Winter 2001

A multigrid method for elliptic grid generation using compact schemes

Balaji S. Iyangar
Louisiana Tech University

A MULTIGRID METHOD FOR ELLIPTIC GRID GENERATION USING COMPACT SCHEMES

by

Balaji S. Iyangar, MSc Physics

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Computational Analysis and Modeling

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

February,2001

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

February 9, 2001
Date

We hereby recommend that the thesis prepared under our supervision by Iyangar Balaji

entitled A MULTIGRID METHOD FOR ELLIPTIC GRID GENERATION USING COMPACT SCHEMES

be accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computational Analysis and Modeling

Chaoyang Liu 2/7/01
Supervisor of Thesis Research

Raja Nassar 2/12/01
Head of Department

Department

Chaoyang Liu 2/7/01

Recommendation concurred in:

Raja Nassar

Weizhong Dai

Boris Gelfand 2/8/01

Natalia Zotov

Advisory Committee

Approved:
Director of Graduate Studies

Approved:
Dean of the Graduate School

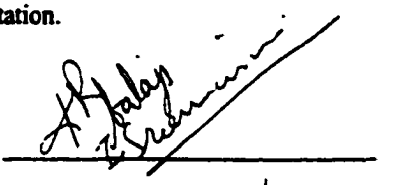
Dean of the College

GS Form 13
(1/00)

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author 

Date 02/23/01

GS Form 14
2/97

ABSTRACT

Traditional iterative methods are stalling numerical processes, in which the error has relatively small changes from one iteration to the next. Multigrid methods overcome the limitations of iterative methods and are computationally efficient. Convergence of iterative methods for elliptic partial differential equations is extremely slow. In particular, the convergence of the non-linear elliptic Poisson grid generation equations used for elliptic grid generation is very slow. Multigrid methods are fast converging methods when applied to elliptic partial differential equations. In this dissertation, a non-linear multigrid algorithm is used to accelerate the convergence of the non-linear elliptic Poisson grid generation method. The non-linear multigrid algorithm alters the performance characteristics of the non-linear elliptic Poisson grid generation method making it robust and fast in convergence. The elliptic grid generation method is based on the use of a composite mapping. It consists of a nonlinear transfinite algebraic transformation and an elliptic transformation. The composite mapping is a differentiable one-to-one mapping from the computational space onto the domains. Compact finite difference schemes are used for the discretization of the grid generation equations. Compared to traditional schemes, compact finite difference schemes provide better representation of shorter length scales and this feature brings them closer to spectral methods.

Keywords: multigrid, elliptic grid generation, compact schemes

TABLE OF CONTENTS

ABSTRACT	iii
LIST OF TABLES	vii
LIST OF FIGURES	viii
ACKNOWLEDGMENTS	xiii
1. INTRODUCTION	1
2. 2D GRID GENERATION	5
2.1 Introduction	5
2.2 Elliptic Generating Systems	8
2.3 Generation Equations	10
2.4 Laplace System	12
2.5 Poisson System	14
2.6 Effect of Boundary Point Distribution	17
2.7 General Poisson-Type Systems	19
2.8 Derivation of the 2D Grid Generation Equations	21
2.9 Illustrations	29
3. DISCRETIZATION USING COMPACT SCHEMES	37

3.1 Approximation of the First Derivative 37

3.2 Approximation of the Second Derivative 44

3.3 Non-Periodic Boundaries 49

3.3.1 Boundary Formulation for the First Derivative 49

3.3.2 Boundary Formulation for the Second Derivative 50

3.4 Numerical Tests 52

4 MULTIGRID SOLUTION FOR ACCELERATING CONVERGENCE 56

4.1 Model Problems 56

4.2 Basic Iterative Methods 62

4.2.1 The Residual Equation 62

4.2.2 Properties of Relaxation Methods 63

4.2.3 An Analytical Approach 73

4.2.4 Some Numerical Experiments 76

4.3 Elements of Multigrid 83

4.3.1 The Multigrid Principle 83

4.3.2 Two Strategies 85

4.3.2.1 Nested Iteration 86

4.3.2.2 Coarse Grid Correction 86

4.3.3 Inter-Grid Transfer Mechanisms 87

4.3.3.1 Interpolation 87

4.3.3.2 Restriction 90

4.3.4 Coarse Grid Correction Scheme 92

4.3.5 A Recursive Formulation 100

4.4 Performance 109

4.5 Discretization Method 116

4.6 Illustrations 118

4.7 Conclusions 122

REFERENCES 124

LIST OF TABLES

Table I Truncation Error for the First Derivative Schemes 40

Table II Truncation Error for the Second Derivative Schemes 46

LIST OF FIGURES

Fig.2.1 The curvilinear coordinate system. 6

Fig.2.2 Boundary-conforming coordinate system. 8

Fig.2.3 The transformed region. 8

Fig.2.4 Examples of grids. 11

Fig.2.5 The coordinate lines. 13

Fig.2.6 Examples of concave and convex boundaries. 13

Fig.2.7 Effect of control function on coordinate lines. 15

Fig.2.8 Effect of control function at the boundary. 15

Fig.2.9 Effects at the boundary. 16

Fig.2.10 Effect of boundary point distribution. 17

Fig.2.11 Effect of control function on boundary point distribution. 18

Fig.2.12 Transformation from computational (ξ, η) space to a domain D in Cartesian (x,y) space. 21

Fig.2.13 Region about NACA0012 airfoil subdivided into four regions. 30

Fig.2.14 Domain boundaries near NACA0012 airfoil. 31

Fig.2.15 Complete O-type Euler grid. 32

Fig.2.16 Grid near NACA0012 airfoil. 33

Fig.2.17 Region about RAE2822 airfoil subdivided into four regions. 34

Fig.2.18 Complete C-type Navier-Stokes grid. 35

Fig.2.19 Grid near RAE2822 airfoil. 36

Fig.3.1 Numerical tests on the derivative scheme; the domain is divided into 128 intervals. The test functions contain Fourier modes up to wave number k_m with equal amplitude and random phases. One realization is plotted for each case with (1) $k_m = 9$; (2) $k_m = 21$; (3) $k_m = 31$; (4) $k_m = 63$; on (a). The corresponding exact derivative (solid line) and the calculated derivative (dashed line) are shown on (b). 53

Fig.3.2 Grid generated using fourth-order standard finite difference scheme. 54

Fig.3.3 Grid generated using fourth-order compact finite difference schemes. 55

Fig.4.1 A one-dimensional grid on the interval $0 \le x \le 1$. The grid spacing is given by $h = 1/N$. The *jth* grid point is $x_j = jh$ for $0 \le j \le N$. 57

Fig.4.2 Two-dimensional grid on the unit square. The circled points show the unknowns that are related at a typical grid point by the discrete equations. 58

Fig.4.3 (a) One-dimensional and (b) two-dimensional grids showing the red points (unmarked) and black points (•) for the red-black. 66

Fig.4.4 The modes $v_j = \sin\left(\frac{jk\pi}{N}\right), 0 \le j \le N$, with wavenumbers k=1,3,6. The *kth* mode consists of $k/2$ full sine waves on the interval. 68

Fig.4.5 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N=64 and with initial guesses of v_1, v_3, and v_6. The maximum norm of the error $\|e\|_\infty$ is plotted against the iteration number for 100 iterations. 69

Fig.4.6 Regular Gauss-Seidel iteration applied to the one-dimensional model problem with N=64 and with initial guesses of v_1, v_3, and v_6. The maximum norm of the error $\|e\|_\infty$ is plotted against the iteration number for 100 iterations. 70

Fig.4.7 Red-black Gauss-Seidel iteration applied to the one-dimensional model problem with N=64 and with initial guesses of v_1, v_3, and v_6. The maximum norm of the error $\|e\|_\infty$ is plotted against the iteration number for 100 iterations. 71

Fig.4.8 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N=64 and with initial guesses consisting of v_1, v_2 and v_6. The log of $\|e\|_\infty$ is plotted against the iteration number for 100 iterations. 72

Fig.4.9 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N = 64 and with initial guesses consisting of $\frac{1}{3}\left[\sin\left(\frac{j\pi}{N}\right)+\sin\left(\frac{6j\pi}{N}\right)+\sin\left(\frac{32j\pi}{N}\right)\right]$. The maximum norm of the error $\|e\|$ is plotted against the iteration number for 100 iterations. 73

Fig.4.10 Graphs of the Fourier modes of A on a grid with N=12. Modes with wavenumbers k= 1,2,3,4,6,8,9 are shown. The wavelength of the kth mode is $l = 24h/k$. 77

Fig.4.11 The weighted Jacobi method with $w = 1$ applied to the one-dimensional model problem with N=64. The initial guesses consist of the modes w_k for $1 \le k \le 63$. The graph shows the number of iterations required to reduce the norm of the initial error by a factor of 100 for each w_k. 79

Fig.4.12 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N=64. The initial guesses consist of the modes w_k for $1 \le k \le 63$. The graph shows the number of iterations required to reduce the norm of the initial error by a factor of 100 for each w_k. Note that for $w = 2/3$ the damping is the strongest for the oscillatory modes $(32 \le k \le 63)$. 80

Fig.4.13 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N=64 and an initial guess consisting of w_3. The figure shows the approximation after one iteration (on the left) and after ten iterations (on the right). 81

Fig.4.14 The weighted Jacobi method with $w = 2/3$ applied to the one dimensional model problem with N=64 and an initial guess consisting of w_{16}. The figure shows the approximation after one iteration (on the left) and after ten iterations (on the right). 81

Fig.4.15 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N=64 and an initial guess consisting of a combination of w_2 and w_{16}. The figure shows the approximation after one iteration (on the left) and after ten iterations (on the right). 82

Fig.4.16 A wave with wavenumber k=4 on Ω^h (N=12) is projected onto Ω^{2h} (N=6). The coarser grid "sees" a wave with k =4 which is more oscillatory on the coarser grid than on the fine grid. 84

Fig.4.17 Interpolation of a vector on the coarse grid Ω^{2h} to the fine grid Ω^h. 88

Fig.4.18 (a) If the exact error on Ω^h (indicated by $^\circ$ and •) is smooth, an interpolant of the coarse grid approximation e^{2h} (indicated by °) should give a good representation of the exact error. (b) If the exact error on Ω^h (indicated by $^\circ$ and •) is oscillatory, an interpolant of the coarse grid approximation e^{2h} (indicated by °) may give poor representation of the exact error. 89

Fig.4.19 Restriction by full weighting of a fine grid vector to the coarse grid. 90

Fig.4.20 The initial guess $v_j = 1/2\left[\sin\left(\frac{12j\pi}{N}\right)+\sin\left(\frac{30j\pi}{N}\right)\right]$ with N = 48 for a coarse grid correction scheme applied with the weighted Jacobi iteration $(w = 2/3)$ $\|e\|_\infty = 1.000$. 95

Fig.4.21 The approximation (and error) after one fine grid relaxation sweep superimposed upon the initial guess. $\|e\| = .430$. 96

Fig.4.22 The error after three fine grid relaxation sweeps superimposed upon the previous iterates. $\|e\|_\infty = .261$. 97

Fig.4.23 The fine grid residual is transferred to the coarse grid by full weighting. This figure shows the error after one relaxation sweep on the coarse grid residual equation, superimposed upon previous approximations. $\|e\|_\infty = .977 \times 10^{-1}$. 98

Fig.4.24 The error after relaxation sweeps on the coarse grid residual equation superimposed upon previous approximations. $\|e\|_\infty = .591 \times 10 - 1$. 99

Fig.4.25 The V-Cycle. 106

Fig.4.26 Schedule of grids for a FMV scheme on four levels. 108

Fig.4.27 The course of a four level (N = 16) V_cycle is illustrated by showing changes in the data array. The V and f arrays hold solution vectors and the right hand side vectors, respectively, on the four grids. 110

Fig.4.28 Maximum norms for the iterative method. 119

Fig.4.29 Maximum norms for the multigrid method. 120

Fig.4.30 Convergence times of iterative vs. multigrid method. 121

Fig.4.31 Convergence of mutigrid in Work units. 122

Fig.4.32 Grid generated near an airfoil. 123

ACKNOWLEDGMENTS

Author acknowledges the contribution of Dr. Richard Greechie, Dr. Chaoqun Liu, Dr. Raja Nassar, Dr. Weizong Dai and Dr. Natalie Zotov in providing support for the work, Dr. T. Srikanth and Mangilal Agarwal for reviewing the manuscript, Ms Frances Welsh and Ms Karen Brown for their sincere support and Miss Wein Tian for the help provided as a student worker.

CHAPTER 1

INTRODUCTION

The starting point of the multigrid method is the following golden rule: the amount of computational work should be proportional to the number of real physical changes in the computed system. Stalling numerical processes are considered wrong. This rule has been put forward as the 'golden rule of computation' by Achi Brandt [5]. Elliptic and hyperbolic partial differential equations are, by and large, at the heart of most mathematical models used in engineering and physics, giving rise to extensive computations. Often the problems that one would like to solve exceed the capacity of even the most powerful computers. Further, the time required is too great to allow inclusion of advanced mathematical models in the design process. Iterative processes for solving the algebraic equations arising from discretizing partial-differential equations are stalling numerical processes, in which the error has relatively small changes from one iteration to the next. The computer grinds very hard for very small or slow real physical effect with the use of too-fine discretization grids. In this case, in large parts of the computational domain the meshsize is much smaller than the real scale of solution changes. The multigrid method, or more generally, the Multi-Level Adaptive Technique (MLAT) can overcome such problems. Stalling numerical processes are usually related to the existence of several solution components with different scales, which conflict with each other. By interactively using several scales of discretization, multigrid techniques

resolve such conflicts, avoiding stalling and also being computationally efficient. Multigrid methods are a prime source of important advances in algorithmic efficiency and have a rapidly increasing number of users [6].

In this dissertation, the multigrid technique has been used for two-dimensional elliptic grid generation. Compact finite difference schemes are used for the discretization of the grid generation equations. The grid generation method is based on an elliptical grid generation method with maximum of robustness and a minimum of grid tuning parameters [3]. The method had been applied successfully to generate boundary conforming Navier-Stokes grids in blocks and block faces with complex shapes.

Since the pioneering work of Thompson on elliptic grid generation, it is known that systems of elliptic second-order partial differential equations produce the best possible grids in terms of smoothness and grid point distribution. These systems of elliptic second-order partial differential equations are Poisson-type systems where control functions have to be specified. The general approach is to compute the control functions at the boundary and then to interpolate them from the boundaries into the field. The disadvantage of this approach is that it is then impossible to prove that the system of Poisson equations defines a one-to-one mapping so that the computed grid may contain grid folding. Thompson's and Warsi's original idea was to define the control functions by a transformation. The advantage of this approach is that the corresponding Poisson system is a one-to-one mapping if the transformation is one-to-one. The one-to-one transformations are constructed using nonlinear transfinite algebraic transformation.

The grid generation method uses a composite mapping. A parameter (coordinate) system is introduced, which only depends on the shape of the domain. This mapping is

the composition of an algebraic transformation and an elliptic transformation based on Laplace equations. The algebraic transformation depends on the prescribed boundary grid point distribution and is a differential one-to-one. The elliptical transformation maps the parameter space onto the domains or surfaces in physical space. The composition of these two mappings is a differential one-to-one mapping from the computational space onto the domains or surfaces in physical space. The elliptic transformation is independent of the prescribed boundary grid point distribution. It is considered as the property of the domain or surface. Thompson [1] and Warsi [2] have shown that the composite mapping obeys an elliptic Poisson system with control functions completely specified by algebraic transformations. The computed grids may not be orthogonal. The algebraic transformation can be redefined to obtain orthogonality at the boundary [3]. The nonlinear elliptic Poisson equations are solved by Picard iteration and using compact finite difference schemes for the discretization of the equations [4].

Compared with traditional finite difference schemes, compact schemes provide better representation of the shorter length scales. This feature brings them closer to the spectral methods, while freeing them to choose the mesh geometry, and maintaining the boundary condition. Direct numerical simulations of many physical phenomena, including turbulent flow, require all the relevant scales to be properly represented in numerical model, thereby requiring the use of spectral methods. The resolution characteristics of compact finite difference schemes reveal the accuracy with which they represent the exact result over a full range of length scales that can be realized on a given mesh. Moreover, compact finite differences can be used to generate higher-order schemes using a smaller stencil size.

In this dissertation a non-linear multigrid algorithm is presented for accelerating the convergence of the non-linear elliptic Poisson grid generation equations used for grid generation. The dissertation is organized as follows: In Chapter 2, the Laplace equations and the algebraic transformation are presented for the domains in two-dimensional physical space. Sections 2.1-2.7 and section 2.9 are quoted from *Numerical Grid Generation* by J.F. Thompson, Z.U.A. Warsi and C.W. Mastin [1]. It is useful in explaining the basic concepts of grid generation. Chapter 3 describes the use of compact finite difference schemes for the discretization of the grid generating equations. The discussion is based on *Compact finite difference schemes with spectral-like resolution* by Sanjiva K.Lele [4]. In Chapter 4, the fundamentals of multigrid method are described and a non-linear multigrid algorithm is presented. In section 4.3.5 an explanation of the non-linear multigrid algorithm has been given. In section 4.4, the performance characteristics of the algorithm is discussed and illustrations are made for comparison with the standard iterative method based on the non-linear elliptic Poisson grid generation equations.

CHAPTER 2

2D GRID GENERATION

2.1 Introduction

The numerical solution of partial differential equations requires some discretization of the field into a collection of points or elemental volumes (cells). The differential equations are approximated by a set of algebraic equations on this collection, and this system of algebraic equations is then solved to produce a set of discrete values that approximates the solution of the partial differential system over the field. The discretization of the field requires some organization for the solution thereon to be efficient; i.e., it must be possible to readily identify the points or cells neighboring the computation site. The discretization must conform to the boundaries of the region in such a way that boundary conditions can be accurately represented. This organization is provided by a coordinate system, and the need for alignment with the boundary is reflected in the routine choice of Cartesian coordinates for rectangular regions, cylindrical coordinates for circular regions, etc

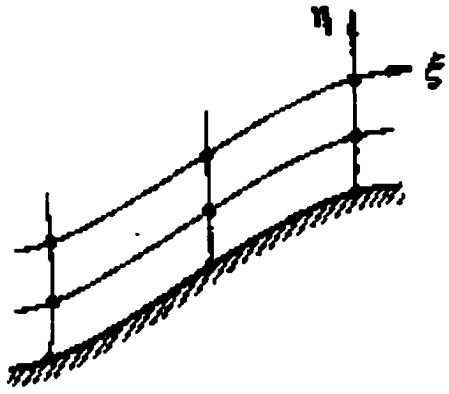


Fig.2.1 The curvilinear coordinate system.

The current interest in numerically-generated, boundary-conforming coordinate systems arises from this need for organization of the discretization of the field for general regions, i.e., to provide computationally for arbitrary regions what is available in the handbook for simple regions. The curvilinear coordinate system covers the field and has coordinate lines (surfaces) coincident with all boundaries. The distribution of lines should be smooth, with concentration in regions of strong solution variation, and the system should ultimately be capable of sensing these variations and dynamically adjusting itself to resolve them.

A numerically generated grid is an organized set of points formed by the intersections of the lines of a boundary-conforming curvilinear coordinate system. The cardinal feature of such a system is that some coordinate line (surface in 3D) is coincident with each segment of the boundary of the physical region. The use of coordinate line intersections to define the grid points provides an organizational structure that allows all computation to be done on a fixed square grid when the partial differential

equations of interest have been transformed so that the curvilinear coordinates replace the Cartesian coordinates as the independent variables.

This grid frees the computational simulation from restriction to certain boundary shapes and allows general codes to be written in which the boundary shape is specified simply by input. The boundaries may also be in motion, either as specified externally or in response to the developing physical solution. Similarly, the coordinate system may adjust to follow variations developing in the evolving physical solution. In any case, the numerically generated grid allows all computation to be done on a fixed square grid in the computational field that is always rectangular by construction.

Basically, the procedures for the generation of curvilinear coordinate systems are of two general types: (1) numerical solution of partial differential equations and (2) construction by algebraic interpolation. In the former, the partial differential system may be elliptic, parabolic or hyperbolic. It is well known that systems of elliptic second-order partial differential equations produce the best possible grids in terms of smoothness and grid-point distribution. These systems of elliptic second-order partial differential equations are Poisson type systems with the specified control functions [1].

2.2 Elliptic Generating Systems

The generation of a boundary-conforming coordinate system is accomplished by the determination of the values of the curvilinear coordinates in the interior of a physical region from specified values (and/or slopes of the coordinate lines intersecting the boundary) on the boundary of the region.

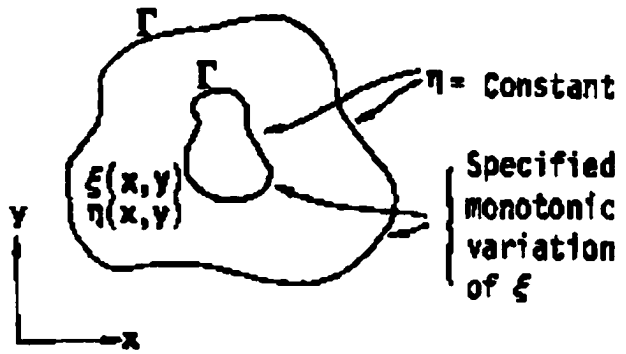


Fig.2.2 Boundary-conforming coordinate system.

One coordinate will be constant on each segment of the physical boundary curve (surface in 3D), while the other varies monotonically along the segment. The equivalent problem in the transformed region is the determination of values of the physical (Cartesian or other) coordinates in the interior of the transformed region from specified values and/or slopes on the boundary of this region.

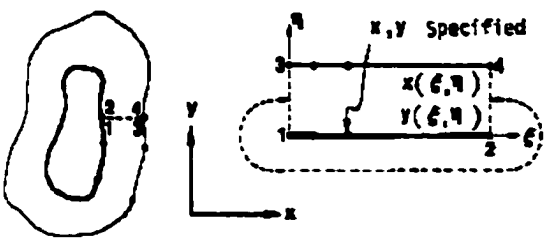


Fig.2.3 The transformed region.

This is a more amenable problem for computation since the boundary of the transformed region has horizontal and vertical segments, so that this region is composed of contiguous rectangular blocks, at least in the sense of being joined by re-entrant boundaries (branch cuts).

The generation of field values of a function from boundary values can be done in various ways, e.g., by interpolation between the boundaries, etc. The solution of such a boundary-value problem, however, is a classic problem of partial differential equations, so it is logical to take the coordinates to be solutions of a system of partial differential equations. If the coordinate points (and/or slopes) are specified on the entire closed boundary of the physical region, the equations must be elliptic, whereas if the specification were on only a portion of the boundary, the equations would be parabolic or hyperbolic. This latter case would occur, for instance, when an inner boundary of a physical region is specified, but a surrounding outer boundary is arbitrary. The present chapter treats the general case of a completely specified boundary, which requires an elliptic partial differential system [1].

Some general discussion of elliptic generation systems has been given in Ref. [10], and numerous references to the application thereof appear in the surveys given by Ref. [11] and [12].

2.3 Generation Equations

The extremum principles-- i.e., that extrema of solutions cannot occur within the field that are exhibited by some elliptic systems-- can serve to guarantee a one-to-one mapping between the physical and transformed regions (Ref. [13] and [14]). Thus, since the variation of the curvilinear coordinate along a physical boundary segment must be monotonic and is over the same range along facing boundary segments, it clearly follows that extrema of the curvilinear coordinates cannot be allowed in the interior of the physical region, else overlapping of the coordinate system will occur. Note that it is the extremum principles of the partial differential system in the physical space, i.e., with the curvilinear coordinates as the dependent variables, that is relevant since it is the curvilinear coordinates, not the Cartesian coordinates, that must be constant or monotonic on the boundaries. Thus it is the form of the partial differential equations in the physical space, i.e., containing derivatives with respect to the Cartesian coordinates, that are important.

Another important property in regard to coordinate system generation is the inherent smoothness that prevails in the solutions of elliptic systems. Furthermore, boundary slope discontinuities are not propagated into the field. Finally, the smoothing tendencies of elliptic operators, and the extremum principles, allow grids to be generated for any configurations without overlap of grid lines. Some examples appear below:

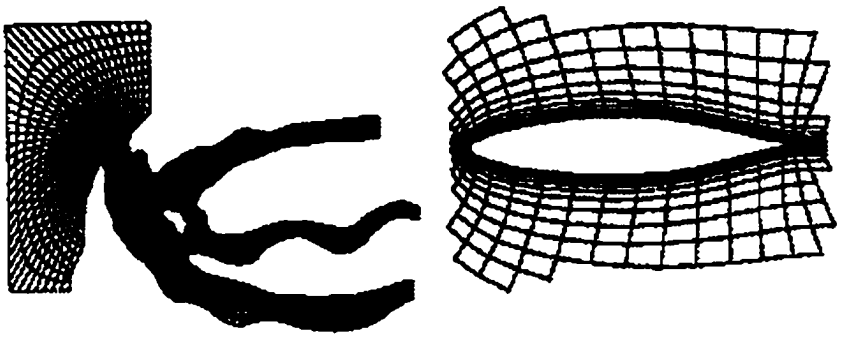


Fig.2.4 Examples of grids.

There are thus a number of advantages to using a system of elliptic partial differential equations as a means of coordinate system generation. A disadvantage, of course, is that a system of partial differential equations must be solved to generate the coordinate system [1].

The historical progress of the form of elliptic systems used for grid generation has been traced in Ref. [11]. Numerous examples of the generation and application of coordinate systems generated from elliptic partial differential equations are covered in the above reference, as well as in Ref. [15].

2.4 Laplace System

The most simple elliptic partial differential system, and one that does exhibit an extremum principle and considerable smoothness, is the Laplace system:

$$\nabla^2 \xi^i = 0$$

This generation system guarantees a one-to-one mapping for boundary-conforming curvilinear coordinate systems on general closed boundaries. These equations can, in fact, be obtained from the Euler equations for the minimization of the integral.

$$I = \iiint \sum_{i=1}^{3} \left| \nabla \xi^i \right|^2 dV$$

Since the coordinate lines are located at equal increments of the curvilinear coordinate, the quantity $\left|\nabla \xi^i\right|$ can be considered a measure of the grid point density along the coordinate line on which ξ^i varies; i.e., ξ^i must change rapidly in physical space where grid points are clustered. Minimization of this integral thus leads to the smoothest coordinate line distribution over the field.

With this generating system the coordinate lines will tend to be equally spaced in the absence of boundary curvature because of the strong smoothing effect of the Laplacian but will become more closely spaced over convex boundaries, and less so over concave boundaries, as illustrated below. (In this and other illustrations and applications in two dimensions, ξ^1 and ξ^2 will be denoted ξ and η, respectively, while x and y will be used for x_1 and x_2.).

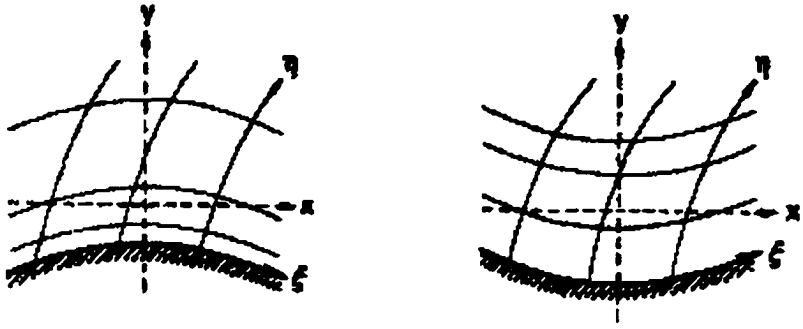


Fig.2.5 The coordinate lines.

In the left figure we have $\eta_{xx} > 0$ because of the convex (to the interior) curvature of the lines of constant η(η-lines). Therefore, it follows that $\eta_{yy} < 0$, and hence the spacing between the η-lines must increase with y. The η-lines thus will tend to be more closely spaced over such a convex boundary segment. For concave segments, illustrated in the right figure, we have $\eta_{xx} < 0$, so that η_{yy} must be positive, and hence the spacing of the η-lines must decrease outward from this concave boundary. Some examples of grids generated from the Laplace system are shown below. The inherent smoothness and the behavior near concave and convex boundaries are evident in these examples [1].

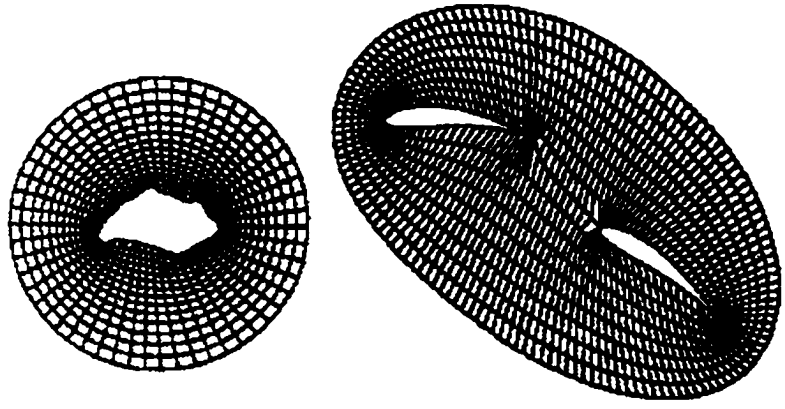


Fig.2.6 Examples of concave and convex boundaries.

2.5 Poisson system

Control of the coordinate line distribution in the field can be exercised by generalizing the elliptic generating system to Poisson equations,

$$\nabla^2 \xi^i = P^i \qquad (2.5.1)$$

in which the "control functions" P^i can be fashioned to control the spacing and orientation of the coordinate lines. The extremum principles may be weakened or lost completely with such a system, but the existence of an extremum principle is a sufficient but not a necessary condition for a one-to-one mapping, so that some latitude can be taken in the form of the control functions.

Considering the equation $\nabla^2 \eta = Q$ and the figures above (P^1 = P and P^2 = Q in the illustrations here), since a negative value of the control function would tend to make η_{yy} more negative, it follows that negative values of Q will tend to cause the coordinate line spacing in the cases shown above to increase more rapidly outward from the boundary. Generalizing, negative values of the control function Q will cause the η-lines to tend to move in the direction of decreasing η, while negative values of P in $\nabla^2 \eta = P$ will cause ξ-lines to tend to move in the direction of decreasing ξ. These effects are illustrated below for a η-line boundary:

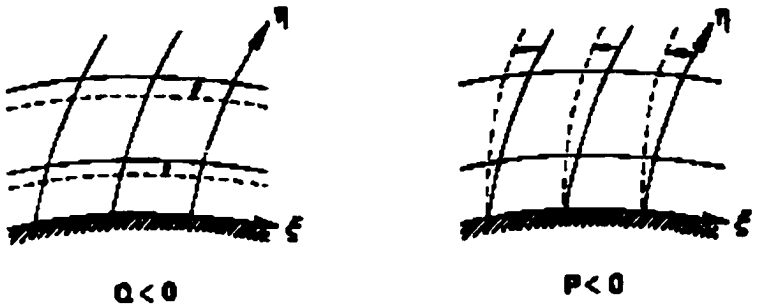


Fig.2.7 Effect of control function on coordinate lines.

With the boundary values fixed, the ξ-lines here cannot change the intersection with the boundary. The effect of the control function P in this case is to change the angle of intersection at the boundary, causing the ξ-lines to lean in the direction of decreasing ξ.

These effects are illustrated in the following figures:

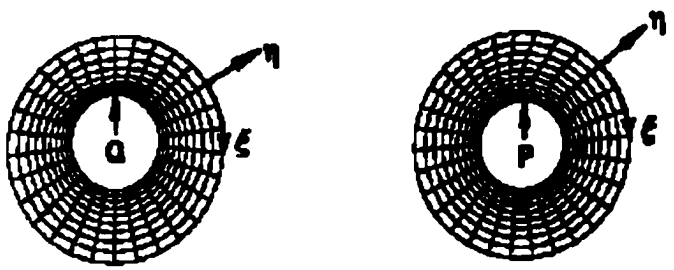


Fig.2.8 Effect of control function at the boundary.

Here the ξ-lines are radial and the η-lines are circumferential. In the left illustration the control function Q is locally non-zero near a portion of the inner boundary as indicated, so the η-lines move closer to that portion of the boundary while in the right figure, P is locally non-zero, resulting in a change in intersection angle of the ξ-lines with that portion of the boundary. If the intersection angle, instead of the point location, on the

boundary is specified, so that the points are free to move along the boundary, then the ξ-lines would move toward lines with lower values of ξ:

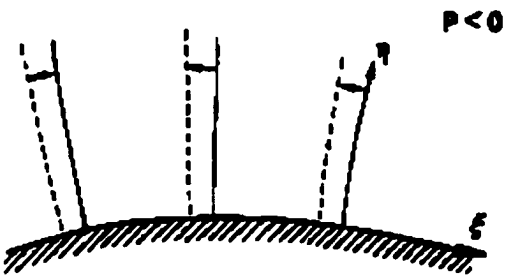


Fig.2.9 Effects at the boundary.

In general, a negative value of the Laplacian of one of the curvilinear coordinates causes the lines on which that coordinate is constant to move in the direction in which that coordinate decreases. Positive values of the Laplacian naturally result in the opposite effect [1].

2.6 Effect of Boundary Point Distribution

Because of the strong smoothing tendencies inherent in the Laplacian operator, in the absence of the control functions, i.e., with $P_i = 0$, the coordinate lines will tend to be generally equally spaced away from the boundaries regardless of the boundary point distribution. For example, the simple case of a coordinate system comprised of horizontal and vertical lines in a rectangular physical region, (the right figure below) cannot be obtained as a solution of Equation (2.5.1) with P=Q=0 unless the boundary points are equally spaced.

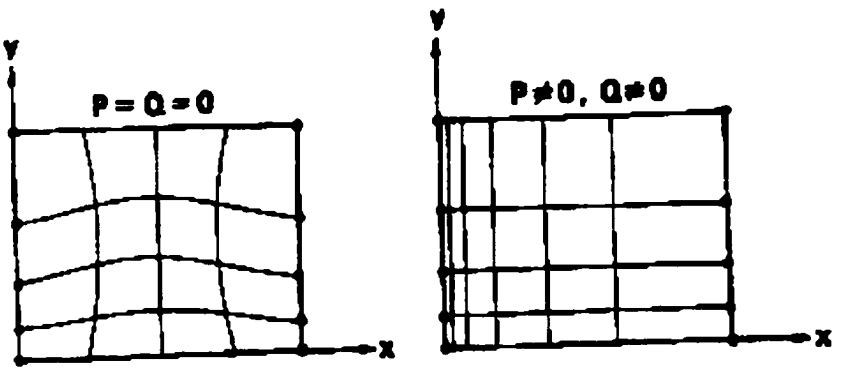


Fig.2.10 Effect of boundary point distribution,

With $\xi_{yy} = \eta_{xx} = 0$, Equation (2.5.1) reduces to

$$\xi_{xx} = P_i \qquad \eta_{yy} = Q$$

and thus P and Q cannot vanish if the point distribution is not uniform on the horizontal and vertical boundaries, respectively. With P=Q=0 the lines tend to be equally spaced away from the boundary. These effects are illustrated further in the figures below. Here the control functions are zero in the left figure.

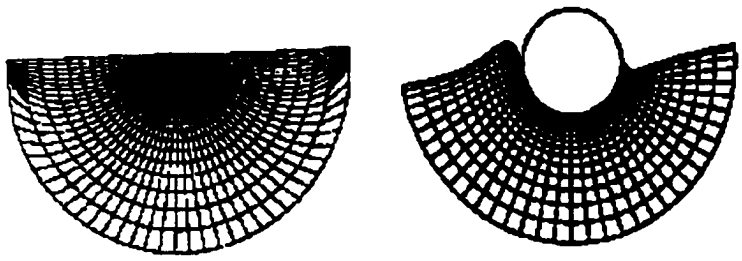


Fig.2.11 Effect of control function on boundary point distribution.

Although the spacing is not uniform on the semicircular outer boundary in this figure, the angular spacing is essentially uniform away from the boundary. By contrast, nonzero control functions in the right figure, evaluated from the boundary point distribution, cause the field spacing to follow that on the boundary. Thus, if the coordinate lines in the interior of the region are to have the same general spacing as the point distributions on the boundaries that these lines connect, it is necessary to evaluate the control functions to be compatible with the boundary point distribution. This evaluation of the control functions from the boundary point distribution is discussed more fully in Section 2.8 of this chapter [1].

2.7 General Poisson-Type Systems

If a curvilinear coordinate system, ξ^i (i=1,2,3), which satisfies the Laplace system

$$\nabla^2 \xi^i = 0$$

is transformed to another coordinate system, ξ^i (i = 1,2,3), then the new curvilinear coordinates, ξ^i satisfy the inhomogeneous elliptic system (cf. Ref. [10])

$$\nabla^2 \xi^l = P^l \qquad (2.7.1)$$

where

$$P^l = \sum_{j=1}^{3}\sum_{k=1}^{3} g^{jk} P^l_{jk} \qquad (2.7.2)$$

with the P^i_{jk} defined by the transformation from ξ^i to ξ^i:

$$P^i_{jk} = \sum_{m=1}^{3}\sum_{n=1}^{3} \frac{\partial \xi^m}{\partial \xi^j}\frac{\partial \xi^n}{\partial \xi^k}\frac{\partial^2 \xi^l}{\partial \xi^m \partial \xi^n} \qquad (2.7.3)$$

(It may be noted that if the subsequent transformation is one-dimensional, i.e., if $\partial \xi^{i'}/\partial \xi^{j'} = \delta^i_j \partial f^i/\partial \xi^i$ then only the three functions P^i_{ii} with i=1,2,3, are nonzero.)

These results show that a grid with lines concentrated by applying a subsequent transformation (often called a "stretching" transformation) to a grid generated as the solution of the Laplace system could have been generated directly as the solution of the Poisson system (2.7.1) with appropriate "control functions", P^i_{jk} derived from the subsequent concentrating transformation according to Equation. (2.7.3). Therefore, it is

appropriate to adopt this Poisson system (2.7.1) as the generation system, but with the control functions specified directly rather than through a subsequent transformation.

Thus an appropriate generation system can be defined by Equations (2.7.1) and (2.7.2):

$$\nabla^2 \xi^l = \sum_{j=1}^{3} \sum_{k=1}^{3} g^{jk} P_{jk}^i \qquad (2.7.4)$$

with the control functions, P_{jk}^i considered to be specified. The basis of the generation system (2.7.4) is that it produces a coordinate system that corresponds to the subsequent application of a stretching transformation to a coordinate system generated for maximum smoothness [1].

2.8 Derivation of the 2D Grid Generation Equations

The grid generation method described uses a composite mapping [3]. It is a composition of an algebraic transformation and an elliptical transformation based on Laplace equations. The algebraic transformation is a differential one-to-one mapping from computational space onto a parameter space. The parameter space and the computational space are unit squares. The algebraic transformation will depend only upon the prescribed boundary grid point distribution. The control functions are defined based on the algebraic transformation. The elliptical transformation is a differential one-to-one mapping from parameter space onto the physical domain. The elliptical transformation depends only on the shape of the domain and is independent of the prescribed boundary grid point distribution. The composition of these two mappings defines the interior grid point distribution and is a differentiable one-to-one mapping from the computational space onto the physical domain.

Consider a simply connected domain D in two dimensions with Cartesian coordinates $\mathbf{x} = (x,y)^T$. Let D be bounded by four edges, E_1, E_2, E_3, E_4. Let (E_1, E_2) and (E_3, E_4) be the two opposite edges as shown in the Fig.2.12. The computational space C and the parameter P space are unit squares with Cartesian coordinates $\xi = (\xi, \eta)^T$ and $\mathbf{s} = (s,t)^T$ respectively.

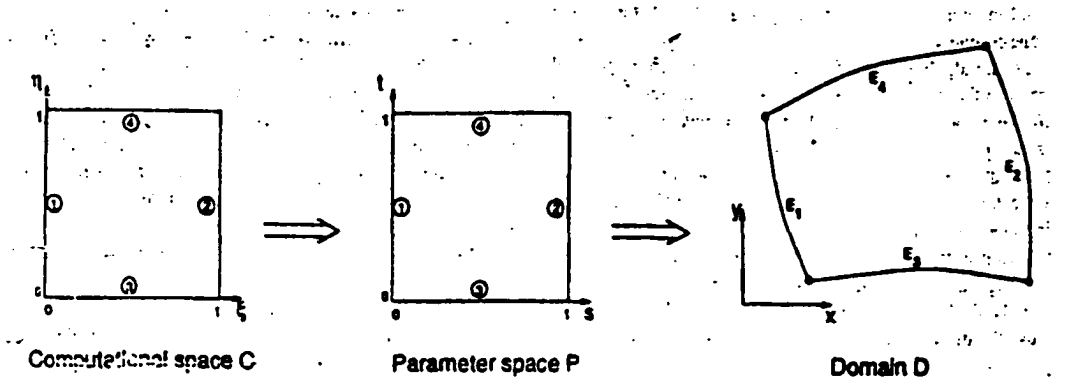


Fig.2.12 Transformation from computational (ξ, η) space to a domain D in Cartesian (x,y) space.

Assume that a mapping $x : \partial C \to \partial D$ is prescribed which maps the boundary of C one-to-one on the boundary of D. This mapping defines the boundary grid point distribution. Assume that

$$\xi \equiv 0 \text{ at edge } E_1 \text{ and } \xi \equiv 1 \text{ at edge } E_2,$$

$$\eta \equiv 0 \text{ at edge } E_3 \text{ and } \eta \equiv 1 \text{ at edge } E_4$$

We will construct a mapping $x : \partial C \to \partial D$ which obeys the boundary grid conditions and which is differential one-to-one mapping. Furthermore, we require that the interior grid point distribution is a good reflection of the prescribed boundary grid point distribution. A natural mapping $x : \partial C \to \partial D$ exists which obeys these requirements. This mapping is the composition of an algebraic transformation and an elliptic transformation based on Laplace equations. The algebraic transformation is a differentiable one-to-one mapping from computational space C onto the parameter space P. The algebraic transformation will depend only upon the prescribed boundary grid point distribution at the four edges of the domain D. The elliptic transformation is a differential one-to-one mapping from parameter space P onto the domain D. The elliptic transformation will only depend on the

shape of the domain D and is thus independent of the prescribed boundary grid point distribution. The elliptic transformation is to be considered as a property of the domain D. The composition of these two mappings defines the interior grid point distribution and is a differential one-to-one mapping from computational domain C onto the domain D. Require the parameters s and t of the parameter space to obey:

$$s \equiv 0 \text{ at edge } E_1 \text{ and } s \equiv 1 \text{ at edge } E_2,$$

$$s \text{ is the normalized arclength along edges } E_3 \text{ and } E_4$$

$$t \equiv 0 \text{ at edge } E_3 \text{ and } t \equiv 1 \text{ at edge } E_4$$

$$t \text{ is the normalized arclength along edges } E_1 \text{ and } E_2$$

Thus $s : \partial D \to \partial P$ is defined by these requirements. In the interior of D we require that s and t are harmonic functions of x and y, and they obey Laplace equations:

$$\Delta s = \frac{\partial^2 s}{\partial x^2} + \frac{\partial^2 s}{\partial y^2} = s_{xx} + s_{yy} = 0, \qquad (2.8.1)$$

$$\Delta t = \frac{\partial^2 t}{\partial x^2} + \frac{\partial^2 t}{\partial y^2} = t_{xx} + t_{yy} = 0 \qquad (2.8.2)$$

The two Laplace equations $\Delta s = 0$ and $\Delta t = 0$, together with the specified boundary conditions, define the mapping $s : \partial D \to \partial P$. This mapping depends only upon the shape of the domain D and is independent of the prescribed boundary grid point distribution. By interchanging the dependent and the independent variables, a non-linear elliptic partial differential equation can be derived for $x : \partial P \to \partial D$. We have to solve a non-linear elliptic boundary value problem in P in order to define this mapping. This mapping defines our elliptic transformation. It is well know that this mapping is differential and one-to-one [16].

The algebraic transformation must be a differential one-to-one mapping from computational space C onto the parameter space P. Because $x:\partial C \to \partial D$ is prescribed and $x:\partial P \to \partial D$ is as defined above, it follows that $s:\partial D \to \partial P$ is also defined.

From the preceding requirements it follows that

$$s(0,\eta)=0, \qquad s(1,\eta)=0,$$

$$s(\xi,0)=s_{E_3}(\xi), \quad s(\xi,1)=s_{E_4}(\xi), \qquad (2.8.3)$$

where the functions s_{E_3}, s_{E_4} are monotonically increasing and

$$t(\xi,0)=0, \quad t(\xi,1)=0,$$

$$t(0,\eta)=t_{E_3}(\eta), \quad t(1,\eta)=t_{E_4}(\eta), \qquad (2.8.4)$$

where the functions t_{E_3}, t_{E_4} are monotonically increasing.

Thus the four functions $t_{E_3}(\eta)$, $t_{E_4}(\eta), s_{E_3}(\xi)$, $s_{E_4}(\xi)$ are defined by the boundary grid point distribution.

The mapping $s:\partial C \to \partial P$ is defined by two algebraic equations:

$$s = s_{E_3}(\xi)(1-t)+s_{E_4}(\xi)t, \qquad (2.8.5)$$

$$t = t_{E_1}(\eta)(1-s)+t_{E_2}(\eta)s \qquad (2.8.6)$$

Equation (2.8.5) implies that a coordinate line $\xi = const$ is mapped to the parameter space P as a straight line: s is a linear function of t; Equation (2.8.6) implies that a grid line $\eta = const$ is also mapped to P as a straight line: t is a linear function of s. For given values of ξ and η, the corresponding s and t values are found as the intersection point of the two straight lines. Hence the system defined by Equations (2.8.5), (2.8.6) is called the 'algebraic straight line transformation' because of the use of straight lines in parameter

space P. It can be easily verified that this system defines a differentiable one-to-one mapping because of the positiveness of the Jacobian: $s_\xi t_\eta - s_\eta t_\xi > 0$.

The algebraic transformations $s:\partial C \to \partial P$ and the elliptic transformation $x:\partial P \to \partial D$ are differential and one-to-one. Thus the composite mapping $x:\partial C \to \partial D$ defined as $x(\xi) = x(s(\xi))$ is also differentiable and one –to –one. Due to the properties of the basic mappings, we may indeed expect that the interior grid point distribution be a good reflection of the boundary grid point distribution.

It has been noted by Warsi and Thompson that the composite mapping will obey an elliptic system of Poisson equations. However, the system of Poisson equations as given in [1,2] is not so useful because it contains control functions that depend also on the derivatives of the inverse mapping $s:\partial P \to \partial C$. It will be shown below the expressions for the control functions which only depend on the derivatives of the mapping $s:\partial C \to \partial P$ itself.

First, introduce the two covariant base vectors

$$a_1 = \frac{\partial x}{\partial \xi} = x_\xi, \qquad a_2 = \frac{\partial x}{\partial \eta} = x_\eta, \qquad (2.8.7)$$

and define the covariant metric tensor components as the inner product of the covariant base vectors

$$a_{ij} = (a_i, a_j), \quad i = \{1,2\},\ j = \{1,2\}. \qquad (2.8.8)$$

Then the contravariant base vectors a^1 and a^2 are defined according to the rules

$$(a^i, a_j) = \delta^i_j, \quad i = \{1,2\},\ j = \{1,2\}. \qquad (2.8.9)$$

with δ^i_j the Kronecker symbol. Define the contravariant metric tensor components

$$a^{ij} = (a^i, a^j), \quad i = \{1,2\},\ j = \{1,2\}, \qquad (2.8.10)$$

so that

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix}\begin{pmatrix} a^{11} & a^{12} \\ a^{12} & a^{22} \end{pmatrix}=\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad (2.8.11)$$

and

$$a^1 = a^{11}a_1 + a^{12}a_2, \quad a^2 = a^{11}a_1 + a^{22}a_2. \qquad (2.8.12)$$

Introduce the determinant J^2 of the covariant metric tensor:

$$J^2 = a_{11}a_{22} - a_{12}^2.$$

Consider an arbitrary function $\phi = \phi(\xi,\eta)$. Then ϕ is also defined in domain D and the Laplacian of ϕ is expressed as

$$\Delta\phi = \phi_{xx} + \phi_{yy} = \frac{1}{J}\left\{\left(Ja^{11}\phi_\xi + Ja^{12}\phi_\eta\right)_\xi + \left(Ja^{12}\phi_\xi + Ja^{22}\phi_\eta\right)_\eta\right\}, \qquad (2.8.13)$$

which may be found in every textbook on tensor analysis and differential geometry (see [17]). Take as special cases respectively $\phi \equiv \xi$ and $\phi \equiv \eta$. Then Equation (2.8.13) yields

$$\Delta\xi = \frac{1}{J}\left\{\left(Ja^{11}\right)_\xi + \left(Ja^{12}\right)_\eta\right\}, \quad \Delta\eta = \frac{1}{J}\left\{\left(Ja^{12}\right)_\xi + \left(Ja^{22}\right)_\eta\right\}. \qquad (2.8.14)$$

Thus the Laplacian of ϕ can also be expressed as

$$\Delta\phi = a^{11}\phi_{\xi\xi} + 2a^{12}\phi_{\xi\eta} + a^{22}\phi_{\eta\eta} + \Delta\xi\phi_\xi + \Delta\eta\phi_\eta. \qquad (2.8.15)$$

Substitution of $\phi \equiv s$ and $\phi \equiv t$ respectively in this equation gives

$$\Delta s = a^{11}s_{\xi\xi} + 2a^{12}s_{\xi\eta} + a^{22}s_{\eta\eta} + \Delta\xi s_\xi + \Delta\eta s_\eta \qquad (2.8.16)$$

$$\Delta t = a^{11}t_{\xi\xi} + 2a^{12}t_{\xi\eta} + a^{22}t_{\eta\eta} + \Delta\xi t_\xi + \Delta\eta t_\eta. \qquad (2.8.17)$$

Using these equations and the requirement that s and t are harmonic in domain D thus, $\Delta s = 0$ and $\Delta t = 0$, we find the expressions for the Laplacian of ξ and η.

$$\begin{pmatrix}\Delta\xi\\ \Delta\eta\end{pmatrix} = a^{11}P_{11} + 2a^{12}P_{12} + a^{22}P_{22}, \qquad (2.8.18)$$

where

$$P_{11} = -T^{-1}\begin{pmatrix}s_{\xi\xi}\\ t_{\xi\xi}\end{pmatrix}, P_{12} = -T^{-1}\begin{pmatrix}s_{\xi\eta}\\ t_{\xi\eta}\end{pmatrix}, P_{22} = -T^{-1}\begin{pmatrix}s_{\eta\eta}\\ t_{\eta\eta}\end{pmatrix} \qquad (2.8.19)$$

and the matrix T is defined as

$$T = \begin{pmatrix}s_{\xi} & s_{\eta}\\ t_{\xi} & t_{\eta}\end{pmatrix} \qquad (2.8.20)$$

The six coefficients of the vectors $P_{11} = (P^1_{11}, P^2_{11})^T$, $P_{12} = (P^1_{12}, P^2_{12})^T$ and $P_{22} = (P^1_{22}, P^2_{22})^T$ are called the control functions. The six control functions are easily computed for a given algebraic transformation $s = s(\xi)$.

Finally substitution of $\phi \equiv x$ in Equation (2.8.15) yields

$$\Delta x = a^{11}x_{\xi\xi} + 2a^{12}x_{\xi\eta} + a^{22}x_{\eta\eta} + \Delta\xi x_{\xi} + \Delta\eta x_{\eta}. \qquad (2.8.21)$$

Substitution of Equation (2.8.18) into this equation and using the fact that $\Delta x \equiv 0$ we arrive at the Poisson grid generating system,

$$a^{11}x_{\xi\xi} + 2a^{12}x_{\xi\eta} + a^{22}x_{\eta\eta} + \left(a^{11}P^1_{11} + 2a^{12}P^1_{12} + a^{22}P^1_{22}\right)x_{\xi} + \left(a^{11}P^2_{11} + 2a^{12}P^2_{12} + a^{22}P^2_{22}\right)x_{\eta} = 0. \qquad (2.8.22)$$

Using Equations (2.8.8) and (2.8.11) we find the following expressions for the contravariant metric tensor components:

$$J^2a^{11} = a_{22} = \left(x_{\eta}, x_{\eta}\right),$$

$$J^2a^{12} = -a_{12} = -\left(x_{\xi}, x_{\eta}\right)$$

$$J^2 a^{22} = a_{11} = (x_\xi, x_\xi),$$

Thus the Poisson grid generating system defined by Equation (2.8.22) can be simplified by multiplication with J^2. Then we get

$$\alpha^{11} x_{\xi\xi} + 2\alpha^{12} x_{\xi\eta} + \alpha^{22} x_{\eta\eta} + (\alpha^{11} P_{11}^1 + 2\alpha^{12} P_{12}^1 + \alpha^{22} P_{22}^1) x_\xi + (\alpha^{11} P_{11}^2 + 2\alpha^{12} P_{12}^2 + \alpha^{22} P_{22}^2) x_\eta = 0, \tag{2.8.24}$$

and

$$\alpha^{11} = (x_\eta, x_\eta), \quad \alpha^{12} = -(x_\xi, x_\eta), \quad \alpha^{22} = (x_\xi, x_\xi). \tag{2.8.25}$$

These equations together with the expressions for the control functions P_{ij}^k given by Equation (2.8.19) form the 2D grid generating system of elliptic partial differential equations. The discretization of this Poisson system is described in chapter 4.

Rewriting the non-linear elliptical Poisson grid generating system we get,

$$P\mathbf{x}_{\xi\xi} + 2Q\mathbf{x}_{\xi\eta} + R\mathbf{x}_{\eta\eta} + S\mathbf{x}_\xi + T\mathbf{x}_\eta = 0 \tag{2.8.26}$$

where, $P = (\mathbf{x}_\eta, \mathbf{x}_\eta)$, $Q = -(\mathbf{x}_\xi, \mathbf{x}_\eta)$,

$$R = (\mathbf{x}_\xi, \mathbf{x}_\xi),$$

$$S = P P^1_{11} + 2Q P^1_{12} + R P^1_{22},$$

$$T = P P^2_{11} + 2Q P^2_{12} + R P^2_{22}.$$

The control functions are given by

$$P_{11} = -T^{-1}\begin{pmatrix} s_{\xi\xi} \\ t_{\xi\xi} \end{pmatrix}, P_{12} = -T^{-1}\begin{pmatrix} s_{\xi\eta} \\ t_{\xi\eta} \end{pmatrix}, P_{22} = -T^{-1}\begin{pmatrix} s_{\eta\eta} \\ t_{\eta\eta} \end{pmatrix} \tag{2.8.27}$$

and the matrix T is defined as

$$T = \begin{pmatrix} s_\xi & s_\eta \\ t_\xi & t_\eta \end{pmatrix} \qquad (2.8.28)$$

The two algebraic equations that define the transformation are given by

$$s = s_{E_3}(\xi)(1-t) + s_{E_4}(\xi)t \qquad (2.8.29)$$

$$t = t_{E_1}(\eta)(1-s) + t_{E_2}(\eta)s \qquad (2.8.30)$$

2.9 Illustrations

Examples of grids in 2D domains are shown in Figs.2.13-2.19. All the grids are grid-folding free and the interior grid point distribution is a good reflection of the prescribed boundary grid point distribution. An initial grid (obtained with algebraic grid generation) is required as the starting solution for the non-linear elliptic Poisson system. The final grid elliptic grid is independent of the initial grid. Moreover, the quality of the initial grid is unimportant and severe grid folding of the initial grid is allowed. Fig.2.13 shows a region about a NACA0012 airfoil subdivided into four domains. The domains have common edges. The total number of edges is 12. The boundary grid point distribution is prescribed at all 12 edges. Fig.2.14 shows a complete O-type Euler grid. A close-up near the airfoil of the domains and the grid is shown in Fig.2.15 and Fig.2.16. Fig.2.17 shows a region about a RAE822 airfoil, also subdivided into four domains. The boundary grid point distribution is prescribed at all 12 edges. Fig.2.18 shows a C-type Navier-Stokes grid. A close-up of the grid near the airfoil is shown in Fig.2.19 [1].

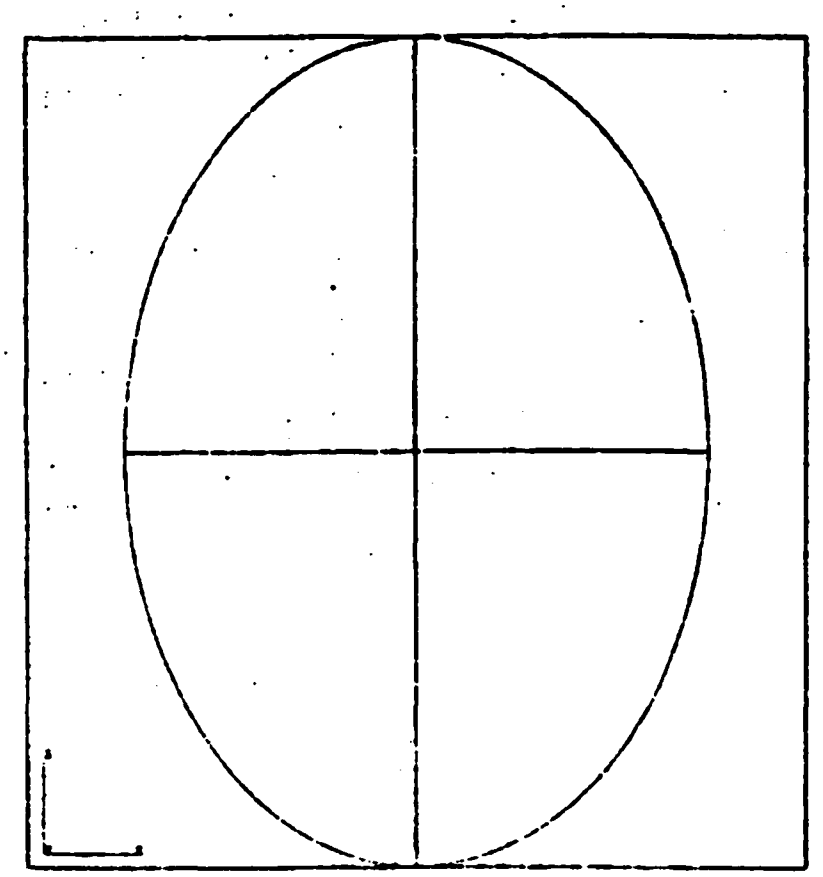


Fig.2.13 Region about NACA0012 airfoil subdived into four regions.

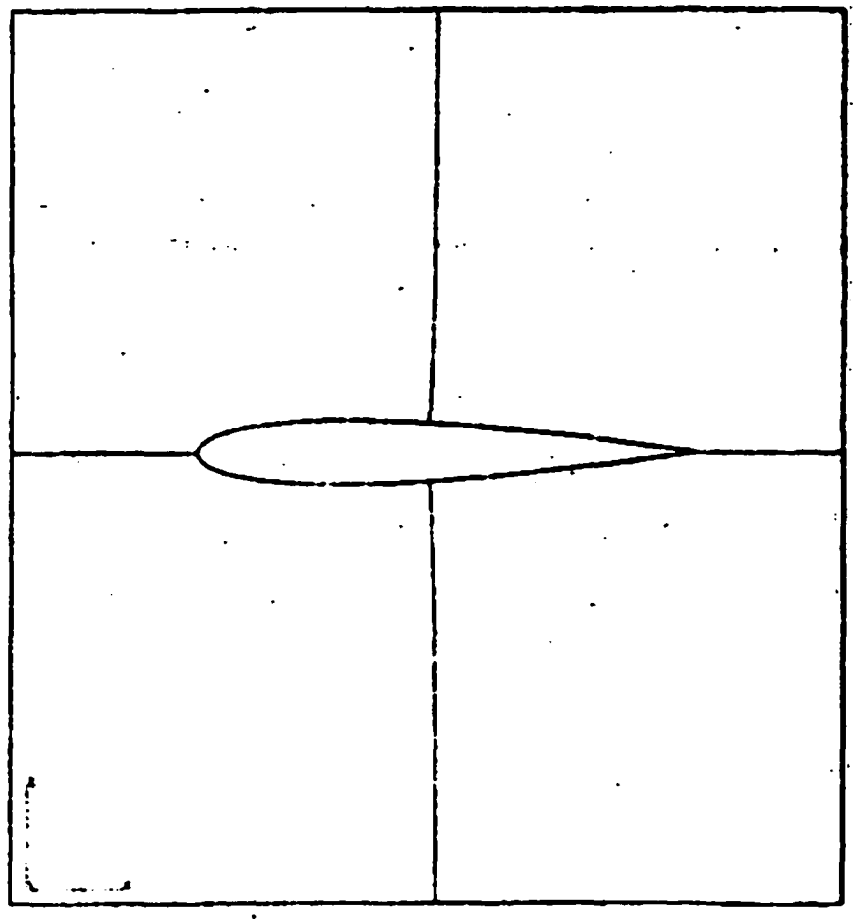


Fig.2.14 Domain boundaries near NACA0012 airfoil.

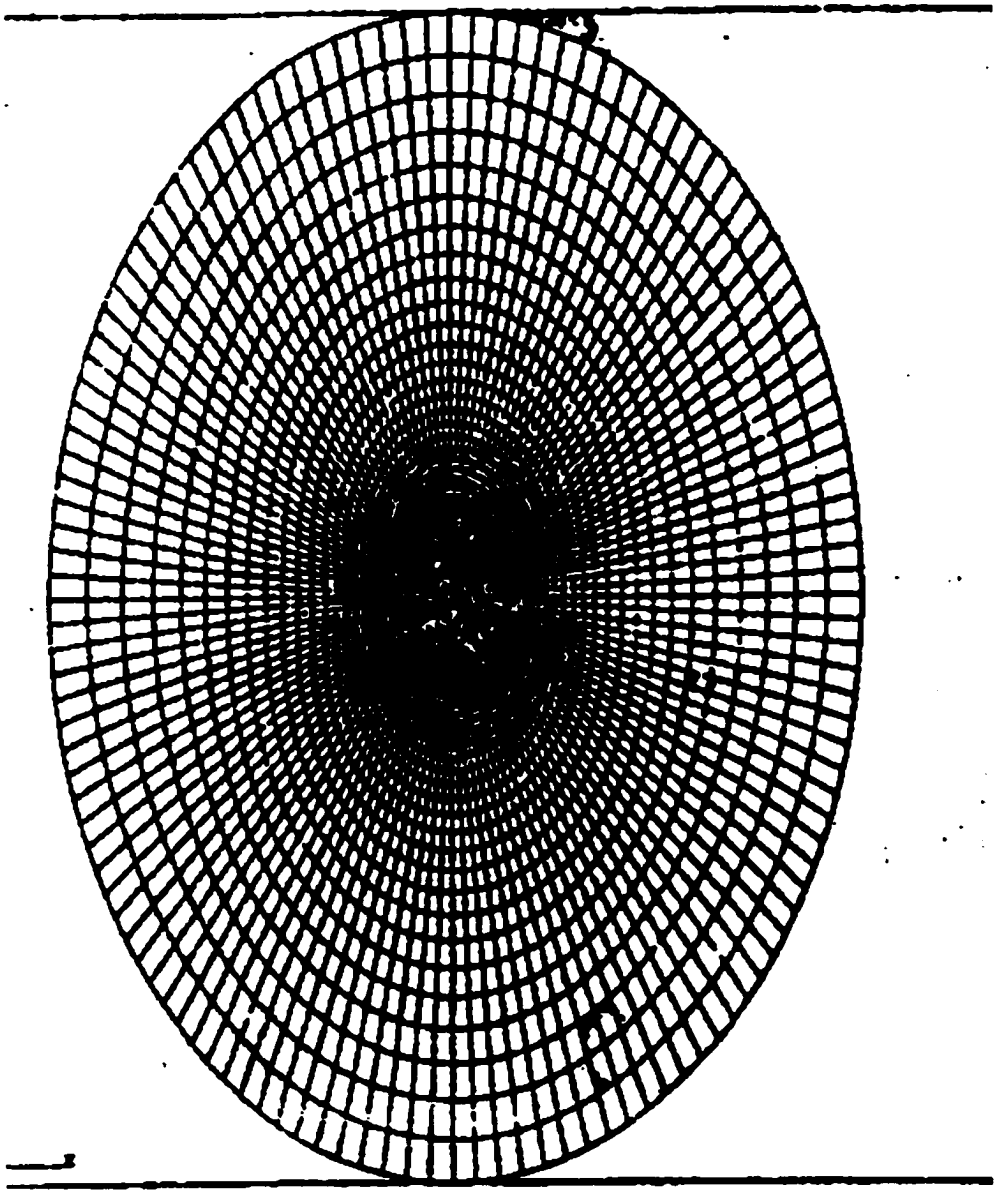


Fig.2.15 Complete O-type Euler grid.

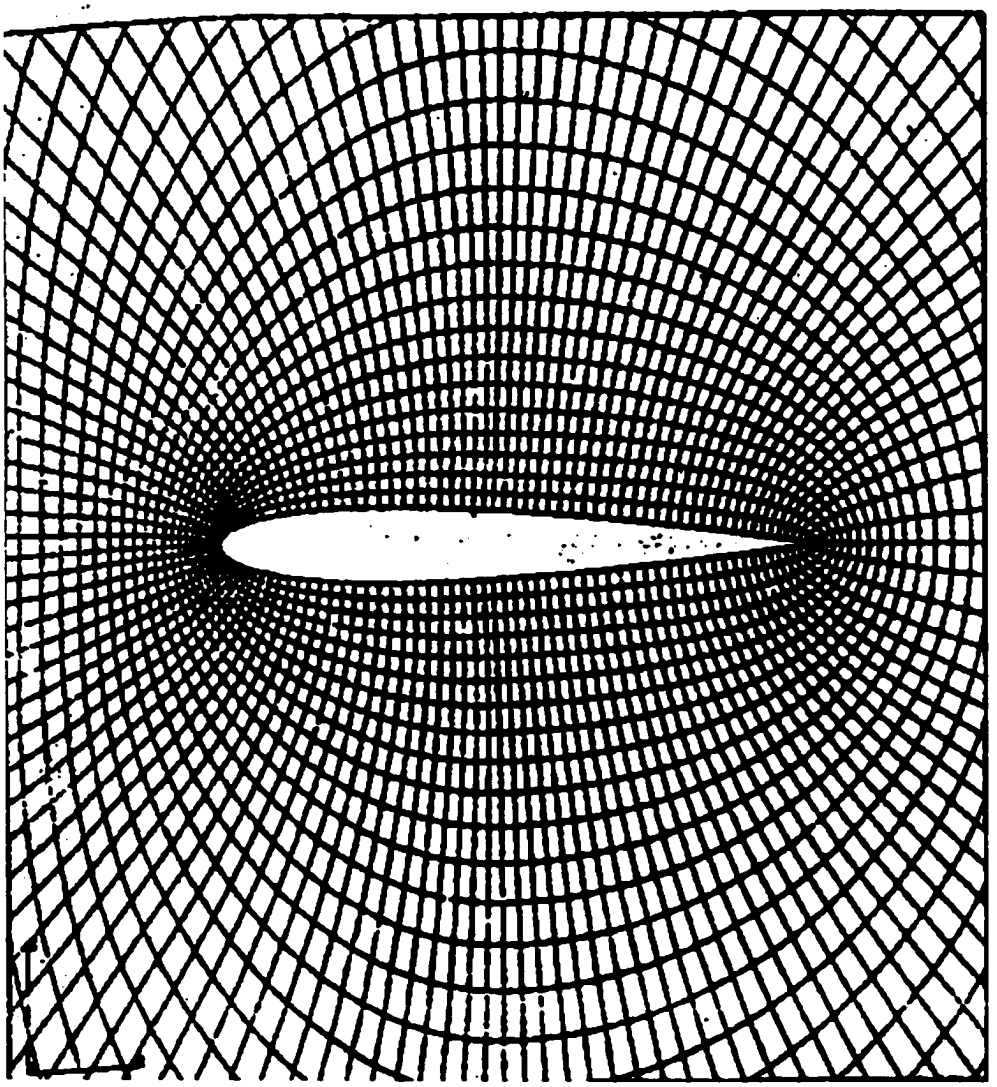


Fig.2.16 Grid near NACA0012 airfoil.

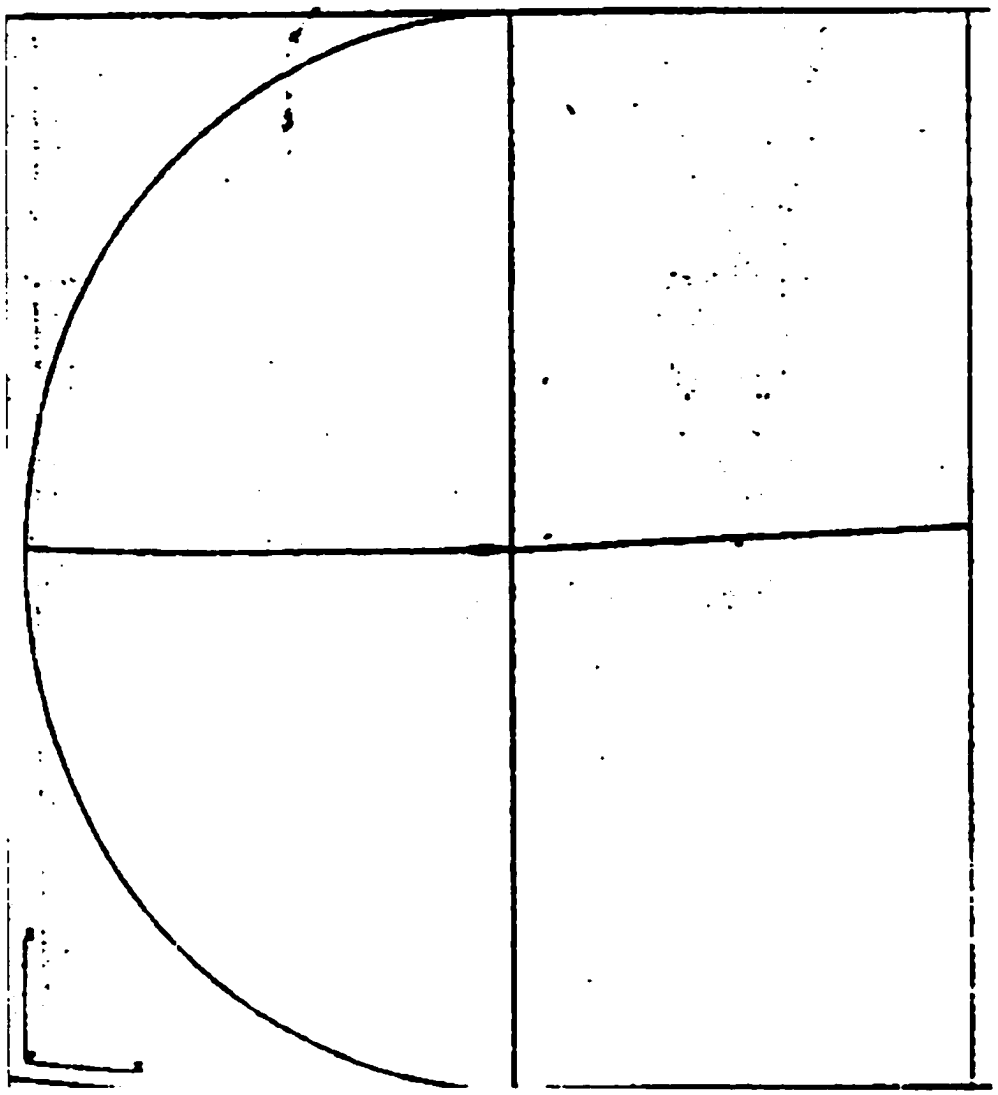


Fig.2.17 Region about RAE2822 airfoil subdivided into four regions.

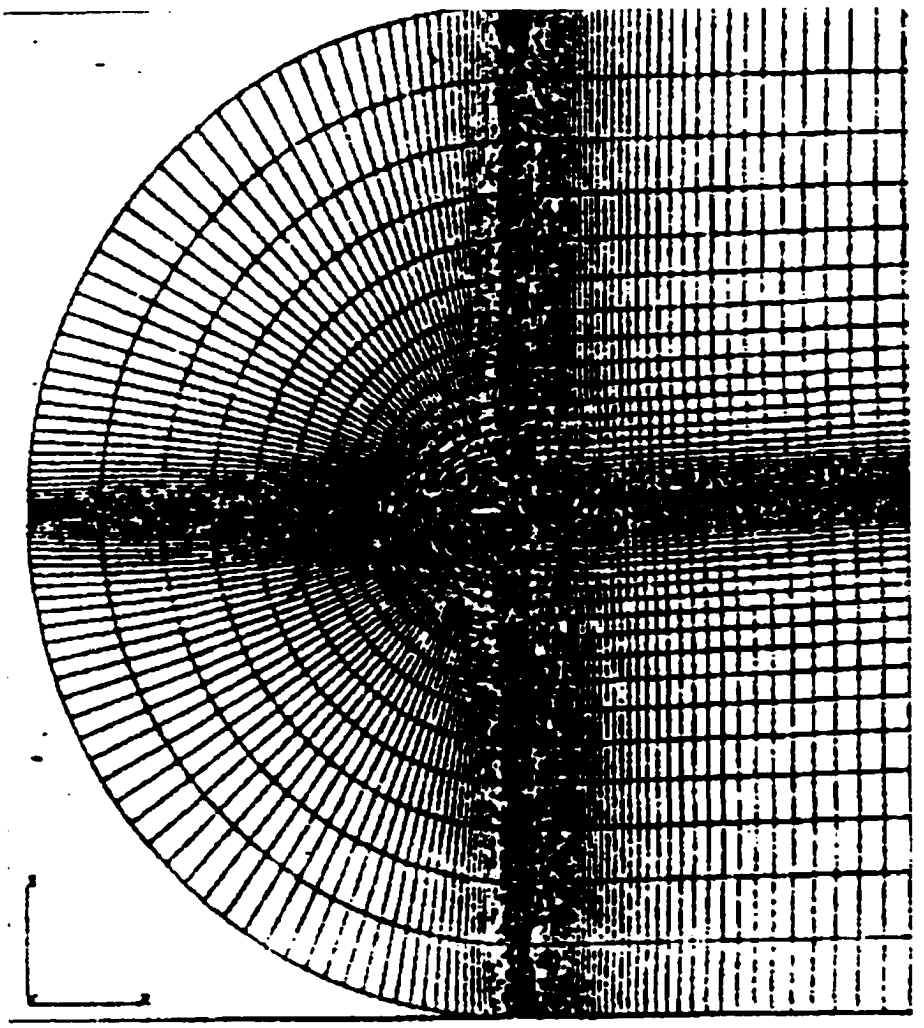


Fig.2.18 Complete C-type Navier-Stokes grid.

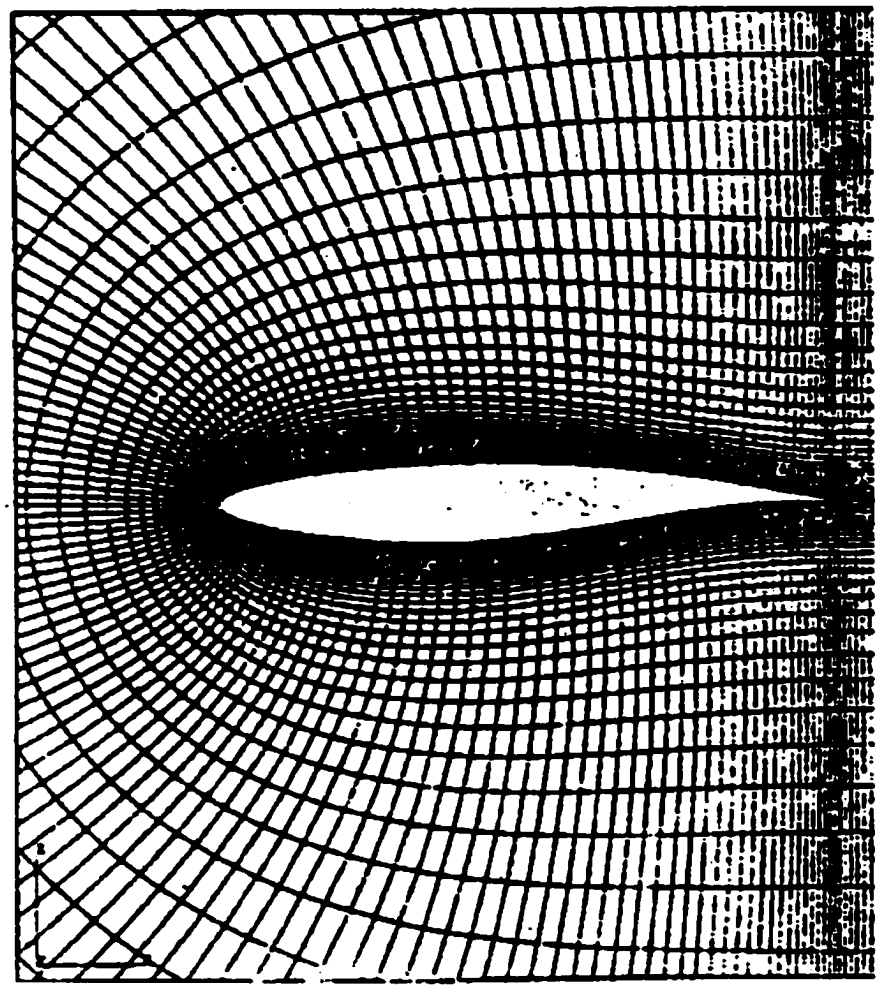


Fig.2.19 Grid near RAE2822 airfoil.

CHAPTER 3

DISCRETIZATION USING COMPACT SCHEMES

Compact finite difference schemes for the evaluation of the first and the second derivatives are presented in [4]. These schemes provide an improved representation of the range of scales (spectral like resolution). The schemes may be used on non-uniform meshes, and a variety of boundary conditions may be imposed. Many applications involve computations in domains with non-periodic boundaries. Approximations for the first and the second derivatives for the near boundary nodes are provided in [4]. These approximations are by nature non-central or one-sided.

3.1 Approximation of First Derivative

Given the values of a function on a set of nodes the finite difference approximation to the derivative of the function is expressed as a linear combination of the given function values. Consider a uniformly spaced mesh where the nodes are indexed by i. The independent variables at the nodes are $x_i = h(i-1)$ for $1 \le i \le N$ and the function values at the nodes $f_i = f(x_i)$ are given. The finite difference approximation f_i' to the first derivative $(df/dx)(x_i)$ at the node i depends on the function values at nodes near i. For second- and fourth-order central differences the approximation f_i' depends on the

sets (f_{i-1}, f_{i+1}) and $(f_{i-2}, f_{i-1}, f_{i+1}, f_{i+2})$, respectively. In the spectral methods, the value of f_i depends on all the nodal values. The Pade or compact difference schemes mimic this global dependence. The schemes presented are generalizations of the Pade scheme.

They are derived from approximations of the form:

$$\beta f_{i-2} + \alpha f_{i-1} + f_i + \alpha f_{i+1} + \beta f_{i+2} = c\frac{f_{i+3} - f_{i-3}}{6h} + b\frac{f_{i+2} - f_{i-2}}{4h} + a\frac{f_{i+1} - f_{i-1}}{2h} \qquad (3.1.1)$$

The relations between the coefficients a, b, c and α, β are derived by matching the Taylor series coefficients of various orders. The first unmatched coefficient determines the formal truncation error of the approximation (3.1.1). The constraints are:

$$a + b + c = 1 + 2\alpha + 2\beta \qquad \text{(second order)} \qquad (3.1.2)$$

$$a + 2^2 b + 3^2 c = 2\frac{3!}{2!}\left(\alpha + 2^2\beta\right) \qquad \text{(fourth order)} \qquad (3.1.3)$$

$$a + 2^4 b + 3^4 c = 2\frac{5!}{4!}\left(\alpha + 2^4\beta\right) \qquad \text{(sixth order)} \qquad (3.1.4)$$

$$a + 2^6 b + 3^6 c = 2\frac{7!}{6!}\left(\alpha + 2^6\beta\right) \qquad \text{(eighth order)} \qquad (3.1.5)$$

$$a + 2^8 b + 3^8 c = 2\frac{9!}{8!}\left(\alpha + 2^8\beta\right) \qquad \text{(tenth order)} \qquad (3.1.6)$$

If the dependent variables are periodic in x, then the systems of Equations (3.1.1) written for each node can be solved together as a linear system of equations for the unknown derivative values. This linear system is a cyclic pentadiagonal (tridiagonal) when β is nonzero (zero). The general non-periodic case requires additional relations appropriate for the near boundary nodes. These are described in Section 3.3.

The relation (3.1.1) along with a mathematically defined mapping between a non-uniform physical mesh and a uniform computational mesh, provides derivatives on a non-uniform mesh. It is also possible to derive relations analogous to (3.1.1) for a non-uniform mesh directly. We will now consider various spectral cases of (3.1.1). At least the first two constraints are imposed. Thus the schemes described have at least fourth order formal accuracy.

The general relation (3.1.1) with (3.1.2), (3.1.3) can be regarded as a three-parameter family of fourth-order schemes. If these schemes are restricted to $\beta = 0$ a variety of tridiagonal systems are obtained. For $\beta \neq 0$ pentadiagonal schemes are generated. If the additional constraint of sixth-order formal accuracy is imposed, a two-parameter family of sixth-order pentadiagonal schemes is obtained. These may be further specialized into a one-parameter family of eighth-order pentadiagonal schemes or a single tenth-order scheme.

We will first consider the tridiagonal schemes. These are generated by $\beta = 0$. If a further choice of $c = 0$ is made, a one-parameter (α) family of fourth-order tridiagonal schemes is obtained. For these schemes

$$\beta = 0, \quad a = \frac{2}{3}(\alpha + 2), \quad b = \frac{1}{3}(4\alpha - 1), \quad c = 0. \qquad (3.1.7)$$

The truncation error on the r.h.s of (3.1.1) for this scheme and for the other schemes to be described are listed in Table 1. The stencil sizes indicated in the table are the maximum stencil sizes needed within a class of schemes.

TABLE I

Truncation Error for the First Derivative Schemes

Scheme	Max. l.h.s. stencil size	Max. r.h.s. stencil size	Truncation error in (2.1)
(2.1.6)	3	5	$\frac{4}{5!}(3\alpha-1)\,h^4f^{(5)}$
(2.1.7)	3	5	$\frac{4}{7!}h^6f^{(7)}$
(2.1.8)	3	7	$\frac{12}{7!}(-8\alpha+3)\,h^6f^{(7)}$
(2.1.8) & $\alpha=\frac{3}{8}$	3	7	$\frac{-36}{9!}h^8f^{(9)}$
(2.1.9)	5	7	$\frac{4}{5}(-1+3\alpha-12\beta+10c)\,h^4f^{(5)}$
(2.1.10)	5	7	$\frac{12}{7!}(3-8\alpha+20\beta)\,h^6f^{(7)}$
(2.1.11)	5	5	$\frac{4}{7!}(9\alpha-4)\,h^6f^{(7)}$
(2.1.12)	5	5	$-\frac{16}{9!}h^8f^{(9)}$
(2.1.13)	5	7	$\frac{144}{9!}(2\alpha-1)\,h^8f^{(9)}$
(2.1.14)	5	7	$\frac{144}{11!}h^{10}f^{(11)}$

As α 0 this family merges into the well-known fourth-order central difference scheme. Similarly for $\alpha = \frac{1}{4}$ the classical Pade scheme is recovered. Furthermore, for $\alpha = \frac{1}{3}$ the leading order truncation error coefficient vanishes and the scheme is formally sixth-order accurate. Its coefficients are

$$\alpha = \frac{1}{3}, \quad \beta = 0, \quad a = \frac{14}{9}, \quad b = \frac{1}{9}, \quad c = 0. \qquad (3.1.8)$$

With $\beta = 0$ and c 0 the family of schemes (3.1.7) is extended to a two parameter family of fourth-order tridiagonal schemes. Contained within these is a one-parameter family of sixth-order tridiagonal schemes. For the sixth-order family

$$\beta = 0, \quad a = \frac{1}{6}(\alpha + 9), \quad b = \frac{1}{15}(32\alpha - 9), \quad c = \frac{1}{10}(-3\alpha + 1). \qquad (3.1.9)$$

The sixth-order tridiagonal scheme (3.1.8) is a member of this family (with $c = 0$, $\alpha = \frac{1}{3}$). This sixth-order family can be further specialized into an eighth-order scheme by choosing $\alpha = \frac{3}{8}$. This tridiagonal scheme ($\beta = 0$) with has the highest formal accuracy within (3.1.1).

Pentadiagonal schemes are generated with $\beta = 0$. In general this fourth-order three-parameter (α.β and c) family is given by

$$a = \frac{1}{3}(4 + 2\alpha - 16\beta + 5c)$$

$$b = \frac{1}{3}(-1 + 4\alpha + 22\beta - 8c) \qquad (3.1.10)$$

Schemes of sixth-order formal accuracy contain two parameters α and β. They are given by

$$a=\frac{1}{6}(9+\alpha-20\beta)$$

$$b=\frac{1}{15}(-9+32\alpha+62\beta)$$

$$b=\frac{1}{10}(1-3\alpha+12\beta) \qquad (3.1.11)$$

The triadiagonal sixth-order family of (3.1.9) is a subclass within (3.1.11). Another subclass is obtained with β 0 and $c=0$. This sixth-order pentadiagonal family has

$$\beta=\frac{1}{12}(-1+3\alpha),$$

$$a=\frac{2}{9}(8-3\alpha),$$

$$b=\frac{1}{18}(-17+57\alpha),$$

$$c=0 \qquad (3.1.12)$$

This family limits to the sixth-order tridiagonal scheme (3.1.8) as β 0 or $\alpha=\frac{1}{3}$. The leading truncation error coefficient for (3.1.12) vanishes for $\alpha=\frac{4}{9}$ yielding an eighth-order scheme. This eighth-order scheme has

$$\alpha=\frac{4}{9},\ \beta=\frac{1}{36},\ a=\frac{40}{27},\ b=\frac{25}{54},\ c=0. \qquad (3.1.13)$$

By choosing $\beta = \frac{1}{20}(-3+8\alpha)$ in (3.1.11) a one-parameter family of eighth-order pentadiagonal schemes is generated.

The eighth-order family has

$$\beta = \frac{1}{20}(-3+8\alpha)$$

$$a = \frac{1}{6}(12-7\alpha)$$

$$b = \frac{1}{150}(568\alpha - 183)$$

$$c = \frac{1}{50}(9\alpha - 4) \qquad (3.1.14)$$

The specific eighth-order schemes, (a) scheme (3.1.9) with $\alpha = \frac{1}{2}$ and (b) scheme (3.1.13), belong to this one-parameter family.

By choosing $\alpha = \frac{1}{2}$ in (3.1.14) a tenth-order scheme is generated. This is the scheme with the highest formal accuracy amongst the schemes defined by (3.1.1). The coefficients of this scheme are

$$\alpha = \frac{1}{2},\ \beta = \frac{1}{20},\ a = \frac{17}{12},\ b = \frac{101}{150},\ c = \frac{1}{100}. \qquad (3.1.15)$$

Among the class of derivative approximations represented by (3.1.1) those which achieve the highest possible formal accuracy within each subclass of schemes (denoted by a specified computational stencil on both the l.h.s and the r.h.s of (3.1.1) are precisely the schemes obtained by a rational (or Pade) approximation of the first derivative operator.

3.2 Approximation of the Second Derivative

The derivation of compact approximations for the second derivative is analogous to the first derivative. We start with relations of the form

$$\beta f_{i-2} + \alpha f_{i-1} + f_i + \alpha f_{i+1} + \beta f_{i+2} = c\frac{f_{i+3} - 2f_i + f_{i-3}}{9h^2} + b\frac{f_{i+2} - 2f_i + f_{i-2}}{4h^2} + a\frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} \tag{3.2.1}$$

where f_i represents the finite difference approximation to the second derivative at the nose i. The relations between the coefficients a,b,c and α,β are derived by matching the Taylor series coefficients of various orders. The first unmatched coefficient determines the formal truncation error of the approximation (3.2.1). The constraints are:

$$a + b + c = 1 + 2\alpha + 2\beta \qquad \text{(second order)} \qquad (3.2.2)$$

$$a + 2^2 b + 3^2 c = \frac{4!}{2!}\left(\alpha + 2^2\beta\right) \qquad \text{(fourth order)} \qquad (3.2.3)$$

$$a + 2^4 b + 3^4 c = \frac{6!}{4!}\left(\alpha + 2^4\beta\right) \qquad \text{(sixth order)} \qquad (3.2.4)$$

$$a + 2^6 b + 3^6 c = 2\frac{8!}{6!}\left(\alpha + 2^6\beta\right) \qquad \text{(eighth order)} \qquad (3.2.5)$$

$$a + 2^8 b + 3^8 c = 2\frac{10!}{8!}\left(\alpha + 2^8\beta\right) \qquad \text{(tenth order)} \qquad (3.2.6)$$

The form of these constraints is very close to that derived for the first derivate approximations but the multiplying factors on the r.h.s are different. At least the first two of these constraints are imposed resulting in schemes with al least fourth-order accuracy. For the dependent variables periodic in x the tridiagonal or pentadiagonal system defined

by (3.2.1) at each node may be solved to yield the second derivatives. For the non-periodic case additional relations are required at the boundary.

By choosing $\beta = 0$ and $c = 0$ a one-parameter family of fourth-order schemes is generated. This family has

$$\beta = 0,\ c = 0,\ a = \frac{4}{3}(1-\alpha),\ b = \frac{1}{3}(-1+10\alpha) \qquad (3.2.7)$$

The truncation error on the r.h.s of (3.2.1) for this and the other schemes described in this section are listed in Table 2. It may be noted that as α 0 this family coincided with the well-known fourth-order central difference scheme. For $\alpha = \frac{1}{10}$ the classical Pade scheme is recovered. For $\alpha = \frac{2}{11}$ a sixth-order tridiagonal scheme is obtained. This scheme has

$$\alpha = \frac{2}{11},\ \beta = 0,\ a = \frac{12}{11},\ b = \frac{3}{11},\ c = 0 \qquad (3.2.8)$$

A three-parameter family of fourth-order schemes is generated from (3.2.1) by considering β 0 and c 0. These are given by

$$a = \frac{1}{3}(4 - 4\alpha - 40\beta + 5c)$$

$$b = \frac{1}{3}(-1 + 10\alpha + 46\beta - 8c) \qquad (3.2.9)$$

This class of schemes can be further classified into a two-parameter family of sixth-order schemes, a one-parameter family of eighth-order schemes or a single tenth-order scheme.

TABLE II

Truncation Error for Second Derivative Schemes

Scheme	Max. l.h.s. stencil size	Max. r.h.s. stencil size	Truncation error in (2.2)
(2.2.6)	3	5	$\frac{-4}{6!}(11\alpha-2)\,h^4f^{(6)}$
(2.2.7)	3	5	$\frac{-8\cdot 23}{11\cdot 8!}h^6f^{(8)}$
(2.2.8)	5	7	$\frac{-4}{6!}(-2+11\alpha-124\beta+20c)\,h^4f^{(6)}$
(2.2.9)	5	7	$\frac{-8}{8!}(9-38\alpha+214\beta)\,h^6f^{(8)}$
(2.2.10)	5	7	$\frac{899\alpha-334}{2696400}h^8f^{(10)}$
(2.2.11)	5	7	$\frac{619}{299043360}h^{10}f^{(12)}$

The two-parameter sixth-order family is defined by

$$a=\frac{6-9\alpha-12\beta}{4}$$

$$b=\frac{-3+24\alpha-6\beta}{5}$$

$$c=\frac{2-11\alpha+124\beta}{20} \qquad (3.2.10)$$

When the eighth-order constraint is imposed (3.2.10) reduces to a one-parameter family of eighth-order schemes. They are defined by

$$\beta = \frac{38\alpha - 9}{214},$$

$$a = \frac{696 - 1191\alpha}{428}$$

$$b = \frac{2454\alpha - 294}{535}$$

$$c = \frac{1179\alpha - 344}{2140} \qquad (3.2.11)$$

When the tenth-order constraint is imposed a single tenth-order scheme is obtained. This scheme is defined as

$$\beta = \frac{43}{1798},\ \alpha = \frac{334}{899},\ a = \frac{1065}{1798},\ b = \frac{1038}{899},\ c = \frac{79}{1798} \qquad (3.2.12)$$

has the highest formal accuracy within the class of schemes defined by (3.2.1).

Among the schemes defined by (3.2.1), those which maximize the formal accuracy correspond precisely to the rational or Pade approximation of the second derivative operator.

An alternate and more effective way of classifying the schemes presented here is provided by their Fourier analysis. It provides a way to "optimize" the scheme from a multi-parameter family. A Fourier analysis of the errors associated with the approximations using compact schemes is presented in [4]. Comparisons are made with the standard finite-difference schemes to note the improvement in the error

characteristics. Fourier analysis of the standard Pade scheme was presented in [9] and comparisons were made with the second and the fourth-order central difference schemes. Fourier analysis provides an effective way of quantifying the resolution characteristics of the differencing approximation schemes. The quantification may be used as a further guide to optimization of the differencing schemes. The analysis in [4] brings out the spectral-like resolution of the compact schemes. The following conclusions can be drawn from the analysis:

Compared with the standard second and the fourth order central difference schemes, the compact schemes are close to the exact differentiation over a wider range of wavenumbers associated with the corresponding Fourier modes. The sixth-order compact scheme is better than the Pade scheme. Similarly the eighth-order schemes and the tenth-order schemes stay close to the exact differentiation over a progressively large wavenumber range.

3.3 Non-Periodic Boundaries

Many applications involve computations in domains with non-periodic boundaries. This section details approximations for the first and second derivatives for the near boundary nodes. These approximations are, of necessity, non-central or one-sided. In developing these boundary formulations emphasis has been placed on maintaining a discrete form of global conservation. For an analysis of the local error inherent in the approximation refer to [4]. It is supplemented by numerical estimates of the global error. Typically the global error is dominated by the boundary error.

3.3.1 Boundary Formulation for the First Derivative.

The first derivative at the boundary $i = 1$ may be obtained by a relation of the form

$$f_1' + \alpha f_2' = \frac{1}{h}\left(a f_1 + b f_2 + c f_3 + d f_4\right), \qquad (3.3.1.1)$$

coupled to the relations (3.1.1) written for the interior nodes. With this choice the boundary schemes can be used with a tridiagonal interior scheme without increasing the bandwidth. Requiring (3.3.1.1) to be at least second-order accurate constraints the coefficients to

$$a = -\frac{3+\alpha+2d}{2},\ b = 2+3d,\ c = -\frac{1-\alpha+6d}{2}. \qquad (3.3.1.2)$$

If higher order formal accuracy is desired schemes of third and fourth order may be derived. These are given by

$$a = -\frac{11+2\alpha}{6},\ b = \frac{6-\alpha}{2},$$

$$c = \frac{2\alpha-3}{2},\ d = \frac{2-\alpha}{6},\quad \text{(third order)} \qquad (3.3.1.3)$$

$$\alpha = 3,\ a = -\frac{17}{6},\ b = \frac{3}{2},\ c = \frac{3}{2},\ d = -\frac{1}{6} \quad \text{(fourth order)} \qquad (3.3.1.4)$$

The leading order of truncation error (on the r.h.s of (3.3.1.1)) for these boundary approximations are given by $((2-\alpha-6d)/3!)h^2 f_1^{(3)}$ for the second-order schemes, by $(2(\alpha-3)/4!)h^3 f_1^{(4)}$ for third-order schemes and by $(6/5!)h^4 f_1^{(5)}$ for the fourth-order scheme. It may be noted that for the even order schemes the leading order truncation error is of dispersive type, while for the third-order schemes it is dissipative.

3.3.2 Boundary Formulation for the Second Derivative.

The relations appropriate for the near boundary nodes between the nodal values of a function and its second derivative may be derived by Taylor series expansions. The compact scheme analogous to (3.3.1.1) is given by

$$f_1'' + 11 f_2'' = \frac{1}{h^2}(13f_1 - 27f_2 + 15f_3 - f_4) \qquad (3.3.2.1)$$

this relation is formally third-order accurate (truncation error on the r.h.s is $h^3 \frac{1}{12} f^{(5)}$). The explicit expressions (at boundary node) with second- and third-order formal accuracy are given by

$$f_1'' = \frac{1}{h^2}(2f_1 - 5f_2 + 4f_3 - f_4) \quad \text{(second order)} \qquad (3.3.2.2)$$

$$f_1'' = \frac{1}{h^2}\left(\frac{35}{12}f_1 - \frac{26}{3}f_2 + \frac{19}{2}f_3 - \frac{14}{3}f_4 + \frac{11}{12}f_5\right) \quad \text{(third)} \qquad (3.3.2.3)$$

Their truncation errors are $\frac{11}{12}h^2 f^{(4)}$ and $\frac{5}{6}h^3 f^{(5)}$. The truncation error of the explicit third-order form is 10 times larger than that of the third-order compact form.

Boundary schemes for the other neighboring nodes may be chosen from (3.2.1). It is possible to extend the global conservation consideration to the second derivatives. For these purposes it becomes necessary to introduce more general compact boundary schemes

$$f_1'' + \alpha f_2'' = \frac{1}{h^2}(af_1 + bf_2 + cf_3 + df_4 + ef_5). \qquad (3.3.2.4)$$

Requiring second order accuracy restricts the coefficients to

$$a = \alpha + 2 + e\ ,\ b = -(2\alpha + 5 + 4e)$$

$$c = \alpha + 4 + 6e,\ d = -(1 + 4e) \qquad (3.3.2.5)$$

The formal truncation error is $\frac{1}{12}(\alpha + 12e - 11)h^2 f^{(4)}$. By requiring third-order formal accuracy the coefficients are reduced to

$$a = \frac{11\alpha + 35}{12},\ b = -\frac{5\alpha + 26}{3},\ c = \frac{\alpha + 19}{2}$$

$$d = \frac{\alpha - 14}{3},\ e = \frac{11 - \alpha}{12} \qquad (3.3.2.6)$$

The formal truncation error is reduced to $((\alpha - 10)/12)h^3 f^{(5)}$.

3.4 Numerical Tests

The global errors may be assessed by direct numerical tests. Such tests can be performed at various levels. The simplest of these is to compare the numerically calculated derivative with the known derivative of various test functions. To test the schemes on functions that contain a range of scales, such functions were numerically synthesized by taking a sum of different Fourier modes (representable on a mesh). The phases of the Fourier modes were chose randomly and their amplitudes were chosen to synthesize a prescribed energy spectrum. For the examples presented here the interval [0,1] was discretized into 128 intervals (i.e., 129 points when counting both the end points). On this mesh Fourier modes [0,63] (for the wavenumber k may be represented). The mode k=64 is not included. In the examples to be discussed Fourier modes with k in the range [0, k_m] were included. The amplitudes of the Fourier were equal and phases were random. For the numerical tests the conservative formulation of the first derivative scheme with $\alpha = 1/3$, $\beta = 0 = c$ (3.1.8) were used. At the end points the third-order compact boundary scheme with $d = 0$ was used. This overall scheme is chosen as it has been used in several practical applications. Fig.3.1 shows test functions containing Fourier modes up to wavenumber k_m (varying from 9,21,31 and 63) with equal amplitude and random phases. One randomly chosen realization of the test functions is displayed for each class in Fig.3.1 (a). In Fig.3.1 (b) the numerically computed first derivatives (shown with a dashed line) and the exact derivative of the test functions (shown with a solid line) are plotted. It may be noted that the dominant error for the derivatives occur at the

boundary. Only for the cases with k_m of 31 and 63 this localized boundary error is visible in the plots, and even in these cases the interior is virtually error-free.

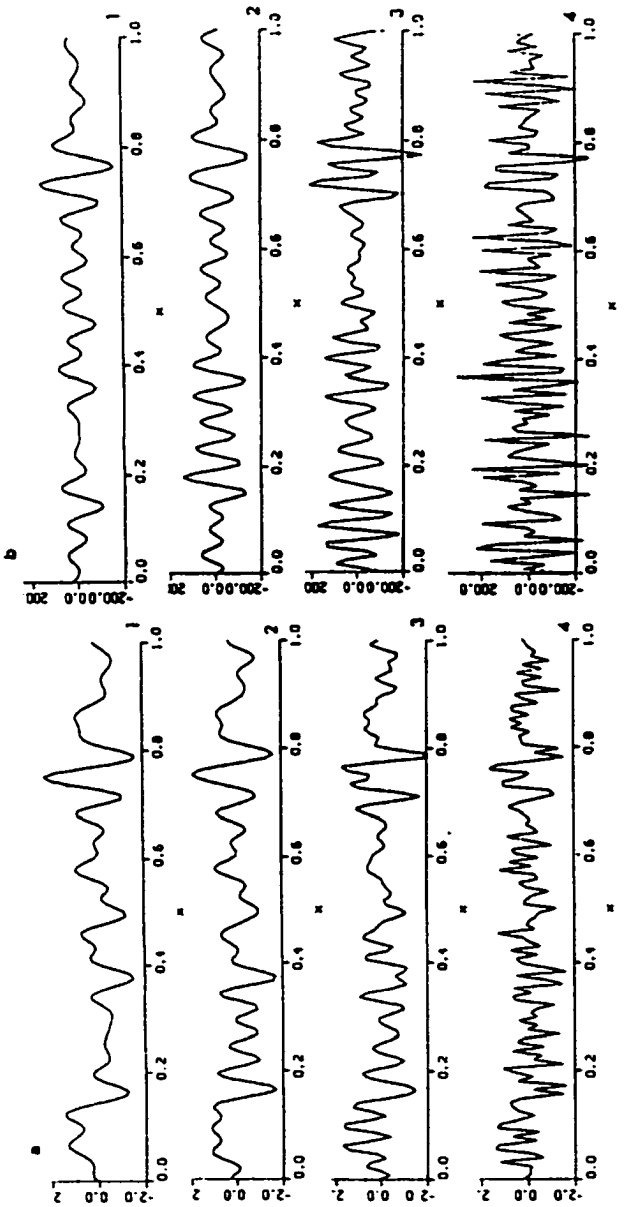


Fig.3.1 Numerical tests on the derivative scheme; the domain is divided into 128 intervals. The test functions contain Fourier modes up to wave-number k_m with equal amplitude and random phases. One realization is plotted for each case with (1) $k_m = 9$; (2) $k_m = 21$; (3) $k_m = 31$; (4) $k_m = 63$; on (a). The corresponding exact derivative (solid line) and the calculated derivative (dashed line) are shown on (b).

The differencing schemes described in [4] provide an improved resolution of the short length scales. These schemes have a pure central difference form (except near the boundaries). Higher-order compact schemes can be used to increase the accuracy of the solution. An example of a grid in 2D domain is shown in Figs.3.2 and 3.3. The figures show the region around an airfoil. The grid in Fig.3.2 is generated using fourth-order standard finite-difference schemes for the discretization of the non-linear elliptical Equations given by (3.2.1).

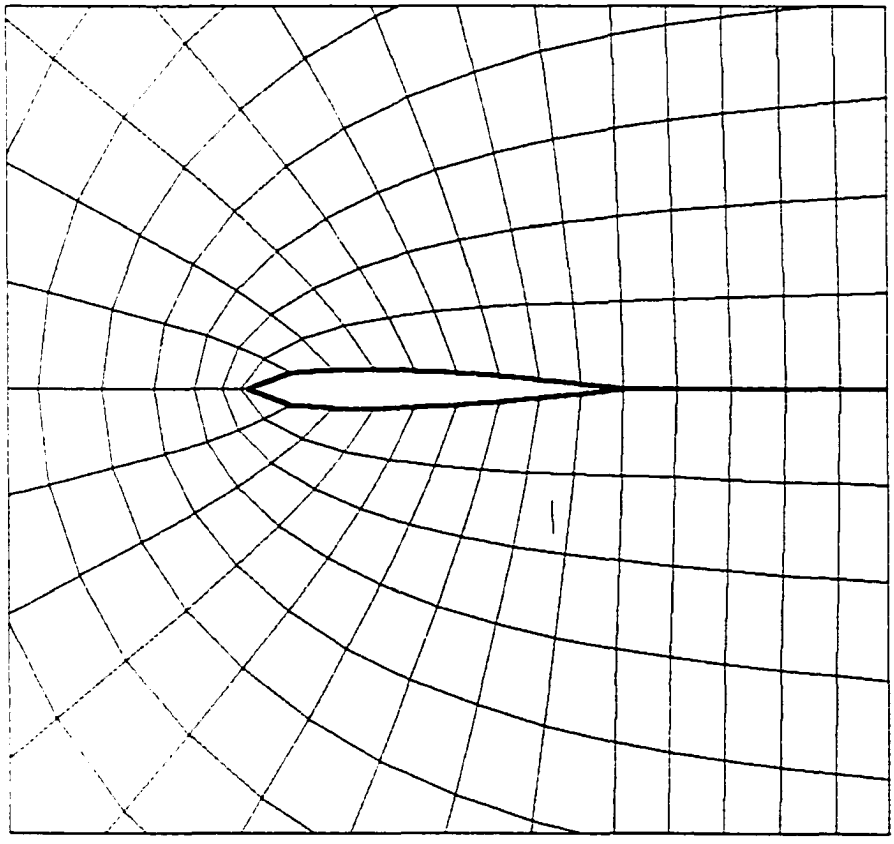


Fig.3.2. Grid generated using fourth-order standard finite difference scheme.

In comparison, Fig.3.3 shows a grid generated using fourth-order compact scheme. A careful look at the two grids will indicate that the grid generated by using the

fourth-order compact scheme has a better grid point distribution than the one generated using fourth-order traditional scheme.

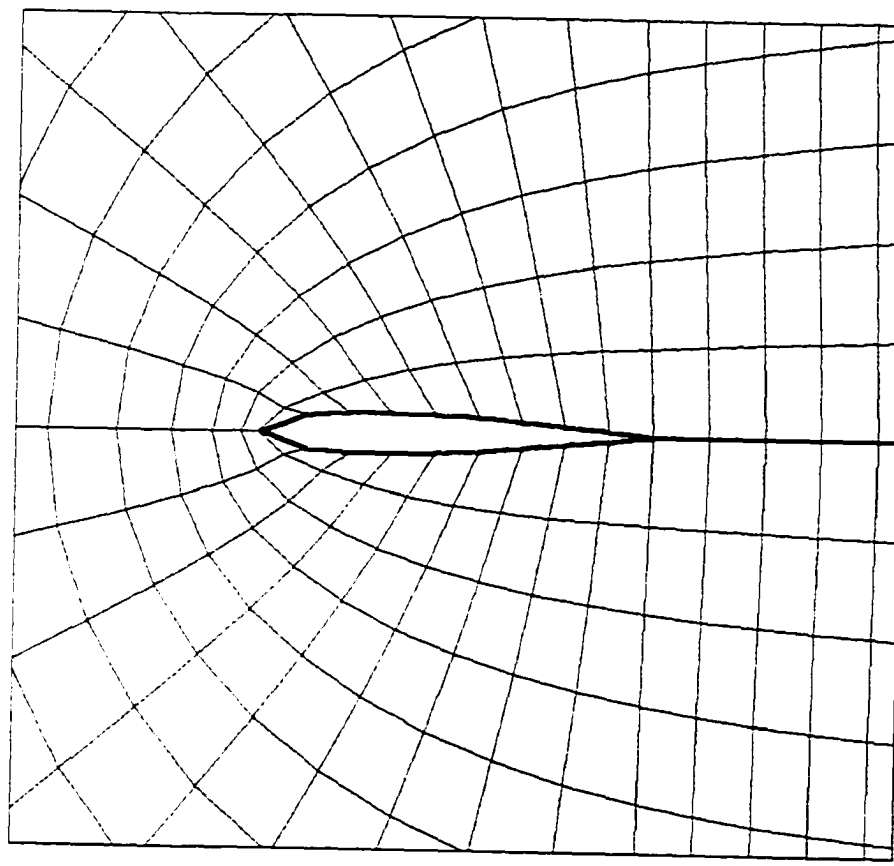


Fig.3.3. Grid generated using fourth-order compact finite difference schemes.

CHAPTER 4

MULTIGRID SOLUTION METHOD FOR ACCELERATING CONVERGENCE

4.1 Model Problems

Multigrid methods were originally applied to simple boundary value problems that arise in many physical applications. These problems provide a natural introduction to multigrid methods. As an example, consider the following two-point boundary value problem that could describe the steady-state temperature distribution in a long uniform rod. It is given by the second-order ordinary differential equation.

$$-u_{xx}(x)+\sigma u(x)=f(x), \qquad 0<x<1, \quad \sigma \geq 0 \qquad (4.1.1)$$

Subject to the boundary conditions $u(0)=u(1)=0$.

Although we can handle this problem analytically, it is a common practice to develop numerical methods for its solution. Many such approaches are possible. The simplest is the finite difference method. The domain of the problem is partitioned into N subintervals by introducing the grid points $x_j = jh$, where $h = 1/N$ is the constant width of the subintervals. This gives the grid shown in Fig.4.1 that we denote Ω^h.

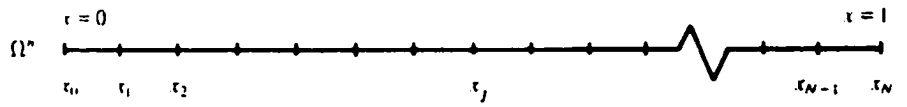


Fig.4.1 A one-dimensional grid on the interval $0 \le x \le 1$. The grid spacing is given by $h = 1/N$. The *jth* grid point is $x_j = jh$ for $0 \le j \le N$.

At each of the *N-1* interior grid points, the original difference equation is replaced by a second-order finite difference approximation. In making this replacement, we also introduce as an approximation to the exact solution $u(x_j)$. This approximate solution may be represented by a vector $v = (v_1, \cdots, v_{N-1})^T$ whose components satisfy the N-1 linear equations

$$\frac{-v_{j-1} + 2v_j - v_{j+1}}{h^2} + \sigma v_j = f(x_j), \qquad 1 \le j \le N\text{-}1. \qquad (4.1.2)$$

$$v_0 = v_N = 0$$

Defining the vector of the right-hand side values $f = (f(x_1), \cdots, f(x_{N-1}))^T = (f_1, \cdots, f_{N-1})^T$, we may also represent this system of linear equations in matrix form as

$$\frac{1}{h^2}\begin{bmatrix} 2+\sigma h^2 & -1 & & & & \\ -1 & 2+\sigma h^2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & -1 \\ & & & & -1 & 2+\sigma h^2 \end{bmatrix}\begin{bmatrix} v_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ v_{N-1} \end{bmatrix} = \begin{bmatrix} f_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ f_{N-1} \end{bmatrix}$$

Or even more compactly as Av = f.

The matrix A is tridiagonal, symmetric positive definite and has dimension (N-1) X (N-1). Although the choice of N-1 unknowns corresponding to the N-1 interior grid points seems awkward at the moment, its usefulness will be apparent soon.

Analogously, it is possible to formulate a two-dimensional version of the problem. Consider the second-order partial differential equation, the Poisson's equation. It is given by

$$-u_{xx} - u_{yy} = f(x,y), \qquad 0 < x < 1, \quad 0 < y < 1. \qquad (4.1.3)$$

We will consider this equation subject to the condition that u =0 on the boundary of the unit square. This problem may be cast in a discrete form by defining the grid points $(x_i, y_j) = (ih_x, jh_y)$, where $h_x = 1/M$ and $h_y = 1/N$. This two-dimensional grid is also denoted Ω^h and is shown in Fig.4.2.

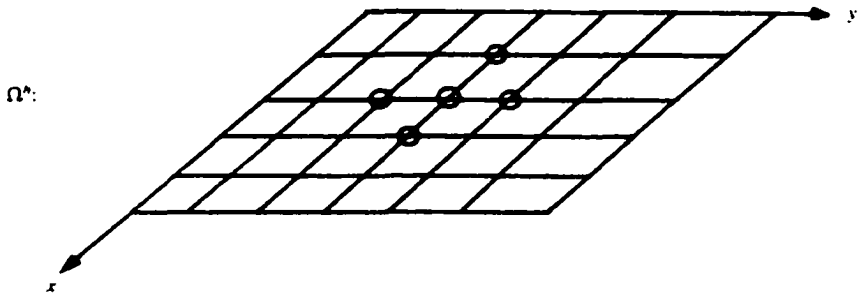


Fig.4.2 Two-dimensional grid on the unit square. The circled points show the unknowns that are related at a typical grid point by the discrete equations.

Replacing the derivates of by second-order finite differences leads to the system of linear equations

$$\frac{-v_{i-1,j} + 2v_{i,j} - v_{i+1,j}}{h_x^2} + \frac{-v_{i,j-1} + 2v_{i,j} - v_{i,j+1}}{h_y^2} = f_{i,j} \qquad (4.1.4)$$

$$v_{i,j} = 0 \quad \text{if } i = 0 \text{ or } i = \text{M or } j = 0 \text{ or } j = \text{N},$$

where $1 \le i \le$ M-1, $1 \le j \le$ N-1, $v_{i,j}$ is an approximation to the exact solution $u(x_i, y_j)$ and $f_{i,j} = f(x_i, y_j)$. There are now (M-1) X (N-1) interior grid points and the same number of unknowns in the problem. Many different ways to order the unknowns exist. Consider the lexicographic ordering by lines of constant i. The unknowns of the *ith* row of the grid may be collected in the vector $v_i = (v_{i1}, \cdots, v_{i,N-1})^T$ for $1 \le i \le$ M-1. Similarly, let $f_i = (f_{i1}, \cdots, f_{i,N-1})^T$. The system of Equations (4.1.4) may then be given in matrix form as

$$\begin{bmatrix} A & -I & & & & \\ -I & A & -I & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & -I \\ & & & & -I & A \end{bmatrix} \begin{bmatrix} v_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ v_{M-1} \end{bmatrix} = \begin{bmatrix} f_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ f_{M-1} \end{bmatrix}$$

This system is symmetric, block tridiagonal and sparse (the matrix entries are predominantly zero). It has block dimension (M-1) X (M-1). Each diagonal block is an (N-1) X (N-1) tridiagonal matrix and each off-diagonal block is a multiple of the (N-1) X (N-1) identity matrix I.

These two model linear systems (4.1.2) and (4.1.4) will be the basis for many of the methods discussed in the following sections. Let us explore the existing methods for solving such systems. The last forty years have produced a tremendous amount of work in devising numerical methods for solving sparse systems of linear equations. Much attention has been given to structured systems such as (4.1.2) and (4.1.4), which arise from boundary value problems. Existing methods of solution fall into two large categories: direct methods and iterative (or relaxation) methods.

Direct methods, of which Gaussian elimination is the prototype, determine a solution exactly (up to machine precision) in a finite number of arithmetic steps. For systems such as (4.1.4) which arise from a two-dimension elliptic problem, very efficiently direct methods have been developed. They are usually based on the fast Fourier transform or the method of cyclic reduction. When applied to problems on an NXN grid, these methods require $O(N^2 \log N)$ arithmetic operations. Therefore, since they approach the minimum operation count of $O(N^2)$ operations, these methods are nearly optimal. However, they are also specialized and can be applied primarily to systems that arise from separable self-adjoint boundary value problems.

Relaxation methods, as represented by the Jacobi and Gauss-Seidel iterations, begin with an initial guess at a solution. They then proceed to improve the current approximation by a succession of simple updating steps or iterations. The sequence of approximations which is generated (ideally) converges to the exact solution of the linear system. Classical relaxation methods are easy to implement and may be successfully applied to more general linear systems than the direct methods. However, these relaxation schemes suffer from some limitations. Multigrid methods evolved from attempts to correct these limitations. These attempts have largely been successful; used in a multigrid setting, relaxation methods are competitive with fast direct methods when applied to the model problems. In addition, the multigrid/relaxation methods have more generality and a wider range of application than the specialized direct methods. It is not possible to say whether direct or iterative methods are ultimately superior. It all depends upon the structure of the problem and the type of computer being used. In hybrid methods, direct and iterative methods are often used in tandem.

The basic multigrid methods have immediate extensions to boundary value problems with more general boundary conditions, operators and geometries. It is safe to state that these methods can be applied to any self-adjoint (symmetric positive definite) problem with no significant modification. However there is no guarantee that such applications will be efficient. The basic multigrid methods are certainly not confined to finite difference formulations. Infact, finite element and finite volume formulations are often more natural, particularly for the analysis of these methods. The basic multigrid methods themselves have been extended to deal with non-linear and time dependent problems. Considerable work has been done in applying them to both algebraic and differential eigenvalue problems.

The original multigrid ideas have been extended to what are more appropriately called multilevel methods. Purely algebraic problems (for example, network problems and structural problems) have led to the development of algebraic multigrid (AMG). Identifying the notion of a grid with the array of pixels has led to multigrid applications in image processing. Multilevel methods have found new applications in control theory, combinational optimization (the traveling salesman problem), statistical mechanics (the Island model) and quantum electrodynamics.

4.2 Basic Iterative Methods

4.2.1 The Residual Equation

Let Au = f denote a system of linear equations. We will use u to denote the exact solution and v to denote an approximation to the exact solution, which is generated by some iterative method. Bold symbols such as u and v, represent vectors, while the *jth* components of these vectors are denoted by u_j and v_j. Clearly the exact solution to the problem Au = f is unknown, while an approximation v, which has just been computed, is known. There are two important measures of v as an approximation to u. One is the error (or algebraic error) and is simply given by

$$e = u - v$$

The error is also a vector and its size may be measured by any of the standard vector norms. The most commonly used norms for this purpose are defined by

$$\|e\|_\infty = \max_{1 \le j \le N} |e_j| \quad \text{and} \quad \|e\|_2 = \left\{ \sum_{j=1}^{N} e_j^2 \right\}^{1/2}$$

Unfortunately, the error is just as inaccessible as the exact solution itself. However, a computable measure of how well v approximates u is the residual, given by

$$r = f - Av$$

The residual is simply the amount by which the approximation v fails to satisfy the original problem Au = f. It is also a vector and its size may be measured by the same norm used for the error. Notice that by the uniqueness of the solution u, r = 0 if and only if e = 0. However, it may not be true that, when r is small in norm, e is also small in norm.

If we rewrite the original problem as

$$Au = f$$

Rearrange the definition of the residual as

$$Av = f - r$$

And then subtract the second expression from the first, we find an extremely important relationship between the error and the residual. It is

$$Ae = r$$

and we call this the residual equation. It says that the error satisfies the same set of equations as the unknown u when f is replaced by the residual r. The residual equation plays a vital role in multigrid methods and it will be used throughout. In fact, we can now anticipate, how the residual can be used to great advantage. Assume that an approximation v has been computed by some unspecified method. It is easy to compute the residual r = f – Av. To improve the approximation v, we can solve the residual equation for e and then compute a new approximation using the definition of the error

$$u = v + e$$

This idea of residual correction is very important.

4.2.2 Properties of Relaxation Methods

We turn to our model problem

$$-u_{j-1} + 2u_j - u_{j+1} = h^2 f_j, \qquad 1 \le j \le N\text{-}1. \qquad (4.2.2.1)$$

$$u_0 = u_N = 0$$

And consider some relaxation methods for its solution. One of the simplest methods is the Jacobi (or simultaneous displacement) method. It is produced by solving the jth

equation of (4.2.2.1) for the jth unknown u_j. This results in an iteration scheme that may be written in component form as

$$v_j^{(1)} = \frac{1}{2}\left(v_{j-1}^{(0)} + v_{j+1}^{(0)} + h^2 f_j\right), \quad 1 \le j \le N\text{-}1.$$

To keep the notation simple, the current approximation (or the initial guess on the first iteration is denoted $v^{(0)}$, while the new, updated approximation is denoted $v^{(1)}$. In practice, once all of the $v^{(1)}$ components have been computed, the procedure is repeated with $v^{(1)}$ playing the role of $v^{(0)}$. These iteration sweeps are continued until (ideally) we obtain satisfactory convergence to the solution. These relaxation schemes can be expressed in matrix form also. We write the matrix A in the form

$$A = D - L - U$$

Where D is the diagonal of A and –L and –U are the strictly lower and upper triangular parts of A, respectively. Including the term h^2 in the vector f, then Au = f becomes

$$(D - L - U)u = f$$

This may be written as

$$Du = (L + U)u + f$$

Or as

$$u = D^{-1}(L + U)u + D^{-1}f$$

This corresponds exactly to solving the jth equation for u_j for $1 \le j \le N\text{-}1$. If we define the Jacobi iteration matrix by

$$P_J = D^{-1}(L + U)$$

The Jacobi method appears in matrix form as

$$v^{(1)} = P_J v^{(0)} + D^{-1} f$$

There is a simple but important modification that can be made to the Jacobi iteration. As before, we compute the new Jacobi iterates using

$$v_j^{(*)} = \frac{1}{2}\left(v_{j-1}^{(0)} + v_{j+1}^{(0)} + h^2 f_j\right), \quad 1 \le j \le N\text{-}1.$$

However $v^{(*)}$ is now only an intermediate value. The new approximation is given by the weighted average

$$v_j^{(1)} = (1-\omega)v_j^{(0)} + \omega v_j^{(*)} = v_j^{(0)} + \omega\left(v_j^{(*)} - v_j^{(0)}\right), \quad 1 \le j \le N\text{-}1.$$

where w is a weighting factor that may be chosen. This generates an entire family of iterations called the weighted or damped Jacobi method. Notice that with w = 1, we have the original Jacobi iteration. In matrix form the weighted Jacobi method is given by

$$v^{(1)} = \left[(1-\omega)I + \omega P_J\right]v^{(0)} + \omega D^{-1} f$$

The weighted Jacobi method waits until all the components of the new approximation have been computed before using them. This requires 2N storage locations for the approximation vector. It also means that new information cannot be used as soon as it is available.

The Gauss –Seidel method incorporates a simple change: components of the new approximation are used as soon as they are computed. This means that components of the approximation vector v are over-written as soon as they are updated. This small change reduces the storage requirement for the approximation vector to only N locations. The Gauss-Seidel method is equivalent to successively setting each component of the solution. In component form this method, when applied to the model problem, may be expressed as

$$v_j \leftarrow \frac{1}{2}\left(v_{j-1} + v_{j+1} + h^2 f_j\right), \qquad 1 \le j \le N\text{-}1.$$

where the arrow notation stands for replacement or over-writing.

In matrix form it is given by

$$u = (D-L)^{-1}Uu + (D-L)^{-1}f$$

Another effective alternative is to update all the even components first by the expression

$$v_{2j} \leftarrow \frac{1}{2}\left(v_{2j-1} + v_{2j+1} + h^2 f_{2j}\right),$$

And then return and update all the odd components using

$$v_{2j+1} \leftarrow \frac{1}{2}\left(v_{2j} + v_{2j+2} + h^2 f_{2j+1}\right),$$

This strategy is called the red-black Gauss-Seidel method that is illustrated in Fig.4.3 for both a one-dimensional and two-dimension grid.

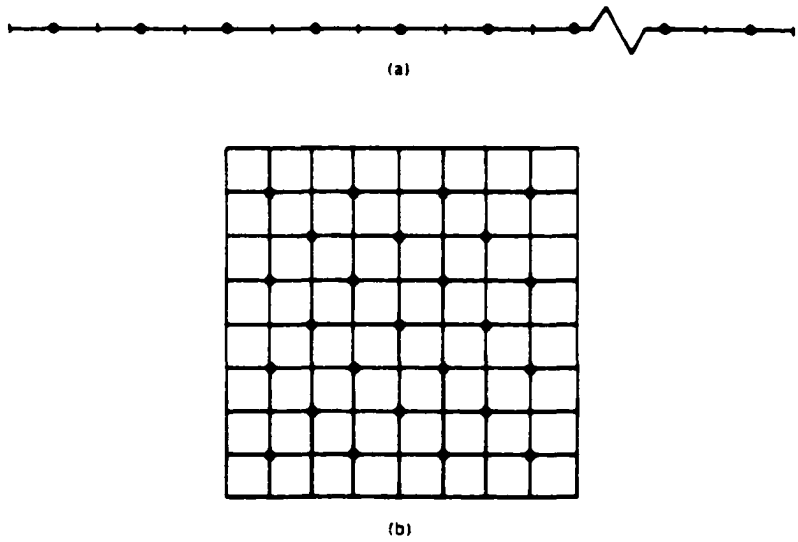


Fig.4.3 (a) One-dimensional and (b) two-dimensional grids showing the red points (unmarked) and black points (•) for the red-black relaxation.

The advantages of red-black versus regular Gauss-Seidel are not apparent. The issue is often problem-dependent. However, red-black Gauss-Seidel does have a clear advantage in terms of implementation on a parallel computer. The red points need only the black points for their updating and therefore may be updated in any order. This work represents independent tasks that can be distributed among several independent processors. Similarly, the black sweep can also be done by several independent processors.

When studying stationary linear iterations, it is sufficient to work with homogeneous linear system Au = 0 and use arbitrary initial guesses to start the relaxation scheme. One reason for doing this is that the exact solution (u = 0) is known and the error in an approximation v is simply –v. Therefore, we return to the one-dimensional model problem with f = 0. It appears as

$$-u_{j-1} + 2u_j - u_{j+1} = 0, \qquad 1 \le j \le N\text{-}1. \qquad (4.2.2.2)$$

$$u_0 = u_N = 0$$

We obtain some valuable insight by applying various iterations to this system of equations with an initial guess consisting of the vectors (or Fourier modes)

$$v_j = \sin\left(\frac{jk\pi}{N}\right)$$

where $0 \le j \le N$ and $1 \le k \le N\text{-}1$. Recall that j denotes the component (or associated grid point) of the vector v. The integer k is called the wavenumber and it indicates the number of half sine waves that constitute v on the domain of the problem. When it becomes necessary to designate the entire vector v with the wavenumber k, we shall use v_k. Fig.4.4 illustrates initial guesses v_1, v_3 and v_6. Notice that small values of k correspond to long, smooth waves, while large values of k correspond to oscillatory waves.

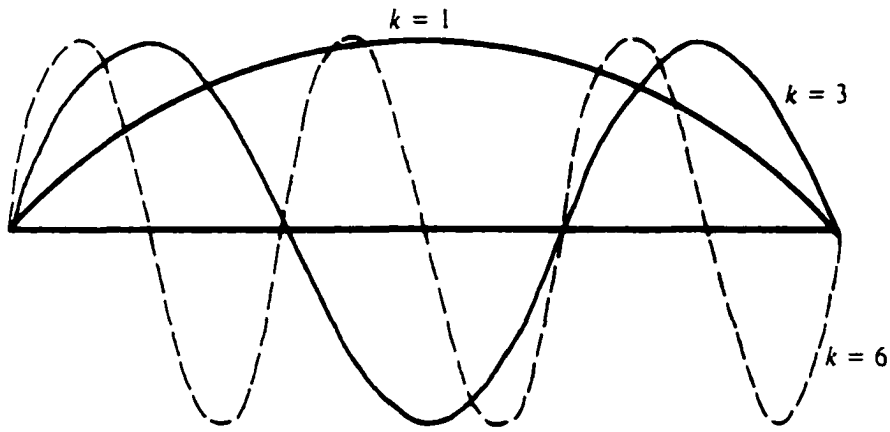


Fig.4.4 . The modes $v_j = \sin\left(\frac{jk\pi}{N}\right)$, $0 \le j \le N$, with wavenumbers k=1,3,6. The k*th* mode consists of $k/2$ full sine waves on the interval.

We begin by applying the weighted Jacobi iteration with w = 2/3 to (4.2.2.2) on a grid with N = 64. Beginning with initial guesses of v_1, v_3 and v_6, the iteration is applied 100 times. After each sweep, we compute the maximum norm of the error (which is –v). Fig.4.5 shows a plot of the error norm versus the iteration number.

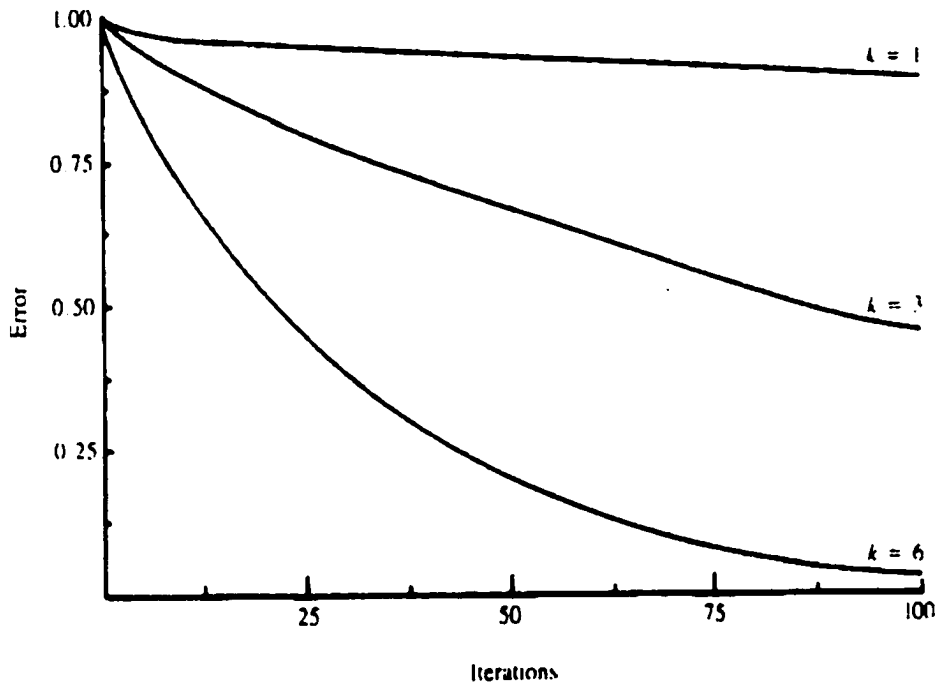


Fig.4.5 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N=64 and with initial guesses of v_1, v_3, and v_6. The maximum norm of the error $\|e\|_\infty$ is plotted against the iteration number for 100 iterations.

For the moment, we will consider only the qualitative behavior of the iteration. The error clearly decreases with each relaxation sweep, and the rate of decrease is larger for the higher wavenumbers. Figs.4.6 and 4.7 show analogous plots for the regular and red-black Gauss-Seidel iterations. We see a similar relationship among the error, the number of iterations and the wavenumber.

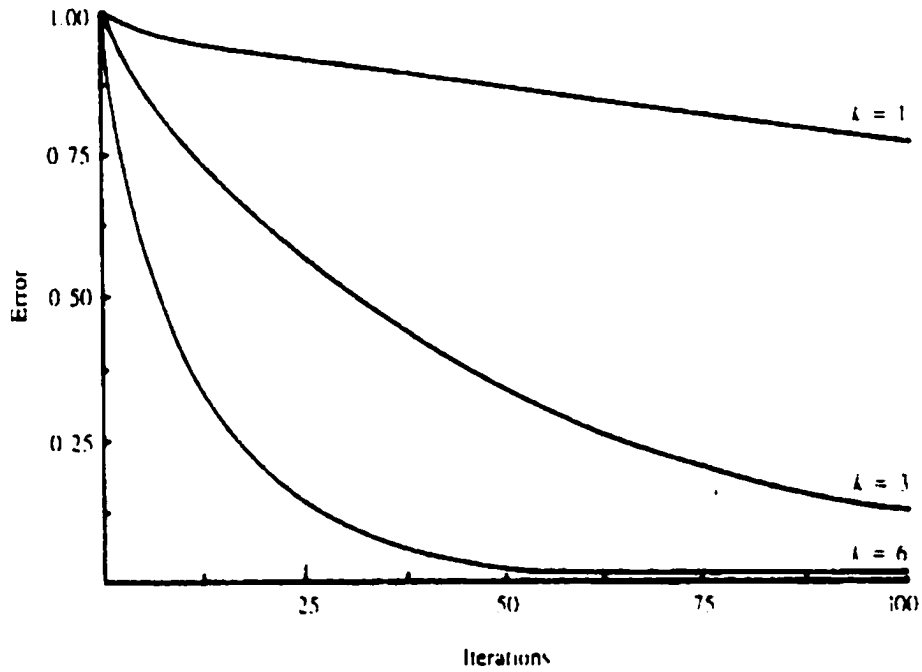


Fig.4.6 Regular Gauss-Seidel iteration applied to the one-dimensional model problem with N=64 and with initial guesses of v_1, v_3, and v_6. The maximum norm of the error $\|e\|_\infty$ is plotted against the iteration number for 100 iterations.

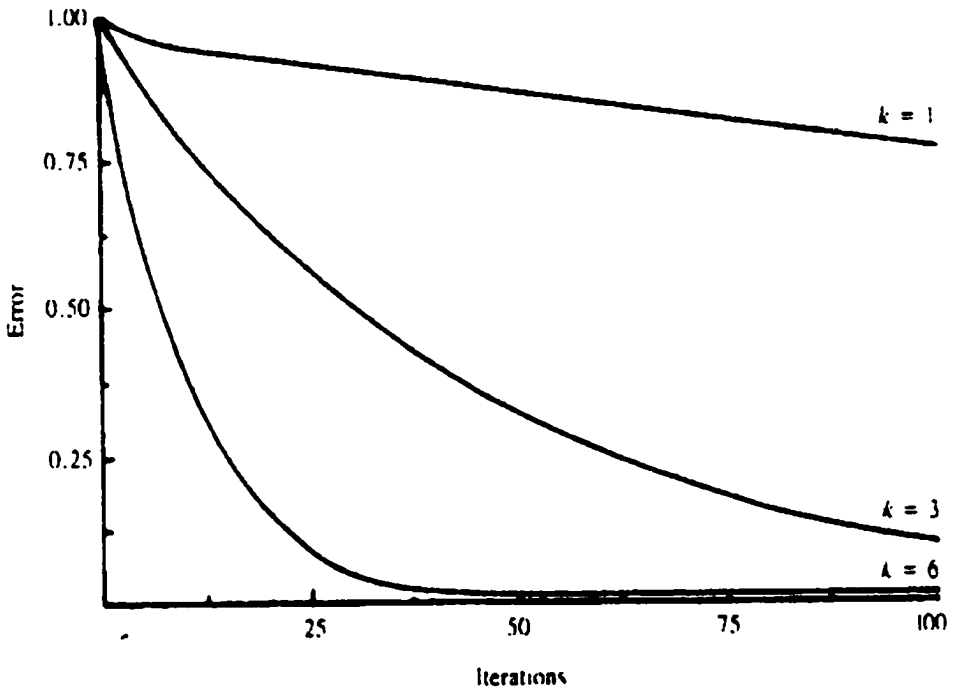


Fig.4.7 Red-black Gauss-Seidel iteration applied to the one-dimensional model problem with N=64 and with initial guesses of v_1, v_3, and v_6. The maximum norm of the error $\|e\|_\infty$ is plotted against the iteration number for 100 iterations.

The experiment of Fig.4.5 is presented in a slightly different light in Fig.4.8. In this figure, the log of the error norm for the weighted Jacobi method is plotted against the iteration number for various wavenumbers. This plot shows a clear linear decrease in the log of the error norm indicating that the error itself decreases geometrically with each iteration. If we let $e^{(0)}$ be the error in the initial guess and $e^{(n)}$ be the error in the nth iterate, then we might expect to describe the error by a relationship of the form

$$\left\|e^{(n)}\right\|_\infty = c_k^n \left\|e^{(0)}\right\|_\infty$$

Where c_k is a constant that depends upon the wavenumber.

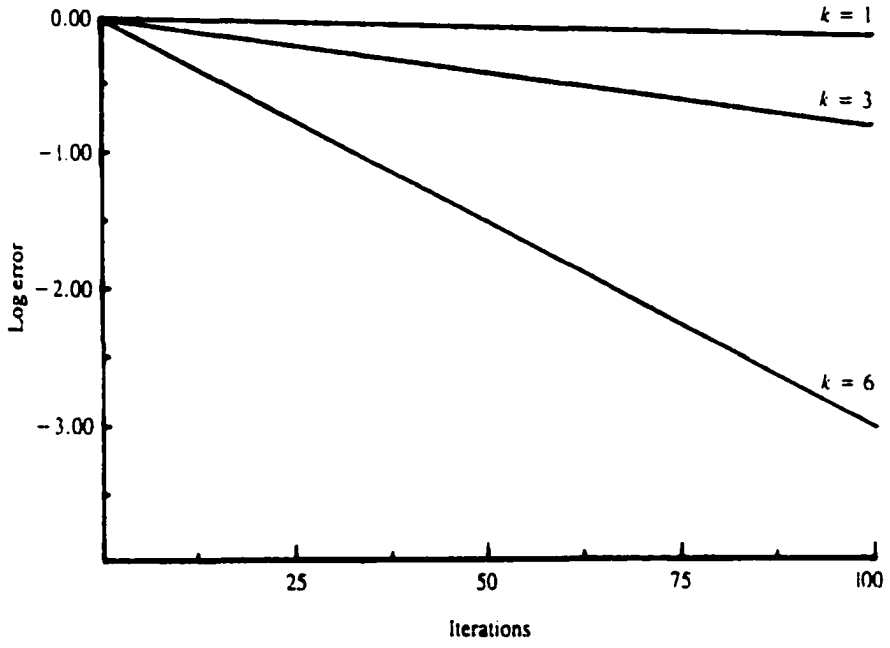


Fig.4.8 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N=64 and with initial guesses consisting of v_1, v_2 and v_6. The log of $\|e\|_\infty$ is plotted against the iteration number for 100 iterations.

In general most initial guesses (or equivalently, most right-hand side vectors f) would not consist of a single mode. Fig.4.9 shows a more realistic situation in which the initial guess consists of three modes: a low frequency wave (k = 1), a medium frequency wave (k = 6) and a high frequency wave (k = 32) on a grid with N = 64. As before, the maximum norm of the error is plotted against the number of iterations. The error decreases rapidly within the first five iterations, after which it decreases much more slowly. The initial decrease corresponds to the quick elimination of the high-frequency modes. The slow decrease is due to the presence of the low frequency modes. The important observation is that the standard iterations converge very quickly as long as the error has high-frequency components. However, the slower elimination of the low-frequency components degrades the performance of these methods.

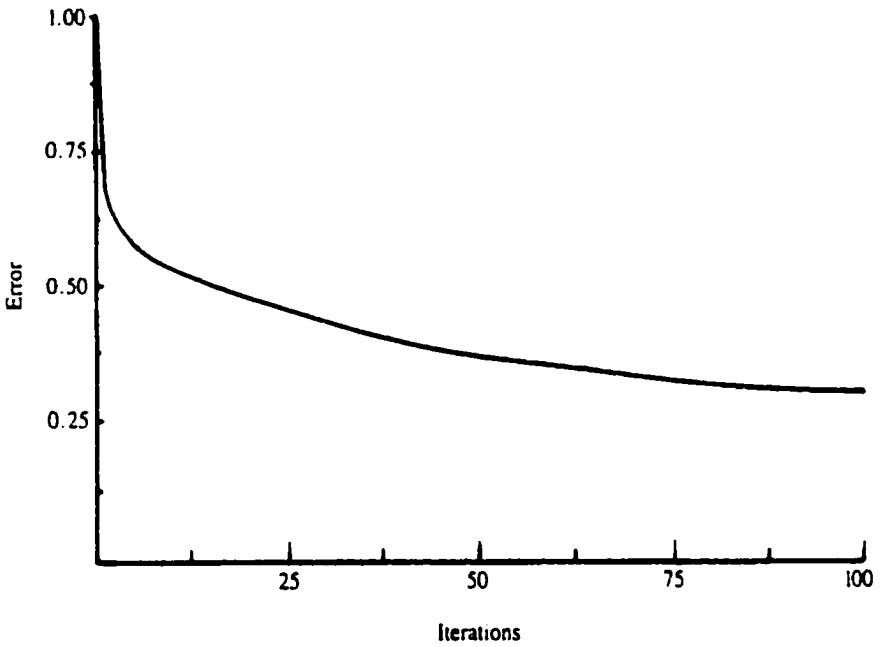


Fig.4.9 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with $N = 64$ and with initial guesses consisting of $\frac{1}{3}\left[\sin\left(\frac{j\pi}{N}\right)+\sin\left(\frac{6j\pi}{N}\right)+\sin\left(\frac{32j\pi}{N}\right)\right]$. The maximum norm of the error $\|e\|$ is plotted against the iteration number for 100 iterations.

4.2.3 An Analytical Approach

Let us turn to a more analytical approach. Consider the following partial differential equation as an example

$$Lu(x,y) \equiv a\frac{\delta^2 u(x,y)}{\delta x^2} + c\frac{\delta^2 u(x,y)}{\delta y^2} = f(x,y)$$

with some boundary conditions. Let u^h be an approximation to the exact solution on a grid with meshsize h. Using second order discretization, we obtain

$$a\frac{u^h_{i+1,j} - 2u^h_{i,j} + u^h_{i-1,j}}{h^2} + c\frac{u^h_{i,j+1} - 2u^h_{i,j} + u^h_{i,j-1}}{h^2} = f_{i,j} \qquad (4.2.3.1)$$

where i,j are integers.

Given an approximation $v_{i,j}$ to u^h, let us apply the Gauss-Seidel iteration to this finite difference equation. The points (i,j) of the grid are scanned in a lexicographic order. At every point the value v^{k-1}_{ij} is replaced by a new value v^k_{ij} such that

$$a\frac{v^{k-1}_{i+1,j} - 2v^k_{i,j} + v^k_{i-1,j}}{h^2} + c\frac{v^{k-1}_{i,j+1} - 2v^k_{i,j} + v^k_{i,j-1}}{h^2} = f_{i,j} \qquad (4.2.3.2)$$

New values v^k_{i-1j} , v^k_{ij-1} are used since by the time (i,j) is scanned, new values have replaced old ones at $(i-1,j)$ and $(i,j-1)$.

The new approximation v does not satisfy Equation (4.2.3.2), and more relaxation sweeps may be needed to improve the approximation. The convergence factor, μ is defined as

$$\mu = \frac{\|e^k\|}{\|e^{k-1}\|}$$

where $e^{k-1} = u^h - v^{k-1}$, $e^k = u^h - v^k$

For the Gauss Seidel scheme, $\mu = 1 - O(h^2)$, so that $O(h^{-2})$ relaxation sweeps are needed to reduce the error of magnitude.

The purpose of relaxation in multigrid methods is not just to reduce the error, but to smooth it out so that it becomes well approximable on a coarser grid. Subtracting Equation (4.2.3.2) from Equation (4.2.3.1),

$$a\left(e^{k-1}_{i+1,j} - 2e^k_{i,j} + e^k_{i-1,j}\right) + c\left(e^{k-1}_{i,j+1} - 2e^k_{i,j} + e^k_{i,j-1}\right) = 0 \qquad (4.2.3.3)$$

And since $e^k_{i,j}$ is a weighted average of neighboring values, if the old error e^{k-1} is not smooth, the new error e^k must be smoother.

To analyze the smoothing effect of a relaxation sweep quantitatively, we consider the local nature (points several meshsizes apart affecting each other exponentially little) of the point, regarding the grid as embedded in a rectangular domain. Expanding e^{k-1} and e^k in Fourier series,

$$e^{k-1}_{i,j} = \sum A^{k-1}_{\theta} e^{m(\theta_1 i + \theta_2 j)}, e^{k}_{i,j} = \sum A^{k}_{\theta} e^{m(\theta_1 i + \theta_2 j)}, \qquad (4.2.3.4)$$

where, $\theta = (\theta_1, \theta_2)$ and the summations are carried over $|\theta| = \max(|\theta_1|, |\theta_2|) \leq \pi$. Substituting Equation (4.2.3.4) in Equation (4.2.3.3) gives,

$$\left(ae^{m\theta_1} + ce^{m\theta_2}\right)A^{k-1}_{\theta} + \left(ae^{-m\theta_1} + ce^{-m\theta_2} - 2a - 2c\right)A^{k}_{\theta} = 0$$

The amplification factor of the θ component due to one relaxation sweep is

$$\mu(\theta) = \frac{\left|A^{k}_{\theta}\right|}{\left|A^{k-1}_{\theta}\right|} = \frac{\left|ae^{m\theta_1} + ce^{m\theta_2}\right|}{\left|2a + 2c - ae^{-m\theta_1} - ce^{-m\theta_2}\right|}$$

It measures the growth or decay of a Fourier mode of the error during an iteration. $\mu(\theta) \rightarrow 1$ as $\theta \rightarrow (0,0)$. In domains of diameter O (1), the lowest non-trivial Fourier components have $|\theta| = O(h)$, thereby $\mu(\theta) = 1 - O(h^2)$. $\mu(\theta)$ shows that the rate of convergence deteriorates as $h \rightarrow 0$. Apart from special cases, for elliptical equations, it is found to be true for all the basic iterative methods in which a grid function value is updated using only neighboring values.

4.2.4 Some Numerical Experiments

Let us develop more familiarity with these Fourier modes. Fig.4.10 shows some of the modes on a grid with N = 12. Notice that the kth mode consists of K/2 full sine waves and has a wavelength of 2/k (the entire interval has length 1). The k = N/2 mode has a wavelength of 4h and the k = N – 1 mode has a wavelength of almost 2h. Waves with wavenumbers greater than N (or wavelengths less than 2h) cannot be represented on the grid. In fact, through the problem of alaising, a wave with a wavelength less than 2h actually appears on the grid with a wavelength greater than 2h.

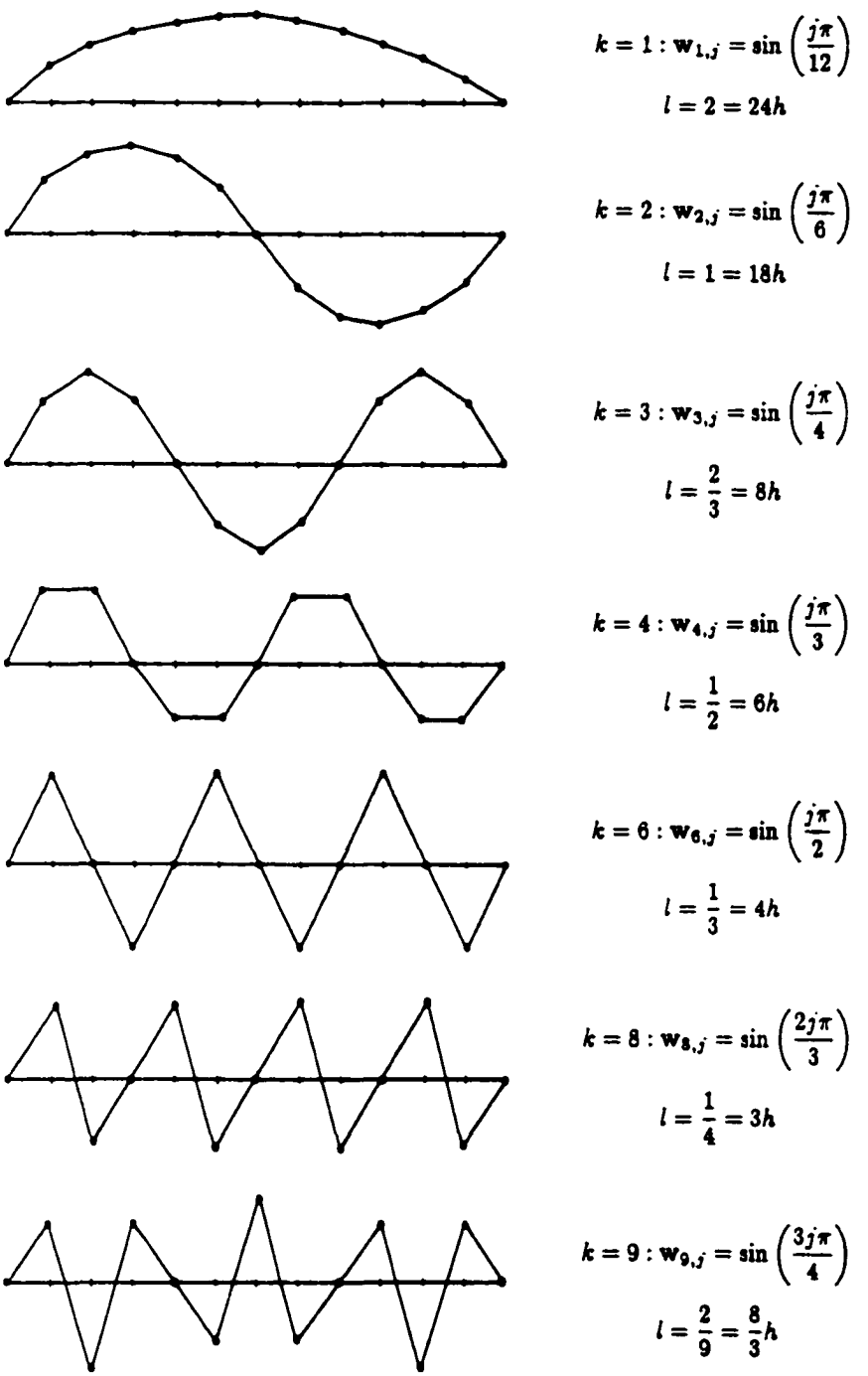


Fig.4.10 Graphs of the Fourier modes of A on a grid with N=12. Modes with wavenumbers k= 1,2,3,4,6,8,9 are shown. The wavelength of the kth mode is $l = 24h/k$.

We will use some terminology now. We need some quantitative terms for the various Fourier modes that have been discussed. The modes in the lower half of the spectrum, with wavenumbers in the range $1 \leq k \leq N/2$, will be called low frequency or smooth modes. The modes in the upper half oh the spectrum, with $N/2 \leq k \leq N-1$, will be called high frequency or oscillatory modes.

We now return to some numerical experiments. Once again, the weighted Jacobi method is applied to the one-dimensional problem $Au = 0$ on a grid with $N = 64$. We will use initial guesses (which are also initial errors) consisting of single modes with wavenumbers $1 \leq k \leq N-1$. Figs.4.11 and 4.12 show how the method performs in terms of different wavenumbers. Specifically, the wavenumber of the initial error is plotted against the number of iterations required to reduce the norm of the initial error by a factor of 100. This experiment is done for weighting factors of $w = 1$ and $w = 2/3$. With $w = 1$, both the high- and low-frequency components of the error are damped very slowly. Components with wavenumbers near n/2 are damped rapidly. We see a quite different behavior in Fig.4.12 with $w = 2/3$. $w = 2/3$ is chosen to give preferential damping to the oscillatory components. Indeed, the smooth waves are damped very slowly, while the upper half of the spectrum ($k \geq N/2$) shows a rapid convergence.

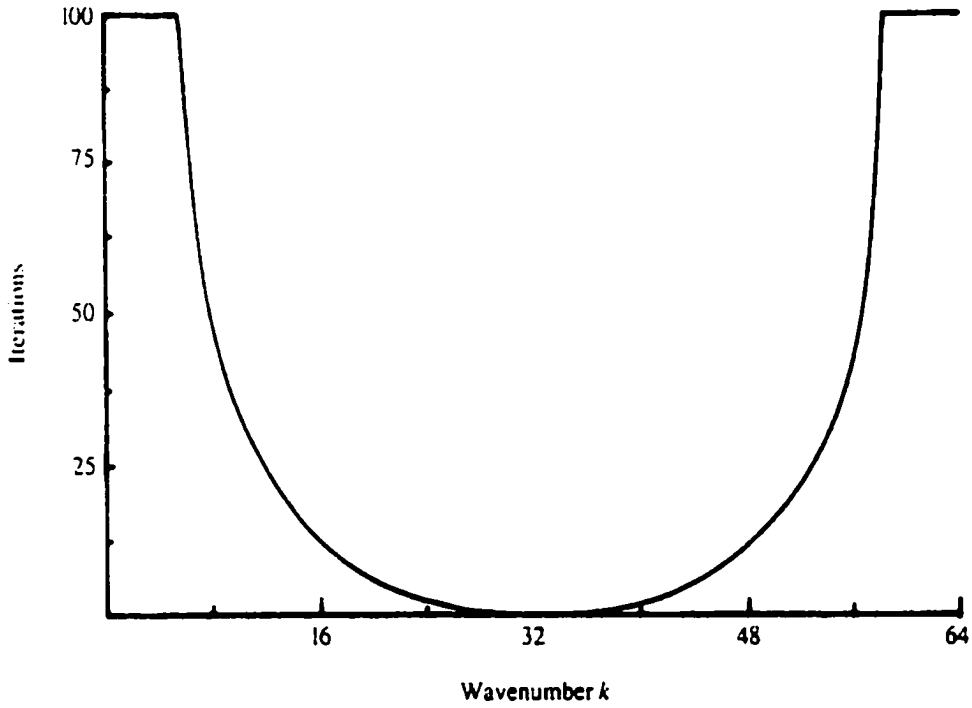


Fig.4.11 The weighted Jacobi method with $w = 1$ applied to the one-dimensional model problem with N=64. The initial guesses consist of the modes w_k for $1 \le k \le 63$. The graph shows the number of iterations required to reduce the norm of the initial error by a factor of 100 for each w_k.

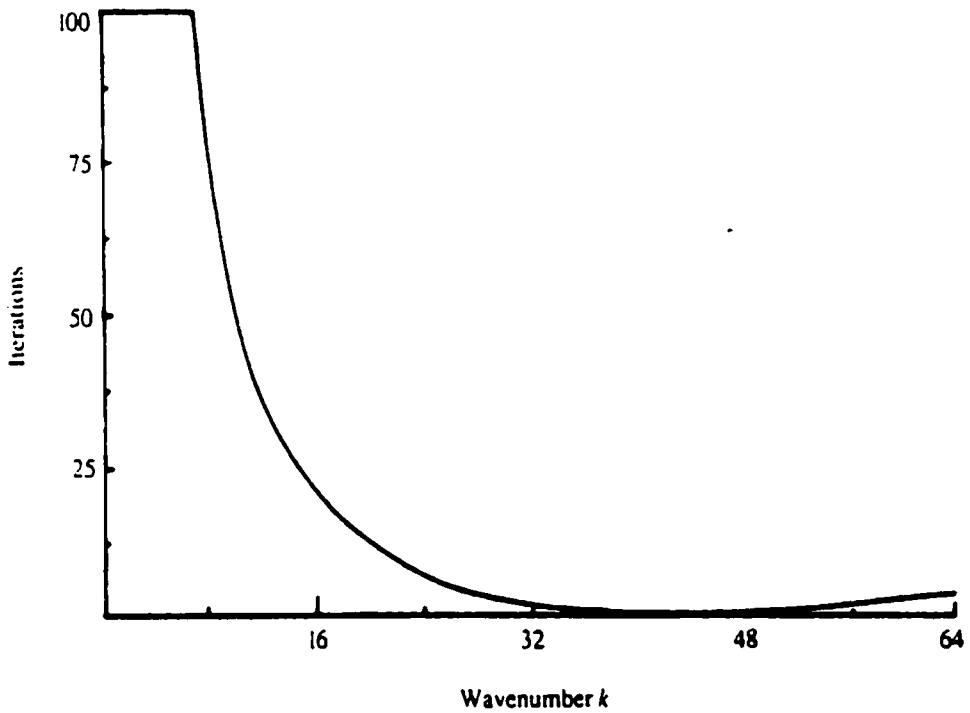


Fig.4.12 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N=64. The initial guesses consist of the modes w_k for $1 \le k \le 63$. The graph shows the number of iterations required to reduce the norm of the initial error by a factor of 100 for each w_k. Note that for $w = 2/3$ the damping is the strongest for the oscillatory modes $(32 \le k \le 63)$.

Another perspective on these convergence properties is provided in Figs.4.13-4.15. This time the actual approximations are plotted. The weighted Jacobi method with w = 2/3 is applied to the same model problem on a grid with N = 64. Fig.4.13 shows the error with wavenumber k = 3 after one relaxation sweep in the left-hand plot and the error after ten relaxation sweeps on the right-hand plot. This long, smooth component is damped very slowly.

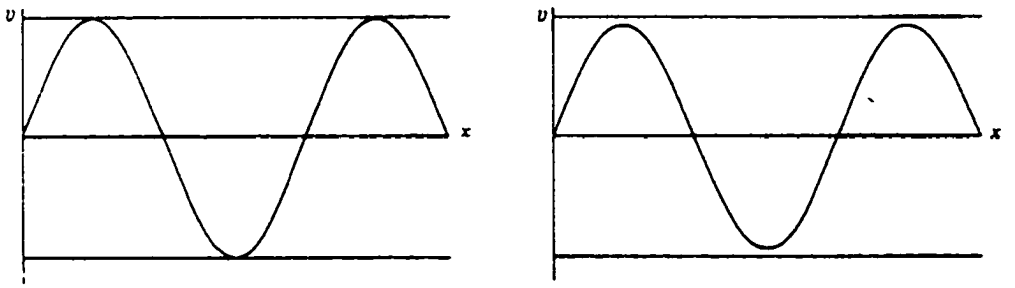


Fig.4.13 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N=64 and an initial guess consisting of w_3. The figure shows the approximation after one iteration (on the left) and after ten iterations (on the right).

Fig.4.14 shows a more oscillatory wave (k = 16) after one and ten iterations. The damping is now much more dramatic. Notice that the weighted Jacobi method preserves modes: once a k = 3 mode, always a k = 3 mode.

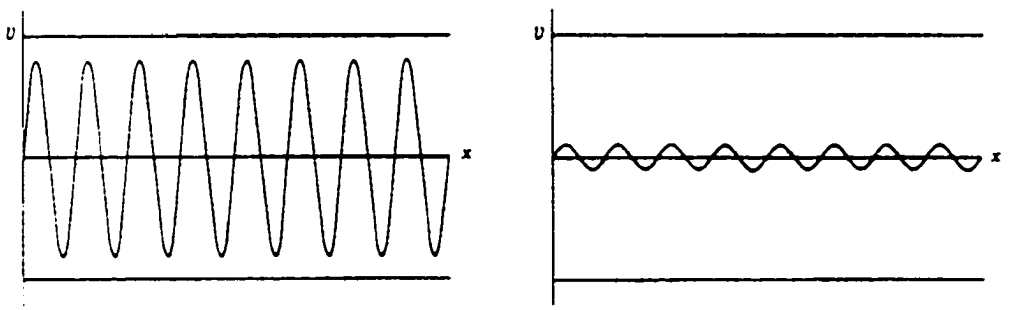


Fig.4.14 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N=64 and an initial guess consisting of w_{16}. The figure shows the approximation after one iteration (on the left) and after ten iterations (on the right).

Fig.4.15 illustrates the selectivity of the damping property. This experiment uses an initial guess consisting of two modes with k = 2 and k = 16. After ten relaxation schemes, the high-frequency modulation on the long wave has been nearly eliminated. However, the original smooth component persists.

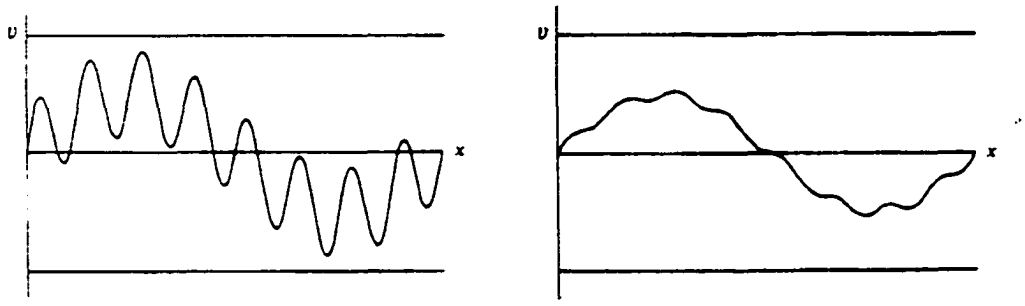


Fig.4.15 The weighted Jacobi method with $w = 2/3$ applied to the one-dimensional model problem with N=64 and an initial guess consisting of a combination of w_2 and w_{16}. The figure shows the approximation after one iteration (on the left) and after ten iterations (on the right).

We have looked in detail at the convergence properties of some basic relaxation schemes. The experiments presented reflect the experience of many practitioners. These schemes work very well for the first several iterations. Inevitably, however, the convergence slows and the entire scheme appears to stall. We have found a simple explanation for this phenomenon: the rapid decrease in error during the early iterations is due to the efficient elimination of the oscillatory modes of the error. Once the oscillatory modes have been removed, the iteration is much less effective in reducing the remaining smooth components.

Many relaxation schemes possess this property of eliminating the oscillatory modes and leaving the smooth modes. We shall call this property the smoothing property. It is a serious limitation of these methods. However, it can be overcome, and the remedy is one of the pathways to multigrid.

4.3 Elements of Multigrid

4.3.1 The Multigrid Principle

We have examined some of the basic iterative methods through analysis and experimentation. It is the beginning of what we might call the spectral (or Fourier mode) picture of relaxation schemes. So far, we have seen that many iterative methods possess the smoothing property. This property makes these methods very effective at eliminating the high frequency or oscillatory components of the error, while leaving the low frequency or smooth components relatively unchanged. The immediate issue is whether these methods can be adapted in some way to make them effective on all error components.

One way to improve a relaxation scheme is to use a good initial guess. A well-known technique for obtaining an improved initial guess is to perform some preliminary iterations on a coarser grid and then use the resulting approximation as an initial guess on the original fine grid. Relaxation on the coarser grid is less expensive since fewer unknowns are to be updated. Also, since the convergence factor behaves like $1-O(h^2)$, the coarser grid will have marginally improved convergence rate. With the coarse grid idea in mind, we recall that most basic relaxation schemes suffer in the presence of smooth components of the error. Assume that a particular relaxation scheme has been applied until only smooth error components remain. We will now see what these smooth components look like on a coarser grid. Fig.4.16 shows the answer. A smooth wave with k = 4 on a grid Ω^h with N = 12 has been projected directly to the grid Ω^{2h} with N = 6. On this coarse grid, the original wave has a wavenumber of k = 4. We see that a smooth wave on Ω^h looks more oscillatory on Ω^{2h}.

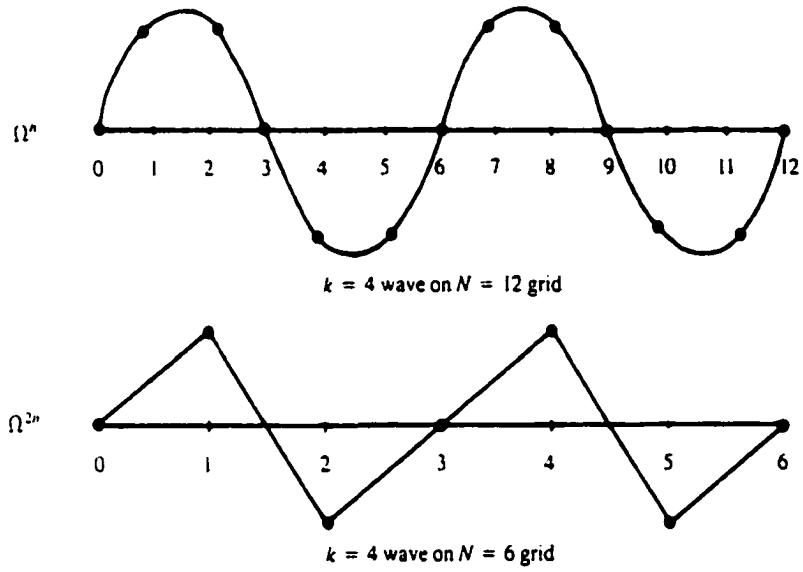


Fig.4.16 A wave with wavenumber k=4 on Ω^h (N=12) is projected onto Ω^{2h} (N=6). The coarser grid "sees" a wave with k =4 which is more oscillatory on the coarser grid than on the fine grid.

To be precise, note that the grid points of the coarse grid Ω^{2h} are the even-numbered grid points of the fine grid Ω^h. Consider the kth mode of the fine grid evaluated at the even-numbered grid points. If $1 \le k < N/2$, its components may be written as

$$\omega^h_{k,2j} = \sin\left(\frac{2jk\pi}{N}\right) = \sin\left(\frac{jk\pi}{N/2}\right) = \omega^{2h}_{k,j}, \qquad 1 \le k < N/2$$

This equation says that the kth mode on Ω^h becomes the kth mode on Ω^{2h}, which means that in passing from the fine grid to the coarse grid, a mode becomes more oscillatory. It is true provided that $1 \le k < N/2$.

The rate of convergence of basic iterative methods can be improved with multigrid methods. The Fourier analysis of the convergence of iterative methods tells us that, although long wavelength Fourier modes decay slowly, short wavelength Fourier modes are reduced rapidly [5]. The essential multigrid principle is to approximate the

smooth (long wavelength) part of the error on coarser grids. The non-smooth, or rough part is reduced with a small number of iterations with a basic iterative method on the fine grid.

4.3.2 Two Strategies

An important point to note is that smooth modes on a fine grid look less smooth on a coarser grid. It suggests that when relaxation begins to stall, signaling the predominance of smooth error modes, it is advisable to move to a coarser grid, on which those smooth error modes appear more oscillatory and relaxation will be more effective. The question is: how do we move to a coarser grid and relax on those oscillatory error modes? At this point we introduce the equation for the error, namely the residual equation. If v is an approximation to the exact solution u, then the error e = u – v satisfies

$$Ae = r = f - Av$$

This suggests that we can relax directly on the error by using the residual equation.

Another argument justifies the use of the residual:

Relaxation on the original equation Au = f with an arbitrary initial guess v is equivalent to relaxing on the residual equation Ae = r with the specific initial guess e =0. This intimate connection between the original and the residual equations further motivates the use of the residual equation.

At this point, we have seen that many relaxation schemes possess the smoothing property. It leads us to consider using coarser grids during the computation to focus the relaxation on the oscillatory components of the error. In addition, there seems to be good reason to involve the residual equation in the picture. We now give these ideas a little more definition by proposing two strategies.

4.3.2.1 Nested Iteration. The first strategy incorporates the idea of using coarse grids to obtain better initial guesses. Let us represent it by the following procedure.

Relax on Au = f on a very coarse grid.

.

.

.

Relax on Au = f on Ω^{4h} to obtain an initial guess for Ω^{2h}.

Relax on Au = f on Ω^{2h} to obtain an initial guess for Ω^{h}.

Relax on Au = f on Ω^{h} to obtain a final approximation to the solution.

The idea of using coarser grids to generate improved initial guesses is the basis of a strategy called nested iteration. Some questions must still be answered. For instance, what does it mean to relax on Au = f on Ω^{2h}? We must somehow define the problem on the coarser grids. Also, what happens if, having reached the fine grid, there are still smooth components in the error? We may have obtained some improvement by using coarse grids, but the final iteration will stall if smooth components still remain. We will soon find some answers that will let us use nested iteration in a very powerful way.

4.3.2.2 Coarse Grid Correction. A second strategy incorporates the idea of using the residual equation to relax on the error. It can be represented by the following procedure.

Relax on Au = f to Ω^{h} obtain an approximation v^{h}.

Compute the residual $r = f - Av^{h}$.

Relax on the residual equation Ae = r on Ω^{2h} to obtain an approximation to the error e^{2h}.

Correct the approximation obtained on Ω^{h} with the error estimate obtained on Ω^{2h}: $v^{h} \leftarrow v^{h} + e^{2h}$.

This procedure is the basis of what is called coarse grid correction. Having relaxed on the fine grid until the convergence deteriorates, we relax on the residual equation on a coarser grid to obtain an approximation to the error itself. We then return to the fine grid to correct the approximation first obtained there.

There are even more questions to be answered. For instance, how do we compute the residual on Ω^{h} and transfer it to Ω^{2h}? How do we compute an error estimate on Ω^{2h} and transfer it to Ω^{h}? And as before, how do we represent the original problem Au =f on a coarser grid? These questions suggest that we need some mechanisms for transferring information between the grids.

4.3.3 Inter-Grid Transfer Mechanisms

In our discussion of intergrid transfers, we consider only the case in which the coarse grid has twice the grid spacing of the next finest grid since there seems to be no advantage in using grid spacings with ratios other than 2.

4.3.3.1 Interpolation. Consider the step in the coarse grid correction scheme that requires transferring the error approximation e^{2h} from the coarse grid Ω^{2h} to the fine grid Ω^{h}. This step is a common procedure in numerical analysis and is generally called interpolation or prolongation. Many interpolating methods can be used, but the most

common is the linear interpolator. It takes coarse grid vectors and produces fine grid vectors and is defined as $I_{2h}^{h} v^{2h} = v^{h}$. In one-dimension, it is given by

$$v_{2j}^{h} = v_{j}^{2h}$$

$$v_{2j+1}^{h} = \frac{1}{2}\left(v_{j}^{2h} + v_{j+1}^{2h}\right) \qquad 0 \le j \le N/2 - 1$$

Fig.4.17 shows graphically the action of I_{2h}^{h}. At even-numbered fine grid points, the values of the vector are transferred directly from Ω^{2h} to Ω^{h}. At odd-numbered points, the value of v^{h} is the average of the adjacent coarse grid values.

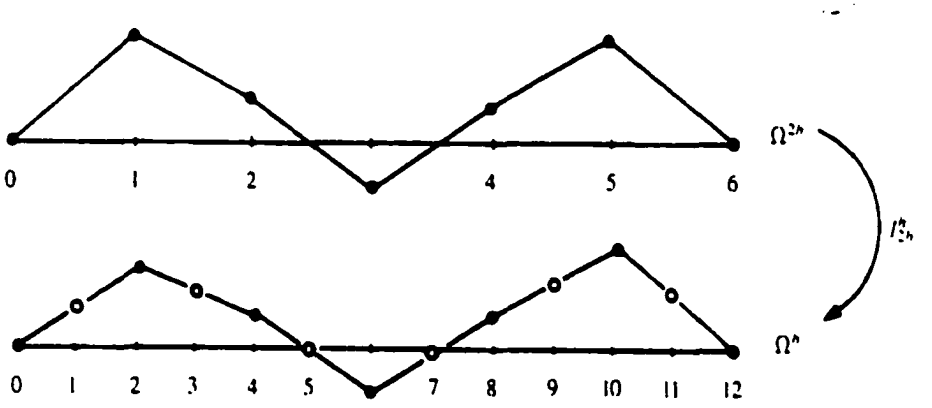


Fig.4.17 Interpolation of a vector on the coarse grid Ω^{2h} to the fine grid Ω^{h}.

To see how well the interpolation process works; assume that the 'real' error (which is not known exactly) is a smooth vector on a fine grid. Assume also that a coarse grid approximation to the error has been determined on Ω^{2h}, and furthermore, that this approximation is exact at the coarse grid points. When the coarse grid approximation is interpolated to the fine grid, the interpolant is also smooth. Hence, we expect a relatively good approximation to the fine grid error as shown in Fig.4.18(a).

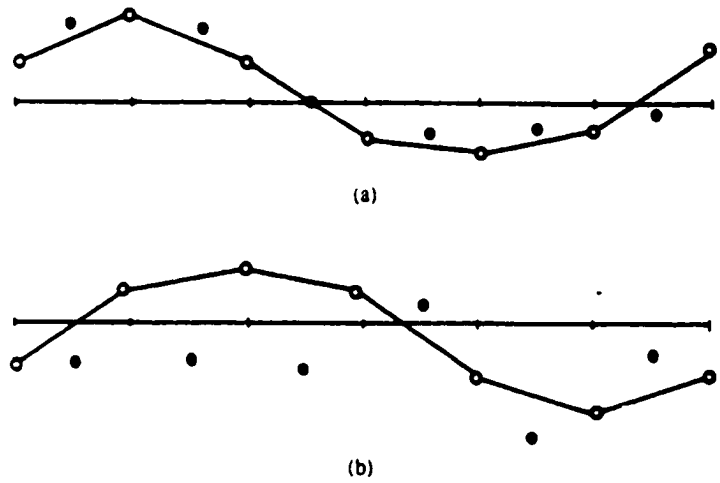


Fig.4.18 (a) If the exact error on Ω^h (indicated by $^\circ$ and •) is smooth, an interpolant of the coarse grid approximation e^{2h} (indicated by °) should give a good representation of the exact error. (b) If the exact error on Ω^h (indicated by $^\circ$ and •) is oscillatory, an interpolant of the coarse grid approximation e^{2h} (indicated by °) may give poor representation of the exact error.

By contrast, if the 'real' error is oscillatory, even a very good coarse grid approximation may produce an interpolant that is not very accurate. This situation is shown in Fig.4.18(b).

In two dimensions it is given by

$$v^h_{2i,2j} = v^{2h}_{i,j}$$

$$v^h_{2i+1,2j} = \frac{1}{2}\left(v^{2h}_{i,j} + v^{2h}_{i+1,j}\right)$$

$$v^h_{2i,2j+1} = \frac{1}{2}\left(v^{2h}_{i,j} + v^{2h}_{i,j+1}\right)$$

$$v^h_{2i+1,2j+1} = \frac{1}{4}\left(v^{2h}_{i,j} + v^{2h}_{i+1,j} + v^{2h}_{i,j+1} + v^{2h}_{i+1,j+1}\right)$$

$$0 \le i,j \le N/2 - 1.$$

Both the interpolation and the restriction processes are most effective when the error is smooth. It explains the purpose of relaxation in multigrid methods to smooth out the error so that it becomes well approximable on the coarser grid. These processes provide a fortunate complement to relaxation most effective when the error is oscillatory.

4.3.3.2 Restriction. The second class of intergrid transfer functions involves moving vectors from a fine grid to a coarser grid. I_h^{2h} is a fine-to-coarse restriction operator, called residual weighting or restriction. The most obvious restriction operator is injection. It is defined by $I_h^{2h} v^h = v^{2h}$, where

$$v_{i,j}^{2h} = v_{2i,2j}^h$$

The value at a coarse grid point takes its value directly from the fine grid point. An alternate restriction operator is called full weighting and uses the weighted average of the neighboring points. The full weighting operator is a linear operator. It is defined by

$$I_h^{2h} v^h = v^{2h}.$$

In one-dimension, it is given by

$$v_j^{2h} = \frac{1}{4}\left(v_{2j-1}^h + 2v_{2j}^h + v_{2j+1}^h\right), \qquad 1 \le j \le N/2-1$$

Fig.4.19 shows $I_h^{2h} v^h$ is a coarse grid function whose value at each point is a certain weighted average of values of v^h at neighboring fine-grid points.

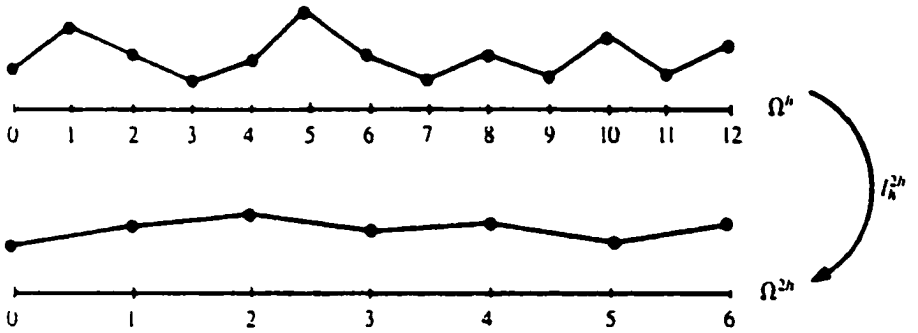


Fig.4.19 Restriction by full weighting of a fine grid vector to the coarse grid.

We will use the full weighting operator for our problem. However, in some instances injection may be a better choice. The issue of intergrid transfers, an important part of multigrid theory, is discussed in the *Multigrid Techniques* [5]. One reason for choosing full weighting, as a restriction operator is the important fact that the interpolation operator and the full weighting operator are transposes of each other up to a constant.

In two dimensions it is given by

$$v_{i,j}^{2h} = \frac{1}{16}\begin{pmatrix} v_{2i-1,2j-1}^{h} + v_{2i-1,2j+1}^{h} + v_{2i+1,2j-1}^{h} + v_{2i+1,2j+1}^{h} \\ +2\left(v_{2i,2j-1}^{h} + v_{2i,2j+1}^{h} + v_{2i-1,2j}^{h} + v_{2i+1,2j}^{h}\right) \\ +4v_{2i,2j}^{h} \end{pmatrix}, \qquad 1 \le i,j \le N/2 - 1.$$

We now have a well defined way to transfer vectors between fine and coarse grids. We will now return to the coarse grid correction strategy. We now define the coarse grid correction scheme.

4.3.4 Coarse Grid Correction Scheme

Relaxation sweeps very quickly reduce all the non-smooth or high-frequency components of the error. The smoother part should be reduced by approximating on a coarser grid. Consider the following scheme:

$v^h \leftarrow CG(v^h, f^h)$

Relax $\mathcal{G}_1$ times on $A^h u^h = f^h$ on Ω^h with initial guess v^h.

Compute $r^{2h} = I_h^{2h}(f^h - A^h v^h)$.

Solve $A^{2h} e^{2h} = r^{2h}$ on Ω^{2h}.

Correct fine grid approximation: $v^h \leftarrow v^h + I_{2h}^h e^{2h}$.

Relax $\mathcal{G}_2$ times on $A^h u^h = f^h$ on Ω^h with initial guess v^h.

For clarity of notation, when it is important to indicate the grid on which a particular vector or matrix is defined, the superscript h or 2h is used. The coarse grid representation of the original matrix A^h is denoted by A^{2h}. The integers $\mathcal{G}_1$ and $\mathcal{G}_2$ are parameters in the scheme which control the number of relaxation sweeps before and after the coarse grid correction.

This procedure is simply the original coarse grid correction idea, proposed earlier, but now refined by the use of intergrid transfer operators. We relax on the fine grid until it ceases to be worthwhile. In practice $\mathcal{G}_1$ is often 1, 2 or 3. The residual of the current approximation is computed on Ω^h and then transferred by a restriction operator to the coarse grid. The procedure calls for the exact solution of the residual equation on Ω^{2h}, which may not be possible. However, if the coarse grid error can at least be approximated, it is then interpolated up to the fine grid, where it is used to correct the fine

grid approximation. It is followed by ϑ_2 additional fine grid relaxation sweeps. It is important to see the complementarity at work in the process. Relaxation on the fine grid will eliminate the oscillatory components of the error, leaving relatively smooth error. Assume that we can solve the residual equation exactly on Ω^{2h}. It is still important to transfer that error accurately back to the fine grid. But since the error is smooth, interpolation will work very well and the error will be represented accurately on the fine grid.

Consider a numerical example, the weighted Jacobi method with $\omega = 2/3$ applied to the one-dimensional model problem Au = 0 on a grid with N = 48. We use an initial guess

$$v_j^h = \frac{1}{2}\left[\sin\left(\frac{12j\pi}{N}\right) + \sin\left(\frac{30j\pi}{N}\right)\right]$$

Consisting of the two modes k =12 and k = 30. The following coarse grid correction has been used.

Relax three times on $A^h u^h = f^h$ on Ω^h with initial guess v^h.

Compute $r^{2h} = I_h^{2h}\left(f^h - A^h v^h\right)$.

Relax three times on $A^{2h}e^{2h} = r^{2h}$ on Ω^{2h} with initial guess $e^{2h} = 0$.

Correct fine grid approximation: $v^h \leftarrow v^h + I_{2h}^h e^{2h}$.

Relax three times on $A^h u^h = f^h$ on Ω^h with initial guess v^h.

Compute $r^{2h} = I_h^{2h}\left(f^h - A^h v^h\right)$.

Relax three times on $A^{2h}e^{2h} = r^{2h}$ on Ω^{2h} with initial guess $e^{2h} = 0$.

Correct fine grid approximation: $v^h \leftarrow v^h + I_{2h}^h e^{2h}$.

The results of this calculation are given in Figs.4.20-4.24. The initial guess with its two modes is shown in Fig.4.20. In Fig.4.21 the approximation v^h after one relaxation sweep is superimposed on the initial guess. Much of the oscillatory component of the initial guess has already been removed. The captions show that the maximum norm of the error has decreased significantly. Fig.4.22 shows the approximation after three relaxation sweeps, again superimposed on the previous approximations. The solution (and in this case, the error) has become smooth and has a significantly reduced norm. Further relaxations on the fine grid would provide only a slow improvement at this point. It signals that it is time to move to the coarse grid. Fig.4.23 shows the fine grid error after one relaxation sweep on the coarse grid residual equation. It is the curve with the smallest amplitude (superimpopsed on previous approximations). Clearly we have achieved another large reduction in the error by moving to the coarse grid because the smooth error components, inherited from the fine grid, appear oscillatory on the coarse grid and are quickly removed. The error after three coarse grid relaxation sweeps is shown in Fig.4.24. The maximum norm of the error is now about 5% of its initial value.

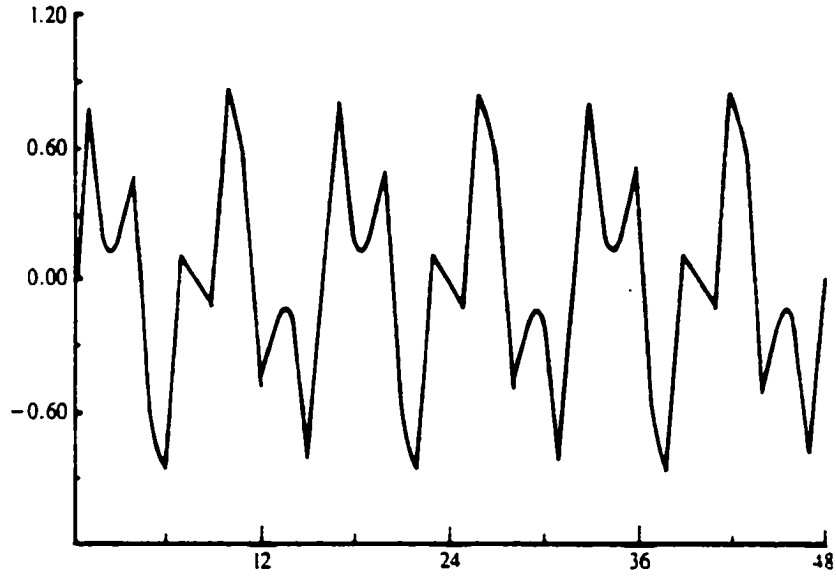


Fig.4.20 The initial guess $v_j = 1/2\left[\sin\left(\frac{12j\pi}{N}\right)+\sin\left(\frac{30j\pi}{N}\right)\right]$ with N = 48 for a coarse grid correction scheme applied with the weighted Jacobi iteration $(w=2/3)$. $\|e\|_\infty = 1.000$.

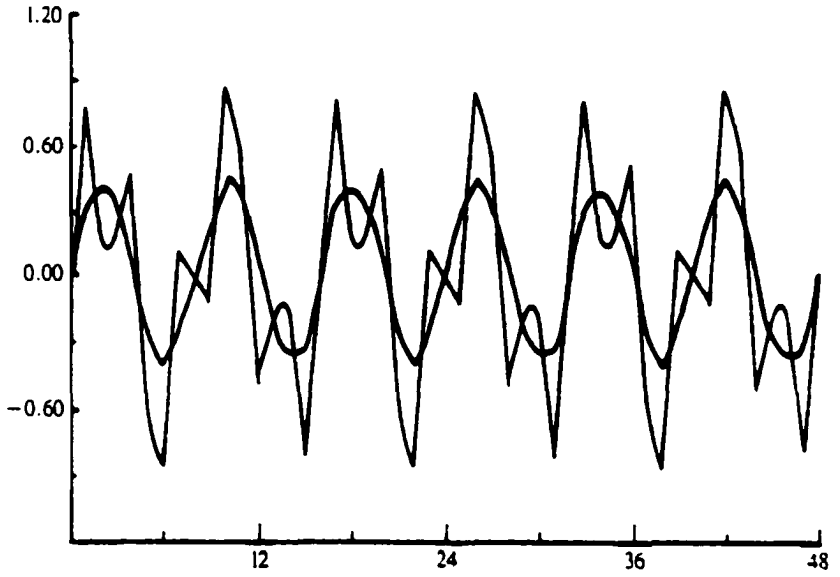


Fig.4.21 The approximation (and error) after one fine grid relaxation sweep superimposed upon the initial guess. $\|e\| = .430$.

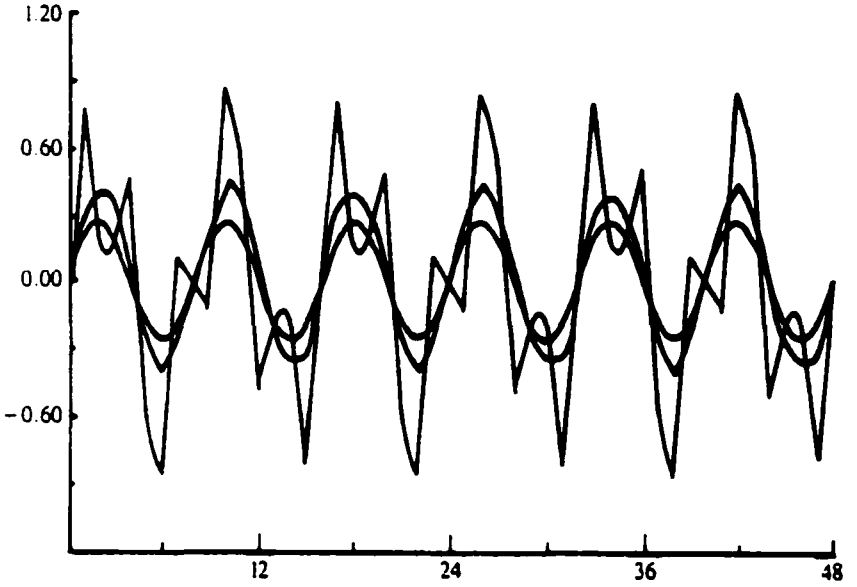


Fig.4.22 The error after three fine grid relaxation sweeps superimposed upon the previous iterates. $\|e\|_{\infty} = .261$.

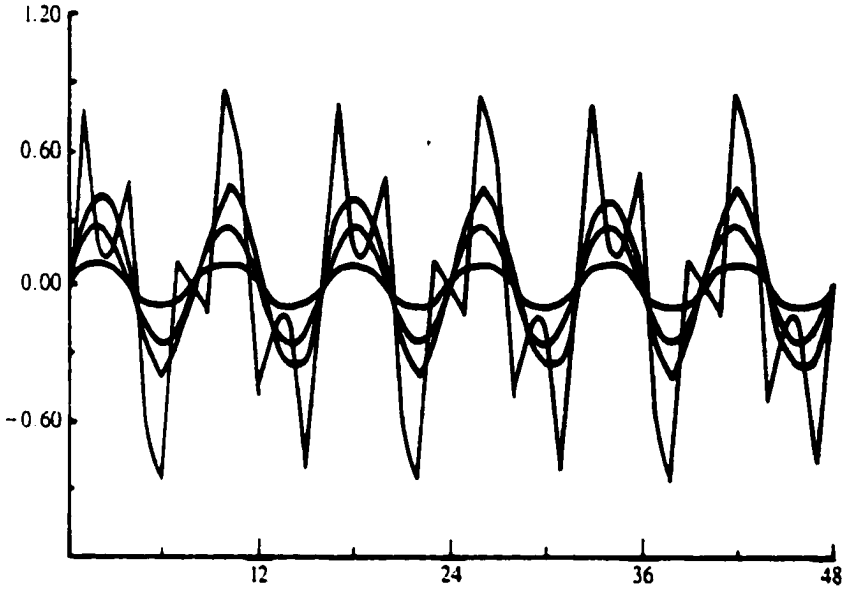


Fig.4.23 The fine grid residual is transferred to the coarse grid by full weighting. This figure shows the error after one relaxation sweep on the coarse grid residual equation, superimposed upon previous approximations. $\|e\|_\infty = .977 \times 10^{-1}$.

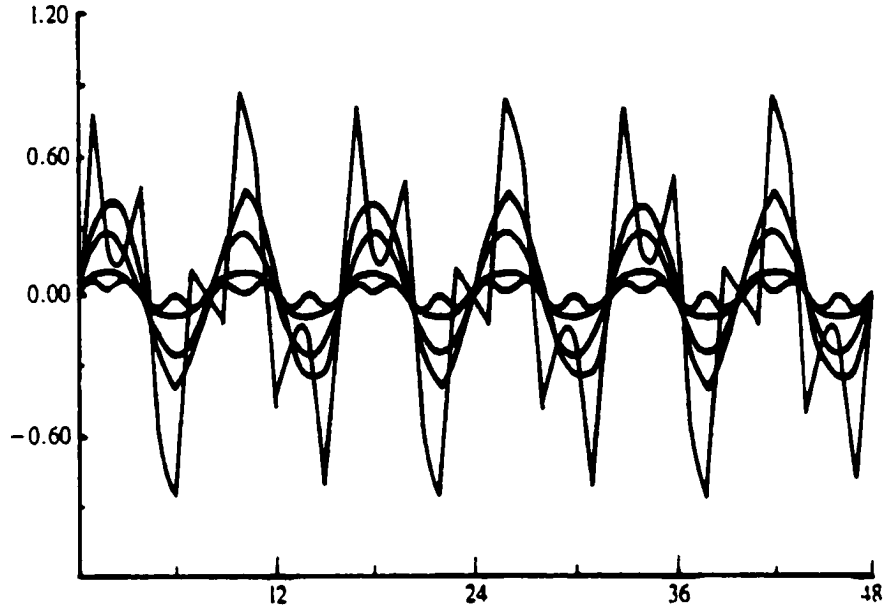


Fig.4.24 The error after relaxation sweeps on the coarse grid residual equation superimposed upon previous approximations. $\|e\|_\infty = .591\times10-1$

The coarse grid approximation to the error is now used to correct the fine grid approximation. After three additional fine grid relaxations, the error norm is reduced to $\|e\|_\infty = .20\times10^{-1}$. The residual is once again transferred to the coarse grid and three coarse grid relaxations follow. At this point the error norm is $\|e\|_\infty = .98\times10^{-2}$ or about 1% of its original value. This experiment demonstrates that relaxation, when done on two grids and applied to both the original and the residual equation, can be very powerful.

For the non-linear elliptic equation, and for any approximation v^h on a grid of meshsize h, the error $e^h = u^h - v^h$ satisfies

$$L^h\left(v^h + e^h\right) - L^h v^h = r^h \qquad (4.3.4.1)$$

where $r^h = f^h - L^h u^h$.

The error can be approximated by a coarse grid function $(v^{2h} + e^{2h})$ that satisfies

$$L^{2h}\left(I_h^{2h}v^h + e^{2h}\right) - L^{2h}\left(I_h^{2h}v^h\right) = I_h^{2h}r^h, \qquad (4.3.4.2)$$

Having obtained an approximation e^{2h} to Equation (4.3.4.2), it can be applied as a correction to the fine grid solution, thereby

$$v^h = v^h + I_{2h}^h e^{2h}$$

At every fine grid point the value of $I^h_{2h}\ e^{2h}$ is interpolated from values of e^{2h} at neighboring coarse grid points. The entire process of calculating $I^{2h}_h\ r^h$, solving Equation (4.3.4.2) and interpolating the residual correction is called coarse-grid correction.

The coarse grid correction scheme leaves one question unanswered: what is the best way to solve the coarse grid problem $A^{2h}e^{2h} = r^{2h}$? The answer is apparent particularly to those who think recursively. The coarse grid problem is not much different from the original problem. Therefore, we can apply the coarse grid correction scheme to the residual equation on Ω^{2h}, which means moving to Ω^{4h} for the correction step. We can repeat this process on successively coarser grids until a direct solution of the residual equation is possible.

4.3.5 A Recursive Formulation

The most elegant method for the basic multigrid algorithm is by means of a recursive formulation. Consider a sequence $\{ G^{nh} : n = 1,2,.....,N \}$ of increasingly coarser grids. The determination of N and the size of the coarsest grid, should be in accordance with the stencil size of the order of discretization that is used. For higher-

order discretization the stencil size increases, thereby limiting the size of the coarsest grid. Compact schemes have an advantage over traditional higher-order schemes in that a smaller stencil size is used to generate higher-order approximations. To facilitate the description of this procedure, the same notation will be used for the computer implementation of the resulting algorithm. We will call the right-hand side vector of the residual equation f^{2h}, rather than r^{2h}, since it is just another right-hand side vector. Instead of calling the solution of the residual equation e^{2h}, we will use v^{2h} since it is just a solution vector. These changes simplify the notation, but it is still important to remember the meaning of these variables.

Here is a coarse grid correction scheme imbedded within itself. We assume that there are grids with the coarsest grid spacing given by Lh, where $L = 2^{Q-1}$.

Relax on $A^h u^h = f^h$ ϑ_1 times with initial guess v^h.

Compute $f^{2h} = I_h^{2h}\left(f^h - A^h v^h\right)$.

Relax on $A^{2h}u^{2h} = f^{2h}$ ϑ_1 times with initial guess $v^{2h} = 0$.

Compute $f^{4h} = I_{2h}^{4h}\left(f^{2h} - A^{2h}v^{2h}\right)$.

Relax on $A^{4h}u^{4h} = f^{4h}$ ϑ_1 times with initial guess $v^{4h} = 0$.

Compute $f^{8h} = I_{4h}^{8h}\left(f^{4h} - A^{4h}v^{4h}\right)$.

.

.

Solve $A^{Lh}u^{Lh} = f^{Lh}$.

.

.

Correct $v^{4h} \leftarrow v^{4h} + I_{8h}^{4h}v^{8h}$.

Relax on $A^{4h}u^{4h} = f^{4h}$ ϑ_2 times with initial guess v^{4h}.

Correct $v^{2h} \leftarrow v^{2h} + I_{4h}^{2h}v^{4h}$. -

Relax on $A^{2h}u^{2h} = f^{2h}$ ϑ_2 times with initial guess v^{2h}.

Correct $v^h \leftarrow v^h + I_{2h}^h v^{2h}$. -

Relax on $A^h u^h = f^h$ ϑ_2 times with initial guess v^h.

For our non-linear problem, let u^{nh} be the set of grid functions on G^{nu}. Let there be given transfer operators P^{nh}: $u^{(n+1)h} \rightarrow u^{nh}$ (interpolation or prolongation) and R^{nh} : $u^{nh} \rightarrow u^{(n+1)h}$ (restriction). The problem to be solved on G^{nh} is given by

$$L^{nh}\left(u^{nh}\right) = b^{nh} \qquad (4.3.5.1)$$

The operator L^{nh} is non-linear. Let there be a smoothing algorithm defined on every grid, denoted by S (u,v,f,ν,nh). S changes an initial guess u^{nh} into an improved approximation v^{nh} with right-hand side f^{nh} by ν iterations using a smoothing method. For the non-linear elliptical equation the discretization method is described in Section 4.5. On the coarsest grid G^{Nh} we solve Equation (4.3.5.1) exactly and in general we write S$(u,v,f,\bullet,nh)$ for smoothing on the coarsest grid. The algorithm has a compact recursive definition. Let some approximation u^{nh} to the solution on G^{nh} be given. The recursive non-linear algorithm is as follows

Subroutine MG (*u,v,f,nh,Nh*)

Comment recursive non-linear multigrid algorithm

Begin

if (nh eq Nh) then

1 S(*u,v,f,•,nh*)

else

2 S(*u,v,f, v,nh*)

3 $r^{nh} = f^{nh} - L^{nh} v^{nh}$

4 $u^{(n+1)h} = R^{(n+1)h} v^{nh}$

5 $f^{(n+1)h} = L^{(n+1)h} (u^{(n+1)h}) + R^{(n+1)h} r^{nh}$

6 MG(*u,v,f,(n+1)h,Nh*)

7 $v^{nh} = v^{nh} + P^{nh} (v^{(n+1)h} - u^{(n+1)h})$

8 S(*u,v,f, v,nh*)

endif

end of MG

The explanation of the subroutine MG is as follows: Statement (1) represents smoothing or solving on the coarsest grid. Statement (2) performs ν smoothing iterations (pre-smoothing), starting with an initial guess u^{nh}. In (3), the residual r^{nh} is computed. r^{nh} will steer the coarse grid correction. The 'short wavelength accuracy' obtained in v^{nh} must be kept, and the correction δv^{nh} (containing 'long wavelength information') is added to v^{nh}. For the non-linear case, r^{nh} cannot be taken for the right-hand side of the problem for δv^{nh} i.e. $L(\delta v^{nh}) = r^{nh}$ might not even have a solution. Similarly, $R^{(n+1)h} r^{nh}$ cannot be the right-

hand side for the coarse grid problem on $G^{(n+1)h}$. Instead, it is added in line 5 to $L^{(n+1)h}$ $(u^{(n+1)h})$ with $u^{(n+1)h}$ as an approximation to the solution of (2). $L^{(n+1)h}$ $(v^{(n+1)h})$ = L^{nh} $(u^{(n+1)h})$ has a solution, and if $R^{(n+1)h}$ r^{nh} is not too large, then $L^{(n+1)h}$ $(v^{(n+1)h})$ = L^{nh} $(u^{(n+1)h})$ + $R^{(n+1)h}$ r^{nh} can be solved. $R^{(n+1)h}$ r^{nh} will be small when u^{nh} is close to the solution where the algorithm is close to convergence. Line (4) obtains the approximation $u^{(n+1)h}$ to the solution on $G^{(n+1)h}$. There are several possibilities for the choice of $u^{(n+1)h}$. One way is

$$u^{(n+1)h} = R^{(n+1)h} v^{nh}$$

where $R^{(n+1)h}$ is a restriction operator. Line (6) recursively calls MG with the size of the next coarsest grid. In line (7) the coarse grid correction is added to v^{nh} . Finally line (8) represents ν smoothing iterations (post-smoothing). The order in which the grids are visited is called the multigrid schedule or multigrid cycle. The algorithm described above telescopes down to the coarsest grid and then works its way back to the finest grid. Fig.4.25 shows the schedule for the grids in the order in which they are visited, Because of the pattern of the diagram, the algorithm is called the V_cycle.

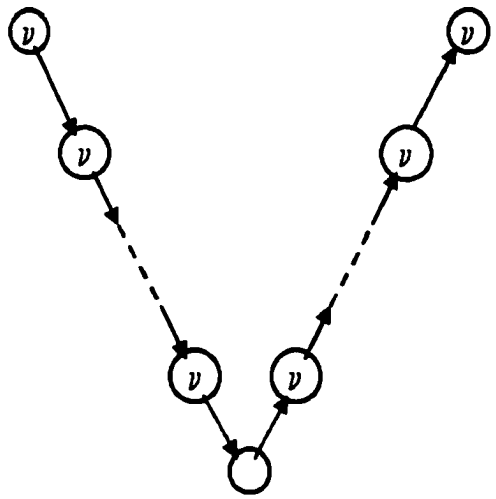


Fig.4.25 The V-Cycle.

We originally stated that two strategies would lead to multigrid. So far we have developed only the coarse grid correction idea. The nested iteration idea has yet to be explored. It uses coarse grids to obtain improved initial guesses for fine grid problems. In looking at the V-cycle, we might ask how to obtain an informed initial guess for the first fine grid relaxation. Nested iteration suggests solving a problem on Ω^{2h}. But how can we obtain a good initial guess for the Ω^{2h} problem? Nested iteration sends us to Ω^{4h}. Clearly we are on another recursive path that leads to the coarsest grid. The algorithm that joins the nested iteration with the V-cycle is called full multigrid (FMV) V-cycle. It appears as follows.

Full Multigrid V-Cycle

$v^h \leftarrow FMV^h\left(v^h, f^h\right)$

Initialize $f^h, f^{2h}, \ldots; v^h, v^{2h}, \ldots$ to zero.

Solve or relax on coarsest grid

.

.

$v^{4h} \leftarrow v^{4h} + I_{8h}^{4h} v^{8h}$

$v^{4h} \quad MV^{4h}\left(v^{4h}, f^{4h}\right)$

$v^{2h} \leftarrow v^{2h} + I_{4h}^{2h} v^{4h}$

$v^{2h} \leftarrow MV^{2h}\left(v^{2h}, f^{2h}\right)$

$v^h \leftarrow v^h + I_{2h}^h v^{2h}$

$v^h \leftarrow MV^h\left(v^h, f^h\right)$

Expressed recursively, the algorithm has the compact form:

$v^h \leftarrow FMV^h\left(v^h, f^h\right)$

1. If Ω^h = coarsest grid, then go to step 3.

Else $\quad f^{2h} = I_h^{2h}\left(f^h - A^h v^h\right)$

$v^h \leftarrow 0$

$v^{2h} \leftarrow FMV^{2h}\left(v^{2h}, f^{2h}\right)$

2. Correct $v^h \leftarrow v^h + I_{2h}^h v^{2h}$

3. $v^h \leftarrow MV^h\left(v^h, f^h\right)$ ϑ_0 times.

Fig.4.26 shows the scheduling of grids for FMV with $\vartheta_0 = 1$. Each V-cycle is preceded by a smaller V-cycle designed to provide the best initial guess possible. The extra work done in these preliminary V-cycles is not only inexpensive, but generally pays for itself. Full multigrid is a remarkable synthesis of ideas and techniques that individually have been well known and used for a long time. Taken alone, many of these ideas have serious defects. Full multigrid is a technique for integrating them so that they can work together in a way that removes these limitations. The result is a simple but a powerful algorithm.

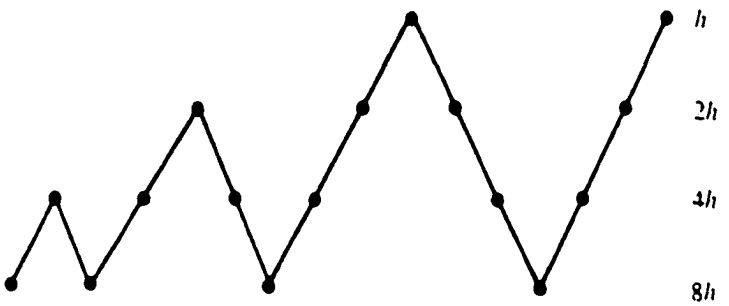


Fig.4.26 Schedule of grids for a FMV scheme on four levels.

4.4 Performance

We now discuss the practical issues of implementation, complexity, and performance of multigrid methods. Writing multigrid programs can be challenging. Many practitioners believe such programs should be highly modular. It allows them to evolve from very simple relaxation programs and makes them easier to check and debug [6]. Also, the various components of the program (relaxation, interpolation or restriction subroutines) can be replaced individually. Choosing an efficient data structure for a multigrid program is important. It is generally accepted that the solution and the right-hand side vectors on the various grids should be stored in contiguously in single arrays.

A typical data structure for a four level (N = 16) V-cycle applied to a one dimensional problem is shown in Fig.4.27. The figure also shows how the data structure changes as the V-cycle progresses. Each grid needs two arrays: one to hold the current approximations on each grid and the other to hold the right-hand side vectors on each grid. Since boundary values must also be stored, the coarsest grid involves three grid points (one interior and two boundary points). In general, the kth coarsest grid involves $2^k + 1$ points. Initially, the entire solution array v may be set to zero that will be the initial guess on each grid. The right-hand side array f will also be set to zero, except for the values of the right-hand side on the finest grid.

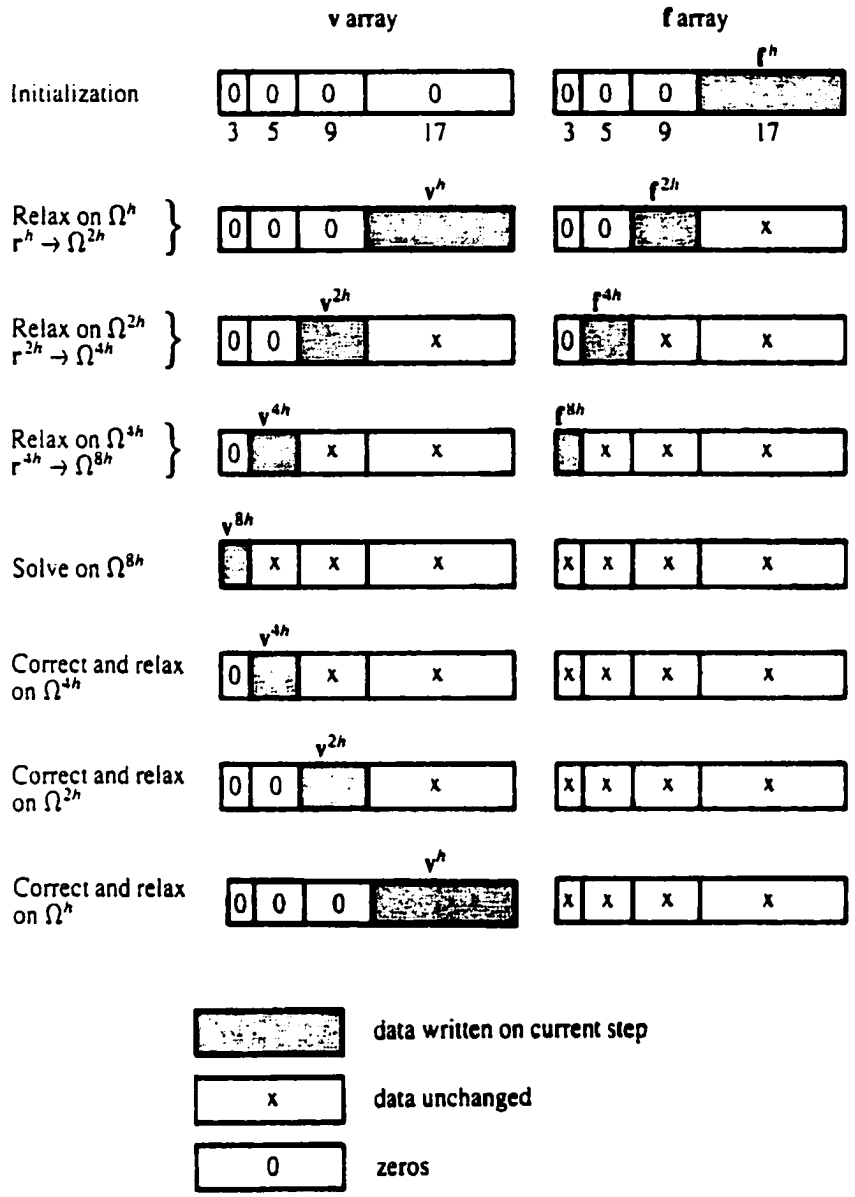


Fig.4.27 The course of s four level (N = 16) V_cycle is illustrated by showing changes in the data array. The V and f arrays hold solution vectors and the right hand side vectors, respectively, on the four grids.

As the V-cycle 'descends' into coarser grids, relaxation fills the segment of the solution array corresponding to each grid. At the same time, the residual transfers fill the segment of the right-hand side array corresponding to the next coarsest grid. As the V-cycle 'ascends' through finer grids, the right-hand side array does not change. However,

the colution array is over-written by additional relaxations on each level. Notice that when a new approximation is computed on one level, the approximation on the previous level is zeroed out. It provides a zero initial guess on each level in case another V-cycle is performed.

We now discuss the complexity. How much do multigrid schemes cost in terms of storage and arithmetic? For the storage, consider a d-dimensional grid with N^d points, where $N = 2^n$. We have just seen that two arrays must be stored on each level. The finest grid Ω^h requires $2N^d$ storage locations; Ω^{2h} has 2^{-d} times as many; Ω^{4h} has 4^{-d} times as many; Ω^{ph} has p^{-d} times as many. Adding these and using the sum of the geometric series as an upper bound gives us

$$Storage = 2N^d\left\{1 + 2^{-d} + 2^{-2d} + \cdots + 2^{-nd}\right\} < \frac{2N^d}{1-2^{-d}}.$$

In particular, for a one-dimensional problem, the storage requirement is less than twice that of the fine grid problem. For problems in two or more dimensions, the requirement drops to less than 4/3 of the fine grid problem alone. Thus, the storage costs of multigrid algorithms decrease relatively as the dimension of the problem increases. We can use a similar reasoning to estimate the computational cost of multigrid methods. A practical measure of the minimum computing work is defined as one work unit (WU) and is the amount of computing work required to evaluate the residual $L^{nh}(u^{nh})$ - b^{nh} on the finest grid G^h. One may also define one work unit as the work involved in one smoothing iteration on the finest grid G^h. Usually convergence histories are given in terms of work units to facilitate comparison between methods, and helps in improving multigrid codes. It is customary to neglect the cost of intergrid transfer operations which could amount to

15-20 percent of the entire cycle. First consider a V-cycle with one relaxation sweep on each level $(\nu_1 = \nu_2 = 1)$. Each level is visited twice and the grid Ω^{ph} requires p^{-d} of a work unit. Adding these costs and again using the geometric series for an upper bound gives

$$MVComputational\cos t = 2\left\{1 + 2^{-d} + 2^{-2d} + \cdots + 2^{-nd}\right\} < \frac{2}{1-2^{-d}} WU$$

A single V-cycle costs 4 WU for a one-dimensional problem. The cost is 8/3 WU for d =2 and 16/7 for d =3. We can also find the computational cost for an FMV cycle. Assume again that one relaxation sweep is done on each level. A full V-cycle beginning from Ω^h costs $2\left(1-2^{-d}\right)^{-1}$ WU. A V-cycle beginning from Ω^{2h} costs 2^{-d} of a full V –cycle. In general, a V-cycle beginning from Ω^{ph} costs p^{-d} of a full V –cycle. Adding these costs gives us

$$FMVComputational\cos t = \left(\frac{2}{1-2^{-d}}\right)\left\{1 + 2^{-d} + 2^{-2d} + \cdots + 2^{-nd}\right\} < \frac{2}{\left(1-2^{-d}\right)^2} WU$$

An FMV costs 8 WU for a one-dimensional problem; the cost is about 7/2 WU for d =2 and 5/2 for d = 3. As expected, a single FMV cycle costs more than a single V-cycle, although the discrepancy is less for higher-dimensional problems.

Now, let us analyze the space and time complexity of the multigrid algorithm MG. Let the finest grid be G^h and the coarse grids G^{mh} : $m = 2,.....,M$ be constructed by successive doubling of the mesh-sizes (standard coarsening). The number of grid-points N_m of G^{mh} is given by

$$N_m = \prod_{\alpha=1}^{d}(1 + n_\alpha)$$

where, n_α is the number of points along each dimension (d).

Writing n_α as $n_\alpha^{(m)} = p_\alpha \bullet 2^m$, the above equation becomes

$$N_m = \prod_{\alpha=1}^{d} \left(1 + p_\alpha 2^m\right) \cong P 2^{md}$$

$$\text{with} \qquad P = \prod_{\alpha=1}^{d} p_\alpha$$

It is expected that the amount of storage required for the computation on the grid G^{mh} is given by $c_1 N_m$, where c_1 is some constant independent of h. Therefore, the total amount of storage required is

$$c_1 \sum_{m=1}^{M} N_m \cong \frac{2^d}{2^d - 1} c_1 N_m$$

In comparison with single grid solutions on G^{mh} with the prescribed smoothing method, the use of multigrid increases the storage requirement by a factor of $2^d/(2^d - 1)$, which is 4/3 in two and 8/7 in three dimensions. The additional storage required by multigrid is modest. Let us estimate the computational work of one multigrid iteration based on the non-linear multigrid algorithm (V-cycle). An approximation to the computational work performed on G^{mh} will be $w_m = c_1 N_m$, assuming that the pre- and post- smoothings are independent of m. Let us define w_m to be all computing work involved in MG, except the recursive call of MG. Let W_m be all the work involved in MG which includes the recursive call in MG. We can then write

$$W_m = c_2 P 2^{md} + W_{m-1} \text{ , thereby}$$

$$W_m = c_2 P 2^{md} \left(1 + (2^{-d} + (2^{-2d} + \ldots + 2^{(1-M)d}) \ldots) \right)$$

$$= c_2 N_m \left(1 + \gamma + \gamma^2 + \ldots + \gamma^{M-1} \right)$$

with $\gamma = 1/2^d$. Hence, it follows that

$$W_m / (c_2 N_m) = (1 - \gamma^M) / (1 - \gamma)$$

Since $(W_m / (c_2 N_m))$ is the ratio of multigrid work and work on the finest grid, the following conclusion may be drawn. The bulk of the work on the finest grid consists of smoothing. Hence $(W_m / (c_2 N_m)) - 1$ is a measure of the additional work required to accelerate smoothing on the finest grid G^h by using multigrid.

Let us discuss the convergence of multigrid methods. We will give heuristic suggesting that the standard multigrid schemes, when applied to well behaved problem (for example symmetric, positive definite systems), not only work, but they work very effectively. Convergence results for such problems can be proved quite rigorously. Where analytical results are lacking, a wealth of computational evidence testifies to the general effectiveness of multigrid methods. Between analysis and experimentation, the multigrid area is slowly being mapped. However, multigrid convergence analysis is still a wild and open area in computational mathematics.

As we have seen, the smoothing rate (the convergence factor for the oscillatory modes) for standard relaxation schemes is small and independent of the grid spacing h. Also, for any given grid, we can apply the coarse grid correction scheme recursively to focus relaxation on the oscillatory modes because the smooth error modes which remain after relaxation on one grid appear more oscillatory on the coarser grids. Therefore, all the error components on the original fine grid eventually appear oscillatory and are

reduced quickly by relaxation. It then follows that the overall convergence factor for a good multigrid scheme should be small and independent of h.

The ideal computing method to approximate the behavior of a given physical problem involves an amount of computing work proportional to the number and size of the physical changes that are modeling. This rule has been put forward as the 'golden rule' by Brandt (1982).

4.5 Discretization Method

Consider a uniform rectangular grid of size $(N+1)$ X $(M+1)$ defined as

$$\xi_{ij} = \xi_i = i/N, \quad \eta_{ij} = \eta_i = j/M, \; i = 0..N; \; j = 0..M$$

Let $\mathbf{x}_{ij}$ be prescribed on the boundary of this grid. We are to compute $\mathbf{x}_{ij}$ in the interior of the computational grid based on the solution of the Poisson system defined by Equation (2.8.26). The solution of this system of nonlinear elliptical equations is obtained by Picard iteration.

$$P^{k-1}\mathbf{x}^k_{\xi\xi} + 2Q^{k-1}\mathbf{x}^k_{\xi\eta} + R^{k-1}\mathbf{x}^k_{\eta\eta} + S^{k-1}\mathbf{x}^k_{\xi} + T^{k-1}\mathbf{x}^k_{\eta} = 0,$$

$$\text{where, } P^{k-1} = (\mathbf{x}^{k-1}_{\eta}, \mathbf{x}^{k-1}_{\eta}), \; Q^{k-1} = -(\mathbf{x}^{k-1}_{\xi}, \mathbf{x}^{k-1}_{\eta}),$$

$$R^{k-1} = (\mathbf{x}^{k-1}_{\xi}, \mathbf{x}^{k-1}_{\xi}),$$

$$S^{k-1} = P^{k-1} P^1_{11} + 2Q^{k-1} P^1_{12} + R^{k-1} P^1_{22},$$

$$T^{k-1} = P^{k-1} P^2_{11} + 2Q^{k-1} P^2_{12} + R^{k-1} P^2_{22}.$$

The six control functions $P^1_{11}, P^1_{12}, P^1_{22}, P^2_{11}, P^2_{12}, P^2_{22}$ are computed according to the equations given by (2.8.27) and by applying higher-order compact schemes for the discretizations of $s_{\xi\xi}, s_{\eta\eta}, s_{\xi\eta}, t_{\xi\xi}, t_{\eta\eta}, t_{\xi\eta}, s_{\xi}, s_{\eta}, t_{\xi}, t_{\eta}$. The arclength normalized variables (s_{ij}, t_{ij}) at the boundary are computed as follows :

- Compute the distance (l) between succeeding points at the each boundary.
- Define the length of the edge (L) along each boundary as the sum of the distances between succeeding points along that boundary.
- The normalized distance (d) between each succeeding point is then given by l/L.

- The arclength normalized variables variables s_{ij} and t_{ij} at the boundary are defined by

$$s_{0j} = 0, \quad s_{Nj} = 1, \quad j = 0 \ldots\ldots M$$

$$t_{i0} = 0, \quad t_{iM} = 1, \quad i = 0 \ldots\ldots N$$

and

$$s_{i0} = s_{i-1,0} + d_{i0}, \quad s_{iM} = s_{i-1,M} + d_{iM}, \quad i = 0 \ldots N$$

$$t_{0j} = t_{0,j-1} + d_{oj}, \quad t_{Nj} = t_{N,j-1} + d_{Nj}, \quad j = 0 \ldots M$$

The arclength normalized variables (s_{ij} , t_{ij}) in the interior of the grid are computed according to the algebraic straight line transformation given by Equations (2.8.29) and (2.8.30). Simultaneously solving the two linear algebraic equations yields (s_{ij} , t_{ij}) .

$$s_{ij} = s_{i0}(1 - t_{ij}) + s_{iM} t_{ij},$$

$$t_{ij} = t_{i0}(1 - s_{ij}) + t_{Nj} s_{ij},$$

at each node (i , j) $\in$ ($1 \ldots N-1; 1 \ldots\ldots M-1$).

The steps for an iteration to improve the current approximation $\mathbf{x}^{k-1}$ is as follows:

- The coefficients P^{k-1}, Q^{k-1}, R^{k-1}, S^{k-1}, T^{k-1} are computed by applying higher-order compact schemes for the discretization of $\mathbf{x}^{k-1}{}_{\xi}$ and $\mathbf{x}^{k-1}{}_{\eta}$. The six control functions remain unchanged during the iterative process.

- Using higher-order compact schemes discretize $\mathbf{x}^{k}{}_{\xi\xi}$, $\mathbf{x}^{k}{}_{\eta\eta}$, $\mathbf{x}^{k}{}_{\xi}$, $\mathbf{x}^{k}{}_{\eta}$. The discretization of the mixed derivative using higher-order compact schemes is done in a way described in [8].

• We then arrive at a linear system of equations for the unknowns $\mathbf{x}^k_{ij}$, $i = 0 \ldots\ldots N; j = 0 \ldots\ldots M$ with Dirichlet boundary conditions. This linear system is solved by a Multigrid solver. The solver is called twice to compute the two components x^k_{ij} and y^k_{ij} of $\mathbf{x}^k_{ij}$.

The initial approximation $\mathbf{x}^0$ is obtained by algebraic grid generation. The above iterative process is repeated until a required approximation to the solution has been obtained.

4.6 Illustrations

An example of a grid in 2D domain is shown in Figs.4.28-4.32. The figure shows the region around an airfoil. The grid generated is grid-folding free and the interior grid point distribution is a good reflection of the prescribed boundary grid point distribution (See Fig.4.32). The initial grid is obtained with algebraic grid generation and is required as the initial solution for the non-linear elliptical Poisson system. The final grid is independent of the initial grid. The quality of the initial grid is unimportant and severe grid folding of the initial grid is allowed. The grid is generated using the non-linear multigrid solver using fourth-order compact scheme (Pade scheme).

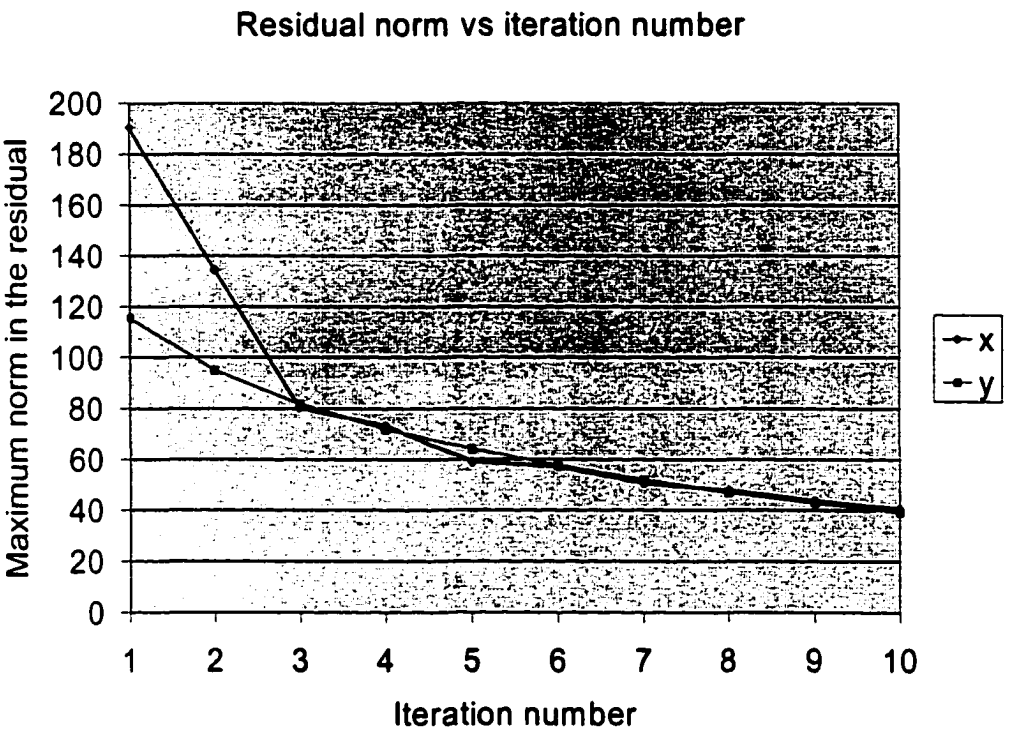


Fig.4.28 Maximum norms for the iterative method.

Figs.4.28 and 4.29 compare the maximum norm in the residual of the iterative and the multigrid method for the first ten iterations. As seen in the figures, the drop in the maximum norm in the residual per iteration using the multigrid method is more pronounced compared with that in the iterative method.

Fig.4.30 shows the convergence times (wall times) of the iterative and the multigrid method. From the figure, the multigrid method is approximately five times faster than the iterative method for the non-linear elliptical grid-generation problem under consideration.

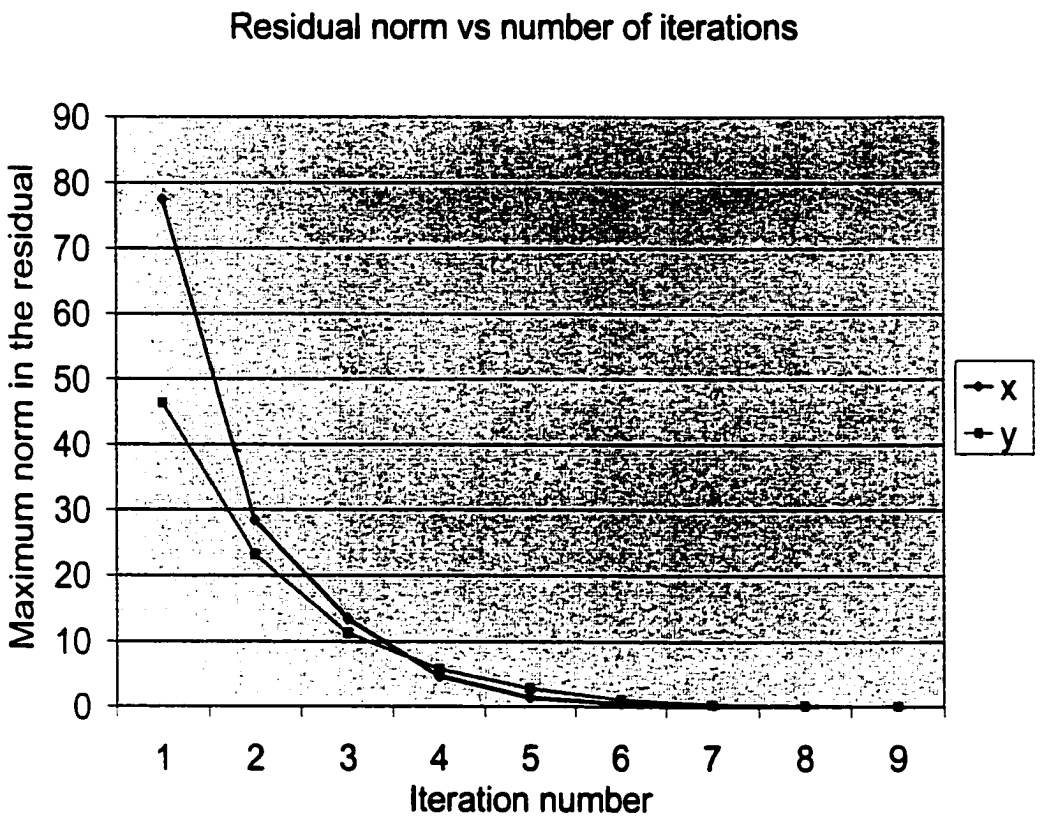


Fig.4.29 Maximum norms for the multigrid method.

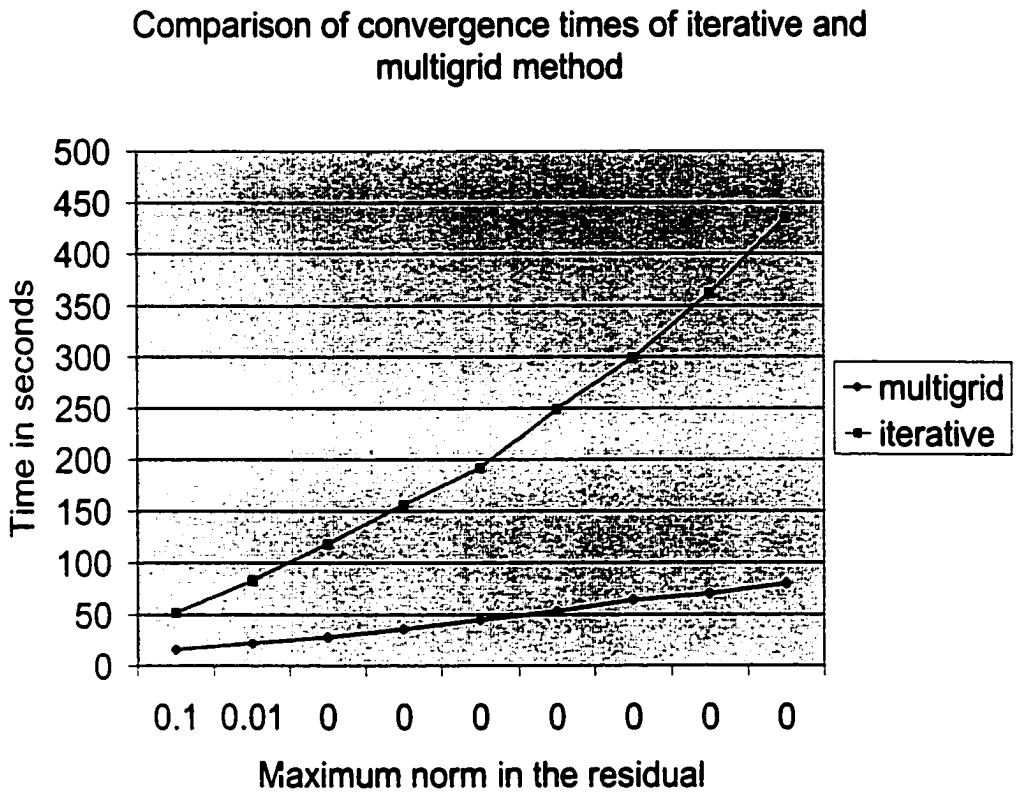


Fig.4.30 Convergence times of iterative vs. multigrid method.

Fig.4.31 shows the convergence of the multigrid method in terms of work units. This can be used for comparison with the other methods.

4.7 Conclusions

Multigrid methods are fast-converging methods when applied to elliptic partial differential equations. As seen from the illustrations, convergence of the two dimensional elliptic grid generation problem is vastly improved by using the multigrid solver. Compact finite difference schemes when used for discretizing the grid generation equations are close to the exact differentiation and provide better grid point distribution. Convergence of iterative methods for three dimensional elliptic grid generation is extremely slow. Future work is in application of the multigrid method for three dimensional elliptic grid generation.

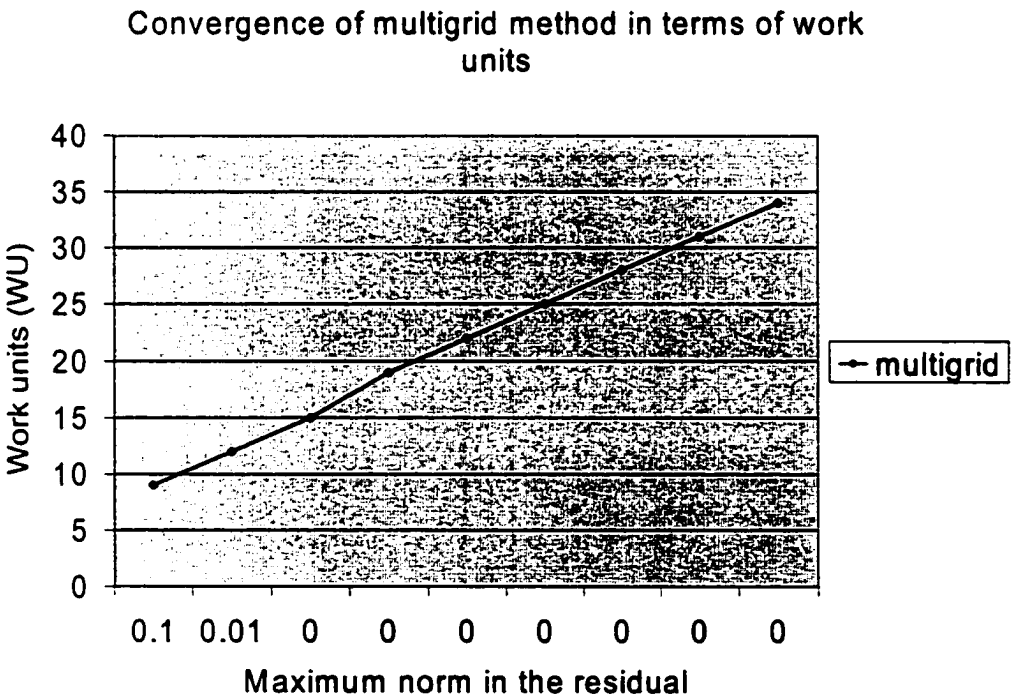


Fig.4.31 Convergence of mutigrid in Work units.

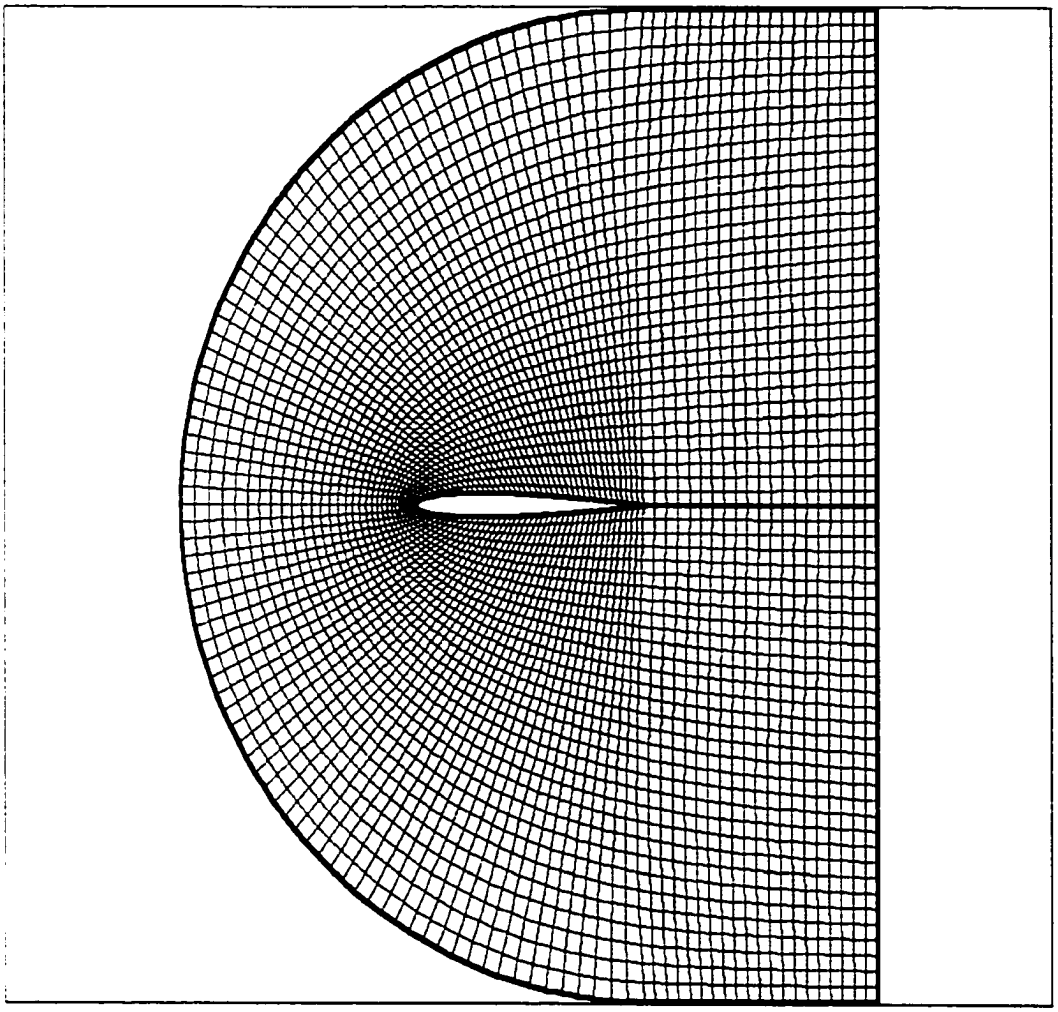


Fig.4.32 Grid generated near an airfoil.

REFERENCES

[1] J.F. Thompson, Z.U.A. Warsi, and C.W. Mastin, *Numerical Grid Generation: Foundations and pplications* (Elsevier,New York,1985).

[2] Z.U.A. Warsi, in *Proceedings, Numerical Grid Generation,* edited by J.F. Thompson, (North-Holland, Amsterdam, 1982), p.41.

[3] S.P. Spekreuse, *Elliptic Grid Generation Based on Laplace Equations and Algebraic Transformations* (Journal of Computational Physics 118, 38-61, 1995).

[4] Sanjiva K.Lele, *Compact finite difference Schemes with Spectral-like resolution* Journal of omputational Physics 103,16-42, 1992).

[5] Achi Brandt *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics* (Department of Applied Mathematics, The Weizmann Institute of Science, Israel)

[6] P. Wesseling *An Introduction To Multigrid Methods* (Delft University of Technology, The Netherlands, 1991)

[7] William L. Briggs *A Multigrid Tutorial* (Society for Industrial and Applied Mathematics. Philadelphia, Pensylvania 1987)

[8] S.R. Chakravarthy, K.Y. Szema, U.C. Goldberg, J.J. Gorski, and S.Osher, in *Proceedings, AIAA 23rd Aerospace Science Meeting, 1985,* AIAA 85-0165

[9] N.N.Mansour, P. Moin, W.C. Reynolds, and J.H. Ferziger, *Turbulent Shear Flows I* (Springer-Verlag, New York/Berlin, 1977), p. 386.

[10] Warsi, Z. U. A., "*Basic Differential Models for Coordinate Generation*", Numerical Grid Generation, Ed. Joe F. Thompson, North-Holland, 41, 1982.

[11] Thompson, Joe F., 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, 47, 1, 1982.

[12] Thompson, Joe F., "*Grid Generation Techniques in Computational Fluid Dynamics*", AIAA Journal, 22, 1505, 1984.

[13] Mastin, C. Wayne and Thompson, Joe F., "*Elliptic Systems and Numerical Transformations*", Journal of Mathematical Analysis and Applications, 62, 52, 1978.

[14] Mastin, C. Wayne and Thompson, Joe F., "*Transformationof Three-Dimensional Regions onto Rectangular Regions by Elliptic Systems*", Numerische Mathematik, 29, 397, 1978.

[15] Thompson, Joe F. (Ed.) *Numerical Grid Generation*, North-Holland 1982. (Also published as Vol. 10 11 of Applied Mathematics and Computation, 1982).

[16] C.W Mastin and J.F. Thompson, *Numer. Math.* 29, 397 (1978)

[17] E.Kreyszig, *Introduction to Differential Geometry and Riemannian Geometry*, Mathematical Expositions No. 16 (University of Toronto Press, Toronto, 1967).

[18] T. Sonar, *Grid Generation Using Elliptic Partial Differential Equations*, 1989 (DFVLR-Forschungsbericht 89-15).

[19] J.W. Boerstoel, S.P Spekreijse, P.L. Vitaliano, in *Proceedings, AIAA 10th Applied Aerodynamics Conference*, 1992, AIAA 92-2619.