A STUDY OF MULTIGRID METHODS WITH APPLICATION TO CONVECTION-DIFFUSION PROBLEMS

A dissertation submitted to the University of Manchester Institute of Science and Technology for the degree of Master of Science

April 1998

By

D A F I K

Department of Mathematics

For my wife, LIS, and son, OZI

Contents

List of Tables **vi**

List of Figures **viii**

Abstract **ix**

Declaration **x**

Acknowledgements **xi**

1 Introduction **1**

1.1 The Convection-Diffusion Problem 3

1.2 The Galerkin Weak Formulation 4

1.3 The Finite Element Formulation 6

2 Stationary Iterative Methods **10**

2.1 The Jacobi and Gauss-Seidel Methods 10

2.2 The Damped Jacobi and Symmetric Gauss-Seidel Methods 16

3 Multigrid Methods **20**

3.1 The Multigrid Idea 20

3.1.1 Prolongation or Interpolation 22

3.1.2 Restriction 23

3.2 The Multigrid Algorithm 24

3.2.1 The Basic Two-Grid Algorithm 24

3.2.2 Multigrid 25

3.3 Complexity and Convergence Analysis 27

3.3.1 Complexity 27

3.3.2 Convergence analysis 28

3.4 The Effectiveness of the Relaxation Methods 30

4 Numerical Results **35**

4.1 Numerical Solution of the Diffusion Problem 36

4.1.1 Convergence Rate of Simple Relaxation Methods 37

4.1.2 Convergence Rate of a Representative Multigrid Met-hod . 39

4.2 Numerical Solution of the Convection-Diffusion Problem 42

4.2.1 Convergence Rate of Relaxation Methods 46

4.2.2 Convergence Rate of Multigrid Methods 48

4.3 Conclusions 52

A The V-CYCLE **Multigrid Program** **53**

List of Tables

4.1 General convergence for Gauss-Seidel method. 37

4.2 General convergence for damped Jacobi method when $\omega = 4/5$. . 38

4.3 General convergence for symmetric Gauss-Seidel method. 38

4.4 General convergence for V–Cycle method with Gauss-Seidel as smoother. 40

4.5 General convergence for V–Cycle method with damped Jacobi as smoother when $\omega = 4/5$. 40

4.6 General convergence for V–Cycle method with symmetric Gauss-Seidel as smoother. 41

4.7 Convergence rate of symmetric Gauss-Seidel method applied to the problem with viscosity parameter 1/10. 46

4.8 Convergence rate of the symmetric Gauss-Seidel method applied to the problem with viscosity parameter 1/100. 47

4.9 Convergence rate of symmetric Gauss-Seidel method applied to the problem with viscosity parameter 1/1000. 47

4.10 Convergence rate of V-Cycle method applied to the convection-diffusion problem with viscosity parameter 1/10. 49

4.11 Convergence rate of V-Cycle method applied to the convection-diffusion problem with viscosity parameter 1/100. 49

4.12 Convergence rate of V-Cycle method applied to the convection-diffusion problem with viscosity parameter 1/1000. 50

4.13 Convergence rate of V-Cycle method applied to the convection-diffusion problem with "optimal" restriction coefficient. 51

List of Figures

1.1 Rectangular Element. 7

3.1 Grids schedule for a V-Cycle with five levels. 26

4.1 The diffusion problem solution for a (64×64) uniform grid with tolerance $\epsilon =$`1e-6`. 36

4.2 The convergence history of the relaxation methods when the tolerance $\epsilon =$`1e-6`. 41

4.3 The convergence history of V-Cycle method with Gauss-Seidel, damped Jacobi, and symmetric Gauss-Seidel as smoothers when the tolerance $\epsilon =$`1e-6`. 42

4.4 The solution for the viscosity parameter 1/10. 43

4.5 The solution for the viscosity parameter 1/100. 44

4.6 The solution for the viscosity parameter 1/1000. 44

4.7 The convergence history of symmetric Gauss-Seidel applied in the convection-diffusion problem without upwinding. 48

4.8 The convergence history of symmetric Gauss-Seidel ($\epsilon =$`1e-6`) applied in the convection-diffusion problem. 50

4.9 The convergence history of V–Cycle method with symmetric Gauss-Seidel as smoother (ϵ =`1e-6`) applied in the convection-diffusion problem. 51

Abstract

The work accomplished in this dissertation is concerned with the numerical solution of linear elliptic partial differential equations in two dimensions, in particular modelling diffusion and convection-diffusion. The finite element method applied to this type of problem gives rise to a linear system of equations of the form $Ax = b$. It is well known that direct and classical iterative (or relaxation) methods can be used to solve such systems of equations, but the efficiency deteriorates in the limit of a highly refined grid. The multigrid methodologies discussed in this dissertation evolved from attempts to correct the limitations of the conventional solution methods. The aim of the project is to investigate the performance of multigrid methods for diffusion problems, and to explore the potential of multigrid in cases where convection dominates.

Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other University or other institution of learning.

Acknowledgements

Throughout the preparation of this dissertation, I have benefited from the kind assistance of my supervisor Dr. D.J. Silvester. I would like to take this opportunity to express my gratitude for all the help and guidance I have been given. Also I would like to thank Dr. David Kay for his help in mastering the Finite Element Method.

I would also like to thank all my family, especially my parents, my wife and my sponsor (DUE Project of Jember University) who have always encouraged my academic endeavours and have been a constant source of inspiration.

My thanks are also due to the academic and secretarial staff here at the University of Manchester Institute of Science and Technology and also at the Victoria University of Manchester for all their help.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this project we consider the numerical solution of linear elliptic partial differential equations in two dimensions, in particular modelling *diffusion* and *convection-diffusion*. We will use the finite element discretisation method, see Johnson [3], for details, which generates a system of linear equations of the form $Ax = b$, where A is a sparse, positive definite matrix (and is symmetric in the diffusion case). It is well known that such systems of equations can be solved efficiently using direct and iterative (or relaxation) methods. In the diffusion case the preconditioned conjugate gradient method is another possibility.

Direct methods, of which Gaussian elimination is the prototype, determine a solution exactly (up to machine precision and assuming a perfectly conditioned matrix) in a finite number of arithmetic steps. They are often based on the fast Fourier transform or the method of cyclic reduction. When applied to PDE problems discretised on an $N \times N$ 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 rather specialised and can be applied primarily to a system which arises from separable self-adjoint boundary value problems.

Relaxation methods, as represented by the Jacobi and Gauss-Seidel iterations, begin with an initial guess at solution. They then proceed to improve the current approximation by a succession of simple updating steps or iterations. The sequence of approximations that 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 also suffer from *some disabling limitations*. Multigrid methods evolved from attempts to correct these limitations. These attempts have been largely successful; used in a multigrid setting, relaxation methods are competitive with the fast direct methods when applied to the model problems.

The basic multigrid methods also 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.

There is actually no simple answer to the question of whether direct or iterative methods are ultimately superior. Since it depends upon the structure of the problem and type of computer being used. We should at least consider the grid size, the number of iterations taken to converge, the approximate work units, and also the computer CPU times, to assess whether one method is more effective than the others. These factors are also to be an indicator in determining the efficiency of those methods.

The basic multigrid methods are certainly not confined to finite difference formulations. In fact, finite element discretisation are often more natural, particularly for analysis of these methods. Herein, we will use the finite element method to discretise the model diffusion and convection-diffusion problems, and then solve the resulting discrete system using multigrid methods. The effectiveness and efficiency will be compared with classical relaxation methods.

1.1 The Convection-Diffusion Problem

The general test problem that we consider is the convection-diffusion problem in a region $\Omega \subset \mathbb{R}^2$

$$-\mu \nabla^2 u + \mathbf{w} \cdot \nabla u = f \quad \text{in} \quad \Omega, \tag{1.1}$$

where μ is viscosity parameter, $-\nabla^2 u$ represents diffusion (defined by $-(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2})$) and $\mathbf{w} \cdot \nabla u$ represents convection (and is given by $\omega_x \frac{\partial u}{\partial x} + \omega_y \frac{\partial u}{\partial y}$), see Morton [1], for details. The conditions on the boundary of Ω are defined by

$$\begin{aligned} u &= 0 \quad \text{on} \quad \partial\Omega_d \neq \emptyset \\ \frac{\partial u}{\partial n} &= 0 \quad \text{on} \quad \partial\Omega_n, \end{aligned}$$

where $\frac{\partial u}{\partial n}$ is the normal derivative, and Ω is the region with the boundary $\partial\Omega_d \cup \partial\Omega_n = \partial\Omega$ and $\partial\Omega_d \cap \partial\Omega_n = \emptyset$. The boundary conditions $u = 0$ are known as the Dirichlet boundary conditions, and $\frac{\partial u}{\partial n} = 0$ as Neumann conditions. We wish to find an approximation to the solution u of this problem. To do this we will use the Galerkin method with a specific choice of basis functions.

1.2 The Galerkin Weak Formulation

The weak formulation of equation (1.1) is to find $u \in V$ such that

$$a_1(u,v) + a_2(u,v) = (f,v) \quad \forall v \in V,$$

where

$$\begin{aligned} a_1(u,v) &= \int_\Omega \mu \nabla u \cdot \nabla v \, d\Omega \\ a_2(u,v) &= \int_\Omega (\mathbf{w} \cdot \nabla u) v \, d\Omega \\ (f,v) &= \int_\Omega f v \, d\Omega, \end{aligned}$$

and

$$V = \{\phi | \phi \in H^1(\Omega), \phi = 0 \text{ on } \partial\Omega_d\},$$

where the Sobolev space $H^1(\Omega)$ is the space of functions with square integrable first derivatives.

To find an approximation to u, we choose an m dimensional subspace V_m of V given by

$$V_m = \{v | v = \sum_{i=1}^{m} \alpha_i \phi_i, \ \alpha_i \in \mathbb{R}, \phi_i \in V\} \quad i = 1, 2, \ldots, m.$$

The Galerkin approximation u_m from the subspace V_m satisfies

$$a_1(u_m, v) + a_2(u_m, v) = (f, v) \quad \forall v \in V_m. \tag{1.2}$$

Since $u_m \in V_m$ we may write u_m in the form

$$u_m = \sum_{i=1}^{m} \alpha_i \phi_i, \quad \alpha_i \in \mathbb{R}, \tag{1.3}$$

and thus we need to find the $\alpha_i \in \mathbb{R}$ for $i = 1, 2, \ldots, m$. Since $\{\phi_j\}_{j=1}^{m}$ are the basis functions for V_m, we need only to test over $\phi_j, j = 1, 2, \ldots, m$ in equation

(1.2). Therefore, the equation (1.2) is equivalent to finding $u_m \in V_m$ such that

$$a_1(u_m, \phi_j) + a_2(u_m, \phi_j) = (f, \phi_j), \quad \forall j = 1, 2, \ldots, m. \tag{1.4}$$

Substituting (1.3) into (1.4) gives

$$a_1(\sum_{i=1}^{m} \alpha_i \phi_i, \phi_j) + a_2(\sum_{i=1}^{m} \alpha_i \phi_i, \phi_j) = (f, \phi_j),$$

and recalling the weak formulation we have

$$\int_\Omega \mu \left(\sum_{i=1}^{m} \alpha_i \nabla \phi_i \cdot \nabla \phi_j \right) d\Omega + \int_\Omega \left(\mathbf{w} \cdot \sum_{i=1}^{m} \alpha_i \nabla \phi_i \right) \phi_j \, d\Omega = \int_\Omega f \phi_j \, d\Omega.$$

We denote

$$\begin{aligned} A_{ij}^{(1)} &= \int_\Omega \mu \nabla \phi_i \cdot \nabla \phi_j \, d\Omega \\ A_{ij}^{(2)} &= \int_\Omega (\mathbf{w} \cdot \nabla \phi_i) \phi_j \, d\Omega \\ b_j &= \int_\Omega f \phi_j \, d\Omega, \end{aligned}$$

where $A^{(1)}$, $A^{(2)}$ are matrices which arise from the diffusion and convection terms respectively, and b is called the force vector and

$$\begin{aligned} \mu \nabla \phi_i \cdot \nabla \phi_j &= \mu \left(\frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial y} \frac{\partial \phi_j}{\partial y} \right) \\ (\mathbf{w} \cdot \nabla \phi_i) \phi_j &= w_x \frac{\partial \phi_i}{\partial x} \phi_j + w_y \frac{\partial \phi_i}{\partial y} \phi_j. \end{aligned}$$

The resulting linear system is of the form

$$A\alpha = b,$$

where A is an $m \times m$ matrix defined by $A := A^{(1)} + A^{(2)}$.

1.3 The Finite Element Formulation

The basic idea of the finite element method is to use a piecewise polynomial approximation. The solution process, referring to Johnson [3], can be divided into steps as follows

1. Split the region of interest into rectangular (or triangular) elements K.
2. Define a local polynomial (for approximating the solution) within each element.
3. Define a set of nodes in each element.
4. Define the local polynomials in terms of the nodal values.
5. Satisfy the essential boundary conditions.
6. Solve the resulting discrete set of algebraic equations.

Since a bilinear function, in x and y is of the form

$$\psi(x,y) = p + qx + ry + sxy \tag{1.5}$$

we see that there are four unknown parameters which can be uniquely determined by values at four nodes $(x_1,y_1),(x_2,y_2),(x_3,y_3),(x_4,y_4)$, (no more than two in a straight line). The bilinear function can then be written in the general form

$$\psi(x,y) = \phi_1(x,y)\psi_1 + \phi_2(x,y)\psi_2 + \phi_3(x,y)\psi_3 + \phi_4(x,y)\psi_4,$$

where the ϕ_i's are the nodal basis function satisfying

$$\phi_j(x_i,y_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j, \quad i,j = 1,\ldots,m, \end{cases}$$

that is $\phi_i(x, y) = \hat{p} + \hat{q}x + \hat{r}y + \hat{s}xy$. These functions are obtained by the solution of

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ 1 & x_3 & y_3 & x_3y_3 \\ 1 & x_4 & y_4 & x_4y_4 \end{bmatrix} \begin{bmatrix} \hat{p} \\ \hat{q} \\ \hat{r} \\ \hat{s} \end{bmatrix} = \begin{bmatrix} \\ e_i \\ \\ \\ \end{bmatrix}$$

where $e_i = 0$ except $(e_i)_i = 1$. This system can be solved provided that not more than two of the nodes lie in a straight line.

Now, given the simplest rectangular element with just four nodes, one at each corner, we choose the local coordinate (ξ, η) as shown in figure 1.1.

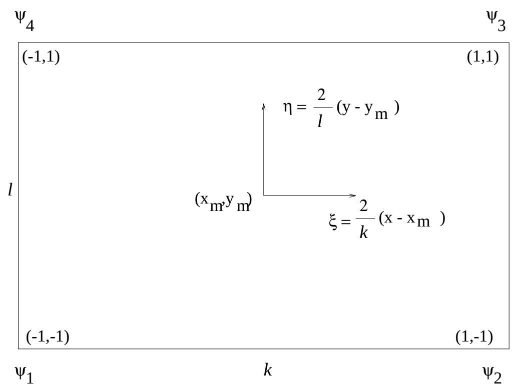

Figure 1.1: Rectangular Element.

Since there are four nodes with one degree of freedom at each node the variation throughout the element is just the bilinear form given in (1.6) below

$$\begin{aligned}\psi(x,y) &= \phi\psi \\ &= \frac{1}{4}[(1-\xi)(1-\eta) \quad (1+\xi)(1-\eta) \quad (1+\xi)(1+\eta) \\ &\qquad (1-\xi)(1+\eta)]\begin{bmatrix}\psi_1\\ \psi_2\\ \psi_3\\ \psi_4\end{bmatrix}. \end{aligned} \tag{1.6}$$

We have

$$\frac{\partial}{\partial x} = \frac{2}{k}\frac{\partial}{\partial \xi} \quad \text{and} \quad \frac{\partial}{\partial y} = \frac{2}{l}\frac{\partial}{\partial \eta}.$$

Now we can obtain the stiffness matrices $A^{(1)}, A^{(2)}$ and force vector b of the convection-diffusion system, say for the special case of constant $f(x,y) = c$. The calculation can be shown by taking an example;

$$A^{(1)}_{11} = \int_{-1}^{1}\int_{-1}^{1} \mu\left(\frac{4}{k^2}\frac{\partial\phi_1}{\partial\xi}\frac{\partial\phi_1}{\partial\xi} + \frac{4}{l^2}\frac{\partial\phi_1}{\partial\eta}\frac{\partial\phi_1}{\partial\eta}\right)\frac{k}{2}\partial\xi\frac{l}{2}\partial\eta.$$

Substituting $\phi_1 = \frac{1}{4}(1-\xi)(1-\eta)$ and doing the integrating process we get

$$A^{(1)}_{11} = \frac{1}{3}\mu(k/l + l/k).$$

By repeating the above process for all entries and noting symmetry for the diffusion matrix we obtain

$$A^{(1)} = \frac{1}{6}\mu\begin{bmatrix} 2(k/l+l/k) & k/l-2l/k & -k/l-l/k & l/k-2k/l \\ & 2(k/l+l/k) & l/k-2k/l & -k/l-l/k \\ \text{symmetric} & & 2(k/l+l/k) & k/l-2l/k \\ & & & 2(k/l+l/k) \end{bmatrix}$$

whilst entries for the convection matrix are obtained by integrating

$$A_{ij}^{(2)} = \int_{-1}^{1}\int_{-1}^{1}\left(w_x \frac{2}{k}\frac{\partial \phi_i}{\partial \xi}\phi_j + w_y \frac{2}{l}\frac{\partial \phi_i}{\partial \eta}\phi_j\right)\frac{k}{2}\partial\xi\frac{l}{2}\partial\eta,$$

leading to the matrix

$$A^{(2)} = \frac{1}{6}\begin{bmatrix} -lw_x - kw_y & lw_x - \frac{1}{2}kw_y & \frac{1}{2}lw_x + \frac{1}{2}kw_y & -\frac{1}{2}lw_x + kw_y \\ -lw_x - \frac{1}{2}kw_y & lw_x - kw_y & \frac{1}{2}lw_x + kw_y & -\frac{1}{2}lw_x + \frac{1}{2}kw_y \\ -\frac{1}{2}lw_x - \frac{1}{2}kw_y & \frac{1}{2}lw_x - kw_y & lw_x + kw_y & -lw_x + \frac{1}{2}kw_y \\ -\frac{1}{2}lw_x - kw_y & \frac{1}{2}lw_x - \frac{1}{2}kw_y & lw_x + \frac{1}{2}kw_y & -lw_x + kw_y \end{bmatrix}.$$

The overall element stiffness matrix is then obtained by the sum of $A^{(1)}, A^{(2)}$ i.e. $A = A^{(1)} + A^{(2)}$.

The force vector has entries obtained by

$$b_j = \int_{-1}^{1}\int_{-1}^{1} c\phi_j \frac{k}{2}\partial\xi\frac{l}{2}\partial\eta,$$

leading to the vector

$$b = \frac{klc}{4}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Assembling the element contributions yields the overall system of equations.

Chapter 2

Stationary Iterative Methods

2.1 The Jacobi and Gauss-Seidel Methods

The simplest iterative scheme is the *Jacobi iteration.* It is defined for matrices that have nonzero diagonal elements, in particular diagonally dominant matrices. This method can be motivated by rewriting the $n \times n$ system $Ax = b$ as follows:

$$\begin{aligned}
x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3 - \ldots - a_{1n}x_n)/a_{11} \\
x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3 - \ldots - a_{2n}x_n)/a_{22} \\
&\vdots \\
x_n &= (b_n - a_{n1}x_1 - a_{n2}x_2 - \ldots - a_{nn-1}x_{n-1})/a_{nn}.
\end{aligned}$$

In the Jacobi method, we choose an initial vector $x^{(0)}$ and then solve, iteratively, to find a new approximation $x^{(i+1)}$ using the algorithm

$$\begin{aligned}
x_1^{(i+1)} &= (b_1 - a_{12}x_2^{(i)} - a_{13}x_3^{(i)} - \ldots - a_{1n}x_n^{(i)})/a_{11} \\
x_2^{(i+1)} &= (b_2 - a_{21}x_1^{(i)} - a_{23}x_3^{(i)} - \ldots - a_{2n}x_n^{(i)})/a_{22} \\
&\vdots
\end{aligned}$$

$$x_n^{(i+1)} = (b_n - a_{n1}x_1^{(i)} - a_{n2}x_2^{(i)} - \ldots - a_{nn-1}x_{n-1}^{(i)})/a_{nn}.$$

We notice here that Jacobi iteration is not updating the most recently available information when computing $x_k^{(i+1)}$, where k is any entry of $x^{(i+1)}$ after the first one. For an example, if we look at the $x_2^{(i+1)}$ calculation we observe that $x_1^{(i)}$ is used even though we know $x_1^{(i+1)}$. If we use this most recent estimate we form the *Gauss-Seidel iteration*

$$\begin{aligned}
x_1^{(i+1)} &= (b_1 - a_{12}x_2^{(i)} - a_{13}x_3^{(i)} - a_{14}x_4^{(i)} - \ldots - a_{1n}x_n^{(i)})/a_{11} \\
x_2^{(i+1)} &= (b_2 - a_{21}x_1^{(i+1)} - a_{23}x_3^{(i)} - a_{24}x_4^{(i)} - \ldots - a_{2n}x_n^{(i)})/a_{22} \\
x_3^{(i+1)} &= (b_3 - a_{31}x_1^{(i+1)} - a_{32}x_2^{(i+1)} - a_{34}x_4^{(i)} - \ldots - a_{3n}x_n^{(i)})/a_{33} \\
&\vdots \\
x_n^{(i+1)} &= (b_n - a_{n1}x_1^{(i+1)} - a_{n2}x_2^{(i+1)} - \ldots - a_{nn-1}x_{n-1}^{(i+1)})/a_{nn}.
\end{aligned}$$

We use $x_1^{(i+1)}$ to calculate $x_2^{(i+1)}, x_2^{(i+1)}$ to calculate $x_3^{(i+1)}$ and hence use $x_{n-1}^{(i+1)}$ to calculate $x_n^{(i+1)}$. The method which updates the most recently available calculation tends to converge faster than the method which does not (see later).

We now let L, D, and U be matrices defined as

$$L = -\begin{bmatrix} 0 & 0 & \ldots & \ldots & 0 \\ a_{21} & 0 & \ddots & & \vdots \\ a_{31} & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 & 0 \\ a_{n1} & \ldots & \ldots & a_{nn-1} & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} a_{11} & 0 & \dots & \dots & 0 \\ 0 & a_{22} & \ddots & & \vdots \\ \vdots & \ddots & a_{33} & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & a_{nn} \end{bmatrix}$$

$$U = -\begin{bmatrix} 0 & a_{12} & \dots & \dots & a_{1n} \\ 0 & 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 & a_{n-1n} \\ 0 & \dots & \dots & 0 & 0 \end{bmatrix},$$

so that $A = D - L - U$. The Jacobi method is of the form

$$x^{(i+1)} = Hx^{(i)} + c_J, \tag{2.1}$$

with $H = D^{-1}(L+U)$ and $c_J = D^{-1}b$. The Gauss-Seidel step has the different form of

$$x^{(i+1)} = Gx^{(i)} + c_G, \tag{2.2}$$

with $G = (D-L)^{-1}U$ and $c_G = (D-L)^{-1}b$.

Both procedures are typical members of a large family of iterations that have the form

$$Qx^{(i+1)} = Vx^{(i)} + b. \tag{2.3}$$

Whether or not (2.3) converges to $x = A^{-1}b$ depends upon the eigenvalues of $Q^{-1}V$. In particular, if the *spectral radius* of an n-by-n matrix Z is defined by

$$\rho(Z) = \max\{|\lambda|, \lambda \in \lambda(Z)\}$$

then it is the size of $\rho(Q^{-1}V)$ that determines the convergence of (2.3). We now introduce the definition of a non-negative matrix.

Definition 2.1.1 *Let $A = (a_{ij})$ and $B = (b_{ij})$ be two $n \times n$ matrices. Then, $A \geq B(> B)$ if $a_{ij} \geq b_{ij}(> b_{ij})$ for all $1 \leq i, j \leq n$. If O is the null matrix and $A \geq O(> O)$, we say that A is a non-negative (positive) matrix. Finally, if $B = (b_{ij})$ is an arbitrary real or complex $n \times n$ matrix, then $|B|$ denotes*[1] *the matrix with entries $|b_{ij}|$.*

A mutually exclusive relation between the Jacobi matrix H and the Gauss-Seidel matrix G is given in the following theorem developed by Varga [5, pp.70].

Theorem 2.1.1 *Let the Jacobi matrix $H = E + F$ (where $E = D^{-1}L$ and $F = D^{-1}U$) be a non-negative $n \times n$ matrix with zero diagonal entries, and $G = (I - E)^{-1}F$. Then, one and only one of the following mutually exclusive relations is valid:*

1. $\rho(H) = \rho(G) = 0.$
2. $0 < \rho(G) < \rho(H) < 1.$
3. $1 = \rho(H) = \rho(G).$
4. $1 < \rho(H) < \rho(G).$

Thus, the Jacobi and the Gauss-Seidel iterations are either both convergent, or both divergent.

[1]This is not to be confused with the determinant of a square matrix B

Definition 2.1.2 *An $n \times n$ real or complex matrix $A = (a_{ij})$ is diagonally dominant if*

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad \text{for all} \quad 1 \leq i \leq n, \tag{2.4}$$

and is strictly diagonally dominant if strict inequality in (2.4) is valid for all $1 \leq i \leq n$. Similarly, A is irreducibly diagonally dominant if A is irreducible (see Varga [5, pp.18-19], for definition of the reducible matrix) and diagonally dominant, with strict inequality in (2.4) for at least one i.

Theorem 2.1.2 *Suppose $b \in \mathbb{R}^n$ and $A = Q - V \in \mathbb{R}^{n\times n}$ is nonsingular. If Q is non singular and the spectral radius of $Q^{-1}V$ satisfies the inequality $\rho(Q^{-1}V) < 1$, then the iterates $x^{(i)}$ defined by $Qx^{(i+1)} = Vx^{(i)} + b$ converge to $x = A^{-1}b$ for any starting vector $x^{(0)}$.*

Proof. Let $e^{(i)} = x^{(i)} - x$ denote the error in the ith iterate. Since $Qx^{(i+1)} = Vx^{(i)} + b$ it follows that $Q(x^{(i+1)} - x) = V(x^{(i)} - x)$, and thus, the error in $x^{(i+1)}$ is given by $e^{(i+1)} = Q^{-1}Ve^{(i)} = (Q^{-1}V)^{(i)}e^{(0)}$. We know that $(Q^{-1}V)^{(i)} \to 0$ iff $\rho(Q^{-1}V) < 1$, see Golub [4, pp.508]. So $x^{(i)}$ defined by $Qx^{(i+1)} = Vx^{(i)} + b$ converge to $x = A^{-1}b$ for any starting vector $x^{(0)}$. $\square$

Another fundamental theorem is concerned with the convergence of the strictly diagonally dominant matrix.

Theorem 2.1.3 *Let $A = (a_{ij})$ be a strictly or irreducibly diagonally dominant $n \times n$ real or complex matrix. Then, both Jacobi and Gauss-Seidel methods are convergent to $x = A^{-1}b$ for any starting vector $x^{(0)}$.*

Proof. If A is a $n \times n$ strictly diagonally dominant real or complex matrix, then it is nonsingular, see Varga [5, pp.23], and the diagonal entries of A are necessarily nonzero. From (2.1) the Jacobi matrix H derived from the matrix A is such that

$$h_{ij} = \begin{cases} 0 & \text{if } i = j \\ \frac{-a_{ij}}{a_{ii}} & \text{if } i \neq j. \end{cases}$$

By Definition 2.1.2 it follows that

$$\sum_{j=1}^{n} |h_{ij}| < 1 \quad \text{for all} \quad 1 \leq i \leq n$$

if A strictly diagonally dominant, or

$$\sum_{j=1}^{n} |h_{ij}| \leq 1 \quad \text{for all} \quad 1 \leq i \leq n$$

with strict inequality for at least one i if A is irreducibly diagonally dominant. Recalling Definition 2.1.1 $|H|$ is a non-negative matrix whose entries are $|h_{ij}|$, and these row sums tell us that $\rho(|H|) < 1$. Applying Theorem 2.8 in [5], we have that for any $n \times n$ matrices A and B with $O \leq |B| \leq A$, then $\rho(B) \leq \rho(A)$. We can conclude here that $\rho(H) \leq \rho(|H|) < 1$, so that the Jacobi iterative method is necessarily convergent. For the Gauss-Seidel matrix G we have

$$G = (D - L)^{-1}U \quad \text{or} \quad G = (I - E)^{-1}F$$

where E and F are respectively strictly lower and upper triangular matrices.

$$\begin{aligned} G &= (I - E)^{-1}F \\ |G| &= |(I + E + E^2 + E^3 + \ldots + E^{n-1})F| \\ &\leq (I + |E| + |E|^2 + |E|^3 + \ldots + |E|^{n-1})|F| \\ &= (I - |E|)^{-1}|F|. \end{aligned}$$

So that $\rho(G) \leq \rho\{(I - |E|)^{-1}|F|\}$. But as $\rho(|H|) < 1$, we have from Theorem 2.1.1 that $\rho\{(I - |E|)^{-1}|F|\} \leq \rho(|H|) < 1$. This implies

$$\rho(G) < 1;$$

thus we conclude that the Gauss-Seidel iterative method is also convergent, which completes the proof. □

2.2 The Damped Jacobi and Symmetric Gauss-Seidel Methods

There is a simple but important modification to the Jacobi iteration. This generates an entire family of iterations called *a weighted or damped Jacobi* method. The formula is given by

$$x^{(i+1)} = [(1-\omega) + \omega H]x^{(i)} + \omega c_J, \tag{2.5}$$

where $\omega \in \mathbb{R}$ is a weighting factor which may be chosen. Notice that with $\omega = 1$ we have the original Jacobi iteration.

We should note in passing that the weighted Jacobi iteration can also be written in the form

$$x^{(i+1)} = x^{(i)} + \omega D^{-1} r^{(i)}.$$

This says that the new approximation is obtained from the current one by adding an appropriate weighting of the residual $r^{(i)} = b - Ax^{(i)}$. In further prediction, this method tends, with a well chosen ω, to converge faster than Jacobi method. In general, we know that the error $e^{(i)} = x^{(i)} - x$ satisfies $Ae^{(i)} = r^{(i)}$, so we have $x^{(i)} - x = A^{-1}r^{(i)}$. From this expression, an iteration may be performed by taking $x^{(i+1)} = x^{(i)} + Br^{(i)}$, where B is some approximation to A^{-1}. If B can be chosen "close" to A^{-1}, then the iteration should be effective.

The weighted Jacobi method waits until all components of the new approximation have been computed before using them. This requires $2N$ storage locations for the approximation vector. It means that the new approximation can not be used as soon as it is available. On the contrary, the *Gauss-Seidel* method incorporates a simple change: components of the new approximation are used as soon as they are computed, so that components of the approximation vector x are over-written as soon as they are updated. Moreover, for the weighted Jacobi method, the order of updating the components (ascending or descending) is immaterial, since they are never over-written. However, for the Gauss-Seidel method, the order is significant.

The Gauss-Seidel method normally generates a nonsymmetric iteration matrix. This can be demonstrated by taking a 2×2 system as an example,

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

If we perform one iteration of Gauss-Seidel,

$$\begin{aligned} a_{11}x_1^{(1)} &= b_1 - a_{12}x_2^{(0)} \\ a_{22}x_2^{(1)} &= b_2 - a_{21}x_1^{(1)}, \end{aligned}$$

and if $x^{(0)}$ is zero, then $x_1^{(1)} = b_1/a_{11}$ and $x_2^{(1)} = b_2/a_{22} - a_{21}b_1/a_{11}a_{22}$ so

$$x^{(1)} = \begin{bmatrix} \frac{1}{a_{11}} & 0 \\ -\frac{a_{21}}{a_{11}a_{22}} & \frac{1}{a_{22}} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

This is equivalent to $x^{(1)} = Kb$. The matrix K here is not symmetric except for the trivial case when a_{21} is zero. However, we also have a symmetric version of the Gauss-Seidel method. If we perform one iteration of Gauss-Seidel again, initialising $x^{(0)}$ to zero, this time with a 3×3 system.

$$a_{11}x_1^* = b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)}$$

$$\begin{aligned} a_{22}x_2^* &= b_2 - a_{21}x_1^* - a_{23}x_3^{(0)} \\ a_{33}x_3^* &= b_3 - a_{31}x_1^* - a_{32}x_2^*. \end{aligned}$$

Therefore

$$\begin{aligned} x_1^* &= b_1/a_{11} \\ x_2^* &= (b_2 - a_{21}b_1/a_{11})/a_{22} \\ x_3^* &= [b_3 - a_{31}b_1/a_{11} - a_{32}(b_2 - a_{21}b_1/a_{11})/a_{22}]/a_{33}. \end{aligned} \tag{2.6}$$

If we then compute $x^{(1)}$ using x^* as an estimate, via

$$\begin{aligned} a_{33}x_3^{(1)} &= b_3 - a_{31}x_1^* - a_{32}x_2^* \\ a_{22}x_2^{(1)} &= b_2 - a_{21}x_1^* - a_{23}x_3^{(1)} \\ a_{11}x_1^{(1)} &= b_1 - a_{12}x_2^{(1)} - a_{13}x_3^{(1)}, \end{aligned}$$

this gives our new solution value of $x^{(1)}$. If we write $x^{(1)} = Kb$, we can show that K is symmetric and positive definite. Rearranging, we obtain

$$\begin{aligned} a_{11}x_1^{(1)} + a_{12}x_2^{(1)} + a_{13}x_3^{(1)} &= b_1 \\ a_{22}x_2^{(1)} + a_{23}x_3^{(1)} &= b_2 - a_{21}x_1^* \\ a_{33}x_3^{(1)} &= b_3 - a_{31}x_1^* - a_{32}x_2^*. \end{aligned}$$

This shows us that what we have performed here is a lower triangular sweep followed by an upper triangular sweep. This is the same as writing

$$(D-U)x^{(1)} = b + Lx^*. \tag{2.7}$$

From the set of equation (2.6), we have

$$\begin{aligned} b_1 &= a_{11}x_1^* \\ b_2 &= a_{21}x_1^* + a_{22}x_2^* \\ b_3 &= a_{31}x_1^* + a_{32}x_2^* + a_{33}x_3^*, \end{aligned}$$

which can be written as $(D-L)x^* = b$. Hence $x^* = (D-L)^{-1}b$, and substituting for x^* in equation (2.7) gives $(D-U)x^{(1)} = b + L(D-L)^{-1}b$.

Therefore

$$\begin{aligned} x^{(1)} &= (D-U)^{-1}(I + L(D-L)^{-1})b \\ &= (D-U)^{-1}((D-L)+L)(D-L)^{-1}b \\ &= (D-U)^{-1}D(D-L)^{-1}b. \end{aligned}$$

Define $K = (D-U)^{-1}D(D-L)^{-1}$. To show the symmetry,

$$K^T = [(D-L)^T]^{-1}D^T[(D-U)^T]^{-1},$$

but $(D-L)^T = (D-U)$ (A is a symmetric matrix in the diffusion case) which implies that $K^T = (D-U)^{-1}D(D-L)^{-1}$. Hence K is symmetric. For positive definiteness

$$x^TKx = x^T(D-U)^{-1}D(D-L)^{-1}x,$$

and if we let $y = (D-L)^{-1}x$ and $y \neq 0$ provided $x \neq 0$ then

$$\begin{aligned} y^T &= x^T[(D-L)^T]^{-1} \\ &= x^T(D-U)^{-1}. \end{aligned}$$

Therefore

$$x^TKx = y^TDy > 0.$$

(Since A is positive definite then D is a diagonal matrix with positive definite diagonal elements). Thus K is positive definite. For more details about symmetric Gauss-Seidel, the reader is referred to Golub [4].

Chapter 3

Multigrid Methods

3.1 The Multigrid Idea

We now introduce the standard multigrid idea for the solution of the algebraic equations arising from finite element discretisation of differential equations. We thus look to multigrid algorithms to provide an efficient iteration for solving a large, sparse system of equations. The efficiency of multigrid algorithms arise from the interaction between the smoothing properties of the relaxation method, such as Gauss-Seidel iteration, and the idea of coarse grid correction, see McCornick [12].

Using standard local Fourier analysis it can be seen that relaxation methods, without convergence acceleration parameters, are very efficient at reducing the amplitude of high-frequency components of the error, and thus of the algebraic residual, whilst their ultimately slow convergence is due to the low-frequency components, typically corresponding to the largest eigenvalues of the iteration operator. Relaxation methods can thus be viewed as efficient smoothers.

The principle of coarse grid correction can be seen by considering linear finite element equations. Denoting the grid by Ω^h, we let $G(\Omega^h)$ be the space of all grid functions defined on Ω^h. Then, assuming that the boundary conditions have been eliminated, the linear equations can be written in the form

$$A^h x^h = b^h$$

where $x^h, b^h \in G(\Omega^h)$, $A^h : G(\Omega^h) \rightarrow G(\Omega^h)$ and we assume that $(A^h)^{-1}$ exists.

Let v^h be the current approximation to x^h and define the algebraic defect (or residual) as

$$r^h = b^h - A^h v^h.$$

Then the exact solution x^h is given by

$$x^h = v^h + s^h$$

where s^h is the solution of the defect equation

$$A^h s^h = r^h. \tag{3.1}$$

If the high-frequency components of the defect are relatively negligible to the low frequency components, we can represent equation (3.1) on a coarser grid, Ω^H (in practice, we define a sequence consisting of $\Omega^{2h}, \Omega^{4h}, \ldots$) of the form:

$$A^H \hat{s}^H = r^H \tag{3.2}$$

where $dim(G(\Omega^H)) \ll dim(G(\Omega^h))$, $A^H : G(\Omega^H) \rightarrow G(\Omega^H)$ and we assume that $(A^H)^{-1}$ exists.

If we solve the equation (3.2) we can interpolate $\hat{s}^H$ to the finer grid giving an approximation $\hat{s}^h$ to s^h and then take

$$\hat{v}^h = v^h + \hat{s}^h$$

as the new approximation to x^h. The assumption here is that the approximation to the solution on the fine grid should be smooth enough to allow the equation to be represented on a coarse grid. This criterion is satisfied by using a suitable relaxation method before transferring the defect equation to the coarser grid.

The multigrid method extends this idea to the solution of the coarse grid equation (3.2) leading to a series of coarser and coarser grids. Thus the complete multigrid method combines a relaxation method on each grid with correction on coarser grids. Neither process alone gives a satisfactory method; it is only when a suitable combination of the two methods is used that good convergence and efficiency properties can be obtained.

The remaining issues are the computation of the residual on Ω^h and it's transfer to Ω^{2h}, and the computation of the error estimate on Ω^{2h} and it's transfer to Ω^h. These questions suggest that we need some mechanism for transferring information between grids. 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 is generally called *interpolation* or *prolongation*. Secondly, the inter-grid transfer function which involves moving the vectors from a fine grid to a coarse grid is generally called *restriction* operator.

3.1.1 Prolongation or Interpolation

The linear interpolation denoted by I_{2h}^h takes coarse grid vectors and fine grid vectors according to the rule $I_{2h}^h v^{2h} = v^h$. For one-dimensional problems, they may be defined as

$$\begin{aligned}
v_{2j}^h &= v_j^{2h} \\
v_{2j+1}^h &= \frac{1}{2}(v_j^{2h} + v_{j+1}^{2h}), \quad \text{where } 0 \le j \le \frac{N}{2} - 1.
\end{aligned}$$

At even-numbered fine grid points, the values of the vector are transferred directly from Ω^{2h} to Ω^h. At odd-numbered fine grid points, the values of the v^h are the average of the adjacent coarse grid values. As for two-dimensional problems, they may be defined in a similar way. If we let $I_{2h}^h v^{2h} = v^h$, then the components of v^h are given by

$$\begin{aligned}
v_{2i,2j}^h &= v_{i,j}^{2h} \\
v_{2i+1,2j}^h &= \frac{1}{2}(v_{i,j}^{2h} + v_{i+1,j}^{2h}) \\
v_{2i,2j+1}^h &= \frac{1}{2}(v_{i,j}^{2h} + v_{i,j+1}^{2h}) \\
v_{2i+1,2j+1}^h &= \frac{1}{4}(v_{i,j}^{2h} + v_{i+1,j}^{2h} + v_{i,j+1}^{2h} + v_{i+1,j+1}^{2h}), \qquad 0 \le i, j \le \frac{N}{2} - 1.
\end{aligned}$$

3.1.2 Restriction

The restriction operator is denoted by I_h^{2h}. The most obvious restriction is *injection* defined (in one dimension) by $I_h^{2h} v^h = v^{2h}$, where

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

In other words, the coarse grid vector simply takes its value directly from the corresponding fine grid point. An alternate restriction operator is called *full weighting* and is defined (in one dimension) by $I_h^{2h} v^h = v^{2h}$, where

$$v_j^{2h} = \frac{1}{4}(v_{2j-1}^h + 2v_{2j}^h + v_{2j+1}^h), \qquad 1 \le j \le \frac{N}{2} - 1. \tag{3.3}$$

In this case, the values of the coarse grid vector are weighted averages of values at neighbouring fine grid points.

The full weighting restriction operator is a linear operator from $\mathbb{R}^{N-1}$ to $\mathbb{R}^{\frac{N}{2}-1}$. It has a rank of $N/2-1$ and null space with dimension $N/2$. One reason for our choice of full weighting as restriction operator is the important fact that

$$I_{2h}^{h} = c(I_h^{2h})^T, \tag{3.4}$$

where $c = 2$. Using the definition (3.3) we can also formulate the restriction operator in two dimensions. Letting $I_h^{2h}v^h = v^{2h}$, we have that

$$\begin{aligned} v_{i,j}^{2h} &= \frac{1}{16}[v_{2i-1,2j-1}^h + v_{2i-1,2j+1}^h + v_{2i+1,2j-1}^h + v_{2i+1,2j+1}^h + 2(v_{2i,2j-1}^h \\ &\quad + v_{2i,2j+1}^h + v_{2i-1,2j}^h + v_{2i+1,2j}^h) + 4v_{2i,2j}^h], \qquad 1 \le i,j \le \frac{N}{2}-1. \end{aligned}$$

3.2 The Multigrid Algorithm

3.2.1 The Basic Two-Grid Algorithm

A well-defined way to transfer vectors between fine and coarse grids can be represented in the following coarse grid correction scheme

$$v^h \leftarrow CG(v^h, b^h).$$

So that the two-grid algorithm can be written as

```
Relax $s_1$ times on $A^h x^h = b^h$ on $\Omega^h$ with initial guess $v^h$.
Compute $r^{2h} = I_h^{2h}(b^h - A^h v^h)$.
    Solve $A^{2h}e^{2h} = r^{2h}$ on $\Omega^{2h}$.
Correct fine grid approximation : $v^h \leftarrow v^h + I_{2h}^h e^{2h}$.
```

Relax s_2 times on $A^h x^h = b^h$ on Ω^h with initial guess v^h.

The procedure defined above is simply the original coarse grid correction idea, which was proposed earlier, now refined by the use of the intergrid transfer operators. We relax on the fine grid until it ceases to be worthwhile. In practice s_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. As it stands, 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. This is followed by s_2 additional fine grid relaxation sweeps.

We can further expand this algorithm to higher grids level, say n levels. This idea is fundamental to the understanding of multigrid.

3.2.2 Multigrid

The multigrid algorithm which is formulated in Briggs [10] can be expressed as

```
Relax on $A^h x^h = b^h$ $s_1$ times with initial guess $v^h$.
Compute $b^{2h} = I_h^{2h} r^h$.
    Relax $A^{2h} x^{2h} = b^{2h}$ $s_1$ times with initial guess $v^{2h} = 0$.
    Compute $b^{4h} = I_{2h}^{4h} r^{2h}$.
        Relax on $A^{4h} x^{4h} = b^{4h}$ $s_1$ times with initial guess $v^{4h} = 0$.
        Compute $b^{8h} = I_{4h}^{8h} r^{4h}$.
                ⋮
            Solve $A^{Lh} x^{Lh} = b^{Lh}$.
                ⋮
            Correct $v^{4h} \leftarrow v^{4h} + I^{4h}_{8h} v^{8h}$.
            Relax on $A^{4h} x^{4h} = b^{4h}$ $s_2$ times with initial guess $v^{4h}$.
        Correct $v^{2h} \leftarrow v^{2h} + I^{2h}_{4h} v^{4h}$.
        Relax $A^{2h} x^{2h} = b^{2h}$ $s_2$ times with initial guess $v^{2h}$.
    Correct $v^{h} \leftarrow v^{h} + I^{h}_{2h} v^{2h}$.
    Relax on $A^{h} x^{h} = b^{h}$ $s_2$ times with initial guess $v^{h}$.
```

Figure 3.1 illustrates the grid schedule of multigrid method.

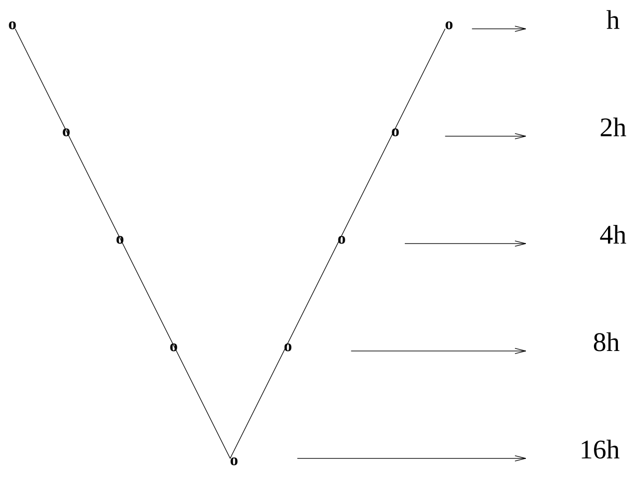

Figure 3.1: Grids schedule for a V-Cycle with five levels.

The algorithm telescopes down to the coarsest grid, which can be a single interior grid point, and then works its way back to the finest grid. Figure 3.1 shows the schedule for the grids in the order in which they are visited. Because of the pattern of this diagram, this algorithm is called the V-cycle. It is our first true multigrid method. For more details of the algorithm, including a compact recursive definition, the reader is referred to Briggs [10].

3.3 Complexity and Convergence Analysis

3.3.1 Complexity

We now analyse a complexity and convergence of multigrid method. These points are important to address due to the practical issues of the effectiveness and efficiency of the overall method. The complexity is considered first.

Each grid in the multigrid scheme needs two arrays: one to hold the current approximation on each grid and one to hold the right-hand side vectors on each grid, see McCormick [12]. Since boundary values must also be stored, the coarsest grid involves three grid points in one dimension (one interior and two boundary points). In general, the kth coarsest grid involves $2^k + 1$ point in one dimension. Now considering a $d-$dimensional grid with N^d points where $N = 2^n$, the finest grid Ω^h requires $2N^d$ storage locations; Ω^{2h} has 2^{-d} times as many; Ω^{4h} has 2^{-2d} times as many; etc. (by assumption two arrays must be stored on each level). Adding these and using the sum of the geometric series as an upper bound give us

$$\text{Storage} = 2N^d\{1 + 2^{-d} + 2^{-2d} + \ldots + 2^{-nd}\} < \frac{2N^d}{1-2^{-d}}.$$

In particular, for a one-dimensional problem, the storage requirement is less than twice that of the fine grid problem alone. For two or more dimensions, the requirement drops to less than 4/3 of the fine grid problem alone. Thus, the storage costs of the multigrid algorithm decrease relatively as the dimension of the problem increases. All the experiments which are going to be presented in chapter 4, are in two dimensions.

The other important thing to measure is the cost in terms of work units (WU).

Here a work unit is defined to be the cost of performing one relaxation sweep on the finest grid. Equivalently, it is the cost of expressing the fine grid operator. It is customary to neglect the cost of intergrid transfer operations which could amount to $15 - 20\%$ of the cost of the entire cycle. First consider a V-cycle with one relaxation sweep on each level ($s1 = s2 = 1$). Each level is visited twice and the grid Ω^{sh} requires s^{-d} work units. Adding these costs and again using the geometric series for an upper bound give

MV computation cost =

$$2\{1 + 2^{-d} + 2^{-2d} + \ldots + 2^{-nd}\} < \frac{2}{1 - 2^{-d}} WU$$

3.3.2 Convergence analysis

We can attempt to give heuristic and qualitative arguments suggesting that standard multigrid schemes, when applied to well-behaved problems (for example, symmetric, positive definite systems), not only work, but they work very efficiently.

We begin with the heuristic argument that captures the spirit of rigorous convergence proofs. Let us denote the original continuous problem (for example, one of our model boundary value problems) $Ax = b$. The associated discrete problem on the fine grid Ω^h will be denoted $A^h x^h = b^h$. As before, we let v^h be an approximation to x^h on Ω^h. The global error is defined by

$$E^h_{i,j} = x(x_{i,j}) - x^h_{i,j}, \quad 1 \leq i, j \leq N - 1$$

and is related to the truncation error of the discretisation process. In general, the global error may be bounded in norm in the form

$$||E^h|| \leq K h^p, \quad K \in \mathbb{R}.$$

The quantity that we have been calling the error, $e^h = x^h - v^h$, will now be called the algebraic error to avoid confusion with the global error. The algebraic error measures how well our approximations (generated by relaxation or multigrid) agree with the exact discrete solution.

The purpose of a typical calculation is to produce an approximation v^h which agrees with the exact solution of the continuous problem. Let us specify a tolerance ϵ with a condition such as

$$||x - v^h|| < \epsilon$$

where $x = (x(x_{1,1}), \ldots, x(x_{N-1,N-1}))^T$ is the vector of exact solution values. This condition can be satisfied if we guarantee that

$$||E^h|| + ||e^h|| < \epsilon$$

(since $||x - v^h|| \leq ||x - x^h|| + ||x^h - v^h|| = ||E^h|| + ||e^h|| < \epsilon$).

One way to ensure that $||E^h|| + ||e^h|| < \epsilon$ is to require $||E^h|| < \epsilon/2$ and $||e^h|| < \epsilon/2$ independently. The first condition determines the grid spacing on the finest grid. It says that we must choose

$$h < h^* = \left(\frac{\epsilon}{2K}\right)^{1/p}$$

The second condition is determined by how quickly our approximations v^h converge to x^h. If relaxation or multigrid cycles have been performed until condition $||e^h|| < \epsilon/2$ is met on grid Ω^h, where $h < h^*$, then we have converged to the *level of truncation*. In summary, the global error determines the critical grid spacing h^*. The only computational requirement is that we converge to the level of truncation error on a grid with $h < h^*$.

We now consider the V-cycle scheme applied to a d-dimensional problem with N^d unknowns and $h = 1/N$. We have to assume (and can generally show rigorously) that with fixed cycling parameters s_1 and s_2, the V-cycle scheme has a convergence rate, γ, which is independent of h. This V-cycle scheme must reduce the algebraic error from $O(1)$ (the error in arbitrary initial guess) to $O(h^p) = O(N^{-p})$ (the order of the global error). Therefore, the number of V-cycles required, v, must satisfy $\gamma^v = O(N^p)$ or $v = O(logN)$. This is comparable to the computational cost of the best fast direct solvers when applied to the model problems as mentioned in chapter 1.

3.4 The Effectiveness of the Relaxation Methods

The main concern of this project is to show the robustness of multigrid methods numerically, and also to assess whether one relaxation method is more effective and efficient than another if it is used within multigrid. In this section we would like to compare the smoothing potential of the damped Jacobi and the Gauss-Seidel methods using a local mode analysis, see Chan [15].

This local mode analysis (originally devised by Brandt, see [14]), is a general and effective tool for analysing and predicting the performance of a smoother. The approach is based on the fact that relaxation is typically a local process in which information propagates by a few mesh-sizes per sweep. Therefore, one can assume the problem to be in an unbounded domain, with constant (frozen) coefficients, in which case the algebraic error can be expanded in terms of a

Fourier series. To do this, let us first recall the discrete Fourier theorem.

Theorem 3.4.1 *Every discrete function* $u : I_n \to \mathbb{R}$, $I_n = \{(i,j) : 0 \leq i,j \leq n\}$, *can be written as*

$$u_{i,j} = \sum_{\theta \epsilon \Theta_n} c_\theta \psi_{i,j}(\theta), \quad \psi_{i,j}(\theta) = e^{\mathbf{i}(i\theta_1 + j\theta_2)}, \quad \mathbf{i} = \sqrt{-1}, \quad \theta = (\theta_1, \theta_2),$$

where

$$c_\theta = \frac{1}{(n+1)^2} \sum_{(k,l)\epsilon I_n} u_{k,l}\psi_{k,l}(-\theta),$$

and

$$\Theta_n = \{\frac{2\pi}{n+1}(k,l), \quad -m \leq k,l \leq m+p\},$$

where $p = 1, m = (n+1)/2$ *for odd* n *and* $p = 0, m = n/2 + 1$ *for even* n.

Consider the pure-diffusion of the form $-\mu\nabla^2 u = f$ with homogeneous Dirichlet condition on a unit square discretised using standard second order difference approximations with uniform grids. The discretised equation can be expressed as

$$\begin{aligned} 4u_{i,j} &= (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) + \frac{k^2}{\mu} f_{i,j}, \quad 1 \leq i,j \leq n-1 \quad (3.5) \\ 4u_{i,j} &= 4u_{i,j} - \omega(4u_{i,j} - (\frac{k^2}{\mu} f_{i,j} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})). \quad (3.6) \end{aligned}$$

The damped Jacobi iteration corresponding to the equation can be expressed as

$$4\hat{u}_{i,j} = 4\bar{u}_{i,j} - \omega(4\bar{u}_{i,j} - (\frac{k^2}{\mu} f_{i,j} + \bar{u}_{i+1,j} + \bar{u}_{i-1,j} + \bar{u}_{i,j+1} + \bar{u}_{i,j-1})), \quad (3.7)$$

where $\hat{u}_{i,j}$ denotes the new value of u while $\bar{u}_{i,j}$ denotes the old value of u. The Gauss-Seidel iteration (with lexigraphical order on nodal points, from left to right and bottom to top) is presented of the form

$$4\hat{u}_{i,j} = (\bar{u}_{i+1,j} + \hat{u}_{i-1,j} + \bar{u}_{i,j+1} + \hat{u}_{i,j-1}) + \frac{k^2}{\mu} f_{i,j}. \quad (3.8)$$

We use the discrete Fourier transform to analyse the damped Jacobi method. Let the global error of the method be defined as $\hat{\epsilon}_{i,j} = u_{i,j} - \hat{u}_{i,j}$ and $\bar{\epsilon}_{i,j} = u_{i,j} - \bar{u}_{i,j}$. Subtracting the equation (3.6) and (3.7) we obtain

$$\hat{\epsilon}_{i,j} = \bar{\epsilon}_{i,j} - \frac{\omega}{4}(4\bar{\epsilon}_{i,j} - (\bar{\epsilon}_{i+1,j} + \bar{\epsilon}_{i-1,j} + \bar{\epsilon}_{i,j+1} + \bar{\epsilon}_{i,j-1})). \tag{3.9}$$

We write

$$\hat{\epsilon}_{i,j} = \sum_{\theta \epsilon \Theta_n} \hat{c}_\theta \psi_{i,j}(\theta) \quad \text{and} \quad \bar{\epsilon}_{i,j} = \sum_{\theta \epsilon \Theta_n} \bar{c}_\theta \psi_{i,j}(\theta). \tag{3.10}$$

Substituting the equation (3.10) into (3.9) we have

$$\begin{aligned} \sum_{\theta \epsilon \Theta_n} \hat{c}_\theta \psi_{i,j}(\theta) &= \sum_{\theta \epsilon \Theta_n} \Big(\bar{c}_\theta \psi_{i,j}(\theta) - \frac{\omega}{4}\Big(4\bar{c}_\theta \psi_{i,j}(\theta) - (\bar{c}_\theta \psi_{i+1,j}(\theta) \\ &\quad + \bar{c}_\theta \psi_{i-1,j}(\theta) + \bar{c}_\theta \psi_{i,j+1}(\theta) + \bar{c}_\theta \psi_{i,j-1}(\theta))\Big)\Big), \qquad (3.11) \\ \hat{c}_\theta \psi_{i,j}(\theta) &= \bar{c}_\theta \Big(\psi_{i,j}(\theta) - \frac{\omega}{4}\Big(4\psi_{i,j}(\theta) - (\psi_{i+1,j}(\theta) \\ &\quad + \psi_{i-1,j}(\theta) + \psi_{i,j+1}(\theta) + \psi_{i,j-1}(\theta))\Big)\Big), \qquad (3.12) \end{aligned}$$

and comparing the coefficient of each $\psi_{i,j}(\theta)$, we obtain that

$$\frac{\hat{c}_\theta}{\bar{c}_\theta} = 1 - \frac{\omega}{4}\Big(4 - (e^{\mathbf{i}\theta_1} + e^{-\mathbf{i}\theta_1} + e^{\mathbf{i}\theta_2} + e^{-\mathbf{i}\theta_2})\Big). \tag{3.13}$$

Applying the expression of the form $e^{\pm \mathbf{i}\theta_k} = cos(\theta_k) \pm \mathbf{i}sin(\theta_k)$ into (3.13) gives

$$\lambda(\theta) = 1 - \omega(1 - \frac{cos\theta_1 + cos\theta_2}{2}), \tag{3.14}$$

where $\lambda(\theta) \equiv \frac{\hat{c}_\theta}{\bar{c}_\theta}$ is called the amplification factor of the local mode $\psi_{i,j}(\theta)$.

The smoothing factor introduced by Brandt is the following quantity

$$\bar{\rho} = \sup\{|\lambda(\theta)|, \pi/2 \leq |\theta_k| \leq \pi, \quad k = 1, 2\}.$$

Roughly speaking, the smoothing factor $\bar{\rho}$ is the maximal amplification factor corresponding to those high frequency local modes that oscillate within a $2h$ range (and hence can not be resolved by the coarse grid of size $2h$).

For damped Jacobi method, it is easy to see that

$$\bar{\rho} = \max\{|1-2\omega|, |1-\omega|, |1-3\omega/2|\}.$$

The optimal ω that minimises the smoothing factor is

$$\omega = 4/5, \quad \bar{\rho} = 3/5$$

(this value of ω is used in the numerical experiments presented in chapter 4). For $\omega = 1$ we have $\bar{\rho} = 1$. This means that the undamped Jacobi method for this model problem, although convergent as an iterative method by itself, should not be used as smoother.

We next examine the smoothing potential of the Gauss-Seidel iteration. Unlike the Jacobi method, Gauss-Seidel method depends on the ordering of the unknown. The most natural ordering is perhaps the lexicographic order. Hence, we also express the Gauss-Seidel method in terms of the global error definition by subtracting the equation (3.5) and (3.8). Thus, the method reads

$$\hat{\epsilon}_{i,j} = \frac{1}{4}(\bar{\epsilon}_{i+1,j} + \hat{\epsilon}_{i-1,j} + \bar{\epsilon}_{i,j+1} + \hat{\epsilon}_{i,j-1}).$$

Again using the Fourier transform (3.10), we obtain that

$$\begin{aligned}
\sum_{\theta\epsilon\Theta_n} \hat{c}_\theta \psi_{i,j}(\theta) &= \sum_{\theta\epsilon\Theta_n} \frac{1}{4}\Big(\bar{c}_\theta \psi_{i+1,j}(\theta) + \hat{c}_\theta \psi_{i-1,j}(\theta) \\
&\quad + \bar{c}_\theta \psi_{i,j+1}(\theta) + \hat{c}_\theta \psi_{i,j-1}(\theta)\Big), &(3.15)\\
\frac{\hat{c}_\theta}{\bar{c}_\theta}\psi_{i,j}(\theta) &= \frac{1}{4}\Big(\psi_{i+1,j}(\theta) + \frac{\hat{c}_\theta}{\bar{c}_\theta}\psi_{i-1,j}(\theta) + \psi_{i,j+1}(\theta) \\
&\quad + \frac{\hat{c}_\theta}{\bar{c}_\theta}\psi_{i,j-1}(\theta)\Big), &(3.16)\\
\frac{\hat{c}_\theta}{\bar{c}_\theta} &= \frac{\psi_{i+1,j}(\theta) + \psi_{i,j+1}(\theta)}{4\psi_{i,j}(\theta) - \psi_{i-1,j}(\theta) - \psi_{i,j-1}(\theta)}. &(3.17)
\end{aligned}$$

Employing the expression $\psi_{i,j}(\theta) = e^{\mathbf{i}(i\theta_1+j\theta_2)}$ we have

$$\frac{\hat{c}_\theta}{\bar{c}_\theta} = \frac{e^{\mathbf{i}((i+1)\theta_1+j\theta_2)} + e^{\mathbf{i}(i\theta_1+(j+1)\theta_2)}}{4e^{\mathbf{i}(i\theta_1+j\theta_2)} - e^{\mathbf{i}((i-1)\theta_1+j\theta_2)} - e^{\mathbf{i}(i\theta_1+(j-1)\theta_2)}}. \tag{3.18}$$

Finally, reducing the exponential term we obtain the local amplification factor as follows:

$$\lambda(\theta) = \frac{e^{\mathbf{i}\theta_1} + e^{\mathbf{i}\theta_2}}{4 - e^{-\mathbf{i}\theta_1} - e^{-\mathbf{i}\theta_2}}. \tag{3.19}$$

It is elementary to determine that

$$\bar{\rho} = |\lambda(\pi/2, cos^{-1}(4/5))| = 1/2.$$

We can see that the smoothing factor of the Gauss-Seidel is less than the damped Jacobi, this means the Gauss-Seidel is slightly better smoother than the damped Jacobi method.

Chapter 4

Numerical Results

We are now in a position to test the relaxation and multigrid methods in a practical application. Numerical results based on the theory discussed in chapters 1–3 are presented here.

The discretisation process for some model diffusion and convection-diffusion problems will be accomplished using some freely available Matlab software for solving such problems, written by D.J. Silvester (available from `http://www.ma.umist.ac.uk/djs/index.htm`). The relaxation and multigrid methods were also written as Matlab programs and these are given in appendix A.

We use the V–Cycle algorithm as a representative multigrid method, and perform one relaxation sweep on each level (s1=s2=1). We will discuss the numerical results separately for diffusion and convection-diffusion, and give some discussion of the V–Cycle method robustness in each case.

4.1 Numerical Solution of the Diffusion Problem

The general test problem in chapter two is $-\mu\nabla^2 u + \mathbf{w}\cdot\nabla u = f$. If we set $\mathbf{w}$=**0** the problem is $-\mu\nabla^2 u = f$, and it is then referred to as the diffusion problem. With the available software we can generate the linear system of equations using *uniform grids* of square elements with $\mu = 1$, $f = 0$. The solution satisfies the condition $u = 1$ on part of the bottom boundary and the right-hand wall, and $u = 0$ on the remainder of the boundary.

Applying the V-Cycle method, the solution of our test problem in the square domain $(0 \le x \le 1; 0 \le y \le 1)$ is presented in figure 4.1.

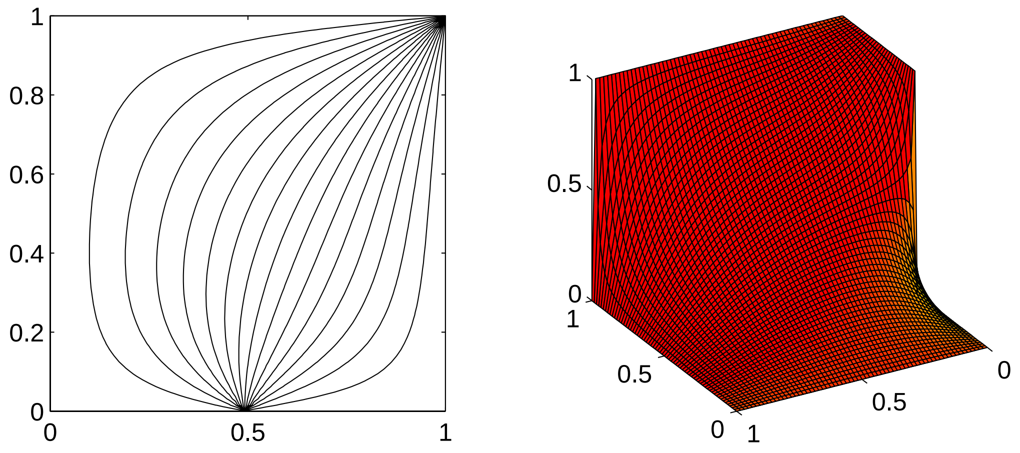

Figure 4.1: The diffusion problem solution for a (64×64) uniform grid with tolerance ϵ =1e-6.

The software generates the stiffness matrices and right hand side force vectors and then assembles into a system of linear equations. We solve this system with relaxation and V–Cycle methods. As mentioned in chapter one, the relaxation methods used are the Gauss-Seidel, damped Jacobi, and symmetric Gauss-Seidel methods. We also use these relaxation methods as smoothers in the V–CYCLE method.

4.1.1 Convergence Rate of Simple Relaxation Methods

The following tables describe the number of iterations to converge, work units and CPU times (in seconds) required in solving the different sized linear systems (which are all sparse, symmetric, and positive-definite) arising from employing an INTEL P150 processor PC.

The numerical results using the Gauss-Seidel, damped Jacobi and symmetric Gauss-Seidel methods are summarised in tables 4.1, 4.2 and 4.3, respectively. The methods are applied on grids N=4, 8, 16, 32, 64, with tolerances ϵ =`1e-2` and ϵ =`1e-6`.

Grids	Gauss-Seidel applied to the diffusion problem					
	ϵ =`1e-2`			ϵ =`1e-6`		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	5	0.002	1.10	14	0.004	1.87
(8×8)	11	0.020	1.36	51	0.091	2.03
(16×16)	21	0.181	1.94	178	1.499	4.11
(32×32)	27	1.007	2.27	614	22.529	20.97
(64×64)	28	4.345	4.32	2069	316.816	238.05

Table 4.1: General convergence for Gauss-Seidel method.

These results suggest that the Gauss-Seidel method is similar to symmetric Gauss-Seidel, and both are better methods than the damped Jacobi method. It is also clear that the rate of convergence of all three relaxation methods is not remarkable; in general we need a large number of iterations and work units to solve the problem, and also every increment of matrix dimension of the stiffness matrix is followed by a rigorous increase in the number of iterations.

Grids	damped Jacobi method applied to the diffusion problem					
	ϵ =1e-2			ϵ =1e-6		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	10	0.005	1.35	34	0.016	2.05
(8×8)	26	0.069	1.87	125	0.329	2.85
(16×16)	51	0.640	2.95	443	5.547	5.77
(32×32)	68	3.703	3.98	1534	83.707	50.78
(64×64)	72	16.231	8.24	5173	1178.100	643.05

Table 4.2: General convergence for damped Jacobi method when $\omega = 4/5$.

Grids	Symmetric Gauss-Seidel method applied to the diffusion problem					
	ϵ =1e-2			ϵ =1e-6		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	3	0.002	1.08	10	0.006	1.24
(8×8)	10	0.031	1.30	31	0.093	1.32
(16×16)	28	0.393	2.03	110	1.528	2.13
(32×32)	85	5.104	4.14	406	24.292	17.15
(64×64)	242	60.213	39.18	1518	377.192	209.18

Table 4.3: General convergence for symmetric Gauss-Seidel method.

These schemes work very well only for the first few iterations, see figure 4.2. Inevitably, however, the convergence slows and the entire scheme appears to stall. We have a simple explanation for this phenomenon earlier; i.e. the rapid decrease in the error during the early iteration is due to the efficient elimination of the highly oscillatory modes of that error. Once the oscillatory modes have been removed, the iteration is much less effective in reducing the remaining smooth components of the error. All three relaxation schemes above possess this property of eliminating the oscillatory modes and leaving the smooth modes. (This is called the smoothing property.)

4.1.2 Convergence Rate of a Representative Multigrid Method

We now present the numerical results of V–CYCLE method applied to the diffusion problem using Gauss-Seidel, damped Jacobi and symmetric Gauss-Seidel as smoothers.

As described in the previous chapter we need to define inter-grid transfer functions of prolongation and restriction to move vectors from a coarse grid to a fine grid, and vice-versa. Here, we set the restriction coefficient (see (3.4)) to different values. For the V–CYCLE method with the Gauss-Seidel and damped Jacobi method as smoothers we set it at 1.5, but for the symmetric Gauss-Seidel at 1.75. Our experience indicates that the convergence rate is strongly sensitive to the value of this coefficient, and the above values are the "optimal" choices.

The numerical results by applying the V–CYCLE method with the Gauss-Seidel, damped Jacobi, and symmetric Gauss-Seidel method, are given in tables 4.4, 4.5 and 4.6, respectively. The tables display the V-CYCLE convergence with one pre-relaxation and post-relaxation sweep done on each level. These show the type of behavior that we aim for in using a multigrid algorithm. If N is the number of points in the finest grid then the work units required to obtain convergence is not far from $0(N)$. Recall from chapter 3 that the work units for each grid are based on the number of operations per grid point. As an example, using the (64×64) grid, there are 3969 grid-points, hence the work units for a tolerance of $\epsilon =$`1e-6` is $4354560/3969 \cong 1097$ WU.

Grids	V–CYCLE method applied to the diffusion problem					
	$\epsilon =$`1e-2`			$\epsilon =$`1e-6`		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	3	0.004	2.46	9	0.012	3.86
(8×8)	3	0.022	3.01	10	0.070	4.98
(16×16)	4	0.131	5.59	13	0.417	9.09
(32×32)	4	0.560	11.58	12	1.648	33.08
(64×64)	4	2.321	30.80	13	7.397	112.80

Table 4.4: General convergence for V–CYCLE method with Gauss-Seidel as smoother.

Table 4.6 then gives the best performance of the method (for the case where the relaxation coefficient equals 1.75). The computation keeps the same number of iterations as the grid size changes i.e. two iterations to achieve the tolerance $\epsilon =$`1e-2` and six iterations to achieve the tolerance $\epsilon =$`1e-6`.

Grids	V–CYCLE method applied to the diffusion problem					
	$\epsilon =$`1e-2`			$\epsilon =$`1e-6`		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	4	0.006	3.35	13	0.019	4.73
(8×8)	4	0.033	3.95	14	0.115	5.05
(16×16)	5	0.193	6.16	15	0.572	11.16
(32×32)	5	0.836	12.55	15	2.478	33.55
(64×64)	5	3.483	37.17	15	10.331	120.17

Table 4.5: General convergence for V–CYCLE method with damped Jacobi as smoother when $\omega = 4/5$.

Grids	V–CYCLE method applied to the diffusion problem					
	ϵ =1e-2			ϵ =1e-6		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	2	0.003	1.03	6	0.009	3.02
(8×8)	2	0.017	2.95	6	0.051	3.95
(16×16)	2	0.082	3.87	6	0.239	6.27
(32×32)	2	0.358	8.12	6	1.041	15.82
(64×64)	2	1.495	20.69	6	4.355	53.83

Table 4.6: General convergence for V–CYCLE method with symmetric Gauss-Seidel as smoother.

The convergence histories of the relaxation and V-CYCLE methods at the fine grid level (64×64) are plotted in figures 4.2 and 4.3.

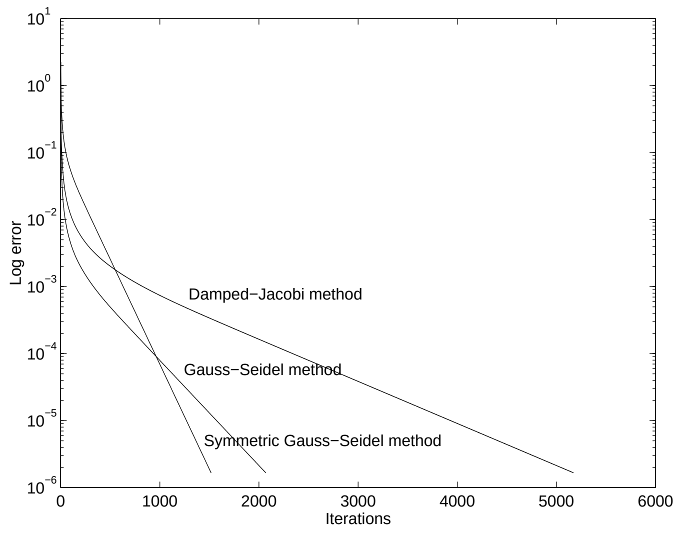

Figure 4.2: The convergence history of the relaxation methods when the tolerance ϵ =1e-6.

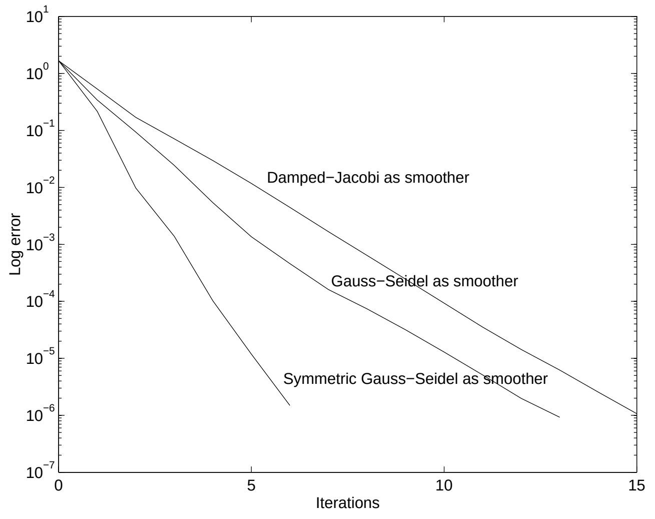

Figure 4.3: The convergence history of V-Cycle method with Gauss-Seidel, damped Jacobi, and symmetric Gauss-Seidel as smoothers when the tolerance $\epsilon =$`1e-6`.

These figures show the number of iterations against the infinity norm of the residual error for each solution method. The graph progresses from left to right in the direction of an increasing number of iterations. It can be observed that the convergence histories are completely different. The graph of relaxation methods decreases rapidly for the first few iterations and then curves upwards to give slower and slower convergence as the iteration proceeds. Unlike the relaxation methods there is not a "deflection" in the convergence history of the V-Cycle method and the error reduction is close to linear in this case.

4.2 Numerical Solution of the Convection-Diffusion Problem

We apply some of these methods to a convection-diffusion problem. In this case we set the viscosity parameter of the equation $-\mu\nabla^2 u + \mathbf{w}\cdot\nabla u = f$ to three different values $\mu = 1/10, 1/100, 1/1000$, and the vector $\mathbf{w}$ to a constant vector $(-\sqrt{2}/2, \sqrt{2}/2)$. The convecting-wind is defined within the available software by setting $\theta = \pi/4$, and f is set to zero.

Using the available software we can generate the linear system of equations whose solution satisfies the condition $u = 1$ on part of the bottom boundary and the right-hand wall, and $u = 0$ on the remainder of the domain. The iteration methods used to solve this problem are the symmetric Gauss-Seidel iteration, and the V–Cycle method with the symmetric Gauss-Seidel as smoother.

Graphs of the solutions that result from applying the V–Cycle method to this problem with viscosity parameters 1/10, 1/100 and 1/1000 are presented in figures 4.4, 4.5 and 4.6, respectively. All the solutions are obtained using a (64×64) uniform grid, and the error tolerance equals $\epsilon =$`1e-6`.

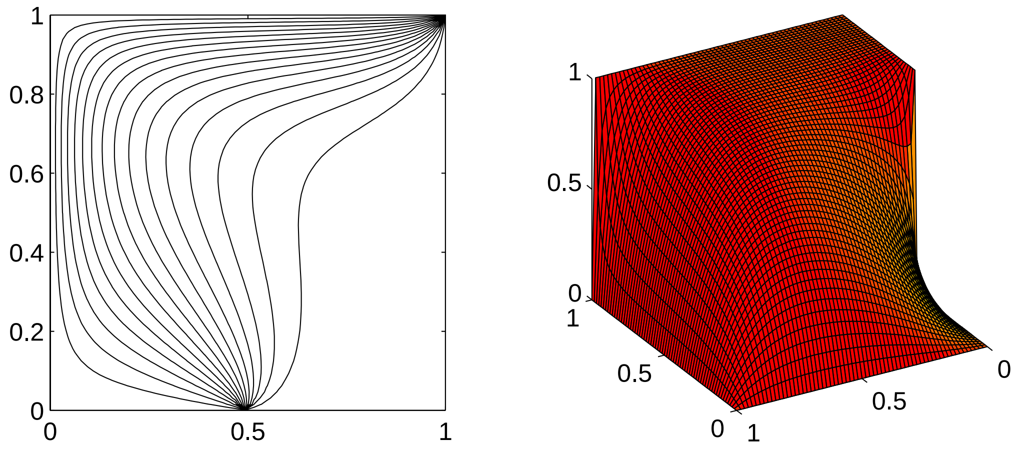

Figure 4.4: The solution for the viscosity parameter 1/10.

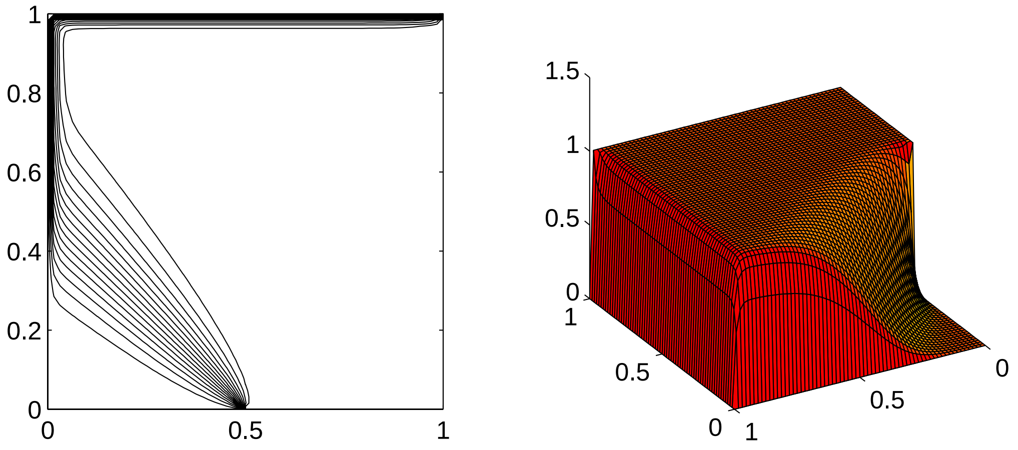

Figure 4.5: The solution for the viscosity parameter 1/100.

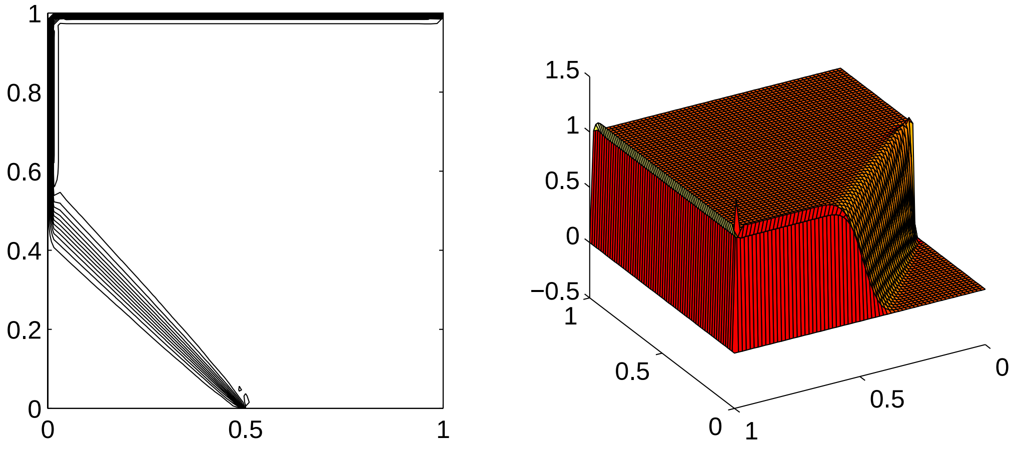

Figure 4.6: The solution for the viscosity parameter 1/1000.

It can be seen that reducing the viscosity parameter μ increases the relative strength of the wind, and that if μ is "small" there is an internal layer generated by the discontinuity on the inflow boundary and a boundary layer at the left hand wall and along the top (which are illustrated in figures 4.5 and 4.6).

These layers are difficult to resolve and are common-place in computational fluid mechanics. Methods which combine a high accuracy with good stability properties in cases where convection dominates diffusion are needed. In particular, the same problem comes up in the vanishing viscosity limit ($\mu = 0$) where the mathematical models for compressible flow take the form of a hyperbolic system of conservation laws and solutions may present discontinuities (shocks) which cannot be fully resolved on any mesh of finite size.

With (small) diffusion $\mu > 0$, discontinuities are replaced by layers with rapid but continuous variations of the flow variables, and we may then either seek to resolve the layers by choosing h sufficiently small, or essentially stay in the vanishing viscosity limit without accurately resolving layers.

The simplest finite difference and standard Galerkin methods fail to resolve this difficulty. They lack stability and may produce approximate solutions with spurious oscillations which do not converge to physically correct solutions, and satisfy only first order accurate (also in regions where the exact solution is smooth) and smear discontinuities (in particular contact discontinuities) over many mesh points.

A special method which eliminates most of the spurious oscillations in the Galerkin method and improves the quality of the numerical solution at discontinuities, giving almost monotone discrete shock profiles is the *streamline diffusion* method, see [3, pp.258-269], for details. This method gives us the possibility of improved accuracy with good stability.

A further feature of this method is the use of space-time finite elements for time-dependent problems with the basis functions being continuous in space and discontinuous in time, which leads to implicit time stepping methods. The discretisation in the available software also uses a streamline diffusion formulation in cases with a small viscosity parameter compared to the mesh size.

4.2.1 Convergence Rate of Relaxation Methods

Tables 4.7, 4.8 and 4.9 present the numerical results arising from using the symmetric Gauss-Seidel method for three different viscosity parameters. It can be observed that the smaller the viscosity parameter, the more effective the solver. We see that the minimum number of iterations, work unit and time elapsing (for the 64×64 grid) occur with the viscosity parameter 1/1000. This suggests that the standard relaxation method is increasingly effective as the viscosity parameter tends to zero.

Grids	Symmetric Gauss-Seidel method applied to the Convection-Diffusion problem					
	ϵ =1e-2			ϵ =1e-6		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	2	0.001	1.08	5	0.003	1.28
(8×8)	6	0.019	1.71	15	0.046	1.98
(16×16)	17	0.241	4.89	54	0.753	3.85
(32×32)	50	3.012	8.14	198	11.859	9.10
(64×64)	131	32.639	22.64	743	184.669	121.22

Table 4.7: Convergence rate of symmetric Gauss-Seidel method applied to the problem with viscosity parameter 1/10.

Grids	Symmetric Gauss-Seidel method applied to the Convection-Diffusion problem					
	$\epsilon =$1e-2			$\epsilon =$1e-6		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	3	0.002	1.10	8	0.005	1.58
(8×8)	3	0.010	1.33	8	0.025	1.62
(16×16)	5	0.075	2.63	10	0.144	2.53
(32×32)	11	0.681	3.86	19	1.159	3.87
(64×64)	30	7.549	12.08	55	13.759	11.44

Table 4.8: Convergence rate of the symmetric Gauss-Seidel method applied to the problem with viscosity parameter 1/100.

Grids	Symmetric Gauss-Seidel method applied to the Convection-Diffusion Problem					
	$\epsilon =$1e-2			$\epsilon =$1e-6		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	3	0.002	1.11	8	0.005	1.58
(8×8)	3	0.010	1.10	9	0.028	1.69
(16×16)	5	0.075	2.73	11	0.158	2.80
(32×32)	7	0.442	3.07	16	0.979	3.14
(64×64)	13	3.326	4.45	23	5.810	7.18

Table 4.9: Convergence rate of symmetric Gauss-Seidel method applied to the problem with viscosity parameter 1/1000.

However, the convergence history, see figure 4.8, again shows that the relaxation method works best for the first few iterations and then the convergence rate decreases in the following iterations. The standard relaxation method, furthermore, is not effective in the absence of *upwinding*. In particular, for a small viscosity parameter the iteration does not converge. Figure 4.7 describes the convergence history using a (16×16) uniform grid for a viscosity parameter of 1/100.

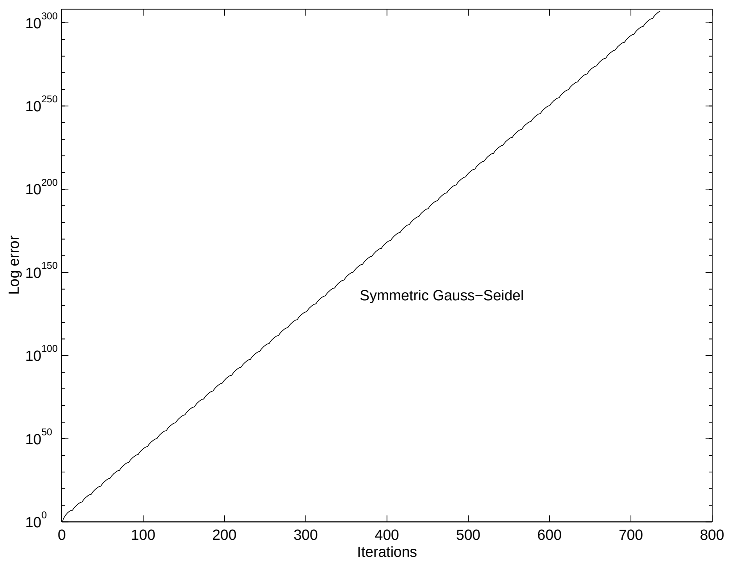

Figure 4.7: The convergence history of symmetric Gauss-Seidel applied in the convection-diffusion problem without upwinding.

4.2.2 Convergence Rate of Multigrid Methods

We now present the numerical results of the V–Cycle method applied to the convection-diffusion problem with the symmetric Gauss Seidel as smoother. We set the restriction coefficient at 1.65 for all viscosity parameters 1/10, 1/100 and 1/1000. This "optimal" coefficient was generated by observing the number of iterations required over the interval $(0.25 \leq c \leq 2)$, where c is the restriction coefficient.

The tables summarising the result of using the V-Cycle scheme applied to this problem are presented in 4.10, 4.11 and 4.12, respectively. These show that there is a small increase in the iteration counts as the viscosity parameter decreases. For the tolerance $\epsilon =$`1e-6` the maximum number of iterations, work unit and time elapsing occur when the viscosity parameter is set to 1/1000.

Grids	V-Cycle method applied to the Convection-Diffusion problem					
	$\epsilon =$`1e-2`			$\epsilon =$`1e-6`		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	2	0.003	1.03	6	0.009	2.02
(8×8)	2	0.017	2.05	6	0.051	3.25
(16×16)	2	0.082	3.87	6	0.239	6.15
(32×32)	2	0.358	6.12	6	1.041	15.92
(64×64)	2	1.495	18.69	6	4.355	50.83

Table 4.10: Convergence rate of V-Cycle method applied to the convection-diffusion problem with viscosity parameter 1/10.

Grids	V-Cycle method applied to the Convection-Diffusion problem					
	$\epsilon =$`1e-2`			$\epsilon =$`1e-6`		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	2	0.003	1.13	8	0.012	2.12
(8×8)	3	0.026	2.95	8	0.067	3.79
(16×16)	2	0.082	3.67	9	0.356	8.27
(32×32)	2	0.358	6.18	9	1.553	18.82
(64×64)	1	0.781	9.69	7	5.069	57.73

Table 4.11: Convergence rate of V-Cycle method applied to the convection-diffusion problem with viscosity parameter 1/100.

Grids	V–Cycle method applied to the Convection-Diffusion problem					
	ϵ =`1e-2`			ϵ =`1e-6`		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	3	0.005	2.24	8	0.012	2.80
(8×8)	3	0.026	3.35	10	0.084	3.84
(16×16)	3	0.121	4.17	11	0.435	8.29
(32×32)	2	0.358	7.12	16	2.749	34.21
(64×64)	2	1.495	18.69	23	16.506	177.65

Table 4.12: Convergence rate of V-Cycle method applied to the convection-diffusion problem with viscosity parameter 1/1000.

In this case, the multigrid method is less efficient than the relaxation method. This method gives improved efficiency for large values of the viscosity parameter because the infinity norm of the residual error decreases abruptly in every V-cycle, but this does not hold at the small values. The convergence histories at the fine grid level (64×64) are illustrated in figures 4.8 and 4.9.

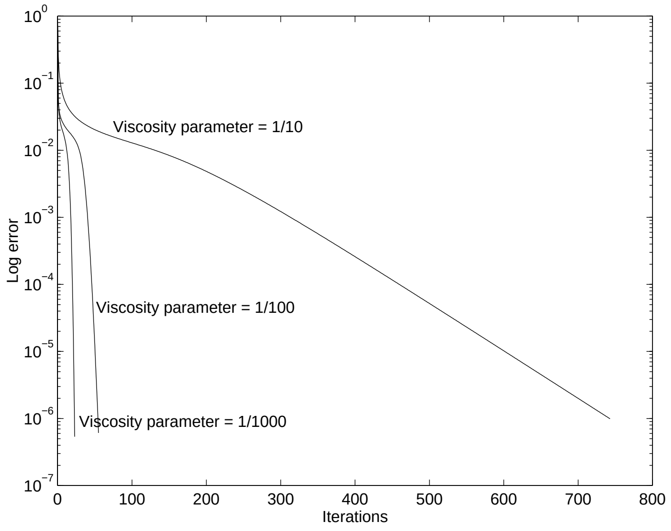

Figure 4.8: The convergence history of symmetric Gauss-Seidel (ϵ =`1e-6`) applied in the convection-diffusion problem.

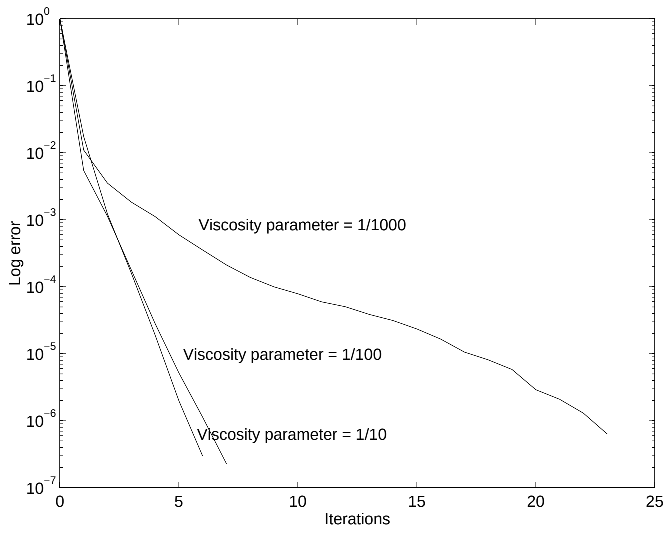

Figure 4.9: The convergence history of V–Cycle method with symmetric Gauss-Seidel as smoother (ϵ =`1e-6`) applied in the convection-diffusion problem.

In addition, if we set the restriction coefficient at 0.95 for the viscosity parameter 1/1000, using multigrid can significantly reduce the number of iterations at the finest level (64×64), see table 4.13.

Grids	V–Cycle method applied to the Convection-Diffusion problem					
	$\epsilon = 1e-2$			$\epsilon = 1e-6$		
	Iteration count	# Mflops	Times (seconds)	Iteration count	# Mflops	Times (seconds)
(4×4)	3	0.005	1.73	8	0.012	2.58
(8×8)	3	0.026	2.65	9	0.076	4.07
(16×16)	2	0.082	3.87	11	0.435	9.27
(32×32)	1	0.187	4.12	15	2.578	24.31
(64×64)	1	0.781	10.68	13	9.358	101.95

Table 4.13: Convergence rate of V-Cycle method applied to the convection-diffusion problem with "optimal" restriction coefficient.

4.3 Conclusions

The relaxation and V–Cycle methods have been implemented to solve diffusion and convection-diffusion problems, with a few modifications involving the restriction coefficient and it's dependence on the viscosity parameter. The numerical results show that the V–Cycle method has given large gains in efficiency in comparison with simple relaxation methods. This scheme worked efficiently for all grid sizes, but the relaxation schemes worked well only for coarse discretisations. Inevitably, however, the relaxation convergence slowed and the entire scheme appeared to stall. It could be deduced that the multigrid method succeeded due to its ability to selectively eliminate the high frequency components of the error.

In particular, the largest gains in efficiency of the multigrid method were in solving the diffusion problem discretised using (64×64) grid size. For different smoothers the minimum and maximum numbers of iterations were respectively 6 and 15 iterations, while the analogous relaxation methods required 1518 and 5173 iterations to reach the same tolerance. However, the performance of multigrid method steadily decreased in solving the convection-diffusion when the viscosity parameter tends to zero. With the viscosity set to $1/1000$ the multigrid method did not dramatically speed up the convergence. The relaxation method, on the contrary, gave a more efficient method than the multigrid method.

Finally, it should be noted that without upwinding (the parameter value set to zero), neither the standard relaxation or multigrid methods converged to the given tolerance. The solutions were becoming highly oscillatory in this case.

Appendix A

```
%%%%%VDRIVERA.M
%V-CYCLE multigrid method driver
%This code calls the function vcyclea.m

t0=clock;flops(0)
fine_level=input('The finest matrix level           : ');
itr       =input('number of levels (usually as above): ');
if itr > fine_level, error('respecify ...too many levels'), end
w_pre     =input('number of pre-smoothing sweeps     : ');
w_post    =input('number of post-smoothing sweeps    : ');
load system1,load system2,load system3;
r=fine_level;

%%% initialisation
zi=zeros(size(eval(['g' num2str(r)])));
init_error=norm(eval(['g' num2str(r)])-eval(['A' num2str(r)])*zi,inf);
error=init_error;
errvec=[init_error];
l=0;
fprintf('\n itn     error ')
fprintf('\n-----   -------')
%%% iteration loop
 while error > 1e-6*init_error
 fprintf('\n %3.0f  %9.3e',l,error)
 [zz]=vcyclea(zi,A1,A2,A3,A4,A5,A6,A7,g1,g2,g3,g4,g5,g6,g7,...
            fine_level,itr,w_pre,w_post,r);
 zi=zz;
 error=norm(eval(['g' num2str(r)])-eval(['A' num2str(r)])*zi,inf);
 errvec=[errvec,error];
 l=l+1;
end
Number_of_iteration = l
semilogy(0:l,errvec)
```

```
xlabel('Iterations');
ylabel('Log error');

n =eval(['n' num2str(r)]);
sx=eval(['sx' num2str(r)]);
sy=eval(['sy' num2str(r)]);
figure
plotadx(zz,n,inf,sx,sy);

calculations = flops %Count of floating point operations
time         = etime(clock,t0) % Time elapsing


%%%%%VCYCLEA.M
%The main program of V-CYCLE method.
%This code calls restr.m, prolo.m

function [zz]=vcyclea(zi,A1,A2,A3,A4,A5,A6,A7,g1,g2,g3,g4,g5,g6,g7,...
     fine_level,itr,w_pre,w_post,r);

%%% restriction
 g_fine=eval(['g' num2str(fine_level)]);
 for grd=fine_level:-1:fine_level-itr+2
%%% Gauss-Seidel Relaxation
     A_fine = eval(['A' num2str(grd)]);
     L=-tril(A_fine,-1);
     U=-triu(A_fine,1);
     D=diag(diag(A_fine));
     M=D-L;
     for s=1:w_pre
       z_fine= M\(U*zi + g_fine);
       zi= z_fine;
     end
%%% store residual and current iterate
     eval(['g' num2str(grd) ' = g_fine;']);
     eval(['Z' num2str(grd) ' = z_fine;']);
%%% restrict to coarser grid
     res_fine = g_fine - A_fine*z_fine;
     r_fine = reshape(res_fine,sqrt(length(res_fine)),...
sqrt(length(res_fine)));
     [r_coarse]=restr(r_fine);
     r_new=reshape(r_coarse,length(eval(['A' num2str(grd-1)])),1);
     g_fine = r_new;
     zi=zeros(size(g_fine));
 end

%%% exact solve on coarsest grid
       g_coarse = r_new;
       A_coarse = eval(['A' num2str(fine_level-itr+1)]);
       u_coarse = A_coarse\g_coarse;

%%% projection and correction
 for grd= fine_level-itr+1: fine_level-2
     u_crsr = reshape(u_coarse,sqrt(length(u_coarse)),...
 sqrt(length(u_coarse)));
     [u_crsr1]=prolo(u_crsr);
     u_crsr2=reshape(u_crsr1,length(eval(['A' num2str(grd+1)])),1);
     zi= eval(['Z' num2str(grd+1)]) + u_crsr2;
%%% Gauss-Seidel relaxation
     g_fine=eval(['g' num2str(grd+1)]);
     A_fine=eval(['A' num2str(grd+1)]);
     L=-tril(A_fine,-1);
     U=-triu(A_fine,1);
     D=diag(diag(A_fine));
     M=D-L;
     for s=1:w_post
       v_latest = M\(U*zi + g_fine);
       zi= v_latest;
    end
    u_coarse=v_latest;
 end

%%% final correction
     u_crsr2= u_coarse;
     u_crsr3 = reshape(u_crsr2,sqrt(length(u_crsr2)),...
 sqrt(length(u_crsr2)));
     [u_crsr4]=prolo(u_crsr3);
     u_crsr5=reshape(u_crsr4,length(eval(['A' num2str(fine_level)])),1);
     zz= eval(['Z' num2str(fine_level)]) + u_crsr5;
return


%%%%%RESTR.M
%The restriction code using full weighting function

function [r_coarse]=restr2b(r_fine);

S=[1 0 0 0;1/2 1/2 0 0;0 1 0 0;...
```

```
   1/2 0 1/2 0;1/4 1/4 1/4 1/4;0 1/2 0 1/2;...
   0 0 1 0;0 0 1/2 1/2;0 0 0 1];
T=(1.75)*S';

s=length(r_fine);
n=0.5*(s-1);
k=n;
for q=1:k
 for r=1:k
  B=r_fine(2*q-1:2*q+1,2*r-1:2*r+1);
  X=reshape(B,9,1);
  Y=T*X;
  C=reshape(Y,2,2);
  r_coarse(q:q+1,r:r+1)=C;
 end
end
m=length(r_coarse);
B=r_coarse(2:(m-1),2:(m-1));
Zer=zeros(m,m);
Zer(2:(m-1),2:(m-1))=B;
r_coarse=Zer;


%%%%PROLO.M
%The interpolation (prolongation) code

function [u_crsr1]=prolo(u_crsr);

S=[1 0 0 0;1/2 1/2 0 0;0 1 0 0;...
   1/2 0 1/2 0;1/4 1/4 1/4 1/4;0 1/2 0 1/2;...
   0 0 1 0;0 0 1/2 1/2;0 0 0 1];

n=length(u_crsr);
k=n-1;
for q=1:k
 for r=1:k
  B=u_crsr(q:q+1,r:r+1);
  X=reshape(B,4,1);
  Y=S*X;
  C=reshape(Y,3,3);
  u_crsr1(2*q-1:2*q+1,2*r-1:2*r+1)=C;
 end
end
```

Bibliography

[1] K. W. Morton. *Numerical Solution of Convection-Diffusion Problems.* Chapman and Hall, 1996.

[2] B. Fisher, A. Ramage, D.J. Silvester and A.J. Wathen. Towards parameter-free streamline upwinding for advection-diffusion problem. Strathclyde University Mathematics Report #37, 1996.

[3] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method.* Cambridge University Press, 1992 .

[4] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Second Edition, Johns Hopkins University Press, Baltimore and London, 1993.

[5] R. S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 1962.

[6] G. Strang. *Linear Algebra and its Applications.* Academic Press, 1988.

[7] M.J.D. Powell. *Approximation Theory and Methods.* Cambridge University Press, 1981.

[8] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM Books, Philadelphia, 1996.

[9] R. L. Burden and J. D. Faires. *Numerical Analysis.* Brooks/Cole Publishing Company, 1997.

[10] W. L. Briggs. *A Multigrid Tutorial.* SIAM Books, Philadelphia, 1987.

[11] K. E. Chen. *Multigrid Algorithm with Linear and Nonlinear Applications.* Ph.D. Thesis, Victoria University of Manchester, 1986.

[12] S. F. McCormick. *Multigrid Methods.* Volume 3 of *Frontiers in Applied Mathematics.* SIAM Books, 1987.

[13] B. Engquist and E. Luo. Convergence of a multigrid method for elliptic equations with highly oscillatory coefficients. SIAM *J. Numer. Anal.,* 34:2254-2273, 1997.

[14] A. Brandt. *Multigrid techniques: 1984 guide with applications to fluid dynamics.* GMD-Studien Nr. 85. Gesellschaft für Mathematik und Datenverarbeitung. St. Augustin, 1984.

[15] R. H. Chan, T. F. Chan and G. H. Golub (Eds.). *Iterative Methods in Scientific Computing.* Springer-Verlag, Singapore, 1997.

[16] J. Penny and G. Lindfield. *Numerical Methods Using Matlab.* Ellis Horwood Limited, 1995.

[17] D. J. Silvester. Incompressible flow and iterative solution Matlab software. IFISS Version 1.2, (http://www.ma.umist.ac.uk/djs/index.htm), UMIST, 1998.