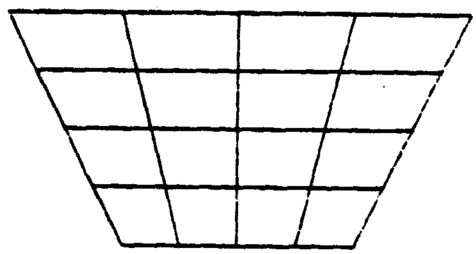

Figure 11a Original grid.

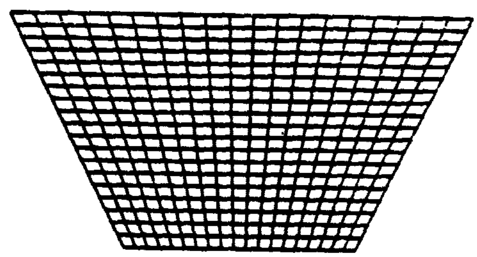

Figure 11b Original grid refined.

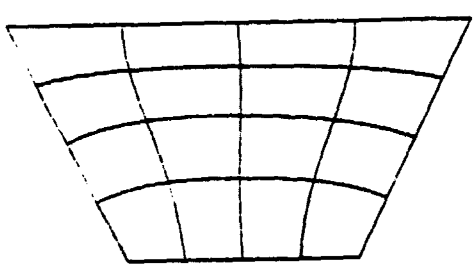

Figure 11c Optimized grid.

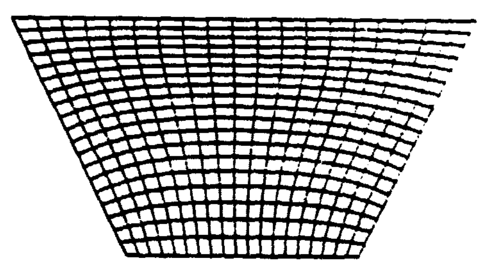

Figure 11d Optimized grid refined.

Figure 11 Grids on trapezoid domain.

the new T of a 20x20 mesh is given in figure 11d. The amount of cpu time used in the optimization process was 1 minute and 25 seconds.

In figure 12 the weights wj and wd have been changed to show what effect they have in the minimization process. Figure 12a shows how the initial 5x5 grid is changed after only three iterations when wj=0 and wd=1. Orthogonality is more pronounced, but the grid spacing is no longer as smooth. In the refined grid in 12b the spacing is very skewed near the top boundary. Figure 12c shows the 5x5 grid after five iterations with wj=1 and wd=0. The spacing is smoother but the grid lines are not orthogonal. Figure 12d shows a finer grid.

5.1.2 Quadrilateral with Unequal Sides

Again, in this example $N_\xi = N_\eta = 5$ which means sequences s and t each contain one interior knot. Also, wj=wd=1. The minimization procedure was applied on the 5x5 grid shown in figure 13a. Five iterations of the technique produced the grid in 13c. Figures 13b and 13d show refined versions of the grids in 13a and 13b, respectively. The five iterations of the minimization procedure required 2 minutes and 16 seconds of cpu time. In figure 14 the optimized grids are concentrated near different parts of the boundary. In 14a an exponential distribution with parameter c=4 has been put on η. Figure 14b shows an exponential distribution on ξ and η with c=4 in each

ORIGINAL PAGE IS
OF POOR QUALITY

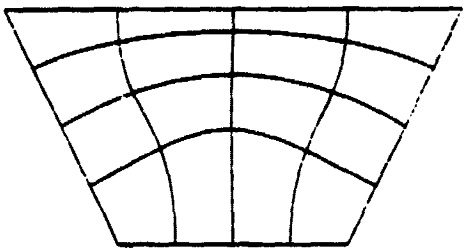

Figure 12a wj=o, wd=l

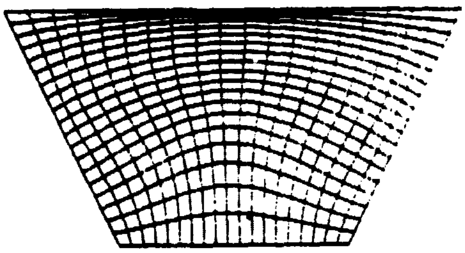

Figure 12b wj=o, wd=l

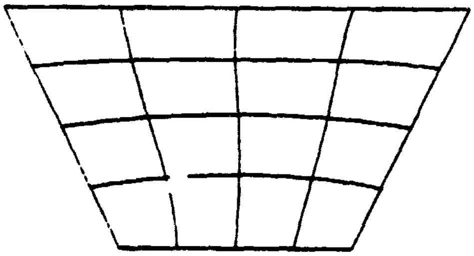

Figure 12c wj=l, wd=o

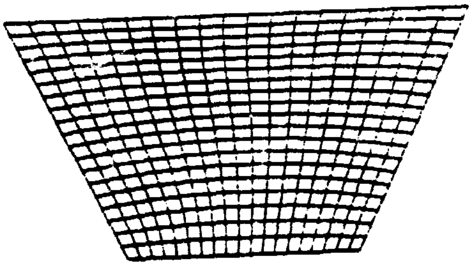

Figure 12d wj=l, wd=o

Figure 12 Effect of weights, wj and wd

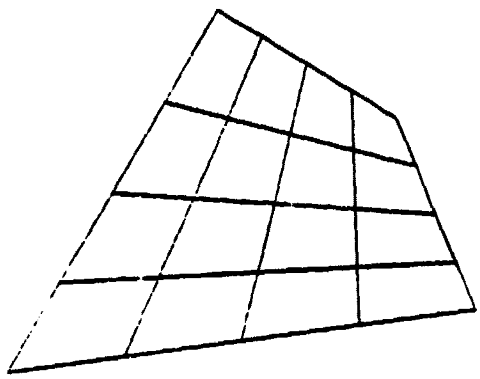

Figure 13a Original grid

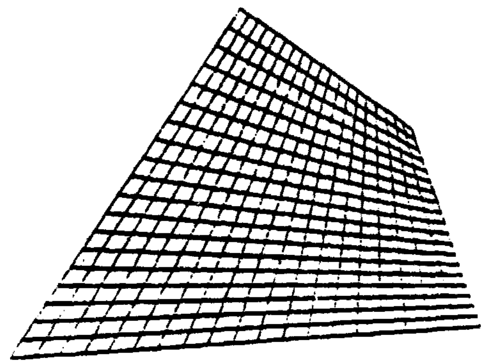

Figure 13b Original grid refined

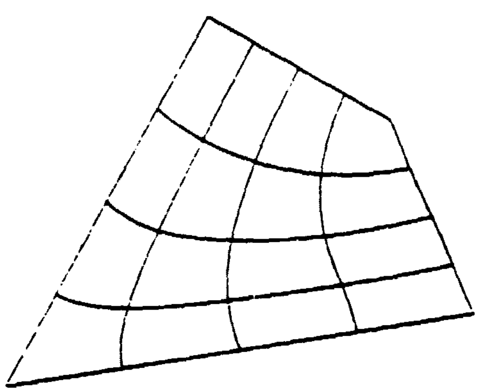

Figure 13c Optimized grid

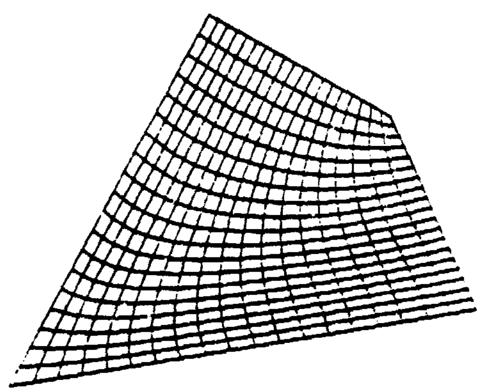

Figure 13d Optimized grid refined

Figure 13 Grids on quadrilateral with unequal sides.

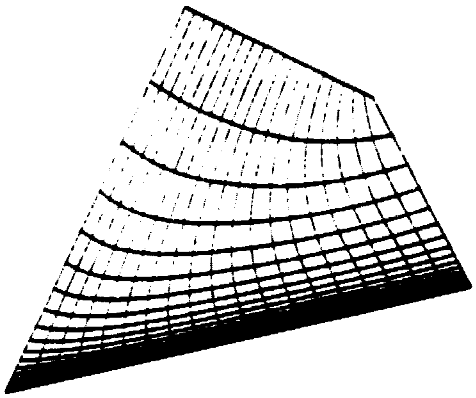

Figure 14a Exponential distribution of η

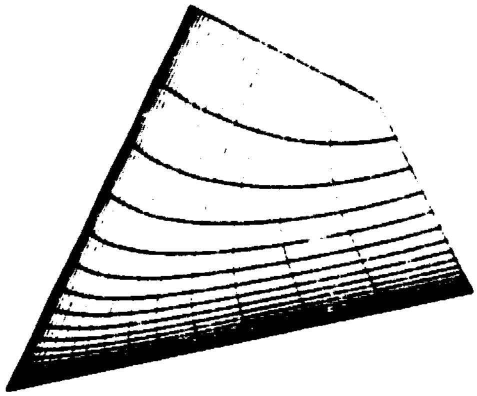

Figure 14b Exponential distribution on ξ and η.

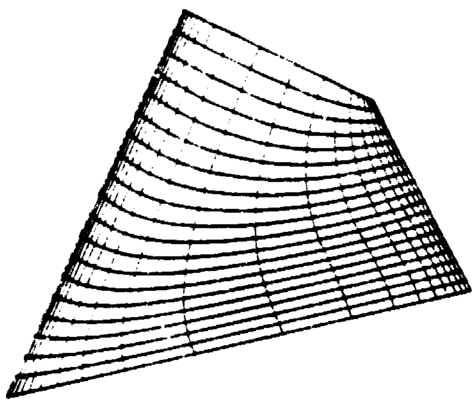

Figure 14c Arctangent distribution on ξ

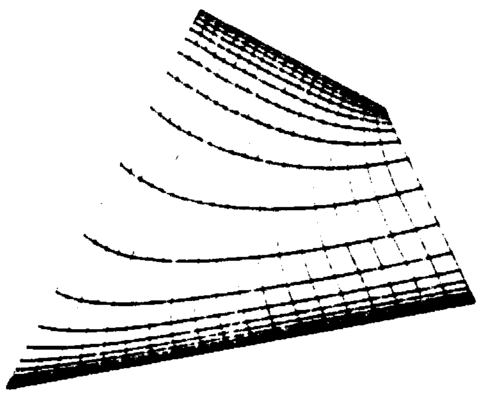

Figure 14d Arctangent distribution on η

Figure 14 Concentrating gridpoints on quadrilateral.

case. In figures 14c and 14d, an arctangent distribution with c=5 has been placed on ξ and η, respectively.

5.1.3 Triangle

In the previous examples, it was clear that each side of the unit square should be mapped to a side of the four-sided physical domain, but in the case of a triangle, which has three sides, this cannot be done. The boundary must be divided into four sections. The simplest thing to do is to divide one of the sides of the triangle into two parts so that two sides of the unit square are mapped onto one side of the triangle as shown in figure 15. Figure 16a shows the initial 5x5 grid constructed with $N_\xi = N_\eta = 5$ and wj=wd=1. Figure 16b shows a 20x20 grid constructed using the same coefficients. After five iterations of the minimization procedure, the initial 5x5 grid is transformed into figure 16c. Figure 16d shows a finer grid. Optimization required 2 minutes and 22 seconds of cpu time.

5.1.4 Circle

Variation diminishing splines reproduce straight lines exactly, but the same cannot be said about their approximation of nonlinear curves. For such curves the accuracy of the approximation depends on the number of knots used to define the spline function. For this reason more knots are needed to obtain a satisfactory mapping of the unit square onto a circular physical domain. For the

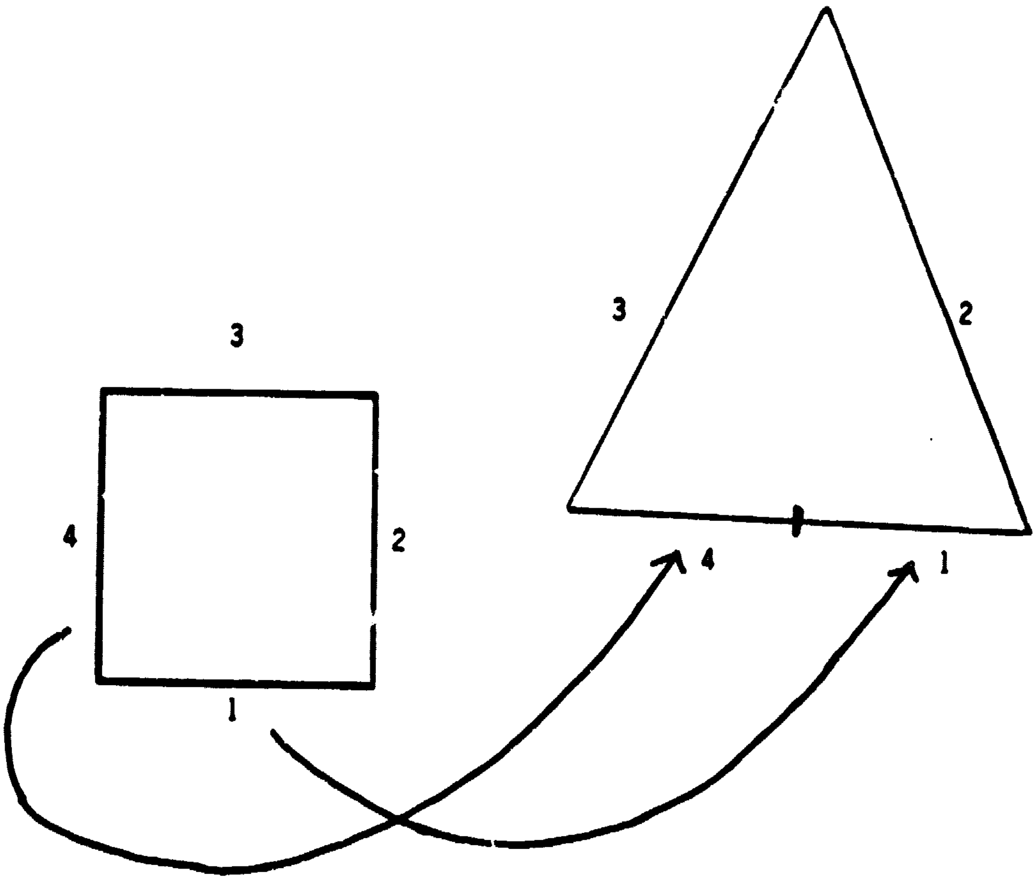

Figure 15 Division of triangular boundary into four sections.

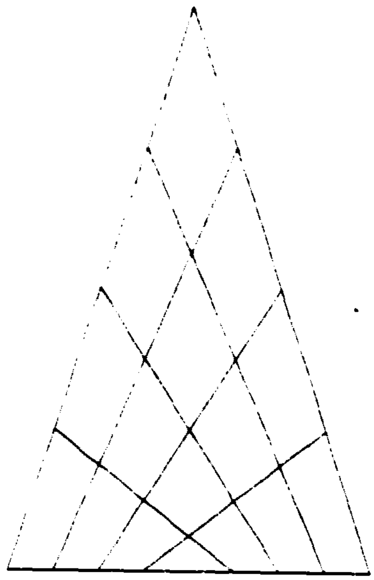

Figure 16a Original grid

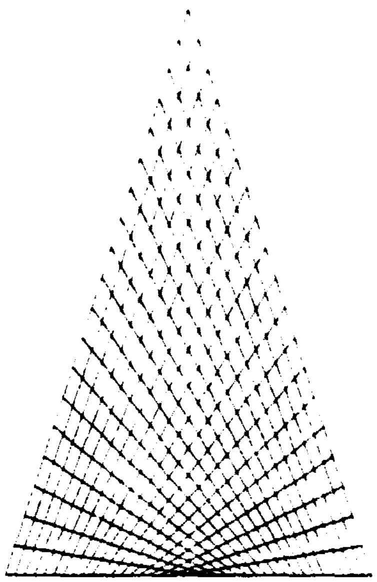

Figure 16b Original grid refined

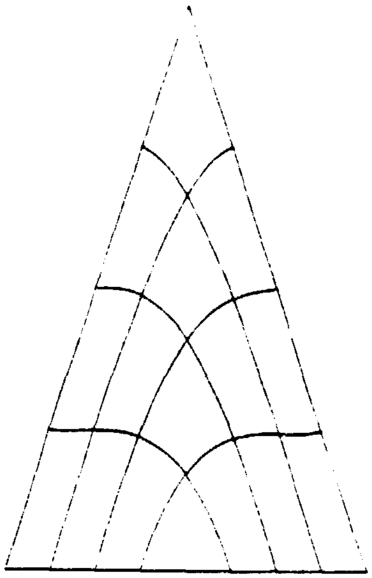

Figure 16c Optimized grid

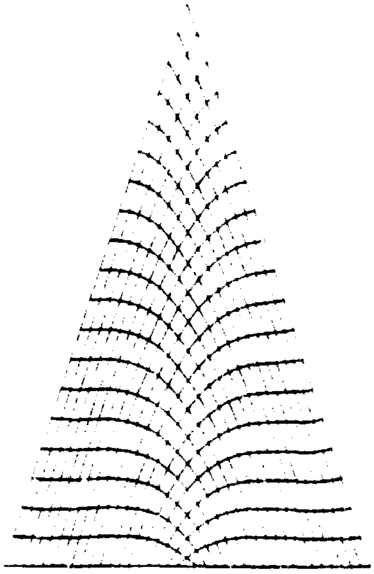

Figure 16d Optimized grid refined

Figure 16 Grids on triangular domain.

grids shown in figure 17, $N_\xi = N_\eta = 9$ and wj=wd=1. Hence, there are five interior knots in both sequence s and sequence t.

Note that $2(N_\xi-2)(N_\eta-2) = 98$. Although this number indicates that a mesh of at least 98 points should be used for the minimization routine, the 8x8 grid shown in figure 17a seems to produce an acceptable grid. One reason for this might be that the initial grid in 17a already appears to be quite smooth and orthogonal at most points. The major problems with orthogonality occur near the areas to which the corners of the square are mapped. These areas are indicated by the arrows in 17a. Figure 18a shows how the initial grid is changed after fifteen iterations of the minimization procedure. Figures 17b and 18b show finer grids. The fifteen iterations of the minimization procedure required 20 minutes and 14 seconds of cpu time.

5.2 Nonconvex Domains

The grids in this section show some of the difficulties in creating grids on domains which are not convex sets.

5.2.1 Nonconvex Quadrilateral

Figure 19 shows the shape of the domain. This example was first mentioned in Section 2.5. The boundary of the unit square is mapped onto the boundary of the domain as indicated in figure 2. Example 2.5-1 shows that T will not map the square homeomorphically onto the domain even

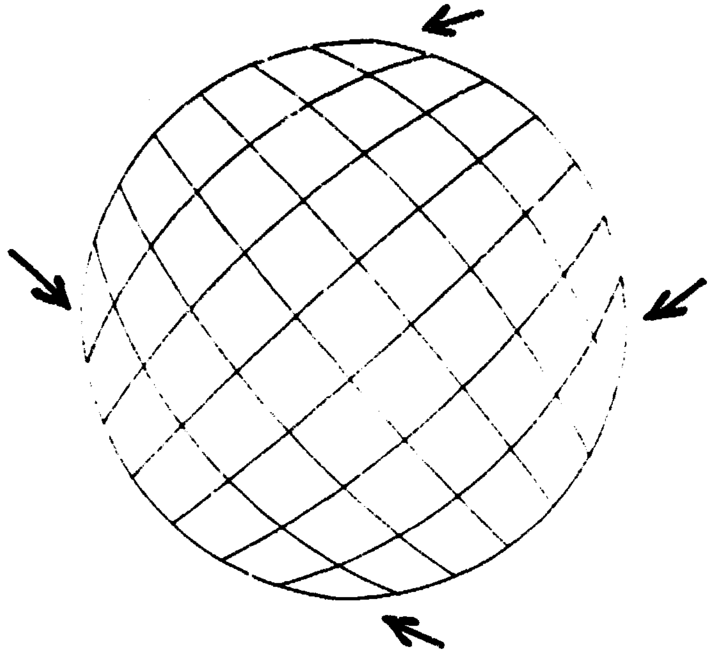

Figure 17a Original grid

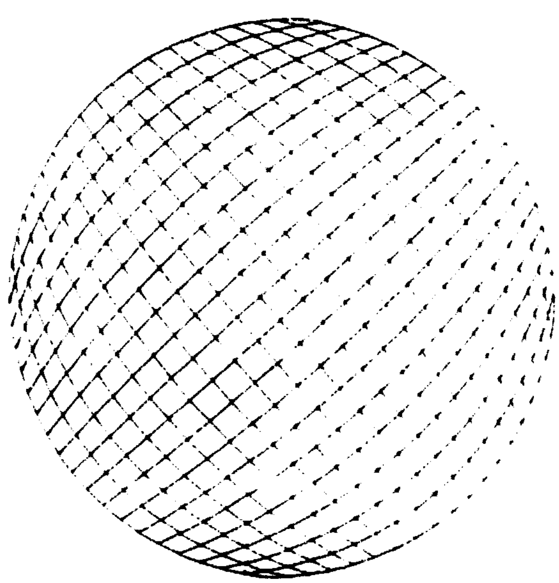

Figure 17b Original grid refined

Figure 17 Grids on circular domain.

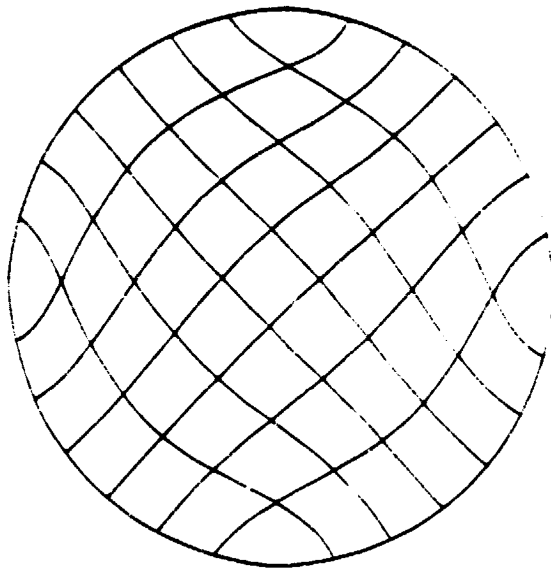

Figure 18a Optimized grid

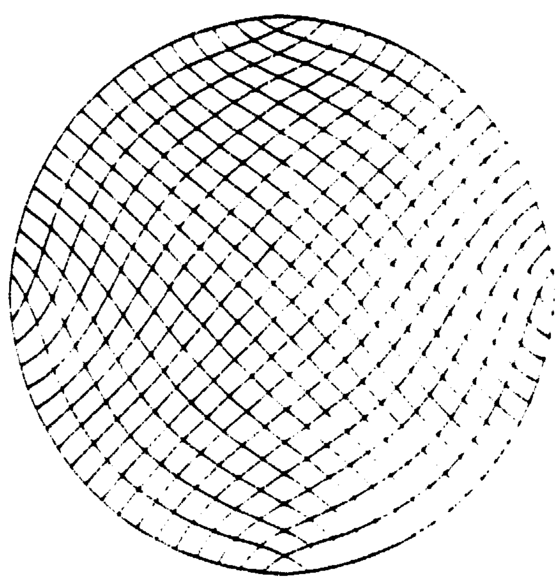

Figure 18b Optimized grid refined

Figure 18 Grids on circular domain after optimization.

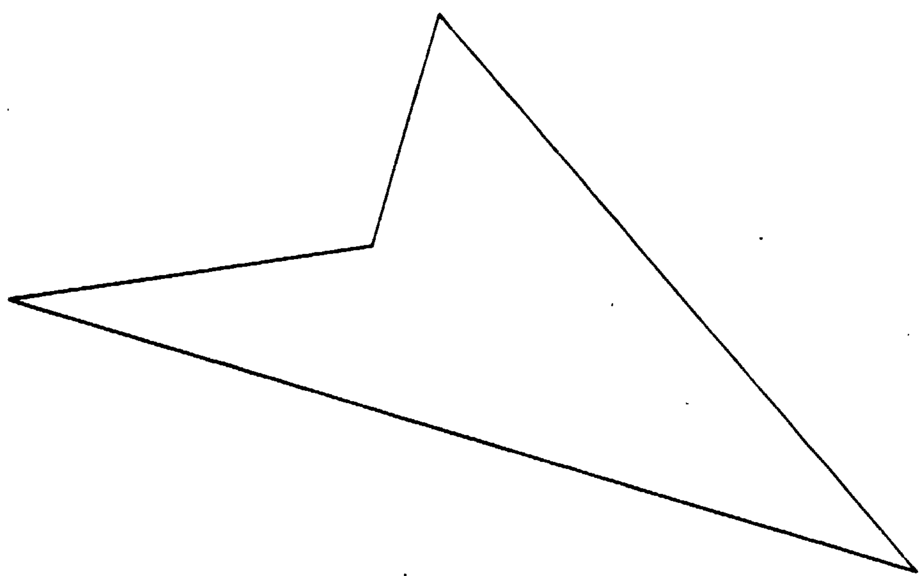

Figure 19 Nonconvex quadrilateral.

after the coefficients are changed. This fact is supported by the negative Jacobian present at one of the corners of the square. The negative sign suggests that points near that corner will be mapped outside of the physical domain. This is confirmed by the grids illustrated. In this example, $N\xi = N\eta = 5$ and wj=wd=1. The 5x5 initial grid shown in figure 20a was used for the minimization procedure. The enlarged picture in figure 20b shows a finer grid. Figure 21a shows the result of four iterations of the minimization procedure. The nonnegative Jacobian requirement pulls the grid lines into the interior of the domain. However, figure 22 shows an enlarged version of the corner which indicates that part of the grid still overlaps the boundary. This means that the minimization routine was unable to restrict all of the coefficients to intervals where the Jacobian of T is nonnegative.

This is further indicated in figure 21b which shows a finer version of the grid in figure 21a. The four iterations of the minimization procedure required 1 minute and 1 second of cpu time.

5.2.2 Puzzle Pieces

The next two domains, illustrated in figure 23, look like pieces from a puzzle. In each case $N\xi = 19$, $N\eta = 5$, wj=1 and wd=10.

Grids on the first domain are shown in figures 24 and 25. The minimization procedure was performed on the

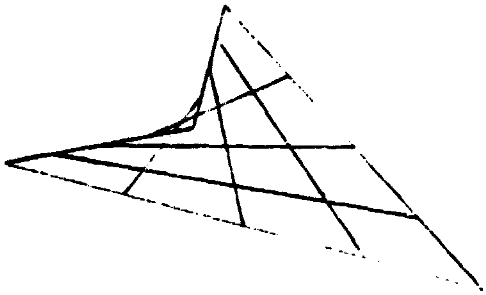

Figure 20a Original grid

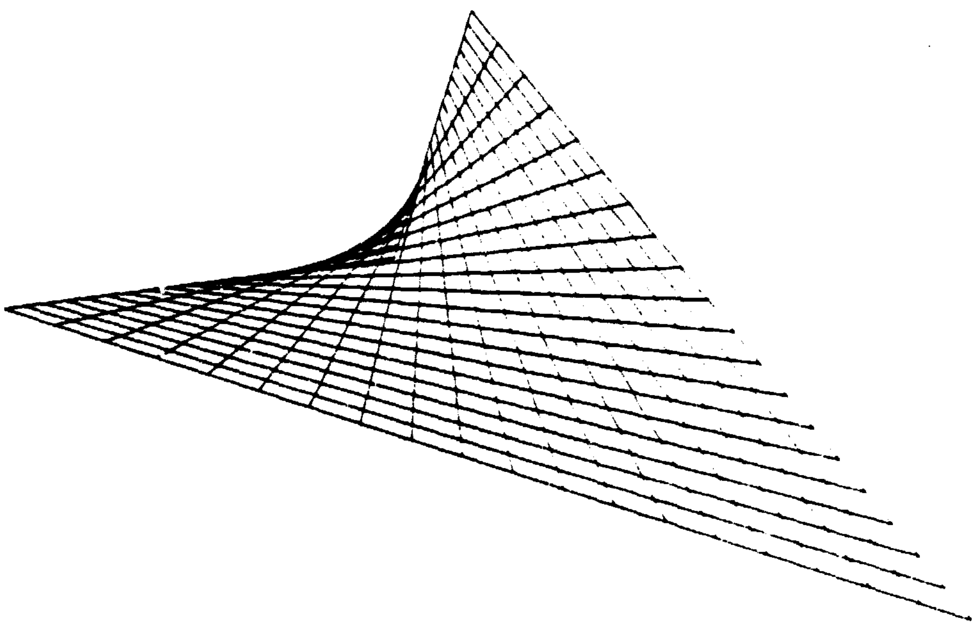

Figure 20b Original grid refined

Figure 20. Grids on nonconvex quadrilateral domain.

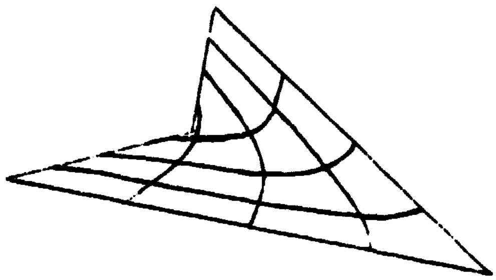

Figure 21a Optimized grid

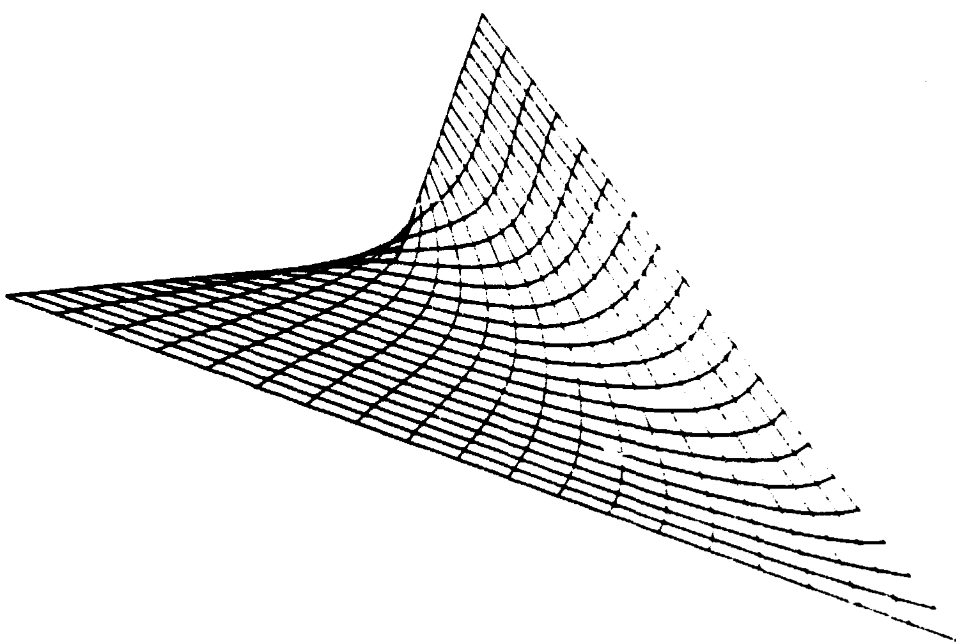

Figure 21b Optimized grid refined

Figure 21 Optimized grids on nonconvex quadrilateral domain.

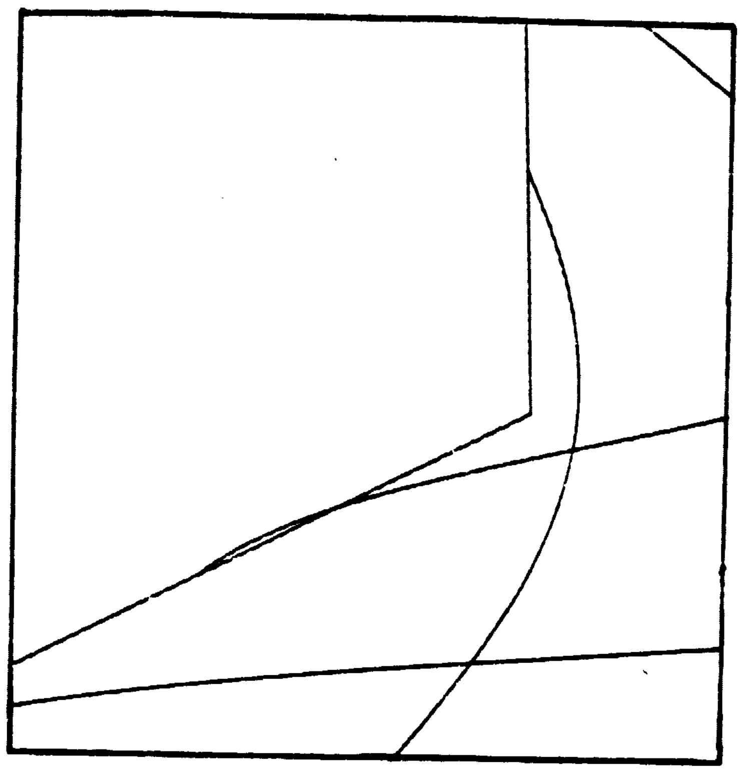

Figure 22 Enlarged corner of optimized grid.

22x8 grid in figure 24a. Figure 24b shows a finer grid. Figure 24c shows the grid obtained after forty iterations and figure 24d shows a finer grid. The grids in figure 25 show how the initial grid changes after two, five and fifteen iterations. The grid obtained after forty iterations is shown again for comparison. On this domain Tentest is able to pull all of the grid lines into the interior of the domain.

Grids on the second domain are shown in figure 26. The initial 22x8 grid is shown in figure 26a and figure 26b shows a finer grid. After forty iterations, the initial grid is transformed into 26c and a finer grid is shown in 26d. Figure 27a shows a grid on the first domain concentrated near the bottom boundary by using an exponential distribution on η with $c=4$. Figure 27b shows a grid on the second domain concentrated near the top by using an exponential distribution on η with $c=-4$.

The forty iterations used for the first domain required 1 hour, 42 minutes and 23 seconds of cpu time, but the second domain required 2 hours, 4 minutes and 46 seconds for forty iterations.

5.3 Grids for Specific Objects

This section deals with grids about particular objects such as an airfoil. The boundaries often have peculiarities which make it difficult to obtain satisfactory grids. In many cases it may be difficult to maintain smoothness in

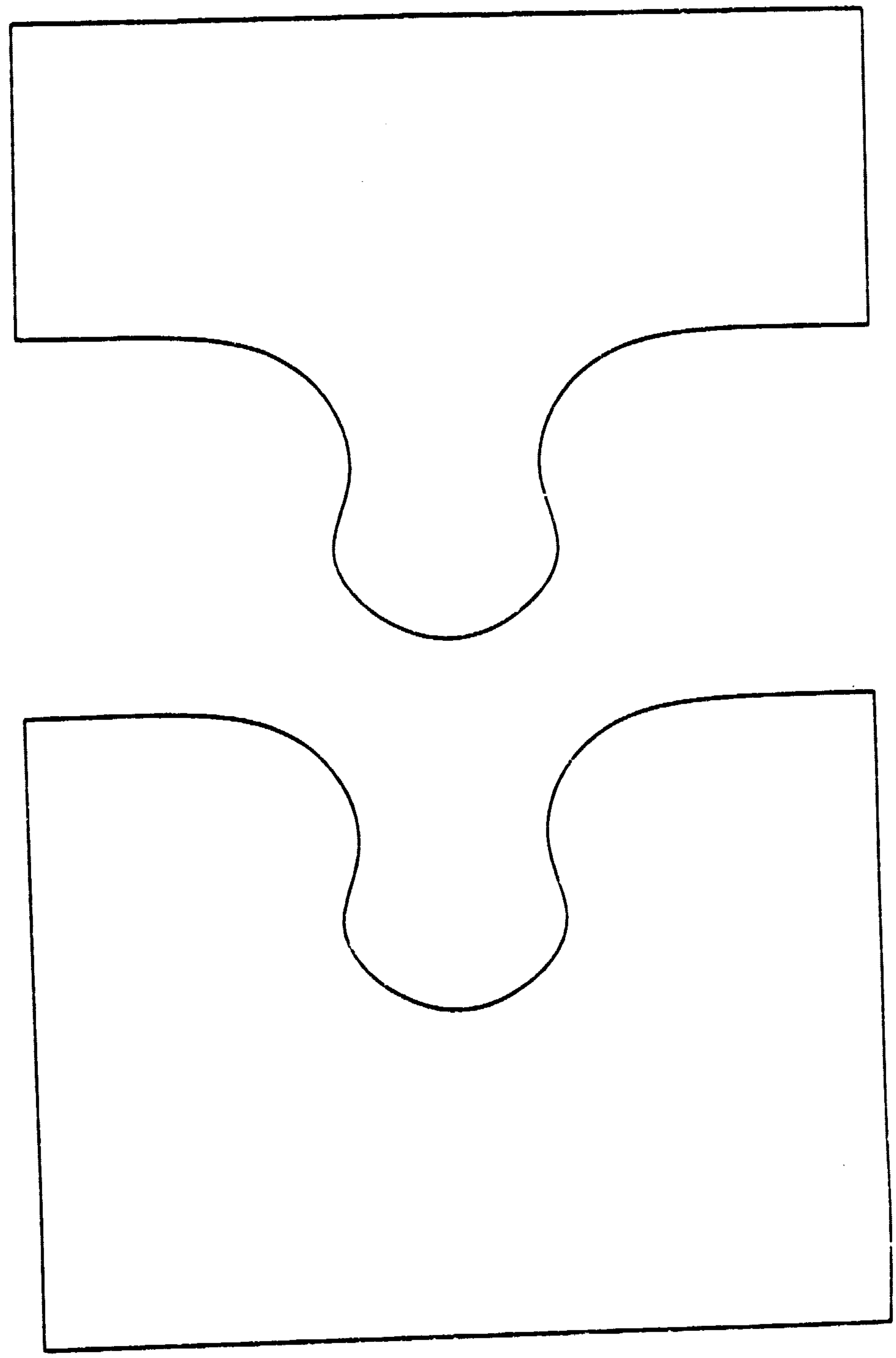

Figure 23 Puzzle shaped domains.

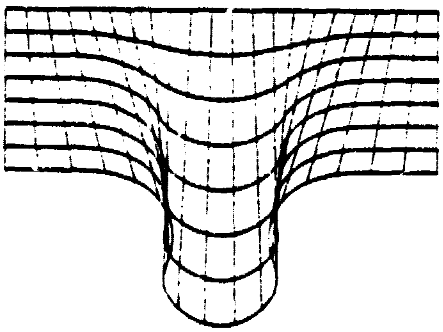

Figure 24a Original grid

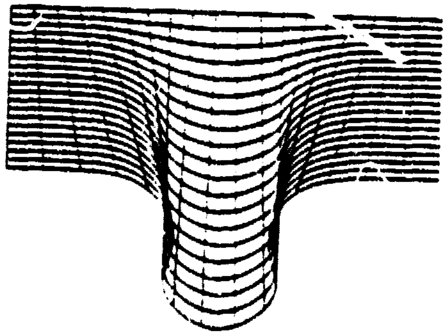

Figure 24b Original grid refined

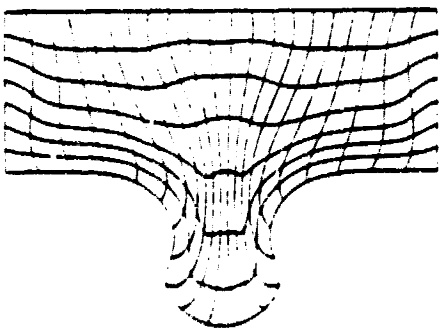

Figure 24c Optimized grid

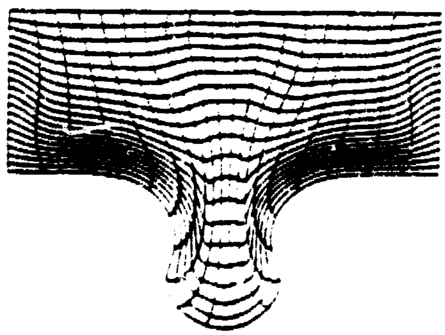

Figure 24d Optimized grid refined

Figure 24 Grids on first puzzle shaped domain.

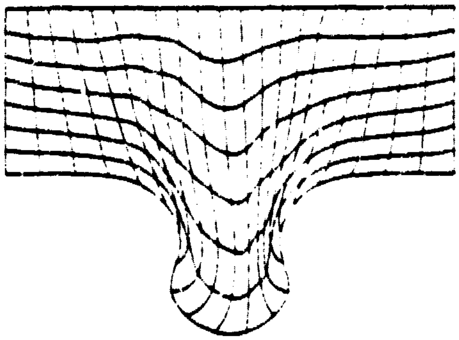

Figure 25a After two iterations

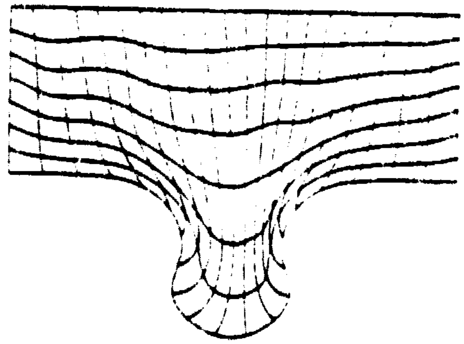

Figure 25b After five iterations

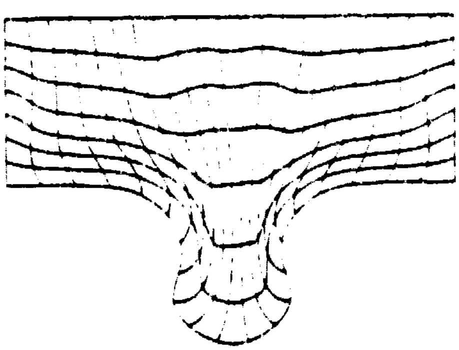

Figure 25c After fifteen iterations

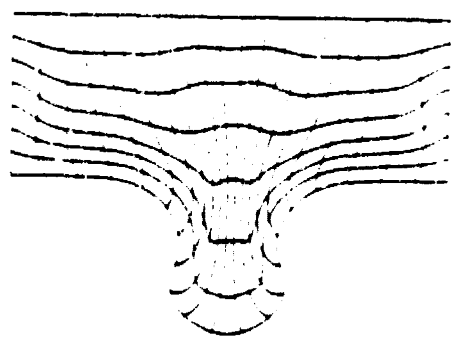

Figure 25d After forty iterations

Figure 25 Grids obtained after various iterations.

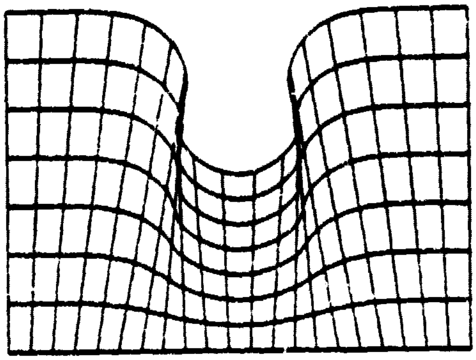

Figure 26a Original grid

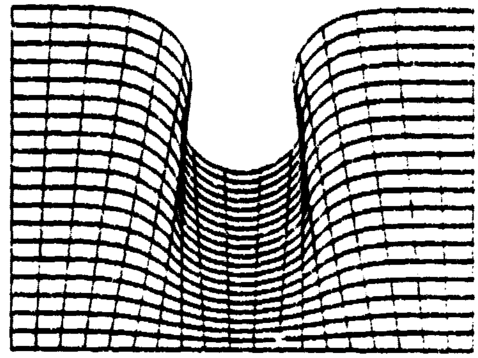

Figure 26b Original grid refined

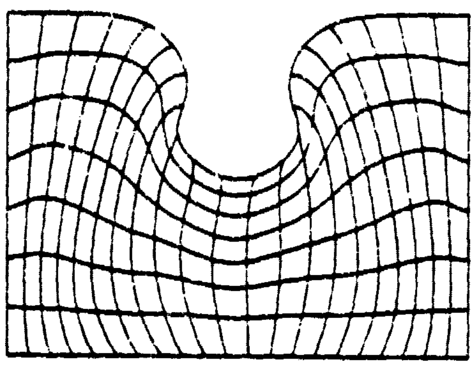

Figure 26c Optimized grid

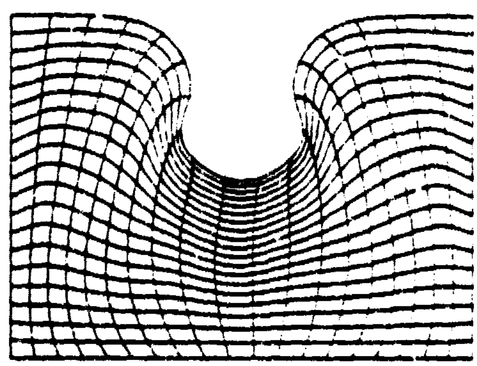

Figure 26d Optimized grid refined

Figure 26 Grids on second puzzle shaped domain.

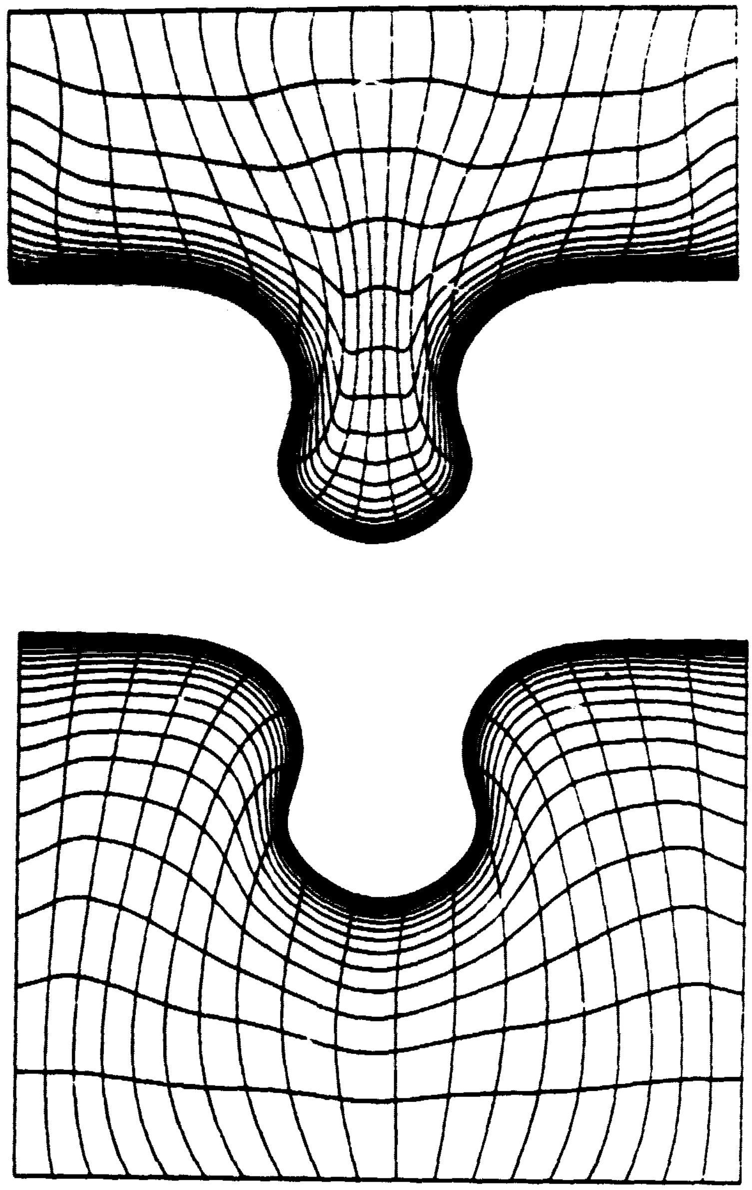

Figure 27 Exponential distributions on puzzle shaped grids.

C-2

the grid while increasing orthogonality. Often the user must try to find an acceptable balance. He must also attempt to concentrate the grids in areas where rapid changes are likely to occur when partial differential equations are solved on the domain.

5.3.1 Airfoil

The grids in this example are for the Kármán - Trefftz airfoil. The parameters $N_\xi = 19$, $N_\eta = 9$, wj=1 and wd=.5. Hence, there are 15 knots in the s sequence and 5 knots in the t sequence. Figure 28 shows how the domain can be viewed as having a boundary consisting of four parts. The minimization procedure was performed on the 21x12 grid in figure 29a. The grid lines appear to be orthogonal everywhere except near boundaries 1, 2 and 4. Note the sharp corners behind the airfoil. After one iteration the corners have been eliminated and the angles of the lines near the airfoil are not as acute. This is shown in figure 29b and in the finer grid in figure 30a.

Solutions on a grid about an airfoil are usually more accurate if a higher concentration of points is placed near the airfoil boundary since this is the area most affected as air moves over the airfoil. Figure 30b shows a 30x30 grid concentrated near the airfoil boundary by using an exponential distribution on η with constant c=4.

The minimization procedure required 6 minutes and 36 seconds of cpu time.

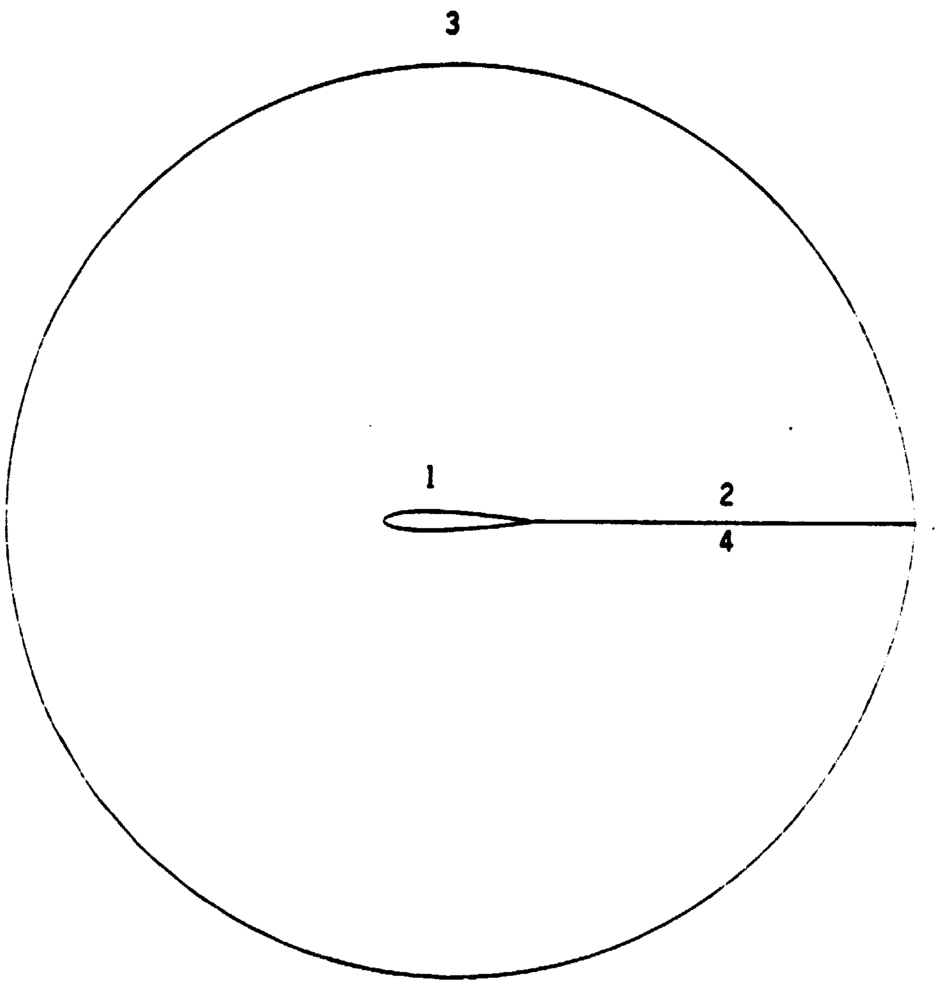

Figure 28 Domain around Kármán - Trefftz airfoil.

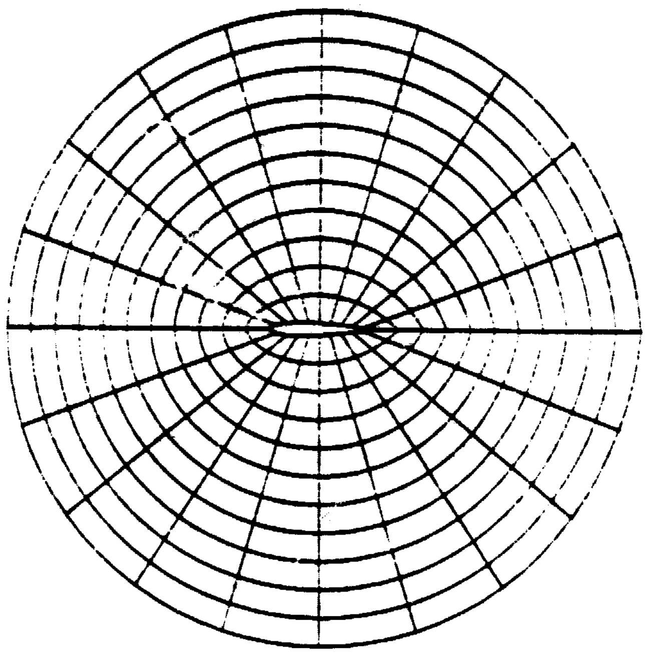

Figure 29a Original grid

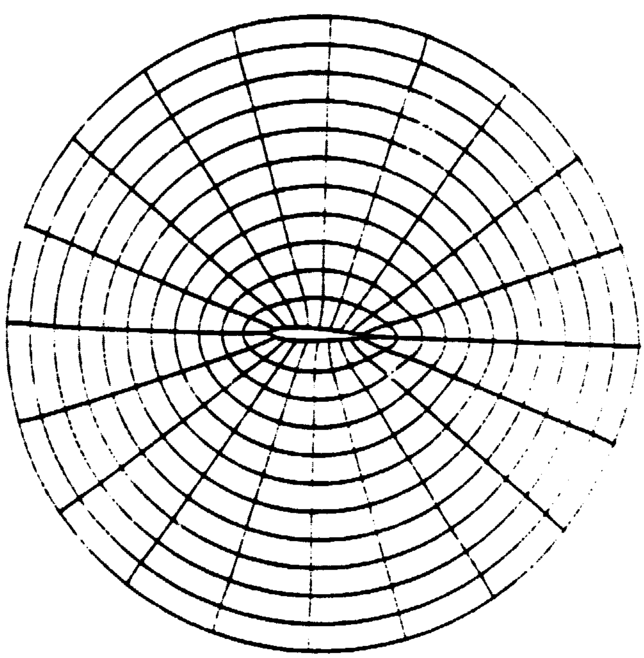

Figure 29b Optimized grid

Figure 29 Grids for Kármán - Trefftz airfoil.

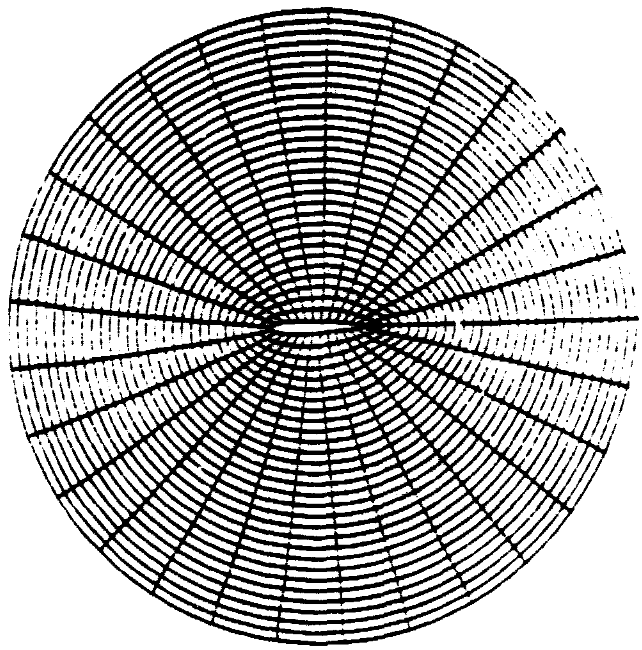

Figure 30a Optimized grid refined

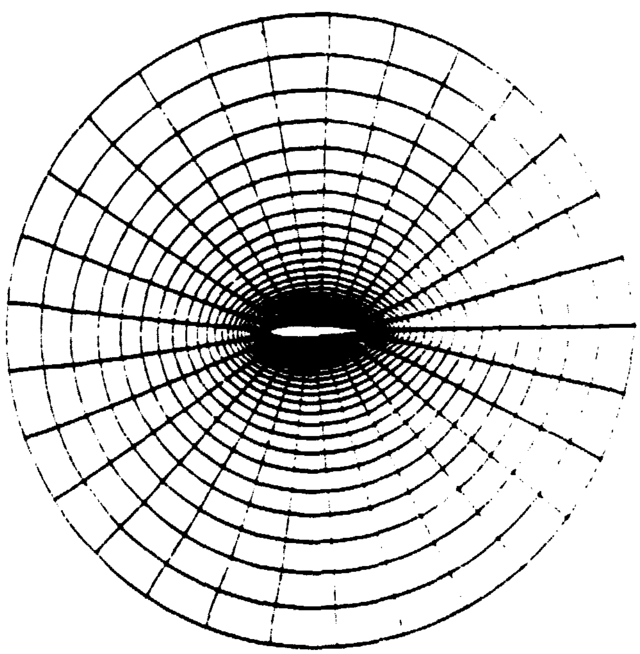

Figure 30b Optimized grid concentrated near airfoil boundary.

Figure 30 Optimized grids for Kármán - Trefftz airfoil.

5.3.2 Spike-Nosed Body

According to [Sm, p. 130], the spike-nosed configuration occurs frequently in supersonic flow. R. E. Smith states that supersonic flow about such bodies is unsteady, with separation occurring near the nose-shoulder region. Therefore, the grids must be concentrated in that area [Sm, p. 48]. The boundary data for the grids shown in this section can be found in [Sm, p. 60]. Rotating the bottom boundary around a horizontal axis of symmetry produces a clearer picture of the actual body. The ratio of the length of the nose to the height of the shoulder is 2.14.

As in the previous example, $N_\xi = 19$, $N_\eta = 9$, wj=1 and wd=0.5. The 21x12 initial grid in figure 31a was used for the minimization procedure. Two iterations produce a small amount of orthogonality near the bottom boundary as shown in figure 31b. Additional iterations produce an undesirable wiggle in the grid lines near the shoulder. Figure 32a shows a finer grid and figure 32b shows a grid concentrated near the bottom boundary by using an exponential distribution on η with c=4.

The two iterations of the minimization procedure required 13 minutes and 1 second of cpu time.

5.3.3 Shuttle

The grids in this example are for a model of the space shuttle. The optimized grids are the result of ten iterations on the 32x12 grid shown in figure 33. Para-

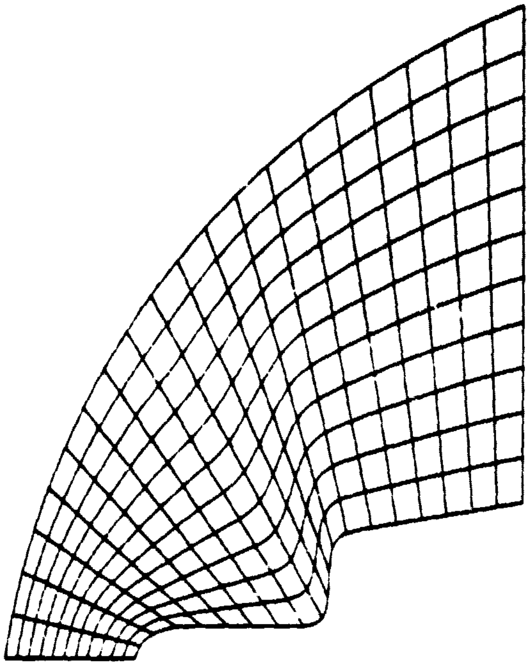

Figure 31a Original grid

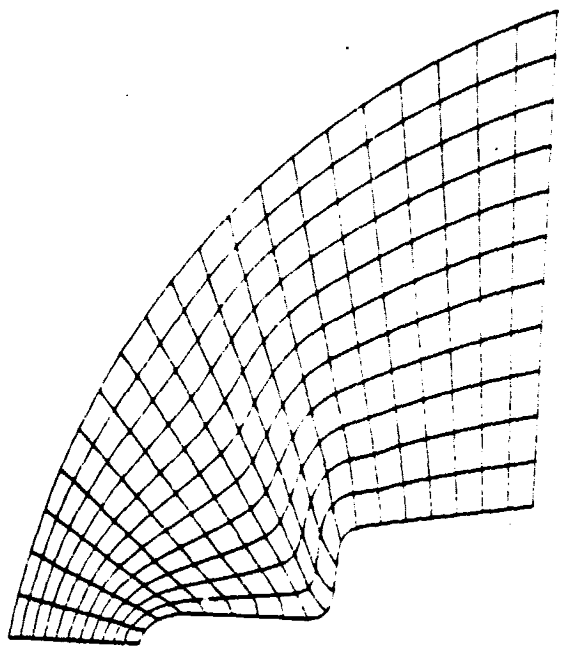

Figure 31b Optimized grid

Figure 31 Grids for spike-nosed body.

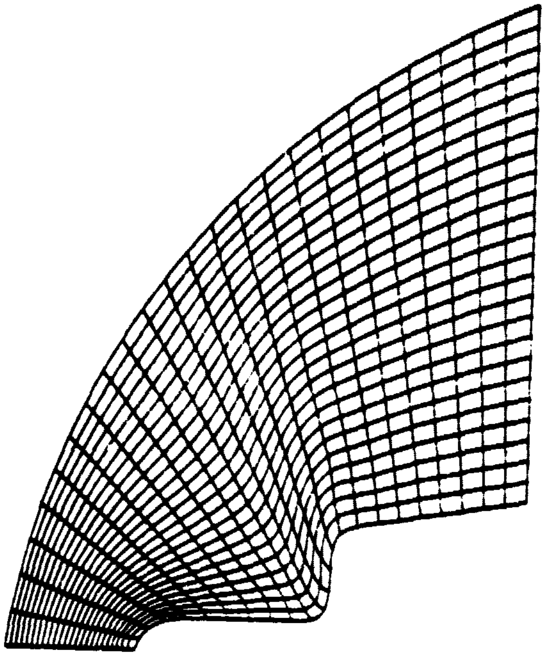

Figure 32a Optimized grid refined

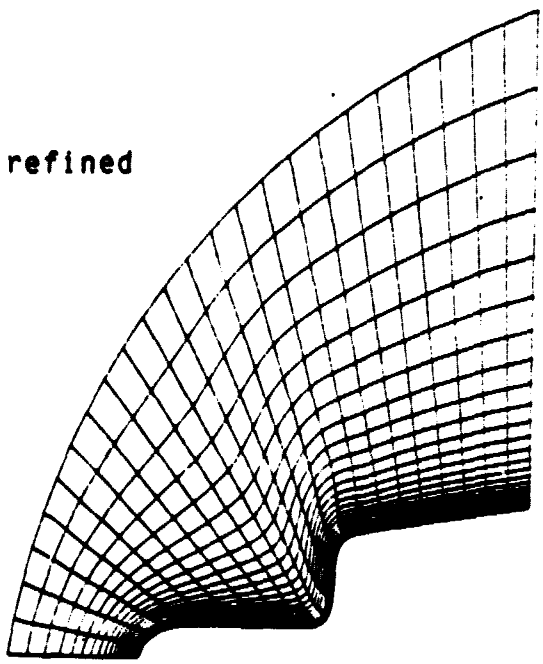

Figure 32b Optimized grid concentrated near boundary of spike-nose body.

Figure 32 Optimized grids for spike-nosed body.

meters N_ξ = 29, N_η = 9, wj=wd=1. Ten iterations of the minimization procedure produce a small amount of orthogonality near the boundary of the shuttle as shown in figure 34. Figure 35 shows an optimized 32x20 grid concentrated near the shuttle boundary using an exponential distribution on η with c=4. The ten iterations of the minimization procedure required 1 hour and 43 minutes of cpu time. The 32x12 grid in figure 33 is the largest grid on which the minimization procedure has been applied.

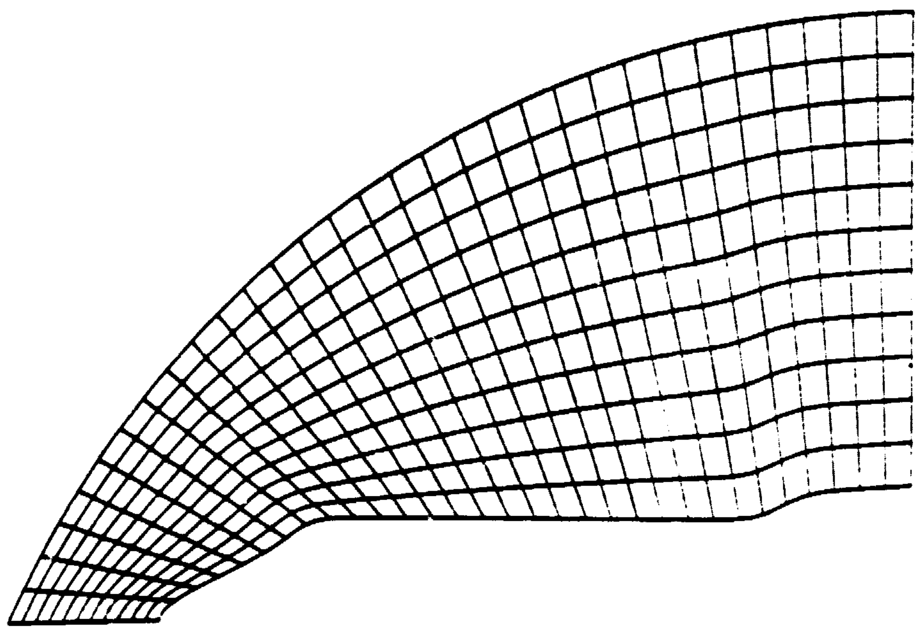

Figure 33 Original grid for shuttle.

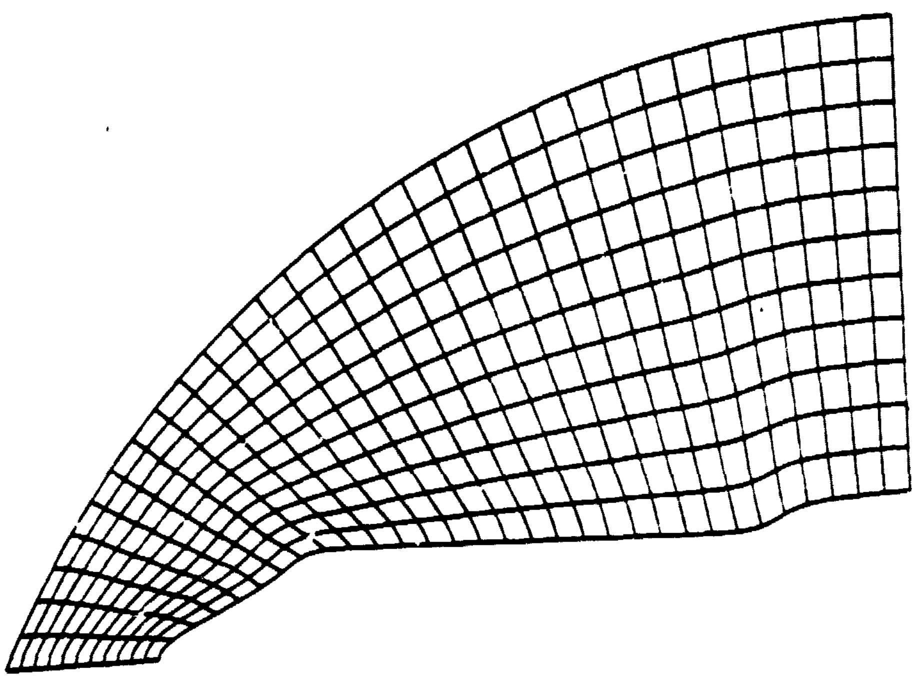

Figure 34 Optimized grid for shuttle.

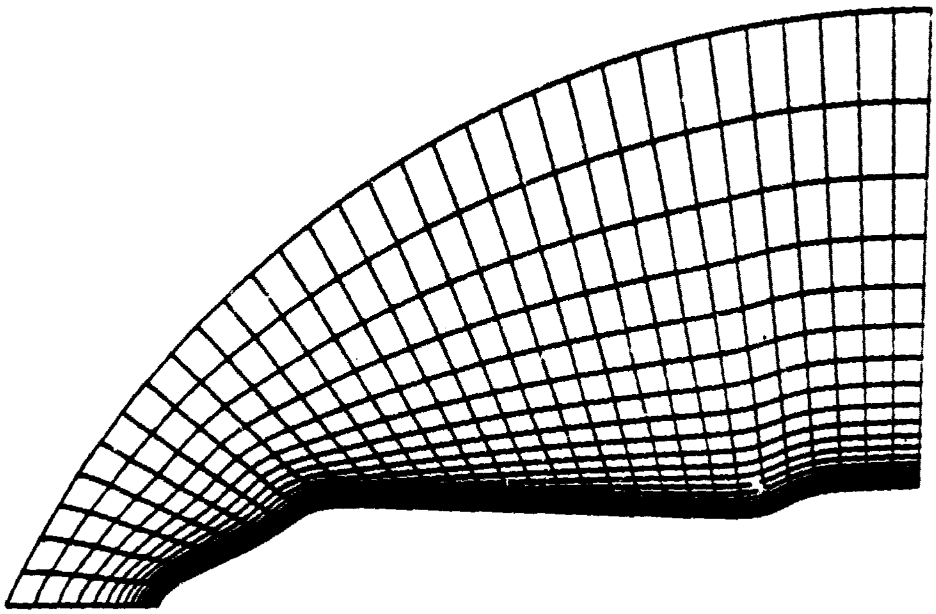

Figure 35 Optimized grid concentrated near shuttle boundary.

6. CONCLUSIONS

This paper has examined an effective algebraic method for creating boundary fitted coordinate systems. The method, which involves a mapping T composed of tensor product B-splines allows one to regulate grid characteristics by adjusting the coefficients of the splines. Modifying the coefficients so that they minimize a smoothing functional enhances the smoothness and orthogonality of the grids generated by T.

The method is implemented in the program TENTEST which gives the user control over the number and concentration of grid points. The user can also regulate the amount of smoothness and orthogonality in the grids by the selection of weight functions for the smoothing functional.

Suggestions for future revisions of TENTEST include the addition of more distribution functions to allow greater control over grid concentration. One might also investigate the possibility of adjusting the boundary coefficients during the optimization process so that the boundary points of the grid are affected by the minimization procedure.

Ultimately, the true test for a grid comes when it is actually used to solve partial differential equations.

Therefore, the next stage of research must include solving problems on several grids produced by TENTEST. Then it may be possible to change the program into an adaptive technique which rearranges the grid points in response to gradient information from the evolving solution.

Once these things are accomplished, one may attempt to use the technique to generate grids on more complicated multiconnected domains. This may involve the study of techniques for grid patching.

Also, the Prime 750 computer is an excellent machine for graphics, but not very fast in solving problems involving a large amount of computations. Hence, the possibility of creating a version of TENTEST which operates efficiently on a vector computer such as the VPS 32 at NASA Langley Research Center should be investigated. This will permit the user to run much larger and more complicated problems.

Finally, once this grid generation technique has been thoroughly developed for two dimensional domains, a three dimensional technique can be attempted. T would become a mapping from the unit cube to the desired physical domain, composed of the tensor product of B-splines in the three coordinate directions. As in the two dimensional case, characteristics of the grid would be changed by changing the coefficients of the tensor product B-splines.

BIBLIOGRAPHY

[B] Bazaraa, Mokhtar and Shetty, C. M. Nonlinear Programming: Theory and Algorithms. New York: John Wiley and Sons, 1979.

[de B] De Boor, Carl. A Practical Guide to Splines. New York: Springer-Verlag, 1978.

[C] Curry, H. B. and Schoenberg, I, J. "On Spline Distributions and Their Limits: The Polya Distribution Functions," Abstract 380t, Bulletin of the American Mathematical Society, Vol. 53, 1947, p. 109.

[D] Dugundji, James. Topology. Boston: Allyn and Bacon, Inc., 1968.

[F] Fleming, Wendell. Functions of Several Variables. New York: Springer-Verlag, 1977.

[G] Goldhaber, Jacob K. and Ehrlich, Gertrude. Algebra. London: The MacMillan Company, 1970.

[H] Heinz, Erhard. "An Elementary Analytic Theory of the Degree of Mapping in n-Dimensional Space." Journal of Mathematics and Mechanics, Vol. 8, No. 2, 1959, p. 231.

[O] Ortega, J. M. and Rheinboldt, W. C. Iterative Solution of Nonlinear Equations in Several Variables. New York: Academic Press, 1970.

[S] Schwartz, J. T. Nonlinear Functional Analysis. New York: Courant Institute of Mathematical Sciences, New York University, 1965.

[Sm] Smith, R. E. "Two-boundary Grid Generation for the Solution of the Three-dimensional Compressible Navier-Stokes Equations," NASA TM 83123, May 1981.

[St] Starius, G. "Constructing Orthogonal Curvilinear Meshes by Solving Initial Value Problems." Numerische Mathematik, Vol. 28, 1977, p. 25.

[SS] Smith, Philip W. and Sritharan, S. S. "On the Grid Generation Methods by Harmonic Mapping on Plane and Curved Surfaces." ICASE Report 84-12, 1984.

[TWM] Thompson, J. E., Warsi, Z. U. A. and Mastin, C. W. "Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - A Review," Journal of Computational Physics, Vol. 47, No. 1, July 1982, p. 1.

APPENDIX

```
C*****************************************************************
      PROGRAM TENTEST
C
C
C        TENTEST MAPS A SQUARE GRID (0,1)X(0,1) ONTO
C  A PHYSICAL DOMAIN OF ARBITRARY SHAPE THROUGH THE USE OF
C  TENSOR PRODUCT B-SPLINES.  THE ORIGINAL KNOT SEQUENCES
C  MAY BE CHOSEN TO HAVE AN EQUALLY SPACED DISTRIBUTION,
C  EXPONENTIAL DISTRIBUTION, OR ARCTANGENT DISTRIBUTION.
C  SIMILAR CHOICES CAN BE MADE FOR THE DISTRIBUTION OF
C  GRIDPOINTS ON THE SQUARE.
C        TENTEST CONSTRUCTS AN INITIAL GRID GENERATION MAPPING
C  CONSISTING OF A LINEAR COMBINATION OF TENSOR PRODUCT
C  B-SPLINES WITH THE COEFFICIENTS CHOSEN SO THAT THE MAPPING
C  YIELDS A VARIATION DIMINISHING SPLINE APPROXIMATION
C  TO THE TRANSFINITE BILINEAR INTERPOLANT OF A
C  FUNCTION WHICH MAPS THE BOUNDARY OF THE UNIT SQUARE
C  ONTO THE BOUNDARY OF THE PHYSICAL DOMAIN.
C        IF THE USER REQUESTS A NEW GRID, TENTEST REARRANGES
C  THE COEFFICIENTS IN AN ATTEMPT TO MINIMIZE A FUNCTIONAL
C  G INVOLVING THE DIFFERENCE IN THE JACOBIAN OF THE GRID
C  GENERATION MAPPING AT ADJACENT MESH POINTS AND THE DOT
C  PRODUCT OF VECTORS TANGENT TO THE GRID LINES ON THE
C  PHYSICAL DOMAIN.
C
C
C  ROUTINES
C
C    EXPONENTIAL
C    ARCTANGENT
C    FIXKNOTS
C    BOUNCOEF
C    INNERCOEF
C    COMSPLINE
C    TENVALF
C    TENSORVAL
C    JACOB
C    CORANGE
C    GF
C    FFMIN
C    CRIT
C    TESTMINO
C    TESTMINR
C    TESTMINL
C    TESTMINB
C    CUBIC
C    EXTREMES
C    NORM
C
C
```

```
C      THE FOLLOWING SUBROUTINES ARE ALSO REQUIRED.
C  THEY MAY BE FOUND IN 'A PRACTICAL GUIDE TO SPLINES'
C  BY CARL DE BOOR, SPRINGER-VERLAG, 1978.
C
C  BSPLVB...  COMPUTES THE VALUE OF ALL POSSIBLE
C             NONZERO B-SPLINES OF A GIVEN ORDER AT
C             A GIVEN POINT.
C
C  BSPLVD...  COMPUTES THE VALUE AND DERIVATIVES
C             OF ALL B-SPLINES WHICH DO NOT VANISH AT
C             A GIVEN POINT
C
C  INTERV...  DETERMINES THE KNOT INTERVAL ON WHICH A
C             GIVEN POINT LIES.  ITS OUTPUT IS THE
C             SUBSCRIPT IDENTIFYING THE KNOT WHICH IS
C             IMMEDIATELY LEFT OF THE POINT.
C
C  BVALUE...  CALCULATES THE JDERIV-TH DERIVATIVE
C             OF A SPLINE FUNCTION WHOSE COEFFICIENTS
C             ARE STORED IN ARRAY BCOEF. THE VALUE OF
C             JDERIV IS PROVIDED BY THE USER.
C
C
C
C      TENTEST USES ROUTINES FROM A PLOT10 GRAPHICS
C  PACKAGE TO PLOT THE GRIDS.
C
C
C  VARIABLES
C
C  NKNOTX,NKNOTY
C      AND
C  NEWNOTX,NEWNOTY..DIMENSIONS FOR SQUARE MESH.
C  NX,NY...         DIMENSION OF SPLINE SPACE IN X
C                   DIRECTION,Y DIRECTION.
C  KX...            QUANTITY OF NUMBERS TO BE ADDED TO THE FRONT
C                   AND BACK OF THE INTERIOR X KNOT SEQUENCE.
C                   ORDER OF B-SPLINES IN X DIRECTION.
C  KY...            QUANTITY OF NUMBERS TO BE ADDED TO THE FRONT
C                   AND BACK OF THE INTERIOR Y KNOT SEQUENCE.
C                   ORDER OF B-SPLINES IN Y DIRECTION.
C  FNX,FNY...       NUMBERS TO BE PLACED AT THE FRONT OF
C                   THE X AND Y KNOT SEQUENCES. RESPECTIVELY.
C  BNX,BNY...       NUMBERS TO BE PLACED AT THE BACK OF THE
C                   X AND Y KNOT SEQUENCES,RESPECTIVELY.
C  INX,INY...       DIMENSIONS OF INTERIOR X AND Y KNOT SEQUENCES,
C                   RESPECTIVELY.
C  INTX,INTY..      INTERIOR X KNOT SEQUENCE,INTERIOR Y KNOT SEQ.
C  TX,TY...         X KNOT SEQUENCE, Y KNOT SEQUENCE.
C  ALPHA,BETA...    ARRAYS CONTAINING COEFFICIENTS OF
C                   TENSOR PRODUCT SPLINE MAPPING.
```

```
C  A,B
C  AND
C  AP,BP...        ARRAYS CONTAINING COORDINATES FOR
C                  SQUARE MESH.
C  LEFTX,LEFTY...  ARRAYS IDENTIFYING KNOT INTERVALS
C                  ON WHICH SQUARE MESH COORDINATES LIE.
C                  (LEFTX(I)=J IMPLIES TX(J)<=A(I)<TX(J+1))
C  W1,W2...        WEIGHTS FOR JACOBIAN,DOT PRODUCT TO
C                  BE USED IN SMOOTHING FUNCTIONAL.
C  X,Y...          TWO-DIMENSIONAL ARRAYS CONTAINING COORDINATES
C                  OF GRID POINTS TO BE PLOTTED.
C  KOUNTE...       VARIABLE USED TO COUNT ITERATIONS OF
C                  MINIMIZATION PROCEDURE, OR
C                  NUMBER OF CALLS TO ROUTINE FFMIN.
C
C
C    AUTHOR:  BONITA VALERIE SAUNDERS
C    DATE:    JULY 1985
C
C
C
C
C***************************************************************
C
C
      COMMON/COEF/ALPHA(100,100),BETA(100,100)
      COMMON/KNOTS/TX(100),TY(100)
      COMMON/PARAM/FKOUNT
      COMMON/PARAM2/A(100),B(100),NX,NY,KX,KY
     * ,LEFTX(100),LEFTY(100)
      COMMON/KNOT/NKNOTX,NKNOTY
      COMMON/WEIGHTS/W1,W2
      COMMON/SPLINES/XSPLINE(50,100,2),YSPLINE(50,100,2)
      COMMON/RANGE/IFIRST(100),ILAST(100),JFIRST(100)
     * ,JLAST(100)
      REAL X(100,100),BCOEF(100),INTX(100),INTY(100)
     *  ,Y(100,100),AP(100),BP(100)
      CHARACTER*10 NAME
      INTEGER*2 STRING(28)
      INTEGER*2 NUMB,DATE(3)
      INTEGER*2 TIME,TIME1,TIME2
      EQUIVALENCE (STRING(1),DATE)
      EQUIVALENCE (STRING(4),TIME)
      EQUIVALENCE (STRING(5),TIME1)
      EQUIVALENCE (STRING(7),TIME2)
      NUMB=28
      CALL TIMDAT(STRING,NUMB)
      WRITE(1,111) DATE
  111 FORMAT(3A2)
      WRITE(1,222) TIME,TIME1,TIME2
  222 FORMAT(I6,I6,I6)
```

```
      PI=3.14159
      OPEN(12,FILE='TENSORDAT')
      OPEN(13,FILE='NEWDATA')
      OPEN(14,FILE='ORGRID')
      OPEN(16,FILE='ORIG2')
      W1=0
      W2=0
      KOUNTE=0
      PRINT*,'INPUT NKNOTX,NKNOTY'
      READ(1,*) NKNOTX,NKNOTY
      NSAVEX=NKNOTX
      NSAVEY=NKNOTY
      PRINT*,'WHAT IS KX'
      READ(1,*) KX
      PRINT*,'WHAT IS KY'
      READ(1,*) KY
      READ(12,*) FNX,BNX,FNY,BNY
      READ(12,*) INX,INY
      READ(12,*) (INTX(I),I=1,INX)
      READ(12,*) (INTY(I),I=1,INY)
      PRINT*,'DISTRIBUTION FOR X KNOTS'
      PRINT*,'1=EQUALLY SPACED,2=EXPONENTIAL,3=ARCTANGENT'
      READ(1,*) KODEX
      IF(KODEX-2) 5,10,15
   10 CALL EXPONENTIAL(INTX,INX,2.)
      GOTO 5
   15 CALL ARCTANGENT(INTX,INX,5.)
    5 PRINT*,'DISTRIBUTION FOR Y KNOTS'
      PRINT*,'1=EQUALLY SPACED,2=EXPONENTIAL,3=ARCTANGENT'
      READ(1,*) KODEY
      IF(KODEY-2) 16,18,19
   18 CALL EXPONENTIAL(INTY,INY,2.)
      GO TO 16
   19 CALL ARCTANGENT(INTY,INY,5.)
   16 CONTINUE
      NX=INX+KX
      NY=INY+KY
      CALL FIXKNOTS(KX,KY,FNX,FNY,BNX,BNY,
     * INX,INY,INTX,INTY)
      CALL BOUNCOEF(KX,KY,NX,NY)
      CALL INNERCOEF(KX,KY,NX,NY)
      WRITE(16,*) ((ALPHA(I,J),J=1,NY),I=1,NX)
      WRITE(16,*) ((BETA(I,J),J=1,NY),I=1,NX)
      DO 50 I=1,NKNOTX
      A(I)=FLOAT(I-1)/(NKNOTX-1)
      DO 20 J=1,NKNOTY
      B(J)=FLOAT(J-1)/(NKNOTY-1)
      IF(A(I).GE.1.0) A(I)=.99999
      IF(B(J).GE.1.0) B(J)=.99999
   20 CONTINUE
   50 CONTINUE
```

```
      PRINT*,'DISTRIBUTION FOR COMPUTATIONAL X'
      PRINT*,'1=EQUALLY SPACED, 2=EXPONENTIAL,3=ARCTANGENT'
      READ(1,*) KODEXX
      IF(KODEXX-2) 22,24,26
   24 CALL EXPONENTIAL(A,NKNOTX,2.)
      GOTO 22
   26 CALL ARCTANGENT(A,NKNOTX,5.)
   22 CONTINUE
      PRINT*,'DISTRIBUTION FOR COMPUTATIONAL Y'
      PRINT*,'1=EQUALLY SPACED,2=EXPONENTIAL,3=ARCTANGENT'
      READ(1,*) KODEYY
      IF(KODEYY-2) 30,32,34
   32 CALL EXPONENTIAL(B,NKNOTY,2.)
      GOTO 30
   34 CALL ARCTANGENT(B,NKNOTY,5.)
   30 CALL COMSPLINE
      CALL CORANGE(NKNOTX,NKNOTY)
      DO 70 I=1,NKNOTX
      DO 60 J=1,NKNOTY
      CALL TENVALF(ALPHA,LEFTX(I),LEFTY(J),KX,KY,I,J,
     * X(I,J),0,0)
      CALL TENVALF(BETA,LEFTX(I),LEFTY(J),KX,KY,I,J
     * ,Y(I,J),0,0)
   60 CONTINUE
   70 CONTINUE
      PRINT*,'JACOBIAN YES(1) OR NO(0)'
      READ(1,*) JCODE
      IF(JCODE.EQ.1) GOTO 700
      PRINT*,'COMPUTE DERIVATIVES YES(1) OR NO(0)'
      READ(1,*) KCODE
      IF(KCODE.EQ.0) GOTO 80
      PRINT*,'INPUT DERIVATIVES DESIRED FOR X,Y DIRECT'
      READ(1,*) JDX,JDY
      PRINT*,'       X               Y          X COMP DERIV  Y COMP DE
      DO 600 II=1,NKNOTX
      DO 500 JJ=1,NKNOTY
      CALL TENVALF(ALPHA,LEFTX(II),LEFTY(JJ),KX,KY,II,JJ,
     * XD,JDX,JDY)
      CALL TENVALF(BETA,LEFTX(II),LEFTY(JJ),KX,KY,II,JJ,
     * YD,JDX,JDY)
      PRINT*,A(II),B(JJ),XD,YD
  500 CONTINUE
  600 CONTINUE
      GOTO 80
  700 CALL JACOB(NX,NY,KX,KY,A,B)
   80 CONTINUE
      CALL EXTREMES(X,Y,TMAX,TMIN,NKNOTX,NKNOTY)
      CALL NORM(X,Y,TMAX,TMIN,NKNOTX,NKNOTY)
      PAUSE
      NN=2
      NAME='PRODUCT GRID'
```

```
      WRITE(13,*) NAME
      WRITE(13,*)NKNOTX,NKNOTY,NN,NN
      WRITE(13,*) (X(I,1),I=1,NKNOTX)
      WRITE(13,*) (Y(I,1),I=1,NKNOTY)
      WRITE(13,*) (X(I,NKNOTY),I=1,NKNOTX)
      WRITE(13,*) (Y(I,NKNOTY),I=1,NKNOTX)
      WRITE(13,*) X(1,1),X(1,NKNOTY)
      WRITE(13,*) Y(1,1),Y(1,NKNOTY)
      WRITE(13,*) X(NKNOTX,1),X(NKNOTX,NKNOTY)
      WRITE(13,*) Y(NKNOTX,1),Y(NKNOTX,NKNOTY)
      CALL INITT(960)
      CALL TWINDO(0,760,0,760)
      CALL DWINDO(-.07,1.07,-.07,1.07)
 338  DO 200 I=1,NKNOTX
      CALL MOVEA(X(I,1),Y(I,1))
      DO 100 J=1,NKNOTY
      CALL DRAWA(X(I,J),Y(I,J))
 100  CONTINUE
 200  CONTINUE
      DO 400 J=1,NKNOTY
      CALL MOVEA(X(1,J),Y(1,J))
      DO 300 I=1,NKNOTX
      CALL DRAWA(X(I,J),Y(I,J))
 300  CONTINUE
 400  CONTINUE
      WRITE(14,*) NKNOTY,NKNOTX
      WRITE(14,*) ((X(I,J),I=1,NKNOTX),J=1,NKNOTY)
      WRITE(14,*) ((Y(I,J),I=1,NKNOTX),J=1,NKNOTY)
      CALL MOVABS(0,760)
      CALL ANMODE
      PRINT*,'ITERATION',KOUNTE
      PRINT*,'  NX=',NX,'    NY=',NY
      IF(KOUNTE.EQ.0) GOTO 410
      PRINT*,'JACOBIAN WEIGHT=',W1
      PRINT*,'ORTHOG WEIGHT=',W2
      PRINT*,'OPTIMIZED ON',NSAVEX,'     BY',NSAVEY,'     GRID'
 410  KOUNTE=KOUNTE+1
      PRINT*,'DO YOU WANT TO CHANGE THE GRID, YES OR NO(0)'
      READ(1,*) KODE
      IF(KODE.EQ.0) GOTO 339
      PRINT*,'CURRENT WEIGHTS ARE',W1,W2
      PRINT*,'NEW WEIGHTS, YES(1) OR NO(0)'
      READ(1,*) KW
      IF(KW.EQ.0) GOTO 401
      PRINT*,'INPUT WEIGHTS FOR JACOB,ORTHOG'
      READ(1,*) W1,W2
 401  CONTINUE
      NKNOTX=NSAVEX
      NKNOTY=NSAVEY
      CALL FFMIN(ERMAX)
      PRINT*,'OUTPUT JACOBIAN,YES(1) OR NO(0)'
```

```
      READ(1,*) JKODE
      IF(JKODE.EQ.0) GOTO 399
      CALL JACOB(NX,NY,KX,KY,A,B)
  399 CONTINUE
      PRINT*,'CHANGE NUM OF GRIDPOINTS, YES(1) OR NO(0)'
      READ(1,*) KODE
      IF(KODE.EQ.0) GOTO 501
      PRINT*,'ENTER NUMBER OF GRIDPOINTS FOR X DIRECTION,
     * Y DIRECTION'
      READ(1,*) NEWNOTX,NEWNOTY
      NKNOTX=NEWNOTX
      NKNOTY=NEWNOTY
      GOTO 502
  501 CONTINUE
      NEWNOTX=NKNOTX
      NEWNOTY=NKNOTY
  502 CONTINUE
      CALL ERASE
      DO 450 I=1,NEWNOTX
      DO 425 J=1,NEWNOTY
      AP(I)=FLOAT(I-1)/(NEWNOTX-1)
      BP(J)=FLOAT(J-1)/(NEWNOTY-1)
      IF(AP(I).GE.1.0) AP(I)=.99999
      IF(BP(J).GE.1.0) BP(J)=.9999
  425 CONTINUE
  450 CONTINUE
      PRINT*,'DISTRIBUTION FOR COMPUTATIONAL X'
      PRINT*,'1=EQUALLY SPACED,2=EXPONENTIAL,3=ARCTAN'
      READ(1,*) KODE3X
      IF(KODE3X-2) 452,454,456
  454 CALL EXPONENTIAL(AP,NEWNOTX,2.)
      GOTO 452
  456 CALL ARCTANGENT(AP,NEWNOTX,5.)
  452 CONTINUE
      PRINT*,'DISTRIBUTION FOR COMPUTATIONAL Y'
      READ(1,*) KODE3Y
      IF(KODE3Y-2) 458,460,462
  460 CALL EXPONENTIAL(BP,NEWNOTY,2.)
      GOTO 458
  462 CALL ARCTANGENT(BP,NEWNOTY,5.)
  458 CONTINUE
      DO 480 I=1,NEWNOTX
      DO 470 J=1,NEWNOTY
      CALL TENSORVAL(ALPHA,NX,NY,KX,KY,AP(I),
     * BP(J),X(I,J),0,0)
      CALL TENSORVAL(BETA,NX,NY,KX,KY,AP(I),BP(J),
     * Y(I,J),0,0)
  470 CONTINUE
  480 CONTINUE
      CALL EXTREME3(X,Y,TMAX,TMIN,NEWNOTX,NEWNOTY)
      CALL NORM(X,Y,TMAX,TMIN,NEWNOTX,NEWNOTY)
```

```
      PAUSE
      GOTO 338
339   CONTINUE
      CALL FINITT(0,760)
      PRINT*,'KX IS ',KX
      PRINT*,'KY IS ',KY
      PRINT*,'DISTRIBUTIONS'
      PRINT*,'1=EQUALLY SPACED,2=EXPONENTIAL,3=ARCTANGENT'
      PRINT*,'X KNOT DIST. IS',KODEX
      PRINT*,'Y KNOT DIST. IS',KODEY
      PRINT*,'COMPUTATIONAL X DIST IS',KODEXX
      PRINT*,'COMPUTATIONAL Y DIST IS',KODEYY
      CLOSE(12)
      CLOSE(13)
      CLOSE(14)
      CALL TIMDAT(STRING,NUMB)
      WRITE(1,222) TIME,TIME1,TIME2
      STOP
      END


C
C
      SUBROUTINE EXPONENTIAL(X,N,AC)
C
C  THIS ROUTINE PRODUCES AN EXPONENTIAL DISTRIBUTION OF
C  POINTS BY SUBSTITUTING AN ORIGINAL SET OF NUMBERS
C  U LYING BETWEEN 0 AND 1 INTO THE EXPONENTIAL
C  FUNCTION (EXP(A*U)-1.)/(EXP(AC)-1) WHERE AC IS A
C  PARAMETER WHOSE VALUE IS SUPPLIED BY THE USER.
C
C  VARIABLES
C
C    X...THIS AN ARRAY WHICH ON INPUT CONTAINS THE ORIGINAL
C        SET OF NUMBERS AND ON OUTPUT CONTAINS THE EXPONENTIAL
C        DISTRIBUTION OF NUMBERS.
C    N...SIZE OF ARRAY X
C    AC..PARAMETER IN EXPONENTIAL FUNCTION
C
      REAL X(100)
      DO 10 I=1,N
      U=X(I)
      X(I)=(EXP(AC*U)-1.)/(EXP(AC)-1.)
      IF (X(I).GE.1.0) X(I)=.99999
10    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE ARCTANGENT(X,N,AC)
```

```
C
C  THIS ROUTINE PRODUCES AN ARCTANGENT DISTRIBUTION
C  OF POINTS BY SUBSTITUTING AN ORIGINAL SET OF NUMBERS
C  U LYING BETWEEN 0 AND 1 INTO THE ARCTANGENT FUNCTION
C  (ATAN(AC)-ATAN(-ATAN(-AC))/(ATAN(AC)-ATAN(-AC))
C  WHERE AC IS A PARAMETER WHOSE VALUE IS
C  SUPPLIED BY THE USER.
C
C  VARIABLES
C
C    X...THIS AN ARRAY WHICH ON INPUT CONTAINS THE ORIGINAL SET
C        OF NUMBERS AND ON OUTPUT CONTAINS THE ARCTANGENT DISTRIBUT
C        OF NUMBERS.
C    N...SIZE OF ARRAY X
C    AC..PARAMETER IN ARCTANGENT FUNCTION
C
C
      REAL X(100)
      DO 10 I=1,N
      U=X(I)
      X(I)=(ATAN(2.*AC*U-AC)-ATAN(-AC))
    * /(ATAN(AC)-ATAN(-AC))
      IF(X(I).GE.1.0) X(I)=.99999
  10  CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE FIXKNOTS(KX,KY,FNX,FNY,BNX,BNY,INX,
    *   INY,INTX,INTY)
C
C    THIS ROUTINE PLACES KX COPIES OF FNX AT THE
C  BEGINNING OF THE INTERIOR X KNOT SEQUENCE,
C  KY COPIES OF FNY AT THE BEGINNING OF THE
C  INTERIOR Y KNOT SEQUENCE, KX COPIES OF BNX AT THE
C  END OF THE INTERIOR X KNOT SEQUENCE AND KY COPIES
C  OF BNY AT THE END OF THE INTERIOR Y KNOT SEQUENCE.
C
C
C    INPUT
C
C
C  KX...          QUANTITY OF NUMBERS TO BE ADDED TO THE FRONT
C                 AND BACK OF THE INTERIOR X KNOT SEQUENCE.
C                 ORDER OF B-SPLINES IN X DIRECTION.
C  KY...          QUANTITY OF NUMBERS TO BE ADDED TO THE FRONT
C                 AND BACK OF THE INTERIOR Y KNOT SEQUENCE. ORDER
C                 OF B-SPLINES IN Y DIRECTION.
C  FNX,FNY...     NUMBERS TO BE PLACED AT THE FRONT OF
C                 THE X AND Y KNOT SEQUENCES, RESPECTIVELY.
```

```
C  BNX,BNY...    NUMBERS TO BE PLACED AT THE BACK OF THE
C                X AND Y KNOT SEQUENCES,RESPECTIVELY.
C  INX,INY...    DIMENSIONS OF INTERIOR X AND Y KNOT SEQUENCES,
C                RESPECTIVELY.
C  INTX,INTY...  INTERIOR X KNOT SEQUENCE,INTERIOR Y KNOT SEQ.
C
C
C    OUTPUT(IN COMMON)
C
C  TX,TY...       X KNOT SEQUENCE, Y KNOT SEQUENCE
C
C
      COMMON/KNOTS/TX(100),TY(100)
      REAL INTX(1),INTY(1)
      NX=INX+KX
      NY=INY+KY
      DO 100 I=KX+1,NX
      J=I-KX
      TX(I)=INTX(J)
  100 CONTINUE
      DO 200 I=KY+1,NY
      J=I-KY
      TY(I)=INTY(J)
  200 CONTINUE
      DO 5 I=1,KX
      TX(I)=FNX
      INDEX=I+NX
      TX(INDEX)=BNX
    5 CONTINUE
      DO 6 I=1,KY
      TY(I)=FNY
      INDEX=I+NY
      TY(INDEX)=BNY
    6 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE BOUNCOEF(KX,KY,NX,NY)
C
C    THIS ROUTINE COMPUTES THE BOUNDARY COEFFICIENTS
C  FOR TWO TENSOR PRODUCT B-SPLINES (X,Y COMPONENTS).
C  SPECIFICALLY, IT COMPUTES ALPHA(I,1),BETA(I,1) AND
C  ALPHA(I,NY),BETA(I,NY) FOR I=1 TO NX;
C  ALPHA(1,J),BETA(1,J) AND ALPHA(NX,J),BETA(NX,J)
C  FOR J=1 TO NY.
C  COEFFICIENTS ARE CHOSEN SO THAT THERE IS A
C  VARIATION DIMINISHING APPROXIMATION ALONG THE
C  BOUNDARY.
```

```
C
C   INPUT
C
C  KX,KY...ORDER OF SPLINE IN X DIRECTION, Y DIRECTION
C  NX,NY...DIMENSION OF SPLINE IN X DIRECT, Y DIRECT.
C
C  BOTH COEFFICIENT SEQUENCES WILL HAVE DIMENSION
C  NX*NY
C
C
C    OUTPUT
C
C  (IN COMMON)
C  BOUNDARY COEFFICIENTS PLACED IN ALPHA,BETA ARRAYS.
C
      COMMON/COEF/ALPHA(100,100),BETA(100,100)
      COMMON/KNOTS/TX(100),TY(100)
      DIMENSION TXSTAR(100),TYSTAR(100)
      G1X(T)=2.*T+1.
      G1Y(T)=0.
      G2X(T)=3.+T
      G2Y(T)=2.*T
      G3X(T)=4.*T
      G3Y(T)=2.
      G4X(T)=1.-T
      G4Y(T)=2.*T
      PI=3.14159
      DO 100 I=1,NX
      SUM=0.
      DO 50 J=1,KX-1
      SUM=SUM+TX(I+J)
   50 CONTINUE
      TXSTAR(I)=SUM/(KX-1)
  100 CONTINUE
      DO 200 J=1,NY
      SUM=0.
      DO 150 K=1,KY-1
      SUM=SUM+TY(J+K)
  150 CONTINUE
      TYSTAR(J)=SUM/(KY-1)
  200 CONTINUE
      DO 300 I=1,NX
      A=TXSTAR(I)
      ALPHA(I,1)=G1X(A)
      BETA(I,1)=G1Y(A)
      ALPHA(I,NY)=G3X(A)
      BETA(I,NY)=G3Y(A)
  300 CONTINUE
      DO 400 J=1,NY
      B=TYSTAR(J)
      ALPHA(NX,J)=G2X(B)
```

```
      BETA(NX,J)=G2Y(B)
      ALPHA(1,J)=G4X(B)
      BETA(1,J)=G4Y(B)
  400 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE INNERCOEF(KX,KY,NX,NY)
C
C     THIS ROUTINE COMPUTES THE INNER COEFFICIENTS
C  FOR TWO TENSOR PRODUCT B-SPLINES (X,Y COMPONENTS)
C  SPECIFICALLY, IT COMPUTES ALPHA(I,J),BETA(I,J) FOR
C  I=2 TO NX-1, J=2 TO NY-1.
C  THE COEFFICIENTS ARE COMPUTED THROUGH THE USE OF
C  TRANSFINITE BILINEAR INTERPOLATION.  THE
C  INTERPOLANTS ARE EVALUATED AT POINTS SO THAT THE
C  RESULTING COEFFICIENTS PRODUCE A VARIATION
C  DIMINISHING SPLINE APPROXIMATION TO THE
C  TRANSFINITE BILINEAR INTERPOLANT.
C
C    INPUT
C
C  KX,KY...ORDER OF B-SPLINES IN X DIRECTION,Y DIRECTION
C  NX,NY...DIMENSION OF SPLINE SPACE IN X DIRECT,Y DIRECT
C
C  BOTH COEFFICIENT SEQUENCES WILL HAVE DIMENSION
C  NX*NY
C  TX,TY(IN COMMON)...KNOT SEQUENCE FOR X DIRECT,Y
C  DIRECTION
C
C
C    OUTPUT(IN COMMON)
C
C  INTERIOR COEFFICIENTS PLACED IN ALPHA,BETA ARRAYS
C
C
      COMMON/COEF/ALPHA(100,100),BETA(100,100)
      COMMON/KNOTS/TX(100),TY(100)
      DIMENSION TXSTAR(100),TYSTAR(100)
      G1X(T)=2.*T+1.
      G1Y(T)=0.
      G2X(T)=3.+T
      G2Y(T)=2.*T
      G3X(T)=4.*T
      G3Y(T)=2.
      G4X(T)=1.-T
      G4Y(T)=2.*T
      FX(X,Y)=G1X(X)*(1.-Y)+G3X(X)*Y
     * +G2X(Y)*X+(1.-X)*G4X(Y)
```

```
     *-G1X(0)*(1.-X)*(1.-Y)-G2X(0)*X*(1.-Y)
     *-G3X(0)*(1.-X)*Y-G2X(1.)*X*Y
      FY(X,Y)=G1Y(X)*(1.-Y)+G3Y(X)*Y
     * +G2Y(Y)*X+(1.-X)*G4Y(Y)
     * -G1Y(0)*(1.-X)*(1.-Y)-G2Y(0)*X*(1.-Y)
     *-G3Y(0)*(1.-X)*Y-G2Y(1.)*X*Y
      PI=3.14159
      DO 100 I=1,NX
      SUM=0.
      DO 50 J=1,KX-1
      SUM=SUM+TX(I+J)
 50   CONTINUE
      TXSTAR(I)=SUM/(KX-1)
 100  CONTINUE
      DO 200 J=1,NY
      SUM=0.
      DO 150 K=1,KY-1
      SUM=SUM+TY(J+K)
 150  CONTINUE
      TYSTAR(J)=SUM/(KY-1)
 200  CONTINUE
      DO 400 I=2,NX-1
      DO 300 J=2,NY-1
      A=TXSTAR(I)
      B=TYSTAR(J)
      ALPHA(I,J)=FX(A,B)
      BETA(I,J)=FY(A,B)
 300  CONTINUE
 400  CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE COMSPLINE
C
C       THIS ROUTINE COMPUTES AND STORES THE FUNCTION
C  VALUES AND FIRST DERIVATIVES OF ALL THE NON-
C  VANISHING B-SPLINES AT EACH POINT OF A SQUARE
C  MESH.
C       IN ADDITION, IT DETERMINES THE KNOT INTERVAL
C  ON WHICH EACH MESH COORDINATE LIES.
C
C  INPUT
C  (IN COMMON)
C    A,B...            ARRAYS CONTAINING COORDINATES FOR SQUARE
C                      MESH, POINTS OF EVALUATION FOR B-SPLINES.
C    NKNOTX,NKNOTY...NUMBER OF ELEMENTS IN A,B.
C    KX,KY...          ORDER OF B-SPLINES IN X DIRECTION, Y
C                      DIRECTION.
C    TX,TY...          X KNOT SEQUENCE FOR B-SPLINES,Y KNOT
```

```
C                          SEQUENCE FOR B-SPLINES.
C     NX,NY...             DIMENSION OF SPLINE SPACE IN X DIRECTION,
C                          Y DIRECTION
C
C    OUTPUT
C    (IN COMMON)
C
C     XSPLINE,YSPLINE..THREE DIMENSIONAL ARRAYS CONTAINING
C                          FUNCTION VALUES AND FIRST DERIVATIVES OF
C                          B-SPLINES IN X DIRECTION, Y DIRECTION AT
C                          EACH ELEMENT OF A,B.  THE FIRST SUBSCRIPT
C                          IDENTIFIES THE B-SPLINE, THE SECOND
C                          SUBSCRIPT REPRESENTS THE POINT OF EVALUATION
C                          AND THE THIRD SUBSCRIPT (1 OR 2) INDICATES
C                          WHETHER THE VALUE REPRESENTS A FUNCTION
C                          EVALUATION OR DERIVATIVE EVALUATION.  HENCE
C                          XSPLINE(3,2,1) WILL CONTAIN THE VALUE OF THE
C                          B-SPLINE IN THE X DIRECTION, B(3), AT A(2).
C
C     LEFTX,LEFTY...       ARRAYS IDENTIFYING KNOT INTERVALS ON
C                          WHICH MESH COORDINATES LIE.  LEFTX(3)=4 WOULD
C                          MEAN THAT A(3) LIES BETWEEN TX(4) AND TX(4+1)
C
C    REQUIRED ROUTINES:
C     BSPLVD
C     BSPLVB
C     INTERV
C
       COMMON/PARAM2/A(100),B(100),NX,NY,KX,KY,LEFTX(100)
     * ,LEFTY(100)
       COMMON/SPLINES/XSPLINE(50,100,2),YSPLINE(50,100,2)
       COMMON/KNOTS/TX(100),TY(100)
       COMMON/KNOT/NKNOTX,NKNOTY
       REAL DBIATX(4,2),WORK(4,4)
       IDERIV=2
       JDERIV=2
C
C    INITIALIZATION
C
       NUMX=NX+KX
       NUMY=NY+KY
       DO 3 I=1,NX
       DO 2 J=1,NKNOTX
       DO 1 KK=1,2
       XSPLINE(I,J,KK)=0.
     1 CONTINUE
     2 CONTINUE
     3 CONTINUE
       DO 9 I=1,NY
       DO 8 J=1,NKNOTY
       DO 7 KK=1,2
```

```
      YSPLINE(I,J,KK)=0.
    7 CONTINUE
    8 CONTINUE
    9 CONTINUE
      DO 25 I=1,NKNOTX
      CALL INTERV(TX,NUMX,A(I),LEFTX(I),MFLAG)
      CALL BSPLVD(TX,KX,A(I),LEFTX(I),WORK,DBIATX,IDERIV)
      DO 252 JJ=1,2
      DO 251 II=1,KX
      IB=LEFTX(I)-KX+II
      IF(IB.LE.0) GOTO 251
      XSPLINE(IB,I,JJ)=DBIATX(II,JJ)
  251 CONTINUE
  252 CONTINUE
   25 CONTINUE
      PRINT*,'DO ALL X SEQ EQUAL ALL Y SEG YES(1) OR NO(0)'
      READ(1,*) KODE
      IF(KODE.EQ.0) GOTO 28
      DO 27 JJ=1,NKNOTY
      LEFTY(JJ)=LEFTX(JJ)
      DO 272 KK=1,2
      DO 271 II=1,NY
      YSPLINE(II,JJ,KK)=XSPLINE(II,JJ,KK)
  271 CONTINUE
  272 CONTINUE
  27  CONTINUE
      GOTO 293
  28  CONTINUE
      DO 29 J=1,NKNOTY
      CALL INTERV(TY,NUMY,B(J),LEFTY(J),MFLAG)
      CALL BSPLVD(TY,KY,B(J),LEFTY(J),WORK,DBIATX,JDERIV)
      DO 292 JJ=1,2
      DO 291 II=1,KY
      JB=LEFTY(J)-KY+II
      IF(JB.LE.0) GOTO 291
      YSPLINE(JB,J,JJ)=DBIATX(II,JJ)
  291 CONTINUE
  292 CONTINUE
  29  CONTINUE
  293 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE TENVALF(ARR,LEFTX,LEFTY,KX,KY
     * ,I,J,VALUE,IDERIV,JDERIV)
C
C     TENVALF COMPUTES PARTIAL DERIVATIVES FOR A TENSOR
C  PRODUCT SPLINE FUNCTION AS INDICATED BY THE PARAMETERS
C  IDERIV,JDERIV.
```

```
C
C  INPUT
C
C    ARR...             ARRAY OF COEFFICIENTS FOR SPLINE
C                       FUNCTION
C    I,J...             INDICIES FOR POINT OF EVALUATION
C                       (A(I),B(J)),WHERE A,B ARE ARRAYS
C                       WHICH CONTAIN COORDINATES OF A
C                       SQUARE MESH.
C    LEFTX...           VALUE INDICATING KNOT INTERVAL
C                       ON WHICH A(I) LIES
C    LEFTY...           VALUE INDICATING KNOT INTERVAL
C                       ON WHICH B(J) LIES
C    KX,KY...           ORDER OF B-SPLINES IN X-DIRECTION,
C                       Y DIRECTION
C    IDERIV...          ORDER OF DERIVATIVE DESIRED FOR X
C                       DIRECTION
C    JDERIV...          ORDER OF DERIVATIVE DESIRED FOR Y
C                       DIRECTION
C    (IN COMMON)
C    XSPLINE,YSPLINE..ARRAYS CONTAINING FUNCTION
C                       VALUES AND FIRST DERIVATIVES OF B-SPLINES IN
C                       X DIRECTION, Y DIRECTION AT EACH POINT
C                       GIVEN IN ARRAYS A,B
C
C  OUTPUT
C
C    VALUE...           VALUE OF TENSOR PRODUCT SPLINE
C                       OR DERIVATIVE
C
      COMMON/SPLINES/XSPLINE(50,100,2),YSPLINE(50,100,2)
      DIMENSION ARR(100,100)
      VALUE=0.
      DO 13 JJ=1,KY
      JB=LEFTY-KY+JJ
      IF(JB.LE.0) GOTO 13
      DO 12 II=1,KX
      IB=LEFTX-KX+II
      IF(IB.LE.0) GOTO 12
      VALUE=VALUE+ARR(IB,JB)*XSPLINE(IB,I,IDERIV+1)
     * *YSPLINE(JB,J,JDERIV+1)
   12 CONTINUE
   13 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE TENSORVAL(ARR,NX,NY,KX,KY,A,B,VALUE
     * ,JDX,JDY)
C  THIS ROUTINE CALCULATES THE VALUE OF A TENSOR PRODUCT
C  SPLINE AT THE POINT (A,B).
```

```
C
C   INPUT
C
C  ARR...  ARRAY OF COEFFICIENTS
C  NX,NY...DIMENSION OF SPLINE SPACE IN X DIRECT.,
C          Y DIRECTION. ARRAY ARR WILL HAVE
C          DIMENSION NX*NY.
C  KX,KY...ORDER OF B-SPLINES IN X DIRECT,Y DIRECT
C  A,B...  POINT OF EVALUATION
C
C  (IN COMMON)
C  TX,TY...KNOT SEQUENCE FOR X DIRECT,
C          Y DIRECT.
C
C
C   OUTPUT
C
C  VALUE...VALUE OF TENSOR PRODUCT SPLINE AT (A,B)
      COMMON/KNOTS/TX(100),TY(100)
      DIMENSION BCOEF(100),ARR(100,100)
      CALL INTERV(TY,NY,B,LEFTY,MFLAG)
      VALUE=0.
      DO 10 J=1,KY
   10 BCOEF(J)=BVALUE(TX,ARR(1,LEFTY-KY+J),NX,KX,A,JDX)
      VALUE=BVALUE(TY(LEFTY-KY+1),BCOEF,KY,KY,B,JDY)
      RETURN
      END
C
C
C
      SUBROUTINE JACOB(NX,NY,KX,KY,A,B)
C
C  THIS ROUTINE COMPUTES THE JACOBIAN OF A
C  TENSOR PRODUCT B-SPLINE MAPPING.
C
C   INPUT
C
C     NX,NY...     DIMENSION OF SPLINE IN X
C                  DIRECTION,Y DIRECTION
C     KX,KY...     ORDER OF B-SPLINES IN X DIRECTION
C                  Y DIRECTION
C     A,B...       ARRAYS CONTAINING EVALUATION
C                  POINTS
C     (IN COMMON)
C     ALPHA,BETA...COEFFICIENTS OF B-SPLINES
C     NKNOTX,NKNOTY.NUMBER OF ELEMENTS IN
C                       ARRAYS A,B
C
C   OUTPUT(TO TERMINAL)
C
C   A,B...           ARRAYS CONTAINING EVALUATION
```

```
C                       POINTS
C     AJCOBIAN...       JACOBIAN AT EACH POINT
C
      COMMON/COEF/ALPHA(100,100),BETA(100,100)
      COMMON/KNOT/NKNOTX,NKNOTY
      REAL A(100),B(100)
      PRINT*,'       X               Y            JACOBIAN'
      PRINT*,' '
      DO 20 I=1,NKNOTX
      DO 10 J=1,NKNOTY
      II=I
      JJ=J
      CALL TENSORVAL(ALPHA,NX,NY,KX,KY,A(II),B(JJ),
     * XDFIRST,1,0)
      CALL TENSORVAL(ALPHA,NX,NY,KX,KY,A(II),B(JJ),
     * YDFIRST,0,1)
      CALL TENSORVAL(BETA,NX,NY,KX,KY,A(II),B(JJ),XDSEC,
     * 1,0)
      CALL TENSORVAL(BETA,NX,NY,KX,KY,A(II),B(JJ),YDSEC,
     * 0,1)
      AJCOBIAN=XDFIRST*YDSEC-XDSEC*YDFIRST
      PRINT*,A(II),B(JJ),AJCOBIAN
   10 CONTINUE
   20 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE CORANGE(NKNOTX,NKNOTY)
C
C      CORANGE DETERMINES THE RANGES OF SUMMATION
C   NEEDED TO MINIMIZE THE SMOOTHING FUNCTIONAL G IN
C   EACH COORDINATE DIRECTION.
C          SPECIFICALLY, FOR EACH COEFFICIENT ALPHA(I,J),
C   OR BETA(I,J), IT DETERMINES THE RANGE OF INDICES FOR
C   THE MESH POINTS LYING ON THE SUPPORT OF THE TENSOR
C   PRODUCT B-SPLINE HAVING SUBSCRIPTS I,J.  IT PLACES
C   THE SMALLEST IN IFIRST(I) AND JFIRST(J) AND THE
C   LARGEST INDICES IN ILAST(I) AND JLAST(J).  THESE VALUES
C   DETERMINE WHICH TERMS IN G SHOULD BE SUMMED WHEN
C   MINIMIZING THE SMOOTHING FUNCTION IN THE DIRECTION
C   REPRESENTED BY THE COEFFICIENT ALPHA(I,J) OR BETA(I,J).
C
C   INPUT
C
C     NKNOTX,NKNOTY...DIMENSIONS FOR SQUARE MESH
C                     OR NUMBER OF ELEMENTS IN ARRAYS A,B
C     (IN COMMON)
C     A,B...          ARRAYS CONTAINING COORDINATES FOR
C                     SQUARE MESH
```

```fortran
C    NX,NY...          DIMENSION OF SPLINE SPACE IN X DIRECT.
C                      ,Y DIRECTION OR
C                      TOTAL NO. OF B-SPLINES IN X DIRECTION,
C                      Y DIRECTION
C    KX,KY...          ORDER OF B-SPLINES IN X DIRECTION,
C                      Y DIRECTION
C    LEFTX,LEFTY...    ARRAYS IDENTIFYING KNOT INTERVALS
C                      ON WHICH SQUARE MESH COORDINATES
C                      LIE. (LEFTX(I)=J IMPLIES
C                      TX(J)<=A(I)<TX(J+1)
C
C  OUTPUT
C
C    IFIRST,JFIRST..   ARRAYS CONTAINING STARTING
C                      POINTS FOR THE RANGES OF SUMMATION
C                      CORRESPONDING TO EACH COEFFICIENT
C                      ALPHA(I,J),BETA(I,J).
C    ILAST,JLAST..     ARRAYS CONTAINING FINAL
C                      POINTS FOR THE RANGES OF SUMMATION
C                      CORRESPONDING TO EACH COEFFICIENT
C                      ALPHA(I,J),BETA(I,J)
C
      COMMON/PARAM2/A(100),B(100),NX,NY,KX,KY,
     * LEFTX(100),LEFTY(100)
      COMMON/RANGE/IFIRST(100),ILAST(100),JFIRST(100),
     * JLAST(100)
      DO 50 I=1,NX
      IF(I.EQ.1) THEN
      II=1
      ELSE
      II=IFIRST(I-1)
      ENDIF
   10 IF(I.GE.LEFTX(II)-(KX-1).AND.I.LE.LEFTX(II))
     *  GOTO 20
      II=II+1
      GOTO 10
   20 IFIRST(I)=II
   30 II=II+1
      IF(II.GT.NKNOTX) GOTO 40
      IF(I.LT.LEFTX(II)-(KX-1)) GOTO 40
      GOTO 30
   40 ILAST(I)=II-1
   50 CONTINUE
      DO 100 J=1,NY
      IF(J.EQ.1) THEN
      JJ=1
      ELSE
      JJ=JFIRST(J-1)
      ENDIF
   60 IF(J.GE.LEFTY(JJ)-(KY-1).AND.J.LE.LEFTY(JJ))
     *   GOTO 70
```

```
      JJ=JJ+1
      GOTO 60
   70 JFIRST(J)=JJ
   80 JJ=JJ+1
      IF(JJ.GT.NKNOTY) GOTO 90
      IF(J.LT.LEFTY(JJ)-(KY-1)) GOTO 90
      GOTO 80
   90 JLAST(J)=JJ-1
  100 CONTINUE
      RETURN
      END
C
C
C
      REAL FUNCTION GF(II,JJ)
C
C
C       FUNCTION GF COMPUTES THE SUM OF THE TERMS IN
C  THE SMOOTHING FUNCTIONAL G OVER THE RANGES INDICATED
C  BY IFIRST(II),JFIRST(JJ) AND ILAST(II), JLAST(JJ).
C
C  INPUT
C
C    II,JJ...          INDICES FOR COEFFICIENT INVOLVED
C                      IN MINIMIZATION.
C    (IN COMMON)
C    NKNOTX,NKNOTY...DIMENSIONS FOR SQUARE MESH
C                    OR NUMBER OF ELEMENTS IN ARRAYS A,B
C    A,B...          ARRAYS CONTAINING COORDINATES FOR
C                    SQUARE MESH
C    NX,NY...        DIMENSION OF SPLINE SPACE IN X DIRECT.
C                    ,Y DIRECTION OR
C                    TOTAL NO. OF B-SPLINES IN X DIRECTION,
C                    Y DIRECTION
C    KX,KY...        ORDER OF B-SPLINES IN X DIRECTION,
C                    Y DIRECTION
C    LEFTX,LEFTY...  ARRAYS IDENTIFYING KNOT INTERVALS
C                    ON WHICH SQUARE MESH COORDINATES
C                    LIE. (LEFTX(I)=J IMPLIES
C                    TX(J)<=A(I)<TX(J+1)
C
C    IFIRST,JFIRST.. ARRAYS CONTAINING STARTING
C                    POINTS FOR THE RANGES OF SUMMATION
C                    CORRESPONDING TO EACH COEFFICIENT
C                    ALPHA(I,J),BETA(I,J).
C    ILAST,JLAST..   ARRAYS CONTAINING FINAL
C                    POINTS FOR THE RANGES OF SUMMATION
C                    CORRESPONDING TO EACH COEFFICIENT
C                    ALPHA(I,J),BETA(I,J)
C    W1,W2...        WEIGHTS FOR JACOBIAN, DOT PRODUCT
C
```

```
C  OUTPUT
C
C    GF...            PARTIAL SUM OF TERMS IN G  OVER THE
C                     APPROPRIATE RANGE.
C
      REAL AJ(30,30),DOT(30,30)
      COMMON/PARAM2/A(100),B(100),NX,NY,KX,KY,
     * LEFTX(100),LEFTY(100)
      COMMON/COEF/ALPHA(100,100),BETA(100,100)
      COMMON/KNOT/NKNOTX,NKNOTY
      COMMON/WEIGHTS/W1,W2
      COMMON/RANGE/IFIRST(100),ILAST(100),JFIRST(100),
     * JLAST(100)
      NUMX=NKNOTX
      NUMY=NKNOTY
      DELX=1./(NUMX-1.)
      DELY=1./(NUMY-1.)
      SDELX=DELX*DELX
      SDELY=DELY*DELY
      SUM=0.0
      IF=IFIRST(II)
      JF=JFIRST(JJ)
      IL=ILAST(II)
      JL=JLAST(JJ)
      IF(IF.GT.1) IF=IF-1
      IF(JF.GT.1) JF=JF-1
      IF(IL.LT.NUMX) IL=IL+1
      IF(JL.LT.NUMY) JL=JL+1
      DO 200 J=JF,JL
      DO 100 I=IF,IL
      CALL TENVALF(ALPHA,LEFTX(I),LEFTY(J),KX,KY,I,J,
     * F1X,1,0)
      CALL TENVALF(BETA,LEFTX(I),LEFTY(J),KX,KY,I,J,
     * F1Y,1,0)
      CALL TENVALF(ALPHA,LEFTX(I),LEFTY(J),KX,KY,I,J,
     * F2X,0,1)
      CALL TENVALF(BETA,LEFTX(I),LEFTY(J),KX,KY,I,J,
     * F2Y,0,1)
      AJ(I,J)=F1X*F2Y-F2X*F1Y
      DOT(I,J)=F1X*F2X+F1Y*F2Y
      SUM=SUM+DOT(I,J)**2
  100 CONTINUE
  200 CONTINUE
      SUM1=0.
      SUM2=0.
      DO 400 J=JF,JL-1
      DO 300 I=IF,IL-1
      IF(I.EQ.1.AND.J.EQ.1) GOTO 300
      IF(I.EQ.1.AND.J.EQ.NUMY-1) GOTO 300
      IF(I.EQ.NUMX-1.AND.J.EQ.1) GOTO 300
      SUM1=SUM1+(AJ(I,J)-AJ(I,J+1))**2
```

```
      SUM2=SUM2+(AJ(I+1,J)-AJ(I,J))**2
  300 CONTINUE
  400 CONTINUE
      SUM1=SUM1*DELX/DELY
      SUM2=SUM2*DELY/DELX
      GF=W1*(SUM1+SUM2)+W2*DELX*DELY*SUM
      RETURN
      END
C
C
C
      SUBROUTINE FFMIN(ERMAX)
C
C        FFMIN SEARCHES FOR THE MINIMUM OF THE SMOOTHING
C     FUNCTIONAL G. EACH CALL TO FFMIN PRODUCES ONE
C     COMPLETE ITERATION OF THE CYCLIC COORDINATE METHOD,
C     A MULTIDIMENSIONAL SEARCH TECHNIQUE FOR MINIMIZING
C     A FUNCTION OF SEVERAL VARIABLES WITHOUT USING
C     DERIVATIVES.  THE ROUTINE SEARCHES FOR A MINIMUM
C     ALONG EACH COORDINATE DIRECTION.
C        IN FFMIN THE COORDINATE DIRECTIONS ARE REPRE-
C     SENTED BY THE TENSOR PRODUCT COEFFICIENTS .FOR EACH
C     COEFFICIENT THE ROUTINE FIRST DETERMINES THE
C     INTERVAL ON WHICH THE COEFFICIENT MUST LIE IF THE
C     JACOBIAN OF THE TENSOR PRODUCT MAPPING IS TO BE
C     NONNEGATIVE AT ALL MESH POINTS AFFECTED BY THE
C     COEFFICIENT.  IT THEN CALLS EITHER TESTMINO, TEST-
C     MINL, TESTMINR, OR TESTMINB DEPENDING ON WHETHER
C     THE INTERVAL IS BIINFINITE, HAS A LEFT ENDPOINT,
C     A RIGHT ENDPOINT, OR TWO ENDPOINTS.  THE CHOSEN
C     SUBROUTINE FINDS THE LOCATION OF THE MINIMUM OF
C     GF ON THE INTERVAL AND CHANGES THE APPROPRIATE
C     COEFFICIENT ACCORDINGLY.
C
C     INPUT
C
C       (IN COMMON)
C       ALPHA,BETA..     ARRAYS CONTAINING COEFFICIENTS OF
C                        TENSOR PRODUCT SPLINE MAPPING.
C       A,B...           ARRAYS CONTAINING COORDINATES FOR
C                        SQUARE MESH.
C       NX,NY...         DIMENSION OF SPLINE SPACE IN X DIRECT.
C                        ,Y DIRECTION OR
C                        TOTAL NO. OF B-SPLINES IN X DIRECTION,
C                        Y DIRECTION.
C       KX,KY...         ORDER OF B-SPLINES IN X DIRECTION,
C                        Y DIRECTION.
C       LEFTX,LEFTY...   ARRAYS IDENTIFYING KNOT INTERVALS
C                        ON WHICH SQUARE MESH COORDINATES
C                        LIE. (LEFTX(I)=J IMPLIES
C                        TX(J)<=A(I)<TX(J+1))
```

```
C
C    IFIRST,JFIRST.. ARRAYS CONTAINING STARTING
C                    POINTS FOR THE RANGES OF SUMMATION
C                    CORRESPONDING TO EACH COEFFICIENT
C                    ALPHA(I,J),BETA(I,J).
C    ILAST,JLAST..   ARRAYS CONTAINING FINAL
C                    POINTS FOR THE RANGES OF SUMMATION
C                    CORRESPONDING TO EACH COEFFICIENT
C                    ALPHA(I,J),BETA(I,J).
C    W1,W2...        WEIGHTS FOR JACOBIAN, DOT PRODUCT
C                    TO BE USED IN GF.
C    NKNOTX,NKNOTY...DIMENSIONS FOR SQUARE MESH
C                    OR NUMBER OF ELEMENTS IN ARRAYS A,B.
C
C  OUTPUT
C
C    ERMAX...        MAXIMUM CHANGE IN THE COEFFICIENTS
C                    AFTER A COMPLETE ITERATION.
C    (IN COMMON)
C    ALPHA,BETA...   ARRAYS CONTAINING NEW COEFFICIENTS
C                    FOR TENSOR PRODUCT SPLINE MAPPING.
C
      COMMON/COEF/ALPHA(100,100),BETA(100,100)
      COMMON/PARAM2/A(100),B(100),NX,NY,KX,KY,
     * LEFTX(100),LEFTY(100)
      COMMON/RANGE/IFIRST(100),ILAST(100),JFIRST(100),
     * JLAST(100)
      COMMON/PARAM/FKOUNT
      COMMON/WEIGHTS/W1,W2
      COMMON/KNOT/NKNOTX,NKNOTY
      REAL LEND,LINT,AJ(2)
      INTEGER CTEST
      FKOUNT=0
      ERMAX=0.
      DO 500 I=2,NX-1
      DO 400 J=2,NY-1
      IF=IFIRST(I)
      IL=ILAST(I)
      JF=JFIRST(J)
      JL=JLAST(J)
      MKOUNT=0
    5 CONTINUE
      HOLD=GF(I,J)
      MTEST=MKOUNT/2*2
      IF(MTEST.EQ.MKOUNT) THEN
      HOCO=ALPHA(I,J)
      ELSE
      HOCO=BETA(I,J)
      ENDIF
      LEND=-10.0E8
      REND=10.0E8
```

```
      KFLAG=0
      IFLAG=0
      DO 200 II=IF,IL
      DO 100 JJ=JF,JL
      IF(MKOUNT/2*2.EQ.MKOUNT) THEN
      ALPHA(I,J)=0.
      ELSE
      BETA(I,J)=0.
      ENDIF
      DO 10 K=1,2
      CALL TENVALF(ALPHA,LEFTX(II),LEFTY(JJ),KX,KY,II,
     * JJ,F1X,1,0)
      CALL TENVALF(BETA,LEFTX(II),LEFTY(JJ),KX,KY,II,
     * JJ,F1Y,1,0)
      CALL TENVALF(ALPHA,LEFTX(II),LEFTY(JJ),KX,KY,II,
     * JJ,F2X,0,1)
      CALL TENVALF(BETA,LEFTX(II),LEFTY(JJ),KX,KY,II,
     * JJ,F2Y,0,1)
      AJ(K)=F1X*F2Y-F2X*F1Y
      IF(MKOUNT/2*2.EQ.MKOUNT) THEN
      ALPHA(I,J)=1.
      ELSE
      BETA(I,J)=1.
      ENDIF
   10 CONTINUE
      D=AJ(1)
      C=AJ(2)-D
      IF(C.GT.1.0E-7) THEN
       LINT=-D/C
       IF(LINT.GT.LEND)THEN
         IF(LINT.LE.REND) THEN
           LEND=LINT
          ELSE
            IFLAG=-1
          ENDIF
       ENDIF
      ELSE IF(C.LT.-1.0E-7) THEN
        RINT=-D/C
        IF(RINT.LT.REND) THEN
          IF(RINT.GE.LEND) THEN
            REND=RINT
          ELSE
            IFLAG=-1
          ENDIF
        ENDIF
      ELSE
        KFLAG=KFLAG+1
      ENDIF
  100 CONTINUE
  200 CONTINUE
      CTEST=(ILAST(I)-IFIRST(I)+1)*(JLAST(J)-JFIRST(J)+1)
```

```
      IF(KFLAG.EQ.CTEST.OR.IFLAG.EQ.-1) THEN
        IF(MKOUNT/2*2.EQ.MKOUNT) THEN
          ALPHA(I,J)=(LEND+REND)/2.
        ELSE
          BETA(I,J)=(LEND+REND)/2.
        ENDIF
      ELSE IF(LEND.LT.-1.0E7.AND.REND.GT.1.0E7) THEN
      CENTER=0.0
      CALL TESTMINO(MKOUNT,I,J)
        ELSE IF(LEND.LT.-10.0E7) THEN
        CENTER=0.0
        CALL TESTMINR(MKOUNT,I,J,LEND,REND)
        ELSE IF(REND.GT.10.E7) THEN
          CENTER=0.0
          CALL TESTMINL(MKOUNT,I,J,LEND,REND)
          ELSE
            CENTER=(LEND+REND)/2.
            CALL TESTMINB(MKOUNT,I,J,LEND,REND)
      ENDIF
      S2=GF(I,J)
      DIFF=HOLD-S2
      IF(HOLD.LT.S2) THEN
      IF(MTEST.EQ.MKOUNT) THEN
      ALPHA(I,J)=HOCO
      ELSE
      BETA(I,J)=HOCO
      ENDIF
      S2=HOLD
      DIFF=0.
      ENDIF
      IF(ABS(DIFF).GT.ERMAX) ERMAX=ABS(DIFF)
      PRINT*,'FUNCTION VALUE IS',S2
      PRINT*,'COUNT IS',FKOUNT
      KFLAG=0
      MKOUNT=MKOUNT+1
      IF(MKOUNT.NE.MKOUNT/2*2) GOTO 5
  400 CONTINUE
  500 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE CRIT(CENTER,C1,C2,C3,C4,C5,NROOTS,R,MKOUNT,
     * I,J)
C
C       CRIT FINDS THE COEFFICIENTS OF THE 4TH DEGREE
C  POLYNOMIAL REPRESENTING GF AND COMPUTES ITS CRITICAL
C  POINTS, I.E., IT FINDS THE POINTS FOR WHICH THE DERIVATIVE
C  OF THE POLYNOMIAL IS 0.
C
```

```
C  INPUT
C
C    CENTER...       NUMBER AT CENTER OF INTERVAL TO BE
C                    CONSIDERED. IF INTERVAL IS INFINITE THEN
C                    CENTER ASSIGNED A VALUE OF 0.
C    MKOUNT...       MKOUNT EVEN MEANS THE COEFFICIENT
C                    INVOLVED IN MINIMIZATION IS IN THE ALPHA
C                    ARRAY. MKOUNT ODD MEANS THE COEFFICIENT
C                    IS IN THE BETA ARRAY.
C    I,J..           SUBSCRIPTS FOR COEFFICIENT INVOLVED
C                    IN MINIMIZATION
C    (IN COMMON)
C    ALPHA,BETA..    ARRAYS CONTAINING COEFFICIENTS OF
C                    TENSOR PRODUCT SPLINE MAPPING.
C    A,B...          ARRAYS CONTAINING COORDINATES FOR
C                    SQUARE MESH.
C    NX,NY...        DIMENSION OF SPLINE IN X DIRECTION
C                    ,Y DIRECTION OR
C                    TOTAL NO. OF B-SPLINES IN X DIRECTION,
C                    Y DIRECTION.
C    KX,KY...        ORDER OF B-SPLINES IN X DIRECTION,
C                    Y DIRECTION.
C    LEFTX,LEFTY...  ARRAYS IDENTIFYING KNOT INTERVALS
C                    IN WHICH SQUARE MESH COORDINATES
C                    LIE. (LEFTX(I)=J IMPLIES
C                    TX(J)<=A(I)<TX(J+1))
C
C    W1,W2...        WEIGHTS FOR JACOBIAN, DOT PRODUCT
C                    TO BE USED IN GF.
C    NKNOTX,NKNOTY...DIMENSIONS FOR SQUARE MESH
C                    OR NUMBER OF ELEMENTS IN ARRAYS A,B.
C    FKOUNT...       PARAMETER CONTAINING NUMBER OF CALLS TO
C                    GF
C    XK...           ARRAY CONTAININ POINTS -2,-1,0,1,2
C                    WHICH ARE USED AS TEST POINTS IN
C                    DETERMINING THE COEFFICIENTS OF THE
C                    4TH DEGREE POLYNOMIAL WHICH APPROXI-
C                    MATES GF.
C
C  OUTPUT
C
C    C1,C2,C3,C4,C5.. COEFFICIENTS OF 4TH DEGREE POLYNOMIAL.
C                    C1 IS THE COEFFICIENT OF THE 4TH DEGREE TERM.
C    NROOTS...       NUMBER OF CRITICAL POINTS
C    R...            ARRAY CONTAINING CRITICAL POINTS
C
      REAL R(3),BK(5)
      COMMON/COEF/ALPHA(100,100),BETA(100,100)
      COMMON/PARAM2/A(100),B(100),NX,NY,KX,KY,LEFTX(100)
     * ,LEFTY(100)
      COMMON/KNOT/NKNOTX,NKNOTY
```

```
      COMMON/WEIGHTS/W1,W2
      COMMON/PARAM/FKOUNT
      COMMON/XKS/XK(5)
      IF(MKOUNT/2*2.EQ.MKOUNT) THEN
      DO 100 IK=1,5
      ALPHA(I,J)=XK(IK)+CENTER
      BK(IK)=GF(I,J)
      FKOUNT=FKOUNT+1
  100 CONTINUE
      ELSE
        DO 200 IK=1,5
        BETA(I,J)=XK(IK)+CENTER
        BK(IK)=GF(I,J)
        FKOUNT=FKOUNT+1
  200 CONTINUE
      ENDIF
      D=XK(4)
      B1=BK(1)
      B2=BK(2)
      B3=BK(3)
      B4=BK(4)
      B5=BK(5)
      C5=B3
      SUM=-B5+8.*(B4-B2)+B1
      C4=1./(12.*D)*SUM
      SUM=-B5+16.*(B4+B2)-30.*B3-B1
      C3=1./(24.*D*D)*SUM
      SUM=B5-2.*(B4-B2)-B1
      C2=1./(12.*D*D*D)*SUM
      SUM=B5-4.*(B4+B2)+6.*B3+B1
      C1=1./(24.*D**4)*SUM
      IF (ABS(C1).LT.1.0E-06) GOTO 300
      CALL CUBIC(4.*C1,3.*C2,2.*C3,C4,NROOTS,R)
      RETURN
  300 CONTINUE
      NROOTS=-1
      RETURN
      END
C
C
C
      SUBROUTINE TESTMINO(MKOUNT,I,J)
C
C      FOR A GIVEN COEFFICIENT ALPHA(I,J) OR BETA(I,J)
C  TESTMINO FINDS AND TESTS THE CRITICAL POINTS OF
C  THE 4TH DEGREE POLYNOMIAL  REPRESENTING GF TO
C  DETERMINE WHICH POINT YIELDS THE SMALLEST VALUE FOR GF
C  WHEN GF IS VIEWED AS A FUNCTION OF THAT COEFFICIENT.
C  THE NUMBER CHOSEN BECOMES THE NEW VALUE FOR
C  ALPHA(I,J) OR BETA(I,J) .
C
```

```
C  INPUT
C
C    MKOUNT...         MKOUNT EVEN MEANS THE COEFFICIENT
C                      INVOLVED IN MINIMIZATION IS IN THE ALPHA
C                      ARRAY. MKOUNT ODD MEANS THE COEFFICIENT
C                      IS IN THE BETA ARRAY.
C    I,J..             SUBSCRIPTS FOR COEFFICIENT INVOLVED
C                      IN MINIMIZATION
C    (IN COMMON)
C    ALPHA,BETA..      ARRAYS CONTAINING COEFFICIENTS OF
C                      TENSOR PRODUCT SPLINE MAPPING.
C    A,B...            ARRAYS CONTAINING COORDINATES FOR
C                      SQUARE MESH.
C    NX,NY...          DIMENSION OF SPLINE IN X DIRECTION
C                      ,Y DIRECTION OR
C                      TOTAL NO. OF B-SPLINES IN X DIRECTION,
C                      Y DIRECTION.
C    KX,KY...          ORDER OF B-SPLINES IN X DIRECTION,
C                      Y DIRECTION.
C    LEFTX,LEFTY...    ARRAYS IDENTIFYING KNOT INTERVALS
C                      IN WHICH SQUARE MESH COORDINATES
C                      LIE. (LEFTX(I)=J IMPLIES
C                      TX(J)<=A(I)<TX(J+1))
C
C    W1,W2...          WEIGHTS FOR JACOBIAN, DOT PRODUCT
C                      TO BE USED IN GF.
C    NKNOTX,NKNOTY...DIMENSIONS FOR SQUARE MESH
C                      OR NUMBER OF ELEMENTS IN ARRAYS A,B.
C    FKOUNT...         PARAMETER CONTAINING NUMBER OF CALLS TO
C                      GF
C    XK...             ARRAY CONTAINING POINTS -2,-1,0,1,2
C                      WHICH ARE USED AS TEST POINTS IN
C                      DETERMINING THE COEFFICIENTS OF THE
C                      4TH DEGREE POLYNOMIAL WHICH APPROXI-
C                      MATES GF.
C
C  OUTPUT
C
C    ALPHA(I,J) OR BETA(I,J)..NEW VALUE FOR COEFFICIENT
C
      COMMON/COEF/ALPHA(100,100),BETA(100,100)
      COMMON/PARAM2/A(100),B(100),NX,NY,KX,KY,LEFTX(100),
    * LEFTY(100)
      COMMON/KNOT/NKNOTX,NKNOTY
      COMMON/WEIGHTS/W1,W2
      COMMON/PARAM/FKOUNT
      COMMON/XKS/XK(5)
      REAL R(3)
      FM(R)=C1*R**4+C2*R*R*R+C3*R*R+C4*R+C5
      DO 50 IK=1,5
      XK(IK)=FLOAT(IK)-3.
```

```
   50   CONTINUE
        CALL CRIT(0,C1,C2,C3,C4,C5,NROOTS,R,MKOUNT,I,J)
        IF(NROOTS.NE.-1) GOTO 55
        RETURN
   55   CONTINUE
        TMIN=10.0E10
        DO 600 IR=1,NROOTS
        FMINN=FM(R(IR))
        IF(FMINN.LT.TMIN) THEN
        TMIN=FMINN
        IMIN=IR
        ENDIF
  600   CONTINUE
        IF(MKOUNT/2*2.EQ.MKOUNT)THEN
        ALPHA(I,J)=R(IMIN)
        ELSE
        BETA(I,J)=R(IMIN)
        ENDIF
        RETURN
        END
C
C
C
        SUBROUTINE TESTMINR(MKOUNT.I,J,LEND,REND)
C
C        FOR A GIVEN COEFFICIENT ALPHA(I,J) OR BETA(I,J)
C   TESTMINR FINDS AND TESTS THE CRITICAL POINTS OF
C   THE 4TH DEGREE POLYNOMIAL  REPRESENTING GF TO
C   DETERMINE WHICH POINT YIELDS THE SMALLEST VALUE FOR GF
C   WHEN GF IS VIEWED AS A FUNCTION OF THAT COEFFICIENT.
C   THE SMALLEST VALUE IS COMPARED WITH THE VALUE AT THE
C   RIGHT ENDPOINT OF THE INTERVAL  (LEND,REND)
C   TO DETERMINE AT WHAT NUMBER THE MINIMUM VALUE
C   OF GF OCCURS. THE NUMBER CHOSEN BECOMES THE NEW
C   VALUE FOR ALPHA(I,J) OR BETA(I,J) .
C
C   INPUT
C
C     MKOUNT...         MKOUNT EVEN MEANS THE COEFFICIENT
C                       INVOLVED IN MINIMIZATION IS IN THE ALPHA
C                       ARRAY. MKOUNT ODD MEANS THE COEFFICIENT
C                       IS IN THE BETA ARRAY.
C     I,J..             SUBSCRIPTS FOR COEFFICIENT INVOLVED
C                       IN MINIMIZATION
C     LEND,REND..       LEFT AND RIGHT ENDPOINTS FOR
C                       INTERVAL. LEND IS A NEGATIVE NO. WITH VERY
C                       LARGE MAGNITUDE INDICATING THAT THE LEFT
C                       ENDPOINT IS INFINITE.
C     (IN COMMON)
C     ALPHA,BETA..      ARRAYS CONTAINING COEFFICIENTS OF
C                       TENSOR PRODUCT SPLINE MAPPING.
```

```
C    A,B...             ARRAYS CONTAINING COORDINATES FOR
C                       SQUARE MESH.
C    NX,NY...           DIMENSION OF SPLINE IN X DIRECTION
C                       ,Y DIRECTION OR
C                       TOTAL NO. OF B-SPLINES IN X DIRECTION,
C                       Y DIRECTION.
C    KX,KY...           ORDER OF B-SPLINES IN X DIRECTION,
C                       Y DIRECTION.
C    LEFTX,LEFTY...     ARRAYS IDENTIFYING KNOT INTERVALS
C                       IN WHICH SQUARE MESH COORDINATES
C                       LIE. (LEFTX(I)=J IMPLIES
C                       TX(J)<=A(I)<TX(J+1))
C
C    W1,W2...           WEIGHTS FOR JACOBIAN, DOT PRODUCT
C                       TO BE USED IN GF.
C    NKNOTX,NKNOTY...DIMENSIONS FOR SQUARE MESH
C                       OR NUMBER OF ELEMENTS IN ARRAYS A,B.
C    FKOUNT...          PARAMETER CONTAINING NUMBER OF CALLS TO
C                       GF
C    XK...              ARRAY CONTAININ POINTS -2,-1,0,1,2
C                       WHICH ARE USED AS TEST POINTS IN
C                       DETERMINING THE COEFFICIENTS OF THE
C                       4TH DEGREE POLYNOMIAL WHICH APPROXI-
C                       MATES GF.
C
C  OUTPUT
C
C    ALPHA(I,J) OR BETA(I,J)..NEW VALUE FOR COEFFICIENT
C
      COMMON/COEF/ALPHA(100,100),BETA(100,100)
      COMMON/PARAM/FKOUNT
      COMMON/PARAM2/A(100),B(100),NX,NY,KX,KY
     * ,LEFTX(100),LEFTY(100)
      COMMON/KNOT/NKNOTX,NKNOTY
      COMMON/WEIGHTS/W1,W2
      COMMON/XKS/XK(5)
      REAL R(3),LEND
      FM(R)=C1*R**4+C2*R*R*R+C3*R*R+C4*R+C5
      XK(1)=REND
      DO 50 IK=2,5
      XK(IK)=REND-FLOAT(IK)+1.
   50 CONTINUE
      CALL CRIT(0,C1,C2,C3,C4,C5,NROOTS,R,MKOUNT,I,J)
      IF(NROOTS.NE.-1) GOTO 55
      PRINT*,'WARNING #1 COEF IS 0'
      RETURN
   55 CONTINUE
      TMIN=10.E10
      IR=0
      DO 600 IROOT=1,NROOTS
      IF(R(IROOT).LE.REND) THEN
```

```
      IR=IR+1
      R(IR)=R(IROOT)
      ENDIF
  600 CONTINUE
      NROOTS=IR
      IF(NROOTS.EQ.0) THEN
        IF(MKOUNT/2*2.EQ.MKOUNT) THEN
        ALPHA(I,J)=REND
        TMIN=FM(REND)
      ELSE
        BETA(I,J)=REND
        TMIN=FM(REND)
      ENDIF
      ELSE
        DO 700 IROOT=1,NROOTS
        FMINN=FM(R(IROOT))
        IF(FMINN.LT.TMIN) THEN
          TMIN=FMINN
          IMIN=IROOT
        ENDIF
  700   CONTINUE
        FMINN=FM(REND)
        IF(FMINN.LT.TMIN) R(IMIN)=REND
        IF(MKOUNT/2*2.EQ.MKOUNT) THEN
          ALPHA(I,J)=R(IMIN)
          TMIN=FM(R(IMIN))
        ELSE
          BETA(I,J)=R(IMIN)
          TMIN=FM(R(IMIN))
        ENDIF
      ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE TESTMINL(MKOUNT,I,J,LEND,REND)
C
C        FOR A GIVEN COEFFICIENT ALPHA(I,J) OR BETA(I,J)
C   TESTMINL FINDS AND TESTS THE CRITICAL POINTS OF
C   THE 4TH DEGREE POLYNOMIAL  REPRESENTING GF TO
C   DETERMINE WHICH POINT YIELDS THE SMALLEST VALUE FOR GF
C   WHEN GF IS VIEWED AS A FUNCTION OF THAT COEFFICIENT.
C   THE SMALLEST VALUE IS COMPARED WITH THE VALUE AT THE
C   LEFT ENDPOINT OF THE INTERVAL  (LEND,REND)
C   TO DETERMINE AT WHAT NUMBER THE MINIMUM VALUE
C   OF GF OCCURS. THE NUMBER CHOSEN BECOMES THE NEW
C   VALUE FOR ALPHA(I,J) OR BETA(I,J) .
C
C   INPUT
C
```

```
C    MKOUNT...          MKOUNT EVEN MEANS THE COEFFICIENT
C                       INVOLVED IN MINIMIZATION IS IN THE ALPHA
C                       ARRAY. MKOUNT ODD MEANS THE COEFFICIENT
C                       IS IN THE BETA ARRAY.
C    I,J..              SUBSCRIPTS FOR COEFFICIENT INVOLVED
C                       IN MINIMIZATION
C    LEND,REND..        LEFT AND RIGHT ENDPOINTS FOR
C                       INTERVAL. REND IS A VERY LARGE NUMBER,
C                       INDICATING THAT THE RIGHT ENDPOINT IS
C                       INFINITE.
C    (IN COMMON)
C    ALPHA,BETA..       ARRAYS CONTAINING COEFFICIENTS OF
C                       TENSOR PRODUCT SPLINE MAPPING.
C    A,B...             ARRAYS CONTAINING COORDINATES FOR
C                       SQUARE MESH.
C    NX,NY...           DIMENSION OF SPLINE IN X DIRECTION
C                       ,Y DIRECTION OR
C                       TOTAL NO. OF B-SPLINES IN X DIRECTION,
C                       Y DIRECTION.
C    KX,KY...           ORDER OF B-SPLINES IN X DIRECTION,
C                       Y DIRECTION.
C    LEFTX,LEFTY...     ARRAYS IDENTIFYING KNOT INTERVALS
C                       IN WHICH SQUARE MESH COORDINATES
C                       LIE. (LEFTX(I)=J IMPLIES
C                       TX(J)<=A(I)<TX(J+1))
C
C    W1,W2...           WEIGHTS FOR JACOBIAN, DOT PRODUCT
C                       TO BE USED IN GF.
C    NKNOTX,NKNOTY...DIMENSIONS FOR SQUARE MESH
C                       OR NUMBER OF ELEMENTS IN ARRAYS A,B.
C    FKOUNT...          PARAMETER CONTAINING NUMBER OF CALLS TO
C                       GF
C    XK...              ARRAY CONTAININ POINTS -2,-1,0,1,2
C                       WHICH ARE USED AS TEST POINTS IN
C                       DETERMINING THE COEFFICIENTS OF THE
C                       4TH DEGREE POLYNOMIAL WHICH APPROXI-
C                       MATES GF.
C
C  OUTPUT
C
C    ALPHA(I,J) OR BETA(I,J)..NEW VALU" FOR COEFFICIENT
C
       COMMON/COEF/ALPHA(100,100),BETA(100,100)
       COMMON/PARAM/FKOUNT
       COMMON/PARAM2/A(100),B(100),NX,NY,KX,KY
     * ,LEFTX(100),LEFTY(100)
       COMMON/KNOT/NKNOTX,NKNOTY
       COMMON/WEIGHTS/W1,W2
       COMMON/XKS/XK(5)
       REAL R(3),LEND
       FM(R)=C1*R**4+C2*R*R*R+C3*R*R+C4*R+C5
```

```
      XK(1)=LEND
      DO 50 IK=2,5
      XK(IK)=LEND+FLOAT(IK)-1.
   50 CONTINUE
      CALL CRIT(0,C1,C2,C3,C4,C5,NROOTS,R,MKOUNT,
     * I,J)
      IF(NROOTS.NE.-1) GOTO 55
      PRINT*,'WARNING #1 COEF IS 0'
      RETURN
   55 CONTINUE
      TMIN=10.0E10
      IL=0
      DO 600 IR=1,NROOTS
      IF(R(IR).GE.LEND)THEN
      IL=IL+1
      R(IL)=R(IR)
      ENDIF
  600 CONTINUE
      NROOTS=IL
      IF(NROOTS.EQ.0) THEN
        IF(MKOUNT/2*2.EQ.MKOUNT) THEN
        ALPHA(I,J)=LEND
        TMIN=FM(LEND)
        ELSE
        BETA(I,J)=LEND
        TMIN=FM(LEND)
        ENDIF
      ELSE
        DO 700 IR=1,NROOTS
        FMINN=FM(R(IR))
        IF(FMINN.LT.TMIN) THEN
        TMIN=FMINN
        IMIN=IR
        ENDIF
  700   CONTINUE
        FMINN=FM(LEND)
        IF(FMINN.LT.TMIN) R(IMIN)=LEND
        IF(MKOUNT/2*2.EQ.MKOUNT) THEN
          ALPHA(I,J)=R(IMIN)
          TMIN=FM(R(IMIN))
        ELSE
          BETA(I,J)=R(IMIN)
          TMIN=FM(R(IMIN))
        ENDIF
      ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE TESTMINB(MKOUNT,I,J,LEND,REND)
```

```
C
C        FOR A GIVEN COEFFICIENT ALPHA(I,J) OR BETA(I,J)
C  TESTMINB FINDS AND TESTS THE CRITICAL POINTS OF
C  THE 4TH DEGREE POLYNOMIAL  REPRESENTING GF TO
C  DETERMINE WHICH POINT YIELDS THE SMALLEST VALUE FOR GF
C  WHEN GF IS VIEWED AS A FUNCTION OF THAT COEFFICIENT.
C  THE SMALLEST VALUE IS COMPARED WITH THE VALUE
C  AT THE ENDPOINTS OF THE INTERVAL  (LEND,REND)
C  TO DETERMINE AT WHAT NUMBER THE MINIMUM VALUE
C  OF GF OCCURS. THE NUMBER CHOSEN BECOMES THE NEW
C  VALUE FOR ALPHA(I,J) OR BETA(I,J) .
C
C  INPUT
C
C    MKOUNT...         MKOUNT EVEN MEANS THE COEFFICIENT
C                      INVOLVED IN MINIMIZATION IS IN THE ALPHA
C                      ARRAY. MKOUNT ODD MEANS THE COEFFICIENT
C                      IS IN THE BETA ARRAY.
C    I,J..             SUBSCRIPTS FOR COEFFICIENT INVOLVED
C                      IN MINIMIZATION
C    LEND,REND..       LEFT AND RIGHT ENDPOINTS FOR
C                      INTERVAL
C    (IN COMMON)
C    ALPHA,BETA..      ARRAYS CONTAINING COEFFICIENTS OF
C                      TENSOR PRODUCT SPLINE MAPPING.
C    A,B...            ARRAYS CONTAINING COORDINATES FOR
C                      SQUARE MESH.
C    NX,NY...          DIMENSION OF SPLINE IN X DIRECTION
C                      ,Y DIRECTION OR
C                      TOTAL NO. OF B-SPLINES IN X DIRECTION,
C                      Y DIRECTION.
C    KX,KY...          ORDER OF B-SPLINES IN X DIRECTION,
C                      Y DIRECTION.
C    LEFTX,LEFTY...    ARRAYS IDENTIFYING KNOT INTERVALS
C                      IN WHICH SQUARE MESH COORDINATES
C                      LIE. (LEFTX(I)=J IMPLIES
C                      TX(J)<=A(I)<TX(J+1))
C
C    W1,W2...          WEIGHTS FOR JACOBIAN, DOT PRODUCT
C                      TO BE USED IN GF.
C    NKNOTX,NKNOTY...DIMENSIONS FOR SQUARE MESH
C                      OR NUMBER OF ELEMENTS IN ARRAYS A,B.
C    FKOUNT...         PARAMETER CONTAINING NUMBER OF CALLS TO
C                      GF
C    XK...             ARRAY CONTAININ POINTS -2,-1,0,1,2
C                      WHICH ARE USED AS TEST POINTS IN
C                      DETERMINING THE COEFFICIENTS OF THE
C                      4TH DEGREE POLYNOMIAL WHICH APPROXI-
C                      MATES GF.
C
C  OUTPUT
```

```
C
C     ALPHA(I,J) OR BETA(I,J)..NEW VALUE FOR COEFFICIENT
C
      COMMON/COEF/ALPHA(100,100),BETA(100,100)
      COMMON/PARAM/FKOUNT
      COMMON/PARAM2/A(100),B(100),NX,NY,KX,KY
     * ,LEFTX(100),LEFTY(100)
      COMMON/KNOT/NKNOTX,NKNOTY
      COMMON/WEIGHTS/W1,W2
      COMMON/XKS/XK(5)
      REAL R(3),LEND
      FM(R)=C1*R**4+C2*R*R*R+C3*R*R+C4*R+C5
      CENTER=(LEND+REND)/2.
      XK(1)=LEND-CENTER
      XK(2)=(LEND-CENTER)/2.
      XK(3)=0.
      XK(4)=(REND-CENTER)/2.
      XK(5)=REND-CENTER
      CALL CRIT(CENTER,C1,C2,C3,C4,C5,NROOTS,R,MKOUNT,
     *  I,J)
      IF(NROOTS.NE.-1) GOTO 55
      RETURN
 55   CONTINUE
      TMIN=10.0E10
      IB=0
      DO 600 IR=1,NROOTS
      IF(R(IR)+CENTER.GE.LEND.AND.R(IR)+CENTER.LE.REND) THEN
      IB=IB+1
      R(IB)=R(IR)
      ENDIF
 600  CONTINUE
      NROOTS=IB
      IF(NROOTS.EQ.0) THEN
        IF(MKOUNT/2*2.EQ.MKOUNT) THEN
        ALPHA(I,J)=CENTER
        TMIN=FM(CENTER)
        ELSE
        BETA(I,J)=CENTER
        TMIN=FM(CENTER)
        ENDIF
      ELSE
        DO 700 IR=1,NROOTS
        FMINN=FM(R(IR))
        IF(FMINN.LT.TMIN)THEN
        TMIN=FMINN
        IMIN=IR
        ENDIF
 700    CONTINUE
        FMINNL=FM(LEND)
        FMINNR=FM(REND)
        IF(FMINNL.LT.FMINNR) THEN
```

```
          IF(FMINNL.LT.TMIN) THEN
          R(IMIN)=LEND
          TMIN=FM(LEND)
          ENDIF
        ELSE IF(FMINNR.LT.TMIN) THEN
          R(IMIN)=REND
          TMIN=FM(REND)
        ENDIF
        IF(MKOUNT/2*2.EQ.MKOUNT) THEN
          ALPHA(I,J)=R(IMIN)+CENTER
        ELSE
          BETA(I,J)=R(IMIN)+CENTER
        ENDIF
      ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE CUBIC(A3,A2,A1,A0,NROOTS,RR)
C
C     CUBIC COMPUTES THE ROOTS OF A CUBIC POLY-
C  NOMIAL USING FORMULAS FROM  'HANDBOOK OF
C  MATHEMATICAL TABLES AND FORMULAS' BY RICHARD
C  STEVENS BURINGTON,PH.D., MCGRAW-HILL ,NEW YORK,
C  1962.
C
C  INPUT
C    A3,A2,A1,A0..   COEFFICIENTS OF CUBIC POLY-
C                    NOMIAL
C
C  OUTPUT
C
C    NROOTS...       NUMBER OF DIFFERENT REAL ROOTS
C    RR...           ARRAY CONTAINING REAL ROOTS
C
      REAL RR(3)
      PI=3.1415
      P=A2/A3
      Q=A1/A3
      R=A0/A3
      A=1./3.*(3.*Q-P*P)
      B=1./27.*(2.*P*P*P-9.*P*Q+27.*R)
      IF (ABS(B).LT.1.0E-06)  THEN
      SIGNB=0.
      ELSE
      SIGNB=B/ABS(B)
      ENDIF
      BB=B*B/4.
      AAA=A*A*A/27.
      TEST=BB+AAA
```

```
      IF(TEST.LT.0) THEN
      NROOTS=3
      PHI=ACOS(-SIGNB*SQRT(BB/(-AAA)))
      SRT=SQRT(-A/3.)
      RR(1)=2.*SRT*COS(PHI/3.)
      RR(2)=2.*SRT*COS(PHI/3.+2.*PI/3.)
      RR(3)=2.*SRT*COS(PHI/3.+4.*PI/3.)
      ELSE IF(TEST.GT.0) THEN
      NROOTS=1
      S1=-.5*B+SQRT(TEST)
      S2=-.5*B-SQRT(TEST)
      IF (ABS(S1).LT.1.0E-06) THEN
      SIGNS1=0.
      ELSE
      SIGNS1=S1/ABS(S1)
      ENDIF
      IF (ABS(S2).LT.1.0E-06) THEN
      SIGNS2=0.
      ELSE
      SIGNS2=S2/ABS(S2)
      ENDIF
      RR(1)=SIGNS1*(ABS(S1)**(1./3.))
     * +SIGNS2*(ABS(S2)**(1./3.))
      ELSE
      NROOTS=2
      RR(1)=-SIGNB*2.*SQRT(-A/3.)
      RR(2)=SIGNB*SQRT(-A/3.)
      END IF
      DO 10 I=1,3
      RR(I)=RR(I)-P/3.
   10 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE EXTREMES(X,Y,TMAX,TMIN,NR,NC)
C
C
C EXTREMES FINDS THE MAXIMUM AND MINIMUM VALUES
C AMONG THE ELEMENTS OF TWO TWO-DIMENSIONAL ARRAYS.
C
C   INPUT
C
C     X...X COMPONENT
C     Y...Y COMPONENT
C     NR...DIMENSION OF X ARRAY
C     NC...DIMENSION OF Y ARRAY
C
C   OUTPUT
C
```

```
C      TMAX...MAXIMUM VALUE IN ARRAYS
C      TMIN...MINIMUM VALUE IN ARRAYS
C
       REAL X(100,100),Y(100,100)
       TMAX=X(1,1)
       TMIN=X(1,1)
       DO 20 I=1,NR
       DO 10 J=1,NC
       IF(X(I,J).GT.TMAX) TMAX=X(I,J)
       IF(X(I,J).LT.TMIN) TMIN=X(I,J)
       IF(Y(I,J).GT.TMAX) TMAX=Y(I,J)
       IF(Y(I,J).LT.TMIN) TMIN=Y(I,J)
    10 CONTINUE
    20 CONTINUE
       RETURN
       END
C
C
C
       SUBROUTINE NORM(X,Y,TMAX,TMIN,NR,NC)
C
C  THIS ROUTINE NORMALIZES THE VALUES OF TWO
C  TWO DIMENSIONAL ARRAYS SO THAT THEY LIE
C  BETWEEN 0 AND 1 INCLUSIVE.
C
C    INPUT
C
C      X...X COMPONENT ARRAY ON INPUT AND
C      NORMALIZED X COMPONENT ARRAY ON OUTPUT
C      Y...Y COMPONENT ARRAY ON INPUT AND
C      NORMALIZED Y COMPONENT ARRAY ON OUTPUT
C      NR...DIMENSION OF X ARRAY
C      NC...DIMENSION OF Y ARRAY
C      TMAX...MAXIMUM VALUE IN ARRAYS
C      TMIN...MINIMUM VALUE IN ARRAYS
C
C    OUTPUT
C
C      X...NORMALIZED X ARRAY
C      Y...NORMALIZED Y ARRAY
C
       REAL X(100,100),Y(100,100)
       DO 20 I=1,NR
       DO 10 J=1,NC
       X(I,J)=(X(I,J)-TMIN)/(TMAX-TMIN)
       Y(I,J)=(Y(I,J)-TMIN)/(TMAX-TMIN)
    10 CONTINUE
    20 CONTINUE
       RETURN
       END
BOTTOM
```

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-177968		

4. Title and Subtitle	5. Report Date
Algebraic Grid Generation Using Tensor Product B-Splines	September 1985
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Bonita V. Saunders	
	10. Work Unit No.
9. Performing Organization Name and Address	11. Contract or Grant No.
Old Dominion University Norfolk, Virginia	NGT 47-003-802
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Contractor Report
National Aeronautics and Space Administration Washington, DC 20546	14. Sponsoring Agency Code 505-31-83-02

15. Supplementary Notes

This report is a dissertation submitted to Old Dominion University for partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computational and Applied Mathematics.
Langley Technical Monitor: R. E. Smith

16. Abstract

In general, finite difference methods are more successful if the accompanying grid has lines which are smooth and nearly orthogonal. This thesis discusses the development of an algorithm which produces such a grid when given the boundary description. Topological considerations in structuring the grid generation mapping are discussed. In particular, this thesis examines the concept of the degree of a mapping and how it can be used to determine what requirements are necessary if a mapping is to produce a suitable grid. The grid generation algorithm uses a mapping composed of bicubic B-splines. Boundary coefficients are chosen so that the splines produce Schoenberg's variation diminishing spline approximation to the boundary. Interior coefficients are initially chosen to give a variation diminishing approximation to the transfinite bilinear interpolant of the function mapping the boundary of the unit square onto the boundary of the grid. The practicality of optimizing the grid by minimizing a functional involving the Jacobian of the grid generation mapping at each interior grid point and the dot product of vectors tangent to the grid lines is investigated. Grids generated by using the algorithm are presented.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
grid generation, Computational Fluid Dynamics, Optimization	Unclassified-Unlimited Subject Category 64

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	152	A08

*For sale by the National Technical Information Service, Springfield, Virginia 22161