DE-FG03-93ER25183

DOE/ER/25183--T3

MULTIGRID METHODS WITH APPLICATIONS TO RESERVOIR SIMULATION

by

Shengyou Xiao

May 1994

CNA-265

HH

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

MULTIGRID METHODS WITH APPLICATION TO RESERVOIR SIMULATION

Publication No. ______________

Shengyou Xiao, Ph.D.
The University of Texas at Austin, 1994

Supervisors: David M. Young and Kamy Sepehrnoori

This dissertation is concerned with the study of multigrid methods for the solution of elliptic partial differential equations. The primary focus is on parallel multigrid methods and the application of multigrid methods to reservoir simulation. Multicolor Fourier analysis is used to analyze the behavior of standard multigrid methods for problems in one and two dimensions. The relationship between multicolor Fourier analysis and standard Fourier analysis is established. Multiple coarse grid methods for solving certain model problems in one and two dimensions are considered. For such methods, at each coarse grid level we use more than one coarse grid to improve convergence. For the application of multiple coarse grid methods to a given Dirichlet problem it is convenient to first construct a related extended problem. For solving an extended problem with a multiple coarse grid method, a "purification" procedure can be used to obtain Moore–Penrose solutions of the singular systems which are encountered. For solving anisotropic equations, semicoarsening and line smoothing techniques are used with multiple coarse grid methods to improve convergence. The two-level convergence factors of the multiple coarse grid methods are estimated by using a multicolor Fourier analysis. In a special case where each of the operators has the same stencil on each of the grid points on one level, the exact multilevel convergence factors of the multiple coarse grid methods can be obtained. For solving partial differential equations with discontinuous coefficients, the interpolation and restriction operators should include information about the coefficients of the equations. Matrix-dependent interpolation and restriction operators based on the Schur complement can be used in nonsymmetric cases. A semicoarsening multigrid solver with matrix-dependent interpolation and restriction operators is used in UTCOMP, a three-dimensional, multiphase, multicomponent, compositional reservoir simulator developed at The University of Texas at Austin. The numerical experiments are carried out on different computing systems. The results obtained from the analysis and the numerical experiments indicate that the multigrid methods are promising.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

For many problems in science and engineering one is faced with the need to solve one or more partial differential equations. The use of discretization methods such as finite-difference methods or finite element methods usually leads to the need to solve one or more large systems of linear (or nonlinear) algebraic equations. The solution of such problems by direct methods or by conventional iterative methods can be very costly.

Multigrid methods offer the possibility of greatly improved convergence, as compared to iterative methods, for some problems. However, rigorous analysis of multigrid methods is available for only a very limited class of problems. Moreover, standard multigrid methods are not suitable, in general, for use with parallel computers.

In this dissertation we are concerned with three aspects of multigrid methods: a rigorous analysis of standard multigrid methods for a class of model problems in one and two dimensions; a description and analysis of multiple coarse grid methods which are actually multigrid methods where at each coarse grid level more than one coarse grid is used; and a description of some applications of multigrid methods to the solution of problems in reservoir simulation.

In Chapter 2, we define the model problems which will be used in later chapters. In Chapter 3, we give a brief description of some basic iterative methods and polynomial acceleration procedures.

In Chapters 4 and 5, we describe the application of standard multigrid methods to certain model problems in one and two dimensions. We present two analyses of these methods: one is based on the use of standard Fourier analysis, the other is based on the use of a two-color Fourier analysis for problems in one dimension and on the use of a four-color Fourier analysis for problems in two dimensions. The new multicolor Fourier analysis is especially effective when certain smoothing iteration methods such as the red/black Gauss–Seidel method are used. We also study the relationship between the standard Fourier analysis and the multicolor Fourier analysis and show that they are equivalent under a similarity transformation.

In Chapters 6 to 9, we consider multiple coarse grid methods for solving certain model problems in one and two dimensions. For such methods, more than one coarse grid is used at every coarse grid level. We consider three types of multiple coarse grid methods including multiple coarse grid multigrid (MCGMG) methods, frequency decomposition multigrid (FDMG) methods, and parallel multigrid (PMG) methods. For each of these methods we first construct a related extended problem as described in Chapter 6. The multiple coarse grid procedures which we consider can be conveniently defined and analyzed for the extended problems. A "purification" procedure is used to obtain Moore–Penrose solutions of singular systems which are usually encountered.

Previous work on parallel multigrid methods by Frederickson and McBryan [28] was applicable to periodic problems. Young and Vona [73] considered parallel multigrid methods for certain non-periodic problems. However, it was necessary to use more complicated operators than those which are involved with the extended problems.

The convergence factors of two-level multiple coarse grid methods are estimated by using the multicolor Fourier analysis. The effects of some red/black smoothing schemes are also described.

For anisotropic problems, the PMG methods based on point smoothing and the standard coarsening schemes are not very efficient. We consider a new variant of the PMG methods using semicoarsening and line smoothing techniques. We extend the convergence analysis of the multilevel PMG procedure described by Frederickson and McBryan [29] to the semicoarsening PMG procedure for anisotropic problems.

In Chapters 10 to 12 we consider the applications of standard multigrid methods to problems in petroleum reservoir simulation. Dendy et al. [24] used multigrid methods to solve some model problems of the type that arise from pressure equations in reservoir simulation. Fogwell and Brakhagen [27] used multigrid methods to solve the equations for incompressible, two phase flow in a porous medium. We developed a semicoarsening multigrid procedure which can be used to solve systems of linear equations arising from the discretization of the governing pressure equation in UTCOMP, a three-dimensional, multiphase, multicomponent, compositional reservoir simulator developed at The University of Texas at Austin [12] [13]. The governing pressure equation in the reservoir simulator is an anisotropic differential equation which may have discontinuous coefficients and the matrices of the linear systems are nonsymmetric. To obtain a fast convergence rate, we use matrix-dependent interpolation and restriction operators constructed in a way analogous to the Schur complement procedure in our multigrid algorithm.

The numerical results show that the multigrid methods compete very well with other iterative methods as well as with direct methods. We examined the performance of the multigrid code on a variety of parallel systems.

Chapter 2

Model Problems

2.1 Introduction

In this chapter, we define the model problems which will be used for the convergence analysis of the multigrid methods discussed in later chapters. We consider elliptic partial differential equations on the unit square (unit interval in each dimension) with Dirichlet boundary conditions. The standard 3-point and 5-point finite-difference discretizations are used for the problems in one dimension and in two dimensions respectively.

2.2 A One-Dimensional Model Problem

The 1D model problem we consider is the Poisson equation defined on the interval $(0,1)$ with Dirichlet boundary conditions:

$$\begin{cases} -\dfrac{d^2u(x)}{dx^2} = f(x) & \text{for } x \in \Omega = (0,1), \\ u(0) = \phi_a, & u(1) = \phi_b. \end{cases} \tag{2.2.1}$$

Let Ω_h be defined as

$$\Omega_h = \left\{ (x_j) \mid j = 0, \ldots, N;\ h = \frac{1}{N} \right\}, \tag{2.2.2}$$

where N is an integer and $x_j = jh$. By using the standard finite difference discretization process we obtain a set of linear equations:

$$\begin{array}{l} \frac{1}{h^2}[-u_{j-1} + 2u_j - u_{j+1}] = f_j, \quad j = 1, \ldots, N-1, \\ u_0 = \phi_a, \quad u_N = \phi_b, \end{array} \tag{2.2.3}$$

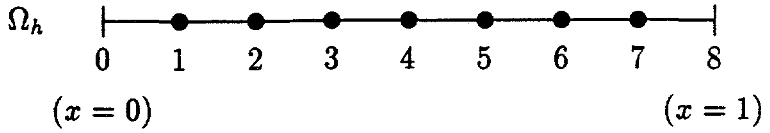


Figure 2.1: Grid Points for a One-Dimensional Problem: $h = 1/8$

where $u_j = u(x_j)$ and $f_j = f(x_j)$. The difference equations can be written in the matrix form as

$$A_h^{(3)} u = b \tag{2.2.4}$$

where the superscript of the matrix A_h indicates the discretization scheme (e.g. "3" indicates the 3-point standard finite-difference scheme).

In the case of $N = 8$, the grid Ω_h is defined as in Figure 2.1 and the corresponding discrete problem (2.2.4) is given by

$$\frac{1}{h^2}\begin{bmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 + \phi_a/h^2 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 + \phi_b/h^2 \end{bmatrix}. \tag{2.2.5}$$

It can be verified that the eigenvectors of $A_h^{(3)}$, for the case of $h = 1/8$, are given by

$$v_h^{(p)} = \begin{bmatrix} \sin(p\pi h) \\ \sin(2p\pi h) \\ \sin(3p\pi h) \\ \sin(4p\pi h) \\ \sin(5p\pi h) \\ \sin(6p\pi h) \\ \sin(7p\pi h) \end{bmatrix}, \qquad p = 1, \ldots, 7 \tag{2.2.6}$$

and the corresponding eigenvalues are given by

$$\nu_h^{(p)} = \frac{1}{h^2}(2 - 2\cos p\pi h), \quad p = 1, \ldots, 7. \tag{2.2.7}$$

2.3 A Two-Dimensional Model Problem

The 2D model problem we consider is an elliptic problem with Dirichlet boundary conditions defined on the unit square as

$$\begin{cases} -\alpha\dfrac{\partial^2 u(x,y)}{\partial x^2} - \dfrac{\partial^2 u(x,y)}{\partial y^2} = f(x,y) & (x,y) \in \Omega = (0,1)^2, \\ u = \phi(x,y) & (x,y) \in \partial\Omega \end{cases} \tag{2.3.8}$$

where $\alpha > 0$. If $\alpha = 1$, we have the *Poisson problem.* If $\alpha > 1$ or $\alpha < 1$, we have an *anisotropic problem.*

As in the one dimensional case, we define an $(N+1) \times (N+1)$ grid Ω_h covering the domain Ω for some integer N. We assume that a uniform step size $h = N^{-1}$ is used for both axis directions. Thus we have

$$\Omega_h = \{(x_j, y_k) \mid j, k = 0, \ldots, N\} \tag{2.3.9}$$

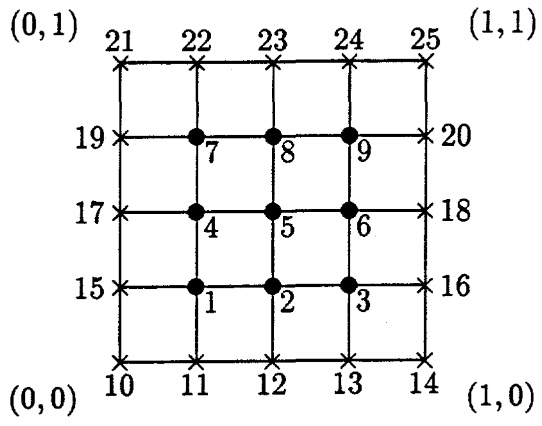


Figure 2.2: Grid Points for a Two-Dimensional Problem: $h = 1/4$

where $x_j = jh$ and $y_k = kh$. The 5–point difference representation of the problem (2.3.8) can be written as

$$\begin{cases} \frac{1}{h^2}[(2+2\alpha)u_{j,k} - \alpha u_{j-1,k} - \alpha u_{j+1,k} - u_{j,k-1} - u_{j,k+1}] = f_{j,k}, \\ u_{j,0} = \phi(x_j, y_0), \\ u_{j,N} = \phi(x_j, y_N), \\ u_{0,k} = \phi(x_0, y_k), \\ u_{N,k} = \phi(x_N, y_k), \\ \qquad j,k = 1, \ldots, N-1 \end{cases} \tag{2.3.10}$$

where $f_{j,k} = f(x_j, y_k)$ and $u_{j,k} = u(x_j, y_k)$. The boundary values can be collected into the right-hand side of the equations. The difference equations can be written in the matrix form

$$A_h^{(5)} u = b. \tag{2.3.11}$$

In the case of $N = 4$, the grid Ω_h is defined as in Figure 2.2. For the model problem (2.3.8) with $\alpha = 1$, the corresponding matrix problem (2.3.11) is given by

$$\frac{1}{h^2}\begin{bmatrix} 4 & -1 & 0 & -1 & & & & & \\ -1 & 4 & -1 & 0 & -1 & & & 0 & \\ 0 & -1 & 4 & 0 & 0 & -1 & & & \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & & \\ & -1 & 0 & -1 & 4 & -1 & 0 & -1 & \\ & & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ & & & -1 & 0 & 0 & 4 & -1 & 0 \\ & 0 & & & -1 & 0 & -1 & 4 & -1 \\ & & & & & -1 & 0 & -1 & 4 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{bmatrix} \tag{2.3.12}$$

with

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{bmatrix} = \begin{bmatrix} f_1 + \frac{1}{h^2}(\phi_{11} + \phi_{15}) \\ f_2 + \frac{1}{h^2}\phi_{12} \\ f_3 + \frac{1}{h^2}(\phi_{13} + \phi_{16}) \\ f_4 + \frac{1}{h^2}\phi_{17} \\ f_5 \\ f_6 + \frac{1}{h^2}\phi_{18} \\ f_7 + \frac{1}{h^2}(\phi_{19} + \phi_{22}) \\ f_8 + \frac{1}{h^2}\phi_{23} \\ f_9 + \frac{1}{h^2}(\phi_{20} + \phi_{24}) \end{bmatrix}. \tag{2.3.13}$$

It can be verified that the eigenvectors of $A_h^{(5)}$, for the case of $h = 1/4$, are given by

$$v_h^{(p,q)} = \begin{bmatrix} \sin(p\pi h)\sin(q\pi h) \\ \sin(2p\pi h)\sin(q\pi h) \\ \sin(3p\pi h)\sin(q\pi h) \\ \sin(p\pi h)\sin(2q\pi h) \\ \sin(2p\pi h)\sin(2q\pi h) \\ \sin(3p\pi h)\sin(2q\pi h) \\ \sin(p\pi h)\sin(3q\pi h) \\ \sin(2p\pi h)\sin(3q\pi h) \\ \sin(3p\pi h)\sin(3q\pi h) \end{bmatrix}, \quad p,q = 1,2,3 \qquad (2.3.14)$$

and the corresponding eigenvalues are given by

$$\begin{aligned} \nu_h^{(pq)} &= \tfrac{1}{h^2}(2\alpha - 2\alpha\cos p\pi h + 2 - 2\cos q\pi h), \\ & \quad p,q = 1,2,3. \end{aligned} \qquad (2.3.15)$$

Chapter 3

Iterative Methods

3.1 Introduction

In this chapter we give a brief description of some basic iterative methods and polynomial acceleration procedures for solving large sparse matrix problems arising from finite difference discretizations of elliptic partial differential equations.

We consider the matrix problem

$$Au = b \tag{3.1.1}$$

where A is an $N \times N$ nonsingular matrix and b is an $N \times 1$ column vector.

3.2 Basic Iterative Methods

Let $u^{(0)}$ be a starting vector. A basic iterative method for solving the linear system (3.1.1) can be written in the form

$$u^{(n+1)} = Gu^{(n)} + k \tag{3.2.2}$$

where

$$\begin{cases} G &= I - Q^{-1}A, \\ k &= Q^{-1}b. \end{cases} \tag{3.2.3}$$

Here Q is a nonsingular matrix which is called the *splitting matrix.*

There are two criteria that need to be considered in choosing the matrix Q. First, Q should be "close" to A in some sense. (When $Q = A$, the method will converge after one step.) Second, Q should be a matrix such that $Qx = y$ can be "easily" solved for x for any given y, since in the iteration, the system $Qx = y$ needs to be solved for x. For example, Q can be the diagonal, the tridiagonal, or the triangular part of A.

3.2.1 Richardson Method

The Richardson method, which is probably the simplest iterative method, is defined by

$$u^{(n+1)} = (I - A)u^{(n)} + b. \tag{3.2.4}$$

Here the identity matrix I is the splitting matrix and the iteration matrix is

$$G = I - A. \tag{3.2.5}$$

3.2.2 Jacobi Method

The Jacobi method is defined by

$$u^{(n+1)} = (I - D^{-1}A)u^{(n)} + D^{-1}b. \tag{3.2.6}$$

The splitting matrix Q is given by $Q = D$ where D is the diagonal part of A and the iteration matrix is given by

$$B = I - D^{-1}A. \tag{3.2.7}$$

3.3 Acçeleration of Basic Iterative Methods

In this section we consider the acceleration process for symmetrizable basic iterative methods. An iterative method with an iteration matrix G (3.2.2) is *symmetrizable* if $I - G$ is similar to a symmetric positive definite (SPD) matrix,* i.e. there exists a nonsingular (symmetrization) matrix W such that $W(I - G)W^{-1}$ is SPD.

3.3.1 Extrapolation

A symmetrizable basic iterative method itself is not necessarily convergent because the eigenvalues of G can be less than -1. However, there always exists a so-called extrapolation method based on (3.2.2) which is convergent whenever the basic method is symmetrizable.

The extrapolation method with extrapolation factor γ for any basic iterative method is defined by

$$\begin{aligned} u^{(n+1)} &= \gamma(Gu^{(n)} + k) + (1-\gamma)u^{(n)} && (3.3.8) \\ &= G_{[\gamma]}u^{(n)} + k_{[\gamma]} && (3.3.9) \end{aligned}$$

where

$$\left\{ \begin{aligned} G_{[\gamma]} &= \gamma G + (1-\gamma)I = I - \gamma Q^{-1}A, \\ k_{[\gamma]} &= \gamma Q^{-1}b. \end{aligned} \right. \qquad (3.3.10)$$

From (3.3.10), the splitting matrix $Q_{[\gamma]}$ for an extrapolation method is given by $Q_{[\gamma]} = \frac{1}{\gamma}Q$ where Q is the splitting matrix of the corresponding basic iterative method. If the basic iterative method is symmetrizable, then the *optimum extrapolation factor* γ, in the sense of minimizing the spectral radius of $G_{[\gamma]}$, is given by

$$\bar{\gamma} = \frac{2}{2 - M(G) - m(G)} \qquad (3.3.11)$$

*A real $N \times N$ matrix A is SPD if A is symmetric and if $(v, Av) > 0$ for any nonzero vector v.

where $M(G)$ and $M(G)$ are the largest and smallest eigenvalues of G respectively. (See e.g. Hageman and Young [35].) From (3.3.11) and (3.3.10), the spectral radius of the optimum extrapolation method is given by

$$\mathcal{S}(G_{[\bar{\gamma}]}) = \frac{M(G) - m(G)}{2 - M(G) - m(G)}. \tag{3.3.12}$$

The number of iterations required to reduce the error by a factor of 0.1 can be estimated as (see Hageman and Young [35])

$$n = -(\log_{10} \mathcal{S}(G_{[\bar{\gamma}]}))^{-1} \approx \frac{\mathcal{K}(I - G)}{2} \tag{3.3.13}$$

where $\mathcal{K}(I - G)$ is the *condition number* of the matrix $I - G$.

3.3.2 Polynomial Acceleration

Let $\bar{u} = A^{-1}b$ be the true solution to equation (3.1.1). We define the *error vector* $e^{(n)}$ associated with the n^{th} iterate $u^{(n)}$ of the basic iterative methods (3.2.2) as

$$e^{(n)} = u^{(n)} - \bar{u}. \tag{3.3.14}$$

Since

$$\begin{aligned} \bar{u} &= G\bar{u} + k, \\ u^{(n)} &= Gu^{(n-1)} + k, \end{aligned} \tag{3.3.15}$$

it is easy to show that

$$e^{(n)} = G^n e^{(0)}. \tag{3.3.16}$$

For a symmetrizable basic iterative method with extrapolation, the error vector is given by

$$\begin{aligned} e^{(n)} &= G^n_{[\gamma]} e^{(0)} \\ &= (\gamma G + (1 - \gamma)I)^n e^{(0)}. \end{aligned} \tag{3.3.17}$$

A natural way to generalize the extrapolation procedure (3.3.9) is to use a different value for γ in each iteration instead of a fixed value. The variable extrapolation procedure can be written as

$$u^{(n+1)} = \gamma_{n+1}(Gu^{(n)} + k) + (1 - \gamma_{n+1})u^{(n)}. \tag{3.3.18}$$

If we let λ_i and v_i, $i = 1, \ldots, N$, be the eigenvalues and the eigenvectors of the matrix G respectively and represent $e^{(0)}$ in the form

$$e^{(0)} = \sum_{i=1}^{N} k_i v_i \tag{3.3.19}$$

then from (3.3.18), the n^{th} error of the variable extrapolation procedure can be written as

$$e^{(n)} = P_n(G)e^{(0)} = \sum_{i=1}^{N} P_n(\lambda_i) k_i v_i \tag{3.3.20}$$

where

$$\begin{aligned} P_n(x) &= \prod_{i=1}^{n} [\gamma_i^{(n)} x + (1 - \gamma_i^{(n)})] \\ &= \prod_{i=1}^{n} \frac{x - a_i^{(n)}}{1 - a_i^{(n)}}. \end{aligned} \tag{3.3.21}$$

Here $a_i^{(n)}$ are the zeros of $P_n(x)$ and are given by

$$a_i^{(n)} = 1 + 1/\gamma_i^{(n)}. \tag{3.3.22}$$

We note that $P_n(x)$ is a polynomial of degree n satisfying $P_n(1) = 1$. We denote by $\mathcal{P}_n$ the set of all such polynomials. We seek a polynomial $P_n(x) \in \mathcal{P}_n$ such that

$$\max_{1 \le i \le N} |P_n(\lambda_i)| \le \max_{1 \le i \le N} |Q_n(\lambda_i)| \tag{3.3.23}$$

for any polynomial $Q_n(x) \in \mathcal{P}_n$. Such a $P_n(x)$ is usually called the *optimal polynomial* and the corresponding $\bar{\gamma}_i^{(n)}$ are called the *optimal variable extrapolation factors.*

One commonly used polynomial acceleration is the *Chebyshev semi-iterative method* which is defined by

$$u^{(n+1)} = \rho_{n+1}[\gamma(Gu^{(n)} + k) + (1-\gamma)u^{(n)}] + (1-\rho_{n+1})u^{(n-1)} \tag{3.3.24}$$

where

$$\gamma = \frac{2}{2 - M(G) - m(G)} \tag{3.3.25}$$

$$\sigma = \frac{M(G) - m(G)}{2 - M(G) - m(G)} \tag{3.3.26}$$

$$\rho_{n+1} = \begin{cases} 1 & n = 0 \\ (1 - \frac{\sigma^2}{2})^{-1} & n = 1 \\ (1 - \frac{\sigma^2}{4}\rho_n)^{-1} & n \geq 2. \end{cases} \tag{3.3.27}$$

It can be shown (e.g. Young [70]) that the error reduction matrix of the Chebyshev semi-iterative method can be written in the polynomial form (3.3.20) with the polynomial given by

$$P_n(G) = T_n\left(\frac{2G - M(G) - m(G)}{M(G) - m(G)}\right) \Big/ T_n\left(\frac{2 - M(G) - m(G)}{M(G) - m(G)}\right) \tag{3.3.28}$$

where $T_n(x)$ is the Chebyshev polynomial of degree n and $M(G)$, $m(G)$ are the largest and the smallest eigenvalues of the matrix G.

The polynomial defined in (3.3.28) has an optimal property in the sense that

$$\max_{m(G)\leq x\leq M(G)} |P_n(x)| \leq \max_{m(G)\leq x\leq M(G)} |Q_n(x)| \tag{3.3.29}$$

for any polynomial $Q_n(x) \in \mathcal{P}_n$ It can also be shown (Young, [70]) that

$$\mathcal{S}(P_n(G)) = \max_{m(G)\leq x\leq M(G)} |P_n(x)| = \frac{2r^{n/2}}{1 + r^n} \approx r^{n/2} \tag{3.3.30}$$

where $\mathcal{S}(P_n(G))$ is the *virtual spectral radius* of $P_n(G)$ and

$$r = \frac{1-\sqrt{1-\sigma^2}}{1+\sqrt{1-\sigma^2}}. \tag{3.3.31}$$

The number of semi-iterations required to reduce the error by a factor of 0.1 can be estimated as

$$n = -(\log_{10}|\mathcal{S}(P_n(G))|)^{-1} \approx \frac{\sqrt{\mathcal{K}(I-G)}}{2} \tag{3.3.32}$$

where $\mathcal{K}(I-G)$ is the *condition number* of the matrix $I-G$. Here we assume that $\mathcal{K}(I-G) \gg 1$.

Generally the eigenvalues $M(G)$ and $m(G)$ are not known. In practice, estimated values are used initially and these estimated values can be improved adaptedly during the process. (See Hageman and Young [35]).

3.4 Optimal Iterative Methods

The vectors defined in (2.2.6) and (2.3.14) are also called *Fourier modes.* The integers p and q represent the number of half sine waves which constitute the Fourier modes. Figure 3.1 shows the relationship between the eigenvalues of the extrapolation Jacobi iteration matrix and the Fourier modes (eigenvectors of A) for the one-dimensional problem (2.2.4) with $N = 64$. It illustrates that changing the value of γ can affect the damping factors $|\lambda_p|$ corresponding to the high-frequency modes ($\frac{N}{2} \leq p \leq N-1$).

Although the polynomial acceleration process can improve the convergence rate of the basic iterative methods, there is an intrinsic limitation. The idea of a classical polynomial acceleration is to choose the $P_n(x)$ so that all the coefficients of $e^{(n)}$ are as small as possible. In other words, each of the coefficients $|P_n(\lambda_i)|$ in (3.3.20) should be small. From (3.3.21), it follows that in order to make $|P_n(\lambda_i)|$ small for a given i one could choose $P_n(x)$ so that there is a root $a_k^{(n)}$ near λ_i. However, if λ_i is close to one (low frequency mode),

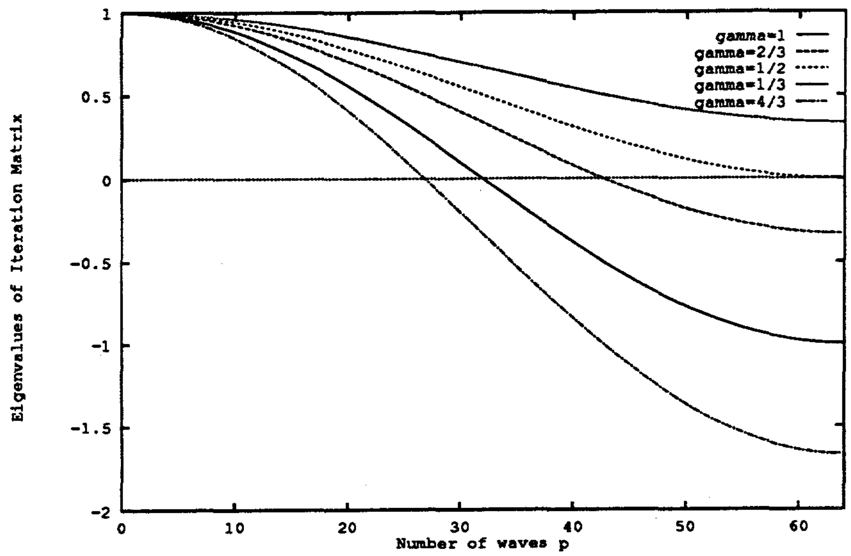


Figure 3.1: Damping Factor of Extrapolation Jacobi vs. Fourier Mode

choosing some $a_k^{(n)}$ near λ_i will introduce large factors $(1 - a_k^{(n)})^{-1}$ for other components. Therefore, to make every $P_n(\lambda_i)$ small, the components related to the eigenvalues close to one cannot be damped rapidly by a polynomial acceleration.

For the 5-point discrete Laplacian, if we use the Richardson method with the optimal polynomial acceleration (Chebyshev acceleration), the number of iterations is on the order of

$$n \approx O(\sqrt{\mathcal{K}(A)}) \approx O(h^{-1}). \tag{3.4.33}$$

3.5 Iterative Methods for Red/Black Systems

In this section, we give a short discussion about some iterative methods for red/black systems. These methods are often used with multigrid methods.

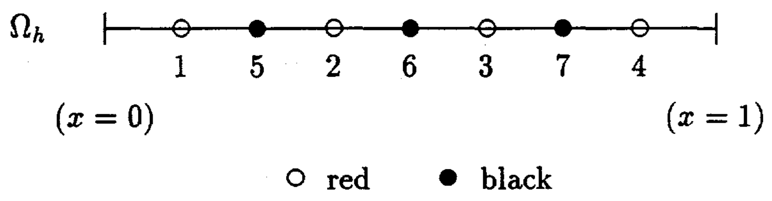


Figure 3.2: Red/Black Ordering of Grid Points in 1D: $h = 1/8$

For the linear systems arising from finite difference discretizations of elliptic partial differential equations, the properties of the matrix A depend on the partitioning and ordering of the unknowns in the problem defined in Section 2.2. The system (2.2.5) corresponds to the *natural ordering* of Ω_h illustrated in Figure 2.1, where x_i follows x_j if $x_i > x_j$. Here we give a brief discussion of an alternate ordering, namely, red/black ordering. In the 2D cases, the standard 5-point discretized scheme is assumed.

In red/black ordering, every other grid point is given the same color (i.e. all the points with an odd sequential index number are marked red and all the points with an even sequential index number are marked black) and then the points are partitioned by their color (e.g. the red points are counted first).

For the model problem (2.2.1) with $N = 8$, one red/black ordering of the grid points is illustrated in Figure 3.2 and the corresponding 3-point finite-difference matrix problem is given by

$$\frac{1}{h^2}\begin{bmatrix} 2 & & & & -1 & 0 & 0 \\ & 2 & & & -1 & -1 & 0 \\ & & 2 & & 0 & -1 & -1 \\ & & & 2 & 0 & 0 & -1 \\ -1 & -1 & 0 & 0 & 2 & & \\ 0 & -1 & -1 & 0 & & 2 & \\ 0 & 0 & -1 & -1 & & & 2 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 + \phi_a/h^2 \\ f_2 \\ f_3 \\ f_4 + \phi_b/h^2 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}. \quad (3.5.34)$$

If we let

$$u_R = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \quad \mathit{and} \quad u_B = \begin{bmatrix} u_5 \\ u_6 \\ u_7 \end{bmatrix} \quad (3.5.35)$$

then (3.5.34) can be written in the form

$$\begin{bmatrix} D_R & H \\ H^T & D_B \end{bmatrix}\begin{bmatrix} u_R \\ u_B \end{bmatrix} = \begin{bmatrix} b_R \\ b_B \end{bmatrix} \quad (3.5.36)$$

where

$$D_R = \frac{2}{h^2} I_4, \quad (3.5.37)$$

$$D_B = \frac{2}{h^2} I_3, \quad (3.5.38)$$

$$H = \frac{1}{h^2}\begin{bmatrix} -1 & 0 & 0 \\ -1 & -1 & 0 \\ 0 & -1 & -1 \\ 0 & 0 & -1 \end{bmatrix}. \quad (3.5.39)$$

Here we use I_n to denote the identity matrix of order n.

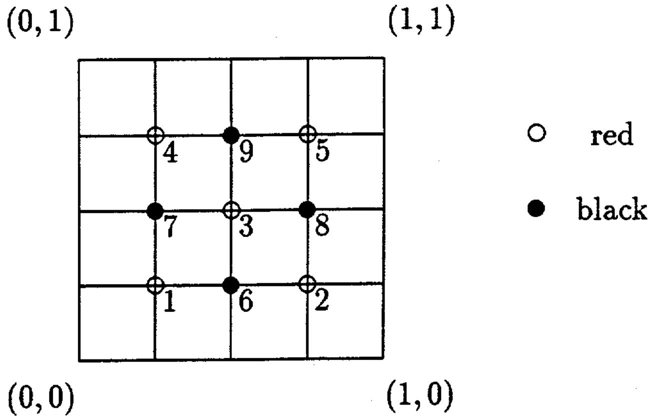


Figure 3.3: Red/Black Ordering of Grid Points in 2D: $h = 1/4$

In the two-dimensional case with $N = 4$, the red/black grid points are illustrated in Figure 3.3. For the model problem (2.3.8) with $\alpha = 1$, the corresponding 5-point finite difference matrix is given by

$$\frac{1}{h^2}\begin{bmatrix} 4 & & & & & -1 & -1 & 0 & 0 \\ & 4 & & & & -1 & 0 & -1 & 0 \\ & & 4 & & & -1 & -1 & -1 & -1 \\ & & & 4 & & 0 & -1 & 0 & -1 \\ & & & & 4 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 4 & & & \\ -1 & 0 & -1 & -1 & 0 & & 4 & & \\ 0 & -1 & -1 & 0 & -1 & & & 4 & \\ 0 & 0 & -1 & -1 & -1 & & & & 4 \end{bmatrix}. \qquad (3.5.40)$$

Red/Black Gauss-Seidel (RBGS) Method

The Gauss-Seidel iteration with red/black ordering is given by

$$\begin{cases} u_R^{(n+1)} &= D_R^{-1} H u_B^{(n)} + D_R^{-1} b_R, \\ u_B^{(n+1)} &= D_B^{-1} H^T u_R^{(n+1)} + D_B^{-1} b_B. \end{cases} \qquad (3.5.41)$$

One can see from (3.5.41) that all the unknowns with the same color can be updated simultaneously with red/black ordering. Therefore the RBGS procedure can be carried out very efficiently on a vector/parallel machine.

Red/Black Successive Overrelaxation (RBSOR) Method

The SOR iteration with red/black ordering is given by

$$\left\{ \begin{array}{lcl} u_R^{(n+1)} & = & \omega(D_R^{-1} H u_B^{(n)} + D_R^{-1} b_R) + (1-\omega) u_R^{(n)}, \\ u_B^{(n+1)} & = & \omega(D_B^{-1} H^T u_R^{(n+1)} + D_B^{-1} b_B) + (1-\omega) u_B^{(n)}. \end{array} \right. \tag{3.5.42}$$

Like the RBGS method, the SOR method with red/black ordering (RBSOR) can be carried out with a high degree of parallelism.

In the next chapter, we will discuss another kind of acceleration technique, namely the standard multigrid technique which can substantially reduce the components of the error corresponding to the low frequencies without amplifying the other components too much. The number of cycles needed for convergence will be $O(1)$ which is independent of h.

Chapter 4

Standard Multigrid Method in 1D

4.1 Introduction

In this chapter, we give a brief introduction to the standard multigrid method (MG), and an analysis of the convergence properties of the method using standard Fourier analysis for the one-dimensional Poisson model problem. We also give an alternative analysis based on a two-color Fourier analysis procedure. We show that this procedure can also be used to analyze the standard multigrid method where a red/black ordering iterative method is used as the smoothing procedure. In later chapters, this alternative analysis will also be used to analyze a multiple coarse grid multigrid method.

4.2 Standard Multigrid Method

The standard multigrid algorithm consists of several pre-smoothing iterations, a *coarse grid correction procedure* and several post-smoothing iterations. The smoothing iterations are carried out by a *smoothing iterative method* which is usually a basic iterative method. The coarse grid correction procedure can be described as follows.

Given an initial guess $u_h^{(0)}$ of the system

$$A_h u_h = b_h, \tag{4.2.1}$$

we wish to solve the correction equation

$$A_h \delta_h = r_h = b_h - A_h u_h^{(0)} \tag{4.2.2}$$

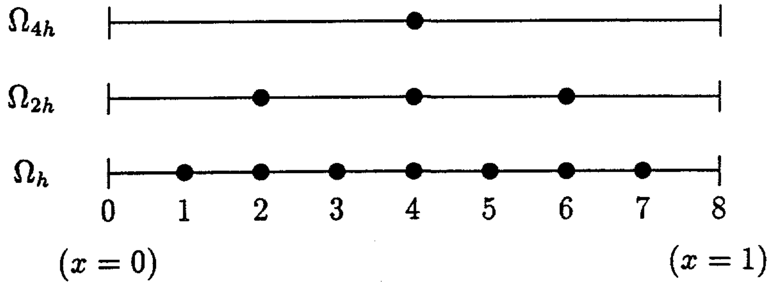


Figure 4.1: Multigrids in 1D: $N = 8$

for the correction

$$\delta_h = \bar{u}_h - u_h^{(0)} \tag{4.2.3}$$

where $\bar{u}_h$ is the true solution of (4.2.1). If we obtain the solution δ_h of (4.2.2), the solution of the original problem (4.2.1) will be $u_h^{(0)} + \delta_h$.

Instead of attempting to solve the correction equation (4.2.2) on the original grid, we solve it on a coarse grid. The coarse grid usually consists of every other point of the fine grid and the distance between two adjacent points is twice as great as on the fine grid. For the case $N = 8$, the coarse grids are shown in Figure 4.1, where there are three levels of grids Ω_h, Ω_{2h}, and Ω_{4h}.

First, we restrict the residual to the coarse grid. The simplest restriction operator is an injection which is defined by

$$r_{2h}(x) = (R_h r_h)(x) = r_h(x), \qquad x \in \Omega_{2h}. \tag{4.2.4}$$

An alternate restriction operator is called *full weighting* which is defined by

$$\begin{aligned} r_{2h}(x) &= (R_h r_h)(x) \\ &= \frac{1}{4}(r_h(x-h) + 2r_h(x) + r_h(x+h)), \qquad x \in \Omega_{2h}. \end{aligned} \tag{4.2.5}$$

The next step is to solve the coarse grid correction equation

$$A_{2h}\delta_{2h} = r_{2h} \tag{4.2.6}$$

for $\delta^{(2h)}$. Here the coarse grid matrix A_{2h} is created by using the standard finite difference discretization for the original partial differential equation on the coarse grid. The coarse grid equation (4.2.6) itself can be solved using this procedure based on an even coarser grid.

Finally, we interpolate the correction $\delta^{(2h)}$ onto the fine grid and add the result vector to the old solution. A commonly used interpolation scheme is linear interpolation which is defined by

$$\delta_h(x) = (P_h\delta_{2h})(x) = \begin{cases} \delta_{2h}(x) & x \in \Omega_{2h} \\ \frac{1}{2}(\delta_{2h}(x-h) + \delta_{2h}(x+h)) & x \notin \Omega_{2h} \end{cases} \tag{4.2.7}$$

where we assume that $\delta_{2h}(0) = 0$ and $\delta_{2h}(1) = 0$.

For the model problem (2.2.4) with $N = 8$, the full weighting restriction of the residual on the finest grid is given by

$$r_{2h} = \begin{bmatrix} r_{2h}(x_2) \\ r_{2h}(x_4) \\ r_{2h}(x_6) \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} r_h(x_1) \\ r_h(x_2) \\ r_h(x_3) \\ r_h(x_4) \\ r_h(x_5) \\ r_h(x_6) \\ r_h(x_7) \end{bmatrix} = R_h r_h. \tag{4.2.8}$$

The coarse grid matrix on Ω_{2h} is given by

$$A_{2h} = \frac{1}{(2h)^2}\begin{bmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \end{bmatrix}. \tag{4.2.9}$$

The interpolation of the correction vector δ_{2h} onto Ω_h can be written in the form

$$\delta_h = \begin{bmatrix} \delta_h(x_1) \\ \delta_h(x_2) \\ \delta_h(x_3) \\ \delta_h(x_4) \\ \delta_h(x_5) \\ \delta_h(x_6) \\ \delta_h(x_7) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & 1 \\ & & 2 \\ & & 1 \end{bmatrix} \begin{bmatrix} \delta_{2h}(x_2) \\ \delta_{2h}(x_4) \\ \delta_{2h}(x_6) \end{bmatrix} = P_h\delta_{2h}. \qquad (4.2.10)$$

This two-level standard multigrid algorithm, for the solution of $A_h u_h = b_h$, starting with an initial guess $u_h^{(0)}$, is described by

Algorithm SMG$(A_h, u_h^{(0)}, b_h)$:

1. Do m_1 pre-smoothing iterations using the smoothing iterative method (a basic iterative method) to obtain u_h'.

2. Compute the residual $r_h = b_h - A_h u_h'$ and restrict the residual to the coarse grid to obtain

$$r_{2h} = R_h r_h. \qquad (4.2.11)$$

3. Solve the coarse grid system

$$A_{2h}\delta_{2h} = r_{2h} \qquad (4.2.12)$$

4. Interpolate the coarse grid correction δ_{2h} onto the fine grid and obtain the new approximate solution

$$u_h'' = u_h' + P_h\delta_{2h}. \qquad (4.2.13)$$

5. Do m_2 post-smoothing iterations using the smoothing iterative method to obtain and return $u_h^{(1)}$.

The procedure from step 2 to step 4 corresponds to the coarse grid correction.

If we let $e_h^{(0)} = u_h^{(0)} - \bar{u}_h$ be the error before the coarse grid correction and $e_h^{(1)} = u_h^{(1)} - \bar{u}_h$ be the error after the coarse grid correction, where $\bar{u}_h = A_h^{-1} b_h$, then from (4.2.11) to (4.2.13) we have

$$\begin{aligned} u_h^{(1)} &= u_h^{(0)} + P_h \delta_{2h} \\ &= u_h^{(0)} + P_h A_{2h}^{-1} r_{2h} \\ &= u_h^{(0)} + P_h A_{2h}^{-1} R_h r_h \\ &= u_h^{(0)} + P_h A_{2h}^{-1} R_h A_h (-e_h^{(0)}) \end{aligned} \tag{4.2.14}$$

and

$$e_h^{(1)} = (I - P_h A_{2h}^{-1} R_h A_h) e_h^{(0)} = C_h e_h^{(0)}. \tag{4.2.15}$$

Here, we use C_h to denote the coarse grid correction matrix. If G is the iterative matrix of the smoothing iterative method, the matrix of the standard two-level multigrid method T_h can then be expressed as

$$T_h = G^{m_2} C_h G^{m_1}. \tag{4.2.16}$$

4.3 Standard Fourier Analysis

In this section, we present the standard Fourier analysis of the two-level standard multigrid method for the matrix problem (2.2.4). We use the full weighting restriction defined in (4.2.5) with the corresponding matrix R_h and the linear interpolation defined in (4.2.7) with the matrix P_h. Most basic iterative methods can be used for smoothing iterations. For simplicity, we use the damped Jacobi method with the iteration matrix

$$B_{[\gamma]} = I - \gamma D^{-1} A_h, \tag{4.3.17}$$

where γ is the damping factor. For any of the matrices which we will consider in the analysis, say Z, it can be shown that

$$Zv^{(p)} = z_{11}^{(p)}v^{(p)} + z_{12}^{(p)}v^{(N-p)} \tag{4.3.18}$$

and

$$Zv^{(N-p)} = z_{21}^{(p)}v^{(p)} + z_{22}^{(p)}v^{(N-p)} \tag{4.3.19}$$

for some values $z_{11}^{(p)}$, $z_{12}^{(p)}$, $z_{21}^{(p)}$, $z_{22}^{(p)}$ depending on p. Therefore we can write

$$Z(v_h^{(p)}, v_h^{(N-p)}) = (v_h^{(p)}, v_h^{(N-p)})\hat{Z}_v^{(p)} \tag{4.3.20}$$

where

$$\hat{Z}_v^{(p)} = \begin{bmatrix} z_{11}^{(p)} & z_{12}^{(p)} \\ z_{21}^{(p)} & z_{22}^{(p)} \end{bmatrix}. \tag{4.3.21}$$

Also, we say that the subspace $E^{(p)}$ spanned by $v_h^{(p)}$ and $v_h^{(N-p)}$ is *invariant* under Z. The matrix $\hat{Z}_v^{(p)}$ is called the *v-transform matrix* because in some sense it can be regarded as a kind of "transform" of the matrix Z on the v-basis vectors $v_h^{(p)}$ and $v_h^{(N-p)}$.

The eigenvectors of the coarse grid matrix A_{2h}, $v_{2h}^{(p)}$, are the projections of the fine grid eigenvectors $v_h^{(p)}$ onto the coarse grid. Thus in the case of $N = 8$, we have the coarse grid eigenvectors

$$v_{2h}^{(p)} = \begin{bmatrix} \sin(2p\pi h) \\ \sin(4p\pi h) \\ \sin(6p\pi h) \end{bmatrix}, \qquad p = 1, 2, 3. \tag{4.3.22}$$

and the corresponding eigenvalues

$$\nu_{2h}^{(p)} = \frac{1}{(2h)^2}(2 - 2\cos 2p\pi h). \tag{4.3.23}$$

Without loss of generality, we assume that N is even. If we let $p' = N - p$, for $p = 1, \ldots, \frac{N}{2}$, we can write

$$A_h(v_h^{(p)}, v_h^{(p')}) = (v_h^{(p)}, v_h^{(p')})\hat{A}_h^{(p)}, \tag{4.3.24}$$

$$R_h(v_h^{(p)}, v_h^{(p')}) = v_{2h}^{(p)}\hat{R}_h^{(p)}, \tag{4.3.25}$$

$$A_{2h}v_{2h}^{(p)} = v_{2h}^{(p)}\hat{A}_{2h}^{(p)}, \tag{4.3.26}$$

$$P_h v_{2h}^{(p)} = (v_h^{(p)}, v_h^{(p')})\hat{P}_h^{(p)}, \tag{4.3.27}$$

$$B_{[\gamma]}(v_h^{(p)}, v_h^{(p')}) = (v_h^{(p)}, v_h^{(p')})\hat{B}_{[\gamma]}^{(p)} \tag{4.3.28}$$

where

$$\hat{A}_h^{(p)} = \frac{1}{h^2}\begin{bmatrix} 2 - 2c_p & 0 \\ 0 & 2 + 2c_p \end{bmatrix}, \tag{4.3.29}$$

$$\hat{R}_h^{(p)} = \frac{1}{2}\begin{bmatrix} 1 + c_p & c_p - 1 \end{bmatrix}, \tag{4.3.30}$$

$$\hat{A}_{2h}^{(p)} = \frac{1}{4h^2}(2 - 2\cos 2p\pi h), \tag{4.3.31}$$

$$\hat{P}_h^{(p)} = \frac{1}{2}\begin{bmatrix} 1 + c_p \\ c_p - 1 \end{bmatrix}, \tag{4.3.32}$$

$$\hat{B}_{[\gamma]}^{(p)} = \begin{bmatrix} 1 - \gamma(1 - c_p) & 0 \\ 0 & 1 - \gamma(1 + c_p) \end{bmatrix} \tag{4.3.33}$$

with

$$c_p = \cos p\pi h. \tag{4.3.34}$$

From (4.2.15) and (4.3.24) through (4.3.33), we obtain the v-transform matrices of the coarse grid correction operator

$$\hat{C}_h^{(p)} = I - \hat{P}_h^{(p)}(\hat{A}_{2h}^{(p)})^{-1}\hat{R}_h^{(p)}\hat{A}_h^{(p)} = \frac{1}{2}\begin{bmatrix} 1-c_p & 1+c_p \\ 1-c_p & 1+c_p \end{bmatrix} \tag{4.3.35}$$

and the two level multigrid operator

$$\hat{T}_h^{(p)} = (\hat{B}_{[\gamma]}^{(p)})^{m_2}\hat{C}_h^{(p)}(\hat{B}_{[\gamma]}^{(p)})^{m_2} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \tag{4.3.36}$$

where

$$\begin{aligned} t_{11} &= \frac{1}{2}(1-c_p)(1-\gamma+\gamma c_p)^{m_1+m_2}, \\ t_{12} &= \frac{1}{2}(1+c_p)(1-\gamma+\gamma c_p)^{m_2}(1-\gamma-\gamma c_p)^{m_1}, \\ t_{21} &= \frac{1}{2}(1-c_p)(1-\gamma+\gamma c_p)^{m_1}(1-\gamma-\gamma c_p)^{m_2}, \\ t_{22} &= \frac{1}{2}(1+c_p)(1-\gamma-\gamma c_p)^{m_1+m_2}. \end{aligned} \tag{4.3.37}$$

From (4.3.35) it is easy to see that the determinant of the matrix $\hat{C}_h^{(p)}$ is zero. Because of (4.3.36), we also have

$$\det(\hat{T}_h^{(p)}) = 0. \tag{4.3.38}$$

Hence the eigenvalues of $\hat{T}_h^{(p)}$ are 0 and trace$(\hat{T}_h^{(p)})$. Therefore, the nonzero eigenvalue of the matrix $\hat{T}_h^{(p)}$ is given by

$$\lambda_p = \mathrm{trace}\,(\hat{T}_h^{(p)}). \tag{4.3.39}$$

Suppose that the initial error has the expansion

$$e^{(0)} = \sum_{p=1}^{N-1} d_p v_h^{(p)}. \tag{4.3.40}$$

Then after one multigrid cycle the new error is given by

$$e^{(1)} = T_h e^{(0)} = \sum_{p=1}^{N-1} d_p^* v_h^{(p)}, \tag{4.3.41}$$

i.e. for $p = 1, \ldots, N/2$, we have

$$\begin{bmatrix} d_p^* \\ d_{N-p}^* \end{bmatrix} = \hat{T}_h^{(p)} \begin{bmatrix} d_p \\ d_{N-p} \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} d_p \\ d_{N-p} \end{bmatrix} \tag{4.3.42}$$

where the t_{ij} are defined in (4.3.37). The value t_{12} (t_{21}) represents the *aliasing* from mode $v_h^{(N-p)}$ ($v_h^{(p)}$) to mode $v_h^{(p)}$ ($v_h^{(N-p)}$).

When the extrapolated Jacobi method with extrapolation factor $\gamma = 2/3$ is used for the smoothing iteration, from (4.3.37) we have

$$\hat{T}_h^{(p)} = \begin{bmatrix} \frac{1-c_p}{2}(\frac{1+2c_p}{3})^{m_1+m_2} & \frac{1+c_p}{2}(\frac{1+2c_p}{3})^{m_2}(\frac{1-2c_p}{3})^{m_1} \\ \frac{1-c_p}{2}(\frac{1+2c_p}{3})^{m_1}(\frac{1-2c_p}{3})^{m_2} & \frac{1+c_p}{2}(\frac{1-2c_p}{3})^{m_1+m_2} \end{bmatrix}. \tag{4.3.43}$$

The trace of the matrix $\hat{T}_h^{(p)}$ is given by

$$\text{trace}\,(\hat{T}_h^{(p)}) = \frac{1-c_p}{2}(\frac{1+2c_p}{3})^{m_1+m_2} + \frac{1+c_p}{2}(\frac{1-2c_p}{3})^{m_1+m_2}. \tag{4.3.44}$$

For trace($\hat{T}_h^{(p)}$), if p is small (corresponding to the low frequency modes), then $c_p \approx 1$, $\frac{1-c_p}{2}$ is small and $|\frac{1-2c_p}{3}| \leq \frac{1}{3}$. Also if p is large (corresponding to the high frequency modes),$\frac{1-c_p}{2} \approx 1$, but $\frac{1+c_p}{2}$ is small and $|\frac{1+2c_p}{3}| \leq \frac{1}{3}$.

The *convergence factor* of the two-level multigrid method is defined as the spectral radius $\rho(T_h)$. Since the spectral radius of the matrix $\hat{T}_h^{(p)}$ is the absolute value of its trace, we have

$$\rho(T_h) = \max_{1 \leq p \leq \frac{N}{2}} \text{trace}\,(\hat{T}_h^{(p)}). \tag{4.3.45}$$

Table 4.1: Two-Level Convergence Factors ρ_m of 1D MG-Jacobi

γ	(m_1, m_2)			
	(0,1)	(1,1)	(1,2)	(1,3)
0.5	.4997	.2499	.1250	.08331
0.6	.3997	.1599	.08798	.06819
0.7	.3984	.2198	.09329	.06016
0.8	.5981	.4293	.2543	.1593
0.9	.7979	.6885	.5391	.4377
1.0	.9977	.9977	.9953	.9953

If we let $m = m_1 + m_2$, the upper bound of the convergence factor of the two-level SMG is given by

$$\rho(T_h) \leq \max_{1 \leq p \leq N/2} \text{trace}\,(\hat{T}_h^{(p)}) = \rho_m. \tag{4.3.46}$$

Table 4.1 lists the two-level convergence factor ρ_m for different values of the extrapolation factor γ in the case of $N = 64$. As the number of smoothing iterations m increases, less convergence improvement is obtained because the smoothing iterations cannot reduce the low frequency modes effectively.

If we let η be the ratio of the work required for carrying out coarse grid correction to the work required for carrying out one smoothing iteration, then the optimal m should be chosen to maximize the function

$$\Phi_\eta(m) = \frac{-\ln \rho_m}{\eta + m}. \tag{4.3.47}$$

When η is large, the optimal m will be large, and when η is small, the optimal m will be small. Table 4.2 lists the values of $\Phi_\eta(m)$ and m for $\eta = 1, 2, 3, 4$. For the cases listed in Table 4.2, $m = 2$ is the best choice.

Table 4.2: $\Phi_\eta(m)$ vs. m

m	$\Phi_1(m)$	$\Phi_2(m)$	$\Phi_3(m)$	$\Phi_4(m)$
1	.5493	.3662	.2747	.2197
2	.7323	.5493	.4394	.3662
3	.6355	.5084	.4237	.3631
4	.5580	.4650	.3986	.3488

4.4 Two-Color Fourier Analysis

In this section we describe a two-color Fourier procedure for analyzing the convergence properties of the two-level standard multigrid method for problem (2.2.4). This is an alternative to the standard Fourier analysis. Although this analysis gives the same result, it will be more effective for the analysis of other schemes.

We consider two sets of grid points, which we refer to as *red points* (Ω_+) and *black points* (Ω_-). Instead of carrying out the analysis in terms of the eigenvectors $v_h^{(p)}$ for $p = 1, 2, \ldots, N-1$ corresponding to the fine grid we work in terms of vectors $w_h^{(+,p)}$, $w_h^{(-,p)}$ where $p = 1, 2, \ldots, N/2$, which are the projections of $v_h^{(p)}$ onto Ω_+ and Ω_- respectively.

4.4.1 The Two Coarse Grids

In the one dimensional case, the fine grid Ω_h defined in (2.2.2) can be partitioned into two coarse grids as *red points*:

$$\Omega_+ = \{x_j \mid x_j \in \Omega_h \text{ and } (j = \text{odd})\}, \tag{4.4.48}$$

and *black points*

$$\Omega_- = \{x_j \mid x_j \in \Omega_h \text{ and } (j = \text{even})\}. \tag{4.4.49}$$

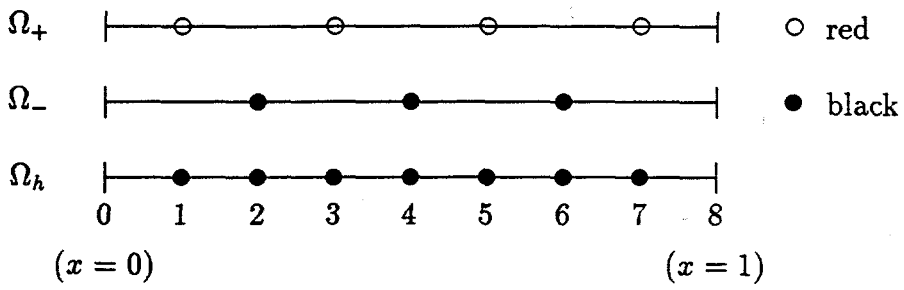


Figure 4.2: Two Coarse Grids for a One-Dimensional Problem: $h = 1/8$

Figure 4.2 illustrates these two sets of grid points for $N = 8$.

The vectors $w_h^{(+,p)}$ and $w_h^{(-,p)}$ are defined by

$$(w_h^{(+,p)})_i = \begin{cases} (v_h^{(p)})_i & \text{if } x_i \text{ is red} \\ 0 & \text{if } x_i \text{ is black} \end{cases} \tag{4.4.50}$$

and

$$(w_h^{(-,p)})_i = \begin{cases} 0 & \text{if } x_i \text{ is red} \\ (v_h^{(p)})_i & \text{if } x_i \text{ is black} \end{cases} \tag{4.4.51}$$

For the case of $N = 8$, we have

$$w_h^{(+,p)} = \begin{bmatrix} \sin(p\pi h) \\ 0 \\ \sin(3p\pi h) \\ 0 \\ \sin(5p\pi h) \\ 0 \\ \sin(7p\pi h) \end{bmatrix}, \qquad p = 1, \ldots, 4 \tag{4.4.52}$$

and

$$w_h^{(-,p)} = \begin{bmatrix} 0 \\ \sin(2p\pi h) \\ 0 \\ \sin(4p\pi h) \\ 0 \\ \sin(6p\pi h) \\ 0 \end{bmatrix}, \qquad p = 1, 2, 3. \tag{4.4.53}$$

For convenience of discussion, we define $w_h^{(-,N/2)}$ to be the zero vector.

We notice that the black grid Ω_- is the same as the coarse grid Ω_{2h} for the standard multigrid method. It is convenient to write the coarse grid matrix in an expanded form so that it can be applied to the vectors $w_h^{(-,p)}$. For example, the expanded coarse grid difference matrix defined in (4.2.9) can be written as

$$A_{2h}^{(E)} = \frac{1}{(2h)^2} \begin{bmatrix} 0 & 0 & 0 & & & & \\ 0 & 2 & 0 & -1 & & 0 & \\ 0 & 0 & 0 & 0 & 0 & & \\ & -1 & 0 & 2 & 0 & -1 & \\ & & 0 & 0 & 0 & 0 & 0 \\ & 0 & & -1 & 0 & 2 & 0 \\ & & & & 0 & 0 & 0 \end{bmatrix}. \tag{4.4.54}$$

Here, we use the superscript "E" to indicate the expanded matrix. Similarly, we can write the expanded restriction matrix and the expanded interpolation matrix as

$$R_h^{(E)} = \frac{1}{4} \left[\begin{array}{ccccccc} 0 & 0 & & & & & \\ 1 & 2 & 1 & & & 0 & \\ & 0 & 0 & 0 & & & \\ & & 1 & 2 & 1 & & \\ & & & 0 & 0 & 0 & \\ & 0 & & & 1 & 2 & 1 \\ & & & & & 0 & 0 \end{array} \right] \tag{4.4.55}$$

and

$$P_h^{(E)} = 2(R_h^{(E)})^T = \frac{1}{2} \left[\begin{array}{ccccccc} 0 & 1 & & & & & \\ 0 & 2 & 0 & & & 0 & \\ & 1 & 0 & 1 & & & \\ & & 0 & 2 & 0 & & \\ & & & 1 & 0 & 1 & \\ & 0 & & & 0 & 2 & 0 \\ & & & & & 1 & 0 \end{array} \right] \tag{4.4.56}$$

respectively.

4.4.2 Convergence Analysis

For the two-color Fourier analysis we use the vectors $w_h^{(+,p)}$ and $w_h^{(-,p)}$ corresponding to the red points (4.4.48) and the black points (4.4.49) respectively as a basis for the invariant subspace $E^{(p)}$ defined in Section 4.3. For any matrix Z with an invariant subspace $E^{(p)}$, we can write

$$Z(w_h^{(+,p)}, w_h^{(-,p)}) = (w_h^{(+,p)}, w_h^{(-,p)})\hat{Z}_w^{(p)}. \tag{4.4.57}$$

where $\hat{Z}_w^{(p)}$ is a 2×2 *w-transform matrix* because it can be regarded as a kind of "transform" of the matrix T corresponding to the w basis. Therefore, for $p = 1, \ldots, \frac{N}{2}$, we have

$$A_h(w_h^{(+,p)}, w_h^{(-,p)}) = (w_h^{(+,p)}, w_h^{(-,p)})\hat{A}_{h,w}^{(p)}, \tag{4.4.58}$$

$$R_h^{(E)}(w_h^{(+,p)}, w_h^{(-,p)}) = w_h^{(-,p)}\hat{R}_{h,w}^{(p)}, \tag{4.4.59}$$

$$A_{2h}^{(E)}w_h^{(-,p)} = w_h^{(-,p)}\hat{A}_{2h,w}^{(p)}, \tag{4.4.60}$$

$$P_h^{(E)}w_h^{(-,p)} = (w_h^{(+,p)}, w_h^{(-,p)})\hat{P}_{h,w}^{(p)}, \tag{4.4.61}$$

$$B_{[\gamma]}(w_h^{(+,p)}, w_h^{(-,p)}) = (w_h^{(+,p)}, w_h^{(-,p)})\hat{B}_{[\gamma],w}^{(p)}, \tag{4.4.62}$$

where

$$\hat{A}_{h,w}^{(p)} = \frac{1}{h^2}\begin{bmatrix} 2 & -2c_p \\ -2c_p & 2 \end{bmatrix}, \tag{4.4.63}$$

$$\hat{R}_{h,w}^{(p)} = \frac{1}{2}\begin{bmatrix} c_p & 1 \end{bmatrix}, \tag{4.4.64}$$

$$\hat{A}_{2h,w}^{(p)} = \frac{1}{4h^2}(2 - 2\cos 2p\pi h), \tag{4.4.65}$$

$$\hat{P}_{h,w}^{(p)} = \begin{bmatrix} c_p \\ 1 \end{bmatrix}, \tag{4.4.66}$$

and

$$\hat{B}_{[\gamma],w}^{(p)} = \begin{bmatrix} 1-\gamma & \gamma c_p \\ \gamma c_p & 1-\gamma \end{bmatrix}. \tag{4.4.67}$$

Here $c_p = \cos p\pi h$.

From (4.4.58) through (4.4.67), the w-transform matrix of the coarse grid correction operator on the subspace $E^{(p)}$ can be written as

$$\hat{C}^{(p)}_{h,w} = I - \hat{P}^{(p)}_{h,w}(\hat{A}^{(p)}_{2h,w})^{-1}\hat{R}^{(p)}_{h,w}\hat{A}^{(p)}_{h,w} = \begin{bmatrix} 1 & -c_p \\ 0 & 0 \end{bmatrix}. \tag{4.4.68}$$

If we let

$$\begin{aligned} \xi &= 1-\gamma+\gamma c_p \\ \eta &= 1-\gamma-\gamma c_p \end{aligned} \tag{4.4.69}$$

we have

$$\hat{B}^{(p)}_{[\gamma],w})^m = \frac{1}{4}\begin{bmatrix} \xi^m+\eta^m & \xi^{m_1}-\eta^m \\ \xi^{m_1}-\eta^m & \xi^m+\eta^m \end{bmatrix} \tag{4.4.70}$$

and the w-transform matrix of the two-level multigrid operator on the subspace $E^{(p)}$ can be written as

$$\hat{T}^{(p)}_{h,w} = (\hat{B}^{(p)}_{[\gamma],w})^{m_2}\hat{C}^{(p)}_{h,w}(\hat{B}^{(p)}_{[\gamma],w})^{m_1} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \tag{4.4.71}$$

where

$$\begin{aligned} t_{11} &= \frac{1}{4}(\xi^{m_2}+\eta^{m_2})((\xi^{m_1}+\eta^{m_1})-c_p(\xi^{m_1}-\eta^{m_1})), \\ t_{12} &= \frac{1}{4}(\xi^{m_2}+\eta^{m_2})((\xi^{m_1}-\eta^{m_1})-c_p(\xi^{m_1}+\eta^{m_1})), \\ t_{21} &= \frac{1}{4}(\xi^{m_2}-\eta^{m_2})((\xi^{m_1}+\eta^{m_1})-c_p(\xi^{m_1}-\eta^{m_1})), \\ t_{21} &= \frac{1}{4}(\xi^{m_2}-\eta^{m_2})((\xi^{m_1}-\eta^{m_1})-c_p(\xi^{m_1}+\eta^{m_1})). \end{aligned} \tag{4.4.72}$$

Since $\det(\hat{C}^{(p)}_{h,w}) = 0$, we have

$$\det(\hat{T}^{(p)}_{h,w}) = 0. \tag{4.4.73}$$

Therefore, for each p, the nonzero eigenvalue of the matrix $\hat{T}_{h,w}^{(p)}$ is the trace of $\hat{T}_{h,w}^{(p)}$. If we let the initial error be expressed in the form

$$e^{(0)} = \sum_{p=1}^{N/2} (k_{+,p} w_h^{(+,p)} + k_{-,p} w_h^{(-,p)}), \tag{4.4.74}$$

then the error $e^{(1)}$ after one multigrid cycle is given by

$$e^{(1)} = T_h e^{(0)} = \sum_{p=1}^{N/2} (k_{+,p}^{*} w_h^{(+,p)} + k_{-,p}^{*} w_h^{(-,p)}), \tag{4.4.75}$$

and for $p = 1, \ldots, N/2$, we have

$$\begin{bmatrix} k_{+,p}^{*} \\ k_{-,p}^{*} \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} k_{+,p} \\ k_{-,p} \end{bmatrix}. \tag{4.4.76}$$

where the t_{ij} are defined in (4.4.72). In the case with the extrapolation factor $\gamma = 2/3$, we have

$$\begin{aligned}
t_{11} &= \frac{1}{4}\left\{\left(\frac{1+2c_p}{3}\right)^{m_1} + \left(\frac{1-2c_p}{3}\right)^{m_1} - c_p\left[\left(\frac{1+2c_p}{3}\right)^{m_1} - \left(\frac{1-2c_p}{3}\right)^{m_1}\right]\right\} \\
&\quad \left\{\left(\frac{1+2c_p}{3}\right)^{m_2} + \left(\frac{1-2c_p}{3}\right)^{m_2}\right\}, \\
t_{12} &= \frac{1}{4}\left\{\left(\frac{1+2c_p}{3}\right)^{m_1} - \left(\frac{1-2c_p}{3}\right)^{m_1} - c_p\left[\left(\frac{1+2c_p}{3}\right)^{m_1} + \left(\frac{1-2c_p}{3}\right)^{m_1}\right]\right\} \\
&\quad \left\{\left(\frac{1+2c_p}{3}\right)^{m_2} + \left(\frac{1-2c_p}{3}\right)^{m_2}\right\}, \\
t_{21} &= \frac{1}{4}\left\{\left(\frac{1+2c_p}{3}\right)^{m_1} + \left(\frac{1-2c_p}{3}\right)^{m_1} - c_p\left[\left(\frac{1+2c_p}{3}\right)^{m_1} - \left(\frac{1-2c_p}{3}\right)^{m_1}\right]\right\} \\
&\quad \left\{\left(\frac{1+2c_p}{3}\right)^{m_2} - \left(\frac{1-2c_p}{3}\right)^{m_2}\right\}, \\
t_{22} &= \frac{1}{4}\left\{\left(\frac{1+2c_p}{3}\right)^{m_1} - \left(\frac{1-2c_p}{3}\right)^{m_1} - c_p\left[\left(\frac{1+2c_p}{3}\right)^{m_1} + \left(\frac{1-2c_p}{3}\right)^{m_1}\right]\right\} \\
&\quad \left\{\left(\frac{1+2c_p}{3}\right)^{m_2} - \left(\frac{1-2c_p}{3}\right)^{m_2}\right\}
\end{aligned} \tag{4.4.77}$$

and the trace $(t_{11} + t_{22})$, as expected, is the same as the trace of the matrix $\hat{T}_h^{(p)}$ defined in (4.3.44). Thus we get the same result as that obtained from the conventional Fourier analysis.

4.5 Relation Between the v-basis and w-basis

In this section, we discuss the relationship between the v-basis corresponding to the standard Fourier analysis and the w-basis used in the two-color Fourier analysis. By comparing the definition of the v-basis (2.2.6) and of the w-basis (4.4.50) and (4.4.51), we have the following relations:

$$\left\{ \begin{array}{lcl} v_h^{(p)} & = & w_h^{(+,p)} + w_h^{(-,p)} \\ v_h^{(N-p)} & = & w_h^{(+,p)} - w_h^{(-,p)} \end{array} \right. \qquad p = 1, \ldots, \frac{N}{2}. \tag{4.5.78}$$

The relation between the v-basis and the w-basis (4.5.78) can be written in the matrix form

$$(v_h^{(p)}, v_h^{(N-p)}) = (w_h^{(+,p)}, w_h^{(-,p)})H_1 \tag{4.5.79}$$

where

$$H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{4.5.80}$$

We note that

$$H_1^{-1} = \frac{1}{2}H_1. \tag{4.5.81}$$

If we let Z be any matrix of order $N-1$ with invariant subspaces $E^{(p)}$ spanned by the eigenvectors $v_h^{(p)}$ and $v_h^{(N-p)}$, and $\hat{Z}_v^{(p)}$ and $\hat{Z}_w^{(p)}$ (4.3.20) be the transform matrices associated with the v-basis and the w-basis respectively, then we have the following result.

Lemma 4.1 *Let $\hat{Z}_v^{(p)}$ and $\hat{Z}_w^{(p)}$ be defined as in (4.3.20) and (4.4.57) respectively. Then the following relation holds:*

$$\hat{Z}_v^{(p)} = \frac{1}{2}H_1\hat{Z}_w^{(p)}H_1. \tag{4.5.82}$$

Proof: By using the relation (4.5.79), we have

$$\begin{aligned} Z(v_h^{(p)}, v_h^{(N-p)}) &= Z(w_h^{(+,p)}, w_h^{(-,p)})H_1 \\ &= (w_h^{(+,p)}, w_h^{(-,p)})\hat{Z}_w^{(p)}H_1 \\ &= (v_h^{(p)}, v_h^{(N-p)})\tfrac{1}{2}H_1\hat{Z}_w^{(p)}H_1, \end{aligned} \tag{4.5.83}$$

when $p \neq N/2$, and $v_h^{(p)}$ and $v_h^{(N-p)}$ are linearly independent. Therefore, the result follows by comparison to (4.3.20). When $p = N/2$, we have

$$v_h^{(N/2)} = w_h^{(+,N/2)}. \tag{4.5.84}$$

Suppose $\lambda_{N/2}$ is the eigenvalue of the matrix Z corresponding to $v_h^{(N/2)}$, and let

$$\hat{Z}_w^{(N/2)} = \hat{Z}_v^{(N/2)} = \lambda_{N/2}I_2, \tag{4.5.85}$$

the relation (4.5.82) still holds.

Lemma 4.1 shows that in any case where the standard Fourier analysis or the two-color Fourier analysis can apply, the other can also apply. Since in a given situation, one of the two Fourier analyses may be easier, it might be appropriate to transform an operator representation form on one basis to the corresponding representation form on another basis. For instance, the RBGS smoothing operator is not easy to write in the v-basis form, but is easy to write in the w-basis form. One can write the v-basis form by using the transformation described in Lemma 4.1. For example, the red iteration operator on the v-basis is

$$\begin{aligned} S_{h,v}^{(+,p)} &= \frac{1}{2}H_1 S_{h,w}^{(+,p)} H_1 \\ &= \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} \cos^2\frac{p\pi h}{2} & -\cos^2\frac{p\pi h}{2} \\ -\sin^2\frac{p\pi h}{2} & \sin^2\frac{p\pi h}{2}. \end{bmatrix}. \end{aligned} \tag{4.5.86}$$

4.6 Red/Black Gauss-Seidel Smoothing

We now consider the use of the red/black Gauss-Seidel (RBGS) method as the smoothing iteration method in the standard multigrid procedure. One RBGS iteration can be regarded as consisting of two sub-iterations: a red sub-iteration followed by a black sub-iteration. Let the red points and the black points be defined in (4.4.48) and (4.4.49) respectively (refer to Figures 3.2 and 4.2). The red sub-iteration operator $S_h^{(+)}$ and the black sub-iteration operator $S_h^{(-)}$ are defined by

$$(S_h^{(+)}u_h)_j = \begin{cases} \frac{1}{2}((u_h)_{j+1} + (u_h)_{j-1}) & j = \text{odd (red)}, \\ (u_h)_j & j = \text{even (black)}, \end{cases} \tag{4.6.87}$$

$$(S_h^{(-)}u_h)_j = \begin{cases} \frac{1}{2}((u_h)_{j+1} + (u_h)_{j-1}) & j = \text{even (black)}, \\ (u_h)_j & j = \text{odd (red)}. \end{cases} \tag{4.6.88}$$

for $j = 1, \ldots, N-1$. These two operators can be written in the w-transform matrix form as

$$S_h^{(+)}(w_h^{(+,p)}, w_h^{(-,p)}) = (w_h^{(+,p)}, w_h^{(-,p)})\hat{S}_h^{(+,p)} \tag{4.6.89}$$

and

$$S_h^{(-)}(w_h^{(+,p)}, w_h^{(-,p)}) = (w_h^{(+,p)}, w_h^{(-,p)})\hat{S}_h^{(-,p)} \tag{4.6.90}$$

where

$$\hat{S}_h^{(+,p)} = \begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix} \tag{4.6.91}$$

and

$$\hat{S}_h^{(-,p)} = \begin{bmatrix} 1 & 0 \\ c_p & 0 \end{bmatrix}. \tag{4.6.92}$$

Let the set of black grid points be used as the coarse grid. If linear interpolation of the corrections and the full weighting restriction of the residuals are used, the two-level standard multigrid method will converge in one cycle with one smoothing iteration on the red grid points that corresponds to the operator $S_h^{(+)}$ defined in (4.6.87). This is because

$$\hat{S}_h^{(+,p)}\hat{C}_{h,w}^{(p)} = \begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -c_p \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.6.93}$$

and

$$\hat{C}_{h,w}^{(p)}\hat{S}_h^{(+,p)} = \begin{bmatrix} 1 & -c_p \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.6.94}$$

where the coarse grid correction operator $\hat{C}_h^{(p)}$ is defined in (4.4.68).

Let us now look more closely at this procedure. If we use one red sub-iteration for pre-smoothing, the full weighting restriction is equivalent to an injection multiplied by $\frac{1}{2}$. On the other hand, if we use one red sub-iteration for post-smoothing, the linear interpolation is equivalent to a pull back injection which is given by

$$\delta_h(x) = (P_h\delta_{2h})(x) = \begin{cases} \delta_{2h}(x) & x \in \Omega_{2h} \\ 0 & x \notin \Omega_{2h}. \end{cases} \tag{4.6.95}$$

To see this, we write

$$\begin{aligned} &\hat{R}_h^{(-,p)}\hat{A}_{h,w}\hat{S}_h^{(+,p)} \\ &= \frac{1}{2}\begin{bmatrix} c_p & 1 \end{bmatrix} \frac{2}{h^2}\begin{bmatrix} 1 & -c_p \\ -c_p & 1 \end{bmatrix} \begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix} c_p & 1 \end{bmatrix} \frac{2}{h^2}\begin{bmatrix} 0 & 0 \\ 0 & 1-c_p^2 \end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix} 0 & 1 \end{bmatrix} \frac{2}{h^2}\begin{bmatrix} 0 & 0 \\ 0 & 1-\cos^2 p\pi h \end{bmatrix} \end{aligned} \tag{4.6.96}$$

and the matrix

$$\begin{bmatrix} 0 & 1 \end{bmatrix} \tag{4.6.97}$$

is the w-transform-space matrix of the injection restriction operator defined in (4.2.4). For the second case, we notice that

$$\begin{aligned} \hat{S}_h^{(+,p)} \hat{P}_h^{(-,p)} &= \begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_p \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{aligned} \tag{4.6.98}$$

and the matrix

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4.6.99}$$

is the w-transform-space matrix of the pull back injection interpolation operator defined in (4.6.95).

Based on the analysis above, it can be shown that the following 1D standard multigrid algorithm converges in one cycle for any given initial guess $u_h^{(0)}$:

1. Do one red iteration: $u'_h = S_h^{(+)} u_h^{(0)} + b_h$.
2. Inject the residual multiplied by $\frac{1}{2}$ on the black points
   $$r_{2h} = \frac{1}{2}(b_h - A_h u'_h).$$
3. Solve the correction equation on the black coarse grid
   $$A_{2h}\delta_{2h} = r_{2h}.$$
4. Get the final solution by linearly interpolating the coarse grid correction and adding the result to the old estimated solution
   $$u_h^{(1)} = u'_h + P_h \delta_{2h},$$
   where $u_h^{(1)}$ is the true solution of the problem $A_h u_h = b_h$.

4.7 Numerical Results

In this section we present some numerical results of the standard multigrid method for the following model equation:

$$\begin{cases} -\dfrac{d^2u(x)}{dx^2} = 64^2 & x \in \Omega = (0,1), \\ u = 1 + x & x \in \partial\Omega. \end{cases} \tag{4.7.100}$$

We use the linear interpolation of correction and the full weighting restriction of the residual in the algorithm. The damped Jacobi method is used for smoothing.

Table 4.3 shows the convergence factors using the two-level scheme where γ is the extrapolation factor of the damped Jacobi method, and m_1 and m_2 are the number of pre-smoothing and post-smoothing iterations respectively. The grid size we used is 64. The convergence factors listed are the average convergence factors of 3 multigrid cycles measured by

$$\left(\frac{\|r^{(3)}\|_2}{\|r^{(0)}\|_2}\right)^{-\frac{1}{3}}. \tag{4.7.101}$$

Table 4.4 shows the convergence factors for the same case with six levels. In the case of more than two levels, the coarse grid problem (4.2.12) is solved by using a similar coarse grid correction procedure based on an even coarser grid (Ω_{4h}). In general, this process can be recursively carried out down to the coarsest grid on which the problem is solved directly. Figure 4.3 shows the schedule for the three-level multigrid method. Because of the shape of the diagram, the multigrid algorithm described here is called the V-cycle.

Comparing these two tables to Table 4.1, one sees that the two-level numerical convergence factors are bounded from above by the estimated upper bounds and the multilevel numerical convergence factors are close to the corresponding two-level ones.

One of the nice properties of multigrid methods is that the convergence factor is independent of the problem size. Figure 4.4 shows that the

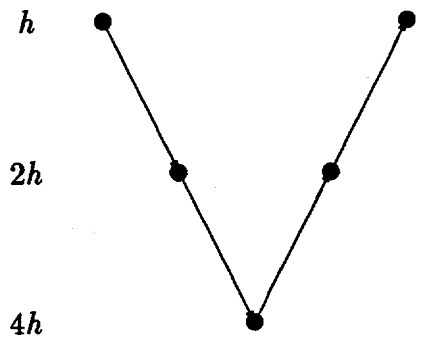


Figure 4.3: Schedule of Grids for Three-Level V-Cycle MG Method

Table 4.3: Numerical Convergence Factors of Two-Level 1D SMG-Jacobi

γ	(m_1, m_2)			
	$(0, 1)$	$(1, 1)$	$(1, 2)$	$(1, 3)$
0.50	**0.3043**	**0.1527**	**0.0822**	**0.0534**
0.60	**0.2455**	**0.1004**	**0.0550**	**0.0410**
0.70	**0.3862**	**0.2096**	**0.0838**	**0.0405**
0.80	**0.5805**	**0.4107**	**0.2434**	**0.1468**
0.90	**0.7758**	**0.6626**	**0.5261**	**0.4212**
1.00	**0.9711**	**0.9603**	**0.9550**	**0.9543**

Table 4.4: Numerical Convergence Factors of Six-Level 1D SMG-Jacobi

γ	(m_1, m_2)			
	(0,1)	(1,1)	(1,2)	(1,3)
0.50	0.4011	0.2334	0.1394	0.1079
0.60	0.2920	0.1869	0.1095	0.0901
0.70	0.3324	0.2724	0.1130	0.0877
0.80	0.5266	0.4370	0.2152	0.1758
0.90	0.7390	0.5708	0.4869	0.4351
1.00	0.9661	0.9603	0.9082	0.9536

convergence factor has only a minor change when the number of points N varies from 32 to 512. In these runs we use $\gamma = 0.6$ and $m_1 = m_2 = 1$.

In our experiments, we found that the solution of the system on the coarsest grid does not need to be exact to obtain fast convergence. The convergence factor of a multigrid method will not degenerate as long as the accuracy of the solution on the coarsest grid is within a certain limit, say 10 times smaller than the average convergence factor. Figure 4.5 shows the residual reduction history for using solutions with different accuracy on the coarsest grid. In the plot, n is the number of smoothing iterations on the coarsest grid. If the number of iterations is more than 8, the result is the same as that using the exact solution on the coarsest grid.

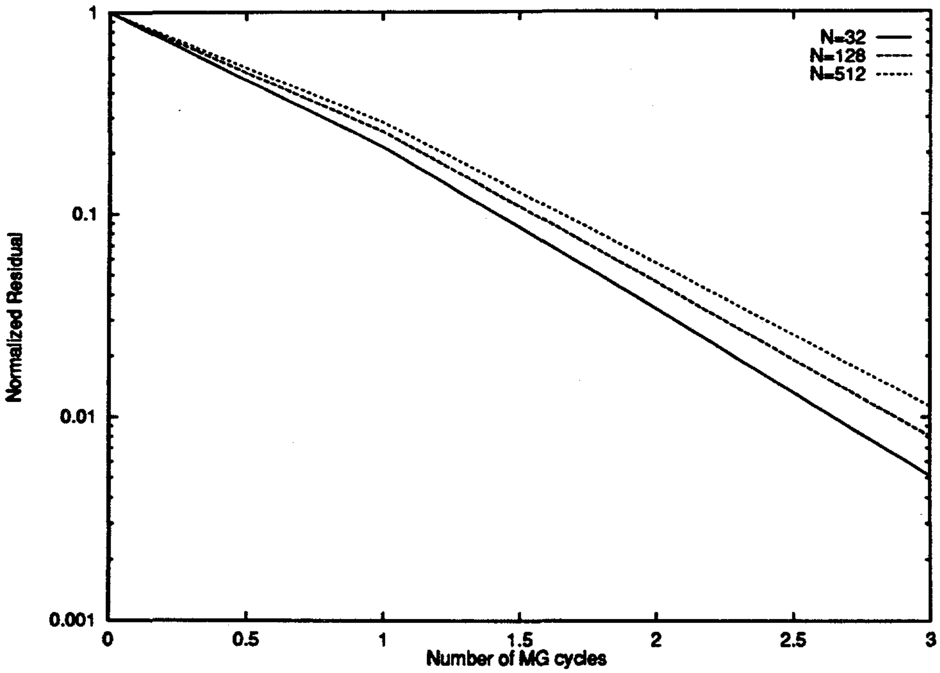


Figure 4.4: Residual Reduction History for Different Problem Sizes (1D)

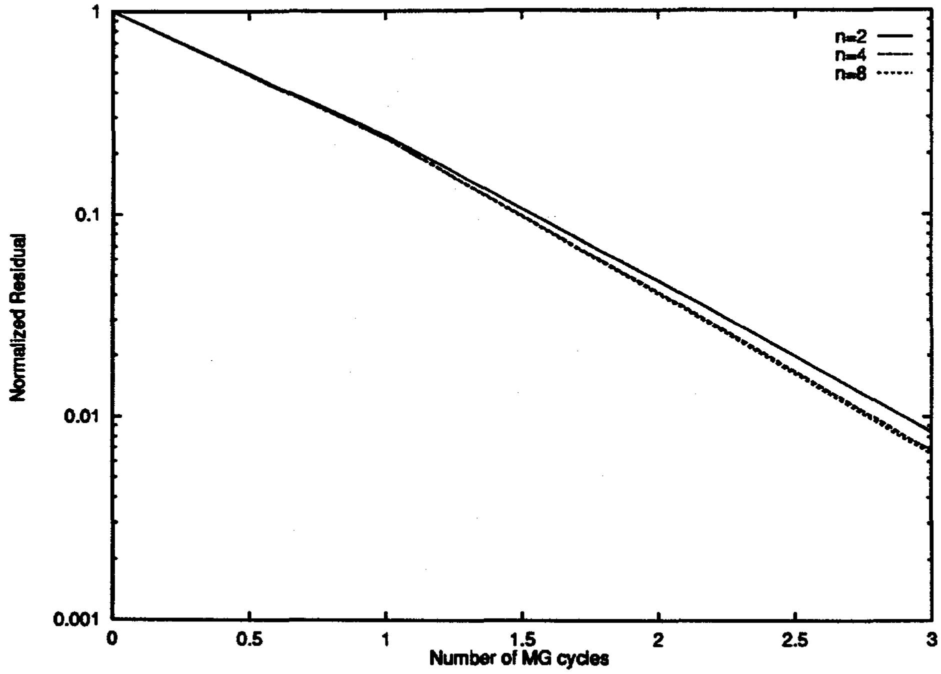


Figure 4.5: Residual Reduction for Different Coarsest Solution Accuracy (1D)

Chapter 5

Standard Multigrid Method in 2D

5.1 Introduction

In this chapter, we describe the two dimensional standard multigrid method, and extend the convergence analysis of the standard multigrid method to the 2D model problem (2.3.11). We again provide both the regular Fourier analysis and the multi-color Fourier analysis.

5.2 Definition of the SMG Algorithm in 2D

The standard multigrid method in 2D is a direct extension of the standard multigrid method in 1D described in the previous chapter. If we let $x_j = jh$ and $y_k = kh$ with $h = 1/N$, the fine grid on the area $(0,1)^2$ is defined by

$$\Omega_h = \{(x_j, y_k) \mid j, k = 1, \ldots, N-1\}. \tag{5.2.1}$$

The fine grid Ω_h contains four subsets which are defined by

$$\left\{ \begin{array}{lcl} \Omega_{++} & = & \{(x_j, y_k) \mid (x_j, y_k) \in \Omega_h \text{ and } (j,k) = (\text{odd, odd})\}, \\ \Omega_{-+} & = & \{(x_j, y_k) \mid (x_j, y_k) \in \Omega_h \text{ and } (j,k) = (\text{even, odd})\}, \\ \Omega_{+-} & = & \{(x_j, y_k) \mid (x_j, y_k) \in \Omega_h \text{ and } (j,k) = (\text{odd, even})\}, \\ \Omega_{--} & = & \{(x_j, y_k) \mid (x_j, y_k) \in \Omega_h \text{ and } (j,k) = (\text{even, even})\}. \end{array} \right. \tag{5.2.2}$$

One of these subsets (usually Ω_{--}) is used as the coarse grid Ω_{2h} for the coarse grid correction in the SMG.

The full weighting restriction operator R_h in 2D is defined by

$$\begin{aligned} r_{2h}(x,y) &= (R_h r_h)(x,y) \\ &= \tfrac{1}{16}(2r_h(x-h,y)+4r_h(x,y)+2r_h(x+h,y) \\ &+ r_h(x-h,y+h)+2r_h(x,y+h)+r_h(x+h,y+h) \\ &+ r_h(x-h,y-h)+2r_h(x,y-h)+r_h(x+h,y-h)) \\ &\quad (x,y)\in\Omega_{2h}. \end{aligned} \tag{5.2.3}$$

The bilinear interpolation operator P_h is defined by

$$\begin{aligned} \delta_h(x,y) &= (P_h\delta_{2h})(x,y) \\ &= \begin{cases} \delta_{2h}(x,y) & (x,y)\in\Omega_{--} \\ \dfrac{1}{2}(\delta_{2h}(x-h,y)+\delta_{2h}(x+h,y)) & (x,y)\in\Omega_{+-} \\ \dfrac{1}{2}(\delta_{2h}(x,y-h)+\delta_{2h}(x,y+h)) & (x,y)\in\Omega_{-+} \\ \dfrac{1}{4}(\delta_{2h}(x-h,y-h)+\delta_{2h}(x-h,y+h) & \\ +\delta_{2h}(x+h,y-h)+\delta_{2h}(x+h,y+h)) & (x,y)\in\Omega_{++}. \end{cases} \end{aligned} \tag{5.2.4}$$

The coarse grid matrix A_{2h} is created by using the standard finite-difference discretization of the original partial differential equation on the coarse grid. It is easy to verify that the eigenvectors of the matrix A_{2h} are given by $v_{2h}^{(pq)}$ for $p,q=1,\ldots,\frac{N}{2}-1$ where

$$(v_{2h}^{(pq)})_{j,k} = \sin(p\pi 2jh)\sin(q\pi 2kh) \qquad j,k=1,\ldots,\frac{N}{2}-1. \tag{5.2.5}$$

The corresponding eigenvalues $\nu_h^{(pq)}$ are given by

$$\nu_h^{(pq)} = \frac{1}{h^2}(2\alpha - 2\alpha\cos(p\pi 2h) + 2 - 2\cos(q\pi 2h)). \tag{5.2.6}$$

We assume the damped Jacobi method is used as the smoothing iterative method and the corresponding matrix is given by

$$B_\gamma = I - \frac{\gamma}{2(1+\alpha)}A_h. \tag{5.2.7}$$

5.3 Standard Fourier Analysis

In the 2D case, for any of the matrices which we will consider in the analysis, say Z, it can be shown that

$$\begin{aligned}
Zv_h^{(p,q)} &= z_{11}^{(p,q)}v_h^{(p,q)} + z_{12}^{(p,q)}v_h^{(p',q)} + z_{13}^{(p,q)}v_h^{(p,q')} + z_{14}^{(p,q)}v_h^{(p',q')}, \\
Zv_h^{(p',q)} &= z_{21}^{(p,q)}v_h^{(p,q)} + z_{22}^{(p,q)}v_h^{(p',q)} + z_{23}^{(p,q)}v_h^{(p,q')} + z_{24}^{(p,q)}v_h^{(p',q')}, \\
Zv_h^{(p,q')} &= z_{31}^{(p,q)}v_h^{(p,q)} + z_{32}^{(p,q)}v_h^{(p',q)} + z_{33}^{(p,q)}v_h^{(p,q')} + z_{34}^{(p,q)}v_h^{(p',q')}, \\
Zv_h^{(p',q')} &= z_{41}^{(p,q)}v_h^{(p,q)} + z_{42}^{(p,q)}v_h^{(p',q)} + z_{43}^{(p,q)}v_h^{(p,q')} + z_{44}^{(p,q)}v_h^{(p',q')}
\end{aligned} \tag{5.3.8}$$

for some values $z_{ij}^{(p,q)}$ depending on (p,q) where $p' = N - p$ and $q' = N - q$. We can also write (5.3.8) in matrix form as

$$ZE_v^{(p,q)} = E_v^{(p,q)}\hat{Z}_v^{(p,q)} \tag{5.3.9}$$

where

$$E_v^{(p,q)} = (v_h^{(p,q)}, v_h^{(p',q)}, v_h^{(p,q')}, v_h^{(p',q')}) \tag{5.3.10}$$

and the v-transform matrix is

$$\hat{Z}_v^{(p,q)} = \begin{bmatrix} z_{11}^{(p,q)} & z_{12}^{(p,q)} & z_{13}^{(p,q)} & z_{14}^{(p,q)} \\ z_{21}^{(p,q)} & z_{22}^{(p,q)} & z_{23}^{(p,q)} & z_{24}^{(p,q)} \\ z_{31}^{(p,q)} & z_{32}^{(p,q)} & z_{33}^{(p,q)} & z_{34}^{(p,q)} \\ z_{41}^{(p,q)} & z_{42}^{(p,q)} & z_{43}^{(p,q)} & z_{44}^{(p,q)} \end{bmatrix}. \tag{5.3.11}$$

Also, we say that the subspace

$$E^{(p,q)} = \mathrm{span}(v_h^{(p,q)}, v_h^{(p',q)}, v_h^{(p,q')}, v_h^{(p',q')}) \tag{5.3.12}$$

is *invariant* under Z. In some sense the matrix $\hat{Z}_v^{(p,q)}$ can be regarded as a kind of "transform" of the matrix Z. Note that the rank of the matrix $E_v^{(p,q)}$ is 2 when one of indices p and q, but not both, is $N/2$, and is 1 when both p and q are $N/2$.

For problem (2.3.11), the matrices used in the standard 2D multigrid algorithm can be written in the forms

$$A_h E_v^{(p,q)} = E_v^{(p,q)} \hat{A}_{h,v}^{(p,q)}, \tag{5.3.13}$$

$$R_h E_v^{(p,q)} = v_{2h}^{(p,q)} \hat{R}_{h,v}^{(p,q)}, \tag{5.3.14}$$

$$A_{2h} v_{2h}^{(p,q)} = v_{2h}^{(p,q)} \hat{A}_{2h,v}^{(p,q)}, \tag{5.3.15}$$

$$P_h v_{2h}^{(p,q)} = E_v^{(p,q)} \hat{P}_{h,v}^{(p,q)}, \tag{5.3.16}$$

$$B_\gamma E_v^{(p,q)} = E_v^{(p,q)} \hat{B}_{\gamma,v}^{(pq)} \tag{5.3.17}$$

where

$$\hat{A}_{h,v}^{(p,q)} = \frac{4}{h^2} \begin{bmatrix} \alpha s_{\frac{p}{2}}^2 + s_{\frac{q}{2}}^2 & 0 & 0 & 0 \\ 0 & \alpha c_{\frac{p}{2}}^2 + s_{\frac{q}{2}}^2 & 0 & 0 \\ 0 & 0 & \alpha s_{\frac{p}{2}}^2 + c_{\frac{q}{2}}^2 & 0 \\ 0 & 0 & 0 & \alpha c_{\frac{p}{2}}^2 + c_{\frac{q}{2}}^2 \end{bmatrix}, \tag{5.3.18}$$

$$\hat{R}_{h,v}^{(p,q)} = \begin{bmatrix} c_{\frac{p}{2}}^2 c_{\frac{q}{2}}^2 & -s_{\frac{p}{2}}^2 c_{\frac{q}{2}}^2 & -c_{\frac{p}{2}}^2 s_{\frac{q}{2}}^2 & s_{\frac{p}{2}}^2 s_{\frac{q}{2}}^2 \end{bmatrix}, \tag{5.3.19}$$

$$\hat{A}_{2h,v}^{(p,q)} = \frac{1}{h^2}(\alpha s_p^2 + s_q^2) \tag{5.3.20}$$

$$\hat{P}_{h,v}^{(p,q)} = \begin{bmatrix} c_{\frac{p}{2}}^2 c_{\frac{q}{2}}^2 \\ -s_{\frac{p}{2}}^2 c_{\frac{q}{2}}^2 \\ -c_{\frac{p}{2}}^2 s_{\frac{q}{2}}^2 \\ s_{\frac{p}{2}}^2 s_{\frac{q}{2}}^2 \end{bmatrix}, \tag{5.3.21}$$

$$\hat{B}_{\gamma,v}^{(p,q)} = I - \frac{\gamma h^2}{2(1+\alpha)} \hat{A}_{h,v}^{(p,q)}. \tag{5.3.22}$$

Table 5.1: Two-Level Convergence Factors of 2D MG-Jacobi

γ	$m=1$	$m=2$	$m=3$	$m=4$
0.50	0.7496	0.5619	0.4212	0.3158
0.60	0.6995	0.4893	0.3423	0.2395
0.70	0.6494	0.4218	0.2740	0.1780
0.80	0.5993	0.3593	0.2154	0.1365
0.90	0.7989	0.6383	0.5100	0.4075
1.00	0.9988	0.9977	0.9965	0.9953

Here $c_{\frac{p}{2}} = \cos(p\pi h/2)$, $c_{\frac{q}{2}} = \cos(q\pi h/2)$, $s_{\frac{p}{2}} = \sin(p\pi h/2)$, and $s_{\frac{q}{2}} = \sin(q\pi h/2)$.

From (4.2.15), the v-transform matrix of the coarse grid correction operator can be computed by

$$\hat{C}_{h,v}^{(p,q)} = I - \hat{P}_{h,v}^{(p,q)}(\hat{A}_{2h,v}^{(p,q)})^{-1}\hat{R}_{h,v}^{(p,q)}\hat{A}_{h,v}^{(p,q)}. \qquad (5.3.23)$$

The v-transform matrix of the two-level multigrid operator with m_1 Jacobi pre-iterations and m_2 Jacobi post-iterations is given by

$$\hat{T}_{h,v}^{(p,q,m_1,m_2)} = (\hat{B}_{\gamma,v}^{(p,q)})^{m_1}\hat{C}_{h,v}^{(p,q)}(\hat{B}_{\gamma,v}^{(p,q)})^{m_2}. \qquad (5.3.24)$$

Therefore, the convergence factor of the two-level 2D standard multigrid method with the damped Jacobi method as the smoothing iterations can be computed by

$$\rho(T_h^{(m_1,m_2)}) = \max_{1\leq p,q\leq \frac{N}{2}} \rho(\hat{T}_{h,v}^{(pq,m_1,m_2)}) = \rho_m \qquad (5.3.25)$$

where $m = m_1 + m_2$. Table 5.1 lists the two-level convergence factor ρ_m in the case of $h = 1/64$ with different values of γ.

5.4 Four-Color Fourier Analysis

The 2D w-basis vectors are defined by

$$\begin{aligned}
(w_h^{(s,p,q)})_{j,k} &= \begin{cases} (v_h^{(p,q)})_{j,k} & \text{if}\,(x_j, y_k) \in \Omega_s \\ 0 & \text{otherwise} \end{cases} \\
& p,q = 1,\ldots,\frac{N}{2} \\
& j,k = 1,\ldots,N-1 \\
& s = ++,\ -+,\ +-,\ -- \qquad (5.4.26)
\end{aligned}$$

where the v-basis $v_h^{(pq)}$ are defined in (2.3.14). Here, (p,q) is the vector index, s is the coarse grid index, and (jk) is the vector element index.

The subspaces $E^{(pq)}$ defined in (5.3.12) can also be represented in terms of

$$E^{(p,q)} = \text{span}(w_h^{(++,p,q)}, w_h^{(-+,p,q)}, w_h^{(+-,p,q)}, w_h^{(--,p,q)}). \qquad (5.4.27)$$

If we let

$$\begin{aligned}
E_w^{(p,q)} &= (w_h^{(++,p,q)}, w_h^{(-+,p,q)}, w_h^{(+-,p,q)}, w_h^{(--,p,q)}) \qquad (5.4.28) \\
& p = 1,\ldots,\frac{N}{2}
\end{aligned}$$

For any matrix Z with an invariant subspace $E^{(p,q)}$, we can write

$$ZE_w^{(p,q)} = E_w^{(p,q)} \hat{Z}_w^{(p,q)} \qquad (5.4.29)$$

where $\hat{Z}_w^{(p,q)}$ is a 4 × 4 matrix which can be regarded as a w-transform matrix of Z.

For each of the operators used in the 2D SMG algorithm, we have

$$A_h E_w^{(p,q)} = E_w^{(p,q)} \hat{A}_{h,w}^{(p,q)} \qquad (5.4.30)$$

$$R_h E_w^{(p,q)} = w_h^{(--,p,q)} \hat{R}_{h,w}^{(p,q)} \qquad (5.4.31)$$

$$A_{2h} w_{2h}^{(--,p,q)} = w_{2h}^{(--,p,q)} \hat{A}_{2h,w}^{(p,q)} \tag{5.4.32}$$

$$P_h w_{2h}^{(p,q)} = E_w^{(p,q)} \hat{P}_{h,w}^{(p,q)} \tag{5.4.33}$$

$$B_\gamma E_w^{(p,q)} = E_w^{(p,q)} \hat{B}_{\gamma,w}^{(p,q)} \tag{5.4.34}$$

where

$$\hat{A}_{h,w}^{(p,q)} = \frac{2}{h^2} \begin{bmatrix} 1+\alpha & -\alpha c_p & -c_q & 0 \\ -\alpha c_p & 1+\alpha & 0 & -c_q \\ -c_q & 0 & 1+\alpha & -\alpha c_p \\ 0 & -c_q & -\alpha c_p & 1+\alpha \end{bmatrix}. \tag{5.4.35}$$

$$\hat{R}_{h,w}^{(p,q)} = \frac{1}{4} \begin{bmatrix} c_p c_q & c_q & c_p & 1 \end{bmatrix}, \tag{5.4.36}$$

$$\hat{A}_{2h,w}^{(p,q)} = \frac{1}{h^2}(\alpha s_p^2 + s_q^2) \tag{5.4.37}$$

$$\hat{P}_{h,w}^{(p,q)} = \begin{bmatrix} c_p c_q \\ c_q \\ c_p \\ 1 \end{bmatrix}, \tag{5.4.38}$$

and

$$\hat{B}_{\gamma,w}^{(p,q)} = I - \frac{\gamma h^2}{2(1+\alpha)} \hat{A}_{h,w}^{(p,q)} \tag{5.4.39}$$

The w-transform matrix of the coarse grid correction operator is given by

$$\hat{C}_{h,w}^{(p,q)} = I - \hat{P}_{h,w}^{(p,q)} (\hat{A}_{2h,w}^{(p,q)})^{-1} \hat{R}_{h,w}^{(p,q)} \hat{A}_{h,w}^{(p,q)} \tag{5.4.40}$$

The w-transform matrix of the two-level multigrid operator with m_1 Jacobi pre-iterations and m_2 Jacobi post-iterations is given by

$$\hat{T}_{h,w}^{(p,q,m_1,m_2)} = (\hat{B}_{\gamma,w}^{(p,q)})^{m_1} \hat{C}_h^{(p,q)} (\hat{B}_{\gamma,w}^{(p,q)})^{m_2} \tag{5.4.41}$$

Therefore the convergence factor can be calculated by

$$\rho(T_h^{(m_1,m_2)}) = \max_{1 \leq p,q \leq \frac{N}{2}} \rho(\hat{T}_{h,w}^{(pq,m_1,m_2)}) = \rho_m \tag{5.4.42}$$

The numerical calculation of $\rho(T_h^{(m_1,m_2)})$ has verified that the convergence factors calculated by the four-color Fourier analysis are the same as those calculated by the standard Fourier analysis.

5.4.1 Relationship Between the v-Basis and w-Basis

Since the w-basis defined in (5.4.26) is constructed from the v-basis defined in (2.3.14), there is a linear transform relationship between these two sets of bases. By comparing the definitions of the two bases, we have

$$\begin{aligned}
v_h^{(p,q)} &= w_h^{(++,p,q)} + w_h^{(-+,p,q)} + w_h^{(+-,p,q)} + w_h^{(--,p,q)}, \\
v_h^{(p',q)} &= w_h^{(++,p,q)} - w_h^{(-+,p,q)} + w_h^{(+-,p,q)} - w_h^{(--,p,q)}, \\
v_h^{(p,q')} &= w_h^{(++,p,q)} + w_h^{(-+,p,q)} - w_h^{(+-,p,q)} - w_h^{(--,p,q)}, \\
v_h^{(p',q')} &= w_h^{(++,p,q)} - w_h^{(-+,p,q)} - w_h^{(+-,p,q)} + w_h^{(--,p,q)}. \\
& p,q = 1, \ldots, \tfrac{N}{2}a.
\end{aligned} \tag{5.4.43}$$

where $p' = N - p$ and $q' = N - q$. Since the wavenumber p' (or q') is larger than $\frac{N}{2}$, the corresponding p' (or q') modes are referred as the *high-frequency modes* in the x-direction (or the y-direction). Equation (5.4.43) can be written in the matrix form

$$\begin{aligned}
&(v_h^{(p,q)}, v_h^{(p',q)}, v_h^{(p,q')}, v_h^{(p',q')}) = \\
&\quad (w_h^{(++,p,q)}, w_h^{(-+,p,q)}, w_h^{(+-,p,q)}, w_h^{(--,p,q)}) H_2
\end{aligned} \tag{5.4.44}$$

where

$$H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \tag{5.4.45}$$

It can be directly verified that the inverse of H_2 is given by

$$H_2^{-1} = \frac{1}{4} H_2. \tag{5.4.46}$$

If we let Z be a matrix of order $(N-1)^2$ with the invariant subspaces $E^{(p,q)}$ defined in (5.4.27), and $\hat{Z}_v^{(p,q)}$ and $\hat{Z}_w^{(p,q)}$ be transform-space matrices associated with the v-basis and the w-basis respectively, then we have the following lemma.

Lemma 5.1 *Let $\hat{Z}_v^{(p,q)}$ and $\hat{Z}_w^{(p,q)}$ be defined as in (5.3.9) and (5.4.29). Then the following relation holds:*

$$\hat{Z}_v^{(p,q)} = \frac{1}{4} H_2 \hat{Z}_w^{(p,q)} H_2. \tag{5.4.47}$$

Proof: From (5.4.44), we have

$$\begin{aligned} & Z(v_h^{(p,q)}, v_h^{(p',q)}, v_h^{(p,q')}, v_h^{(p',q')}) \\ & \quad = Z(w_h^{(++,p,q)}, w_h^{(-+,p,q)}, w_h^{(+-,p,q)}, w_h^{(--,p,q)}) H_2 \\ & \quad = (w_h^{(++,p,q)}, w_h^{(-+,p,q)}, w_h^{(+-,p,q)}, w_h^{(--,p,q)}) \hat{Z}_w^{(p,q)} H_2 \\ & \quad = (v_h^{(p,q)}, v_h^{(p',q)}, v_h^{(p,q')}, v_h^{(p',q')}) \frac{1}{4} H_2 \hat{Z}_w^{(p,q)} H_2 \end{aligned} \tag{5.4.48}$$

The result follows by comparing this to (5.3.9).

We remark that if $\hat{Z}_w^{(p,q)}$ has the form

$$\hat{Z}_w^{(p,q)} = \begin{bmatrix} a & b & c & d \\ b & a & d & c \\ c & d & a & b \\ d & c & b & a \end{bmatrix}, \tag{5.4.49}$$

the corresponding matrix $\hat{Z}_w^{(p,q)}$ is given by

$$\hat{Z}_w^{(p,q)} = \begin{bmatrix} a+b+c+d & 0 & 0 & 0 \\ 0 & a-b+c-d & 0 & 0 \\ 0 & 0 & a+b-c-d & 0 \\ 0 & 0 & 0 & a-b-c+d \end{bmatrix}. \tag{5.4.50}$$

5.5 Numerical Results

The problem we used in our numerical experiments with the 2D standard multigrid method is given by

$$\begin{cases} -\dfrac{\partial^2 u(x,y)}{\partial x^2} - \dfrac{\partial^2 u(x,y)}{\partial y^2} = 64^2 & (x,y) \in \Omega = (0,1)^2, \\ u = 1 + xy & (x,y) \in \partial\Omega \end{cases} \tag{5.5.51}$$

We use 2D linear interpolation of correction and full weighting restriction of the residual in the algorithm. Again, the damped Jacobi method is used for smoothing.

Tables 5.2 and 5.3 list the convergence factors of the two-level multigrid algorithm and the multilevel multigrid algorithm respectively. The grid size we used is 64 × 64. The convergence factors are obtained by averaging the convergence factors in 3 multigrid cycles.

The two-level numerical convergence factors of the two-dimensional multigrid method, as in the one-dimensional case, are bounded from above by

Table 5.2: Numerical Convergence Factors of Two-Level 2D SMG-Jacobi

γ	(m_1, m_2)			
	$(0,1)$	$(1,1)$	$(1,2)$	$(1,3)$
0.50	0.6643	0.4621	0.3209	0.2088
0.60	0.6061	0.3346	0.2595	0.1788
0.70	0.5511	0.3384	0.1793	0.1194
0.80	0.3984	0.2911	0.1691	0.1371
0.90	0.4584	0.1956	0.1134	0.0827
1.00	0.3185	0.2173	0.1335	0.1196

Table 5.3: Numerical Convergence Factors of Six-Level 2D SMG-Jacobi

γ	(m_1, m_2)			
	$(0,1)$	$(1,1)$	$(1,2)$	$(1,3)$
0.50	0.7702	0.6296	0.4390	0.3237
0.60	0.7189	0.4918	0.3332	0.2494
0.70	0.6671	0.4957	0.2987	0.2150
0.80	0.5303	0.3561	0.2346	0.1827
0.90	0.5599	0.3845	0.2119	0.1631
1.00	0.4227	0.2580	0.1863	0.1525

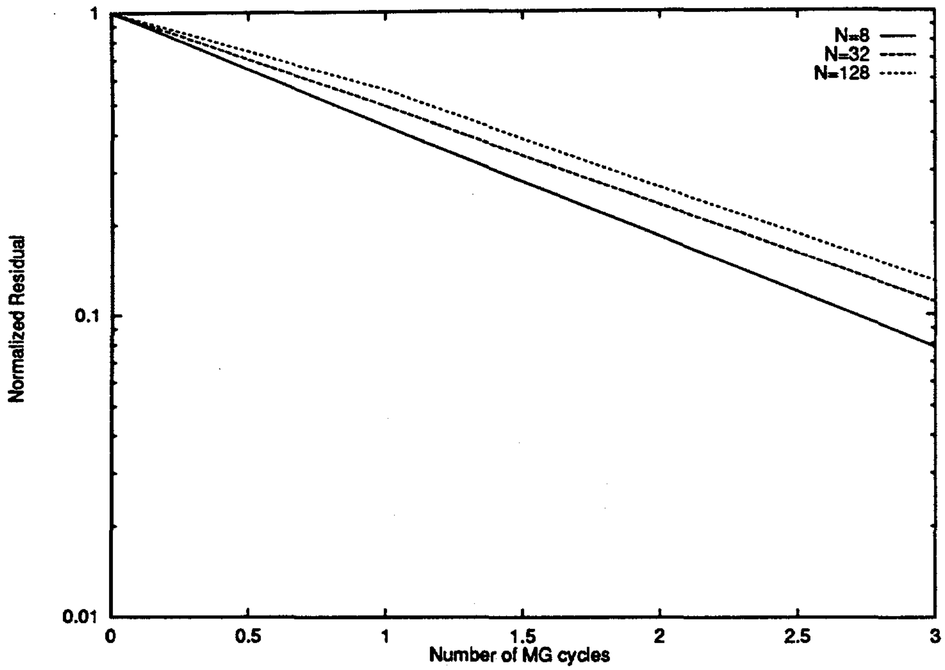


Figure 5.1: Residual Reduction History for Different Problem Sizes (2D)

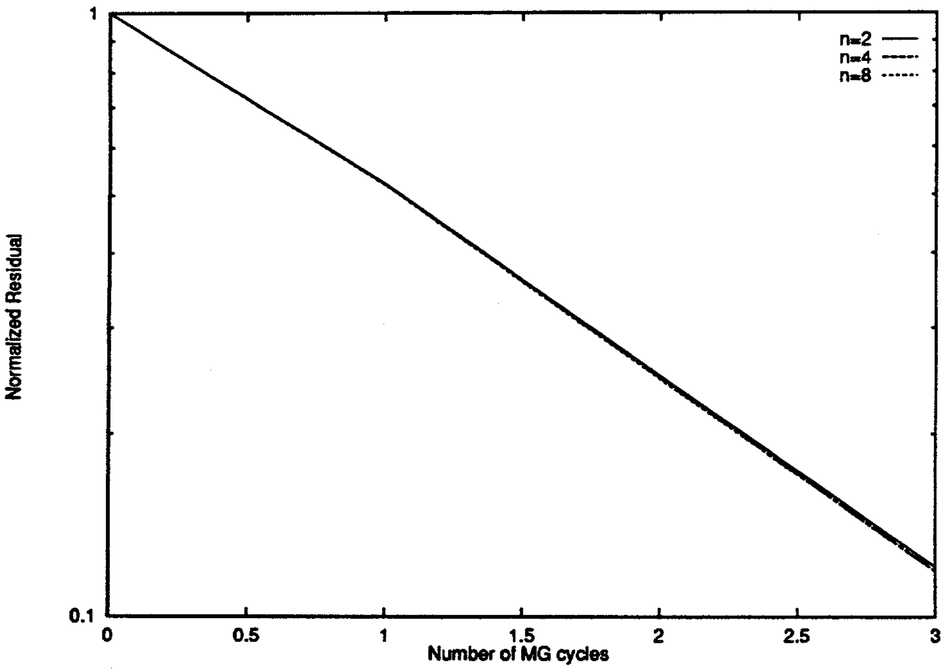


Figure 5.2: Residual Reduction for Different Coarsest Solution Accuracy (2D)

the estimated upper bounds presented in Table 5.1 and the multilevel numerical convergence factors are close to the corresponding two-level ones.

Figure 5.1 illustrates the residual reduction factors of the multigrid method with different grid sizes. It confirms that the convergence factor of multigrid methods is independent of the size of problems. In these runs we used $\gamma = 0.6$ and $m_1 = m_2 = 1$. The problem sizes are from 8×8 to 128×128. The system on the coarsest grid does not have to be solved exactly. In practice, the system on the coarsest grid can be solved approximately by performing a few smoothing iterations. Figure 5.2 shows that the difference between the convergence factors when the system on the coarsest grid is solved to different accuracy. Here n is the number of smoothing iterations performed on the coarsest grid. In this case two smoothing iterations are enough.

Chapter 6

The Construction of Extended Systems

6.1 Introduction

In the next two chapters, we will discuss multigrid methods involving the use of more than one coarse grid for the 1D problem (2.2.1) and for the 2D problem (2.3.8). Since such kinds of methods can be more conveniently applied to periodic systems, we will first consider periodically extended systems corresponding to the problems (2.2.1) and (2.3.8).

6.2 A Sample Problem in 1D

We consider the numerical solution of the model problem with the Dirichlet boundary condition defined by

$$\begin{cases} -\dfrac{d^2u(x)}{dx^2} = f(x) & \text{for } x \in \Omega = (0,1), \\ u(0) = \phi_a, & u(1) = \phi_b. \end{cases} \tag{6.2.1}$$

This problem was considered in Chapter 2.

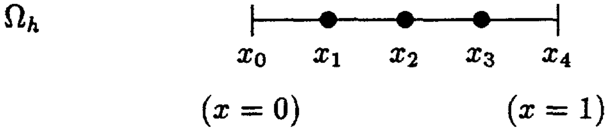


Figure 6.1: The Original Grid with $h = 1/4$

For a grid with grid size $h = 1/4$ (see Figure 6.1), the standard 3-point finite-difference equation system is given by

$$\left\{ \begin{array}{lclcl} u_0 & = & \phi_a & = & b_0 \\ -u_0 + 2u_1 - u_2 & = & h^2 f_1 & = & b_1 \\ -u_1 + 2u_2 - u_3 & = & h^2 f_2 & = & b_2 \\ -u_2 + 2u_3 - u_4 & = & h^2 f_3 & = & b_3 \\ u_4 & = & \phi_b & = & b_4 \end{array} \right. \tag{6.2.2}$$

where $f_j = f(x_j)$.

We now consider the "modified system" given by

$$\left\{ \begin{array}{lclcl} \tilde{u}_0 & = & 0 & = & \tilde{b}_0 \\ -\tilde{u}_0 + 2\tilde{u}_1 - \tilde{u}_2 & = & h^2 f_1 + \phi_a & = & \tilde{b}_1 \\ -\tilde{u}_1 + 2\tilde{u}_2 - \tilde{u}_3 & = & h^2 f_2 & = & \tilde{b}_2 \\ -\tilde{u}_2 + 2\tilde{u}_3 - \tilde{u}_4 & = & h^2 f_3 + \phi_b & = & \tilde{b}_3 \\ \tilde{u}_4 & = & 0 & = & \tilde{b}_4 \end{array} \right. \tag{6.2.3}$$

Evidently, if u, given by

$$u = \begin{bmatrix} \phi_a \\ u_1 \\ u_2 \\ u_3 \\ \phi_b \end{bmatrix} \tag{6.2.4}$$

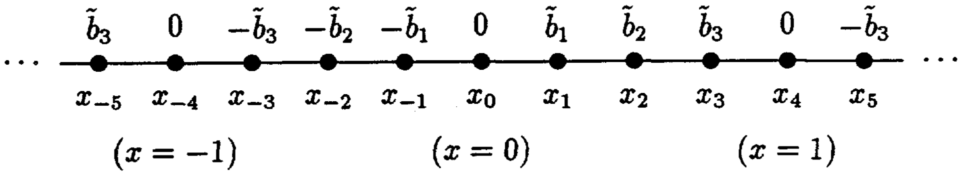


Figure 6.2: The Periodically Extended Grid and the Vector $\tilde{\tilde{b}}$

satisfies (6.2.2), then $\tilde{u}$, given by

$$\tilde{u} = \begin{bmatrix} 0 \\ u_1 \\ u_2 \\ u_3 \\ 0 \end{bmatrix} \tag{6.2.5}$$

satisfies (6.2.3).

6.3 The Periodically Extended System in 1D

We now replace the modified system (6.2.3) by a "periodically extended system" involving the entire real line. We construct a vector $\tilde{\tilde{b}}$ on the entire real line by first extending $\tilde{b}$ asymmetrically from the interval $[0, 1]$ to the interval $[-1, 1]$ and then extending it periodically with period 2 to the entire real line as shown in Figure 6.2.

We now define the periodically extended system by requiring that $\tilde{\tilde{u}}$ satisfy the following conditions:

1. At every grid point $x_j = jh$ (including $j = 0, \pm 1, \pm 2, \ldots$) we have

$$-\tilde{\tilde{u}}_{j-1} + 2\tilde{\tilde{u}}_j - \tilde{\tilde{u}}_{j+1} = \tilde{\tilde{b}}_j \tag{6.3.6}$$

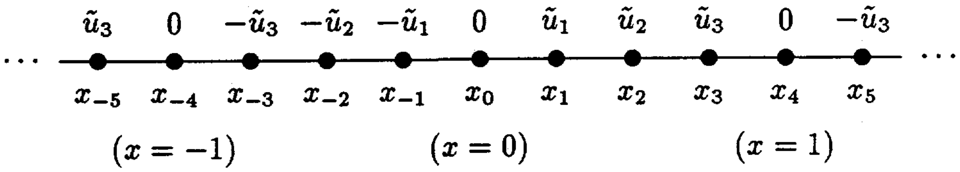


Figure 6.3: A Solution of the Periodically Extended System

where $\tilde{\tilde{u}}_j = \tilde{\tilde{u}}(x_j)$;

2. $\tilde{\tilde{u}}$ is periodic with period 2, i.e.

$$\tilde{\tilde{u}}_{j+2N} = \tilde{\tilde{u}}_j \tag{6.3.7}$$

for $j = 0, \pm 1, \pm 2, \ldots$;

3. The sum of the values of $\{\tilde{\tilde{u}}_j\}$ over any period of length 2, excluding one of the end points, vanishes.

Let $\tilde{u}^*$ given by

$$\tilde{u}^* = \begin{bmatrix} 0 \\ \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ 0 \end{bmatrix}. \tag{6.3.8}$$

be the solution of the modified system (6.2.3). We claim that $\tilde{\tilde{u}}^*$ is a solution of the periodically extended system where $\tilde{\tilde{u}}^*$ is obtained from $\tilde{u}^*$ by extending $\tilde{u}^*$ asymmetrically to the interval $[-1, 1]$ and then extending the asymmetric vector periodically, with period 2, to the entire real line as shown in Figure 6.3.

Suppose now that $\tilde{\tilde{v}}$ is any other solution of the periodically extended system and let

$$\tilde{\tilde{w}} = \tilde{\tilde{v}} - \tilde{\tilde{u}}^*. \tag{6.3.9}$$

Evidently, $\tilde{\tilde{w}}$ is periodic and the sum of the $\{\tilde{\tilde{w}}_j\}$ taken over any period of length 2, excluding one of the end points vanish. Moreover

$$-\tilde{\tilde{w}}_{j-1} + 2\tilde{\tilde{w}}_j - \tilde{\tilde{w}}_{j+1} = 0, \quad j = 0, \pm 1, \pm 2, \ldots \tag{6.3.10}$$

where $\tilde{\tilde{w}}_j = \tilde{\tilde{w}}(x_j)$ for all j. The solution of (6.3.10) can be obtained by rewriting (6.3.10) in the form

$$\tilde{\tilde{w}}_{j+1} = 2\tilde{\tilde{w}}_j - \tilde{\tilde{w}}_{j-1}, \quad j = 0, \pm 1, \pm 2, \ldots \tag{6.3.11}$$

Thus, we have

$$\begin{aligned} \tilde{\tilde{w}}_2 &= 2\tilde{\tilde{w}}_1 - \tilde{\tilde{w}}_0 \\ &= 2(\tilde{\tilde{w}}_1 - \tilde{\tilde{w}}_0) + \tilde{\tilde{w}}_0, \end{aligned} \tag{6.3.12}$$

$$\begin{aligned} \tilde{\tilde{w}}_3 &= 2\tilde{\tilde{w}}_2 - \tilde{\tilde{w}}_1 \\ &= 2(2(\tilde{\tilde{w}}_1 - \tilde{\tilde{w}}_0) + \tilde{\tilde{w}}_0) - \tilde{\tilde{w}}_1 \\ &= 3(\tilde{\tilde{w}}_1 - \tilde{\tilde{w}}_0) + \tilde{\tilde{w}}_0 \end{aligned} \tag{6.3.13}$$

and so on. In general, we have

$$\tilde{\tilde{w}}_n = n(\tilde{\tilde{w}}_1 - \tilde{\tilde{w}}_0) + \tilde{\tilde{w}}_0. \tag{6.3.14}$$

By periodicity, $\tilde{\tilde{w}}_1 - \tilde{\tilde{w}}_0 = 0$. Also since the sum of the $\{\tilde{\tilde{w}}_j\}$ over a period must vanish it follows that $\tilde{\tilde{w}}_0 = 0$. Hence $\tilde{\tilde{w}} = 0$ and

$$\tilde{\tilde{v}} = \tilde{\tilde{u}}^*. \tag{6.3.15}$$

Therefore, the periodically extended system has a unique solution $\tilde{\tilde{u}}^*$. Moreover, from $\tilde{\tilde{u}}^*$ we can obtain the solution of the modified system.

6.4 The Extended System in 1D

We will now show that one can derive a finite-number of equations from the periodically extended system such that the solution of the periodically extended system, the solution of the modified system and thus the solution of the original system can be obtained.

To derive the extended system, we first consider the equations of the periodically extended system corresponding to the points $x_{-4}, x_{-3}, \ldots, x_4$. We obtain the system

$$\left\{\begin{array}{rcll} -\tilde{\tilde{u}}_{-5} + 2\tilde{\tilde{u}}_{-4} - \tilde{\tilde{u}}_{-3} & = & \tilde{\tilde{b}}_{-4} & = 0 \\ -\tilde{\tilde{u}}_{-4} + 2\tilde{\tilde{u}}_{-3} - \tilde{\tilde{u}}_{-2} & = & \tilde{\tilde{b}}_{-3} & \\ -\tilde{\tilde{u}}_{j-1} + 2\tilde{\tilde{u}}_{j} - \tilde{\tilde{u}}_{j+1} & = & \tilde{\tilde{b}}_{j} & j = -2, -1, \ldots, 3 \\ -\tilde{\tilde{u}}_{3} + 2\tilde{\tilde{u}}_{4} - \tilde{\tilde{u}}_{5} & = & \tilde{\tilde{b}}_{4} & = 0 \end{array}\right. \qquad (6.4.16)$$

We then use the periodicity to replace $\tilde{\tilde{u}}_5$ by $\tilde{\tilde{u}}_{-3}$, $\tilde{\tilde{u}}_{-5}$ by $\tilde{\tilde{u}}_3$ and $\tilde{\tilde{u}}_{-4}$ by $\tilde{\tilde{u}}_4$ and we obtain

$$\left\{\begin{array}{rcll} -\tilde{\tilde{u}}_{3} + 2\tilde{\tilde{u}}_{4} - \tilde{\tilde{u}}_{-3} & = & \tilde{\tilde{b}}_{-4} & \\ -\tilde{\tilde{u}}_{4} + 2\tilde{\tilde{u}}_{-3} - \tilde{\tilde{u}}_{-2} & = & \tilde{\tilde{b}}_{-3} & \\ -\tilde{\tilde{u}}_{j-1} + 2\tilde{\tilde{u}}_{j} - \tilde{\tilde{u}}_{j+1} & = & \tilde{\tilde{b}}_{j} & j = -2, -1, \ldots, 3 \\ -\tilde{\tilde{u}}_{3} + 2\tilde{\tilde{u}}_{4} - \tilde{\tilde{u}}_{-3} & = & \tilde{\tilde{b}}_{4} & \end{array}\right. \qquad (6.4.17)$$

Since $\tilde{\tilde{b}}_{-4} = \tilde{\tilde{b}}_4$. We then discard the first equation since it is the same as the last equation and obtain the conditions

$$\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} \tilde{\tilde{u}}_{-3} \\ \tilde{\tilde{u}}_{-2} \\ \tilde{\tilde{u}}_{-1} \\ \tilde{\tilde{u}}_{0} \\ \tilde{\tilde{u}}_{1} \\ \tilde{\tilde{u}}_{2} \\ \tilde{\tilde{u}}_{3} \\ \tilde{\tilde{u}}_{4} \end{bmatrix} = \begin{bmatrix} \tilde{\tilde{b}}_{-3} \\ \tilde{\tilde{b}}_{-2} \\ \tilde{\tilde{b}}_{-1} \\ \tilde{\tilde{b}}_{0} \\ \tilde{\tilde{b}}_{1} \\ \tilde{\tilde{b}}_{2} \\ \tilde{\tilde{b}}_{3} \\ \tilde{\tilde{b}}_{4} \end{bmatrix} \qquad (6.4.18)$$

or

$$A^{(E)}u^{(E)} = b^{(E)}. \qquad (6.4.19)$$

In addition, since $\sum_{j=-3}^{4} \tilde{\tilde{u}}_j = 0$ for the periodically extended problem we have

$$\sum_{j=-3}^{4} u_j^{(E)} = 0. \qquad (6.4.20)$$

We refer to the system defined by (6.4.19) and (6.4.20) as the *extended system.*

It is easy to show that any $N \times N$ matrix of the form of $A^{(E)}$ has rank $N-1$ and has as its null space the one-dimensional subspace spanned by the vector

$$z = (1, 1, \ldots, 1)^T. \qquad (6.4.21)$$

Moreover, the vector

$$(u^{(E)})^* = \begin{bmatrix} -\tilde{u}_3 \\ -\tilde{u}_2 \\ -\tilde{u}_1 \\ 0 \\ \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ 0 \end{bmatrix}. \tag{6.4.22}$$

satisfies (6.4.18). Hence the general solution of (6.4.18) is

$$u^{(E)} = (u^{(E)})^* + \alpha z. \tag{6.4.23}$$

If one requires that the sum of the components of $u^{(E)}$ vanish, then α must vanish, since the sum of the components of $(u^{(E)})^*$ vanishes.

We remark that the process of replacing a vector w by a vector $w' = w + \alpha z$ such that the sum of the components of w' vanishes is referred to as *purification*. Thus if w is a vector of order N and if w' is given by

$$w' = w - (\frac{1}{N}\sum_{j=1}^{N} w_j)z, \tag{6.4.24}$$

then w' is the purification vector of w and we let

$$w' = \mathcal{P}(w) \tag{6.4.25}$$

Suppose now that we want to solve an $N \times N$ system of the form

$$\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & -1 \\ -1 & 2 & -1 & \cdots & 0 & 0 \\ & & & \cdots & & \\ & & & \cdots & & \\ -1 & 0 & 0 & \cdots & -1 & 2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_N \end{bmatrix} \tag{6.4.26}$$

or

$$Aw = b \tag{6.4.27}$$

where the sum of the components of b do not necessarily vanish. Since the sums of the rows of A vanish, the system is inconsistent. However, the following procedure can be used to obtain an approximate solution

1. Purify b to obtain $b' = \mathcal{P}(b)$. Thus

   $$b'_j = b_j - \alpha, \quad j = 1, 2, \ldots, N \tag{6.4.28}$$

   where

   $$\alpha = \frac{1}{N} \sum_{j=1}^{N} b_j; \tag{6.4.29}$$

2. Find a solution, $\hat{w}$, of the consistent system $Aw = b'$;

3. Purify $\hat{w}$ to obtain $\hat{w}'\mathcal{P}(\hat{w})$. Thus

   $$\hat{w}'_j = \hat{w}_j - \beta, \quad j = 1, 2, \ldots, N \tag{6.4.30}$$

   where

   $$\beta = \frac{1}{N} \sum_{j=1}^{N} \hat{w}_j; \tag{6.4.31}$$

It can be shown (see Appendix A), that the approximate solution of (6.4.26) obtained in this way is the same as the "Moore-Penrose solution" of (6.4.26). In general the Moore-Penrose solution of a linear system $Au = b$ minimizes $||b - Au||$ and $||u||$.

In later chapters we will need to solve linear systems of the form (6.4.26) which may not be consistent.

From the discussion above, the procedure for solving the difference equation system

$$\begin{cases} -u_{j-1} + 2u_j - u_{j+1} = h^2 f(x_j), & j = 1, \ldots, N-1 \\ u_0 = \phi_a, \quad u_N = \phi_b, \end{cases} \tag{6.4.32}$$

where $h = 1/N$, $u_j = u(x_j)$, $x_0 = 0$ and $x_N = 1$, can be described as follows.

1. Construct the modified system

$$\begin{cases} -\tilde{u}_{j-1} + 2\tilde{u}_j - \tilde{u}_{j+1} = \tilde{b}_j, & j = 1, \ldots, N-1 \\ \tilde{u}_0 = 0, \quad \tilde{u}_N = 0, \end{cases} \tag{6.4.33}$$

   where

$$\begin{cases} \tilde{b}_1 = h^2 f(x_1) + \phi_a 2, \\ \tilde{b}_j = h^2 f(x_j) & j = 2, \ldots, N-2 \\ \tilde{b}_{N-1} = h^2 f(x_{N-1}) + \phi_b \end{cases} \tag{6.4.34}$$

2. Construct the periodically extended system
   - Extend $\tilde{b}$ to get $\tilde{\tilde{b}}$;
   - $-\tilde{\tilde{u}}_{j-1} + 2\tilde{\tilde{u}}_j - \tilde{\tilde{u}}_{j+1} = \tilde{\tilde{b}}_j$;
   - $\tilde{\tilde{u}}$ is periodic;
   - sum of $\{\tilde{\tilde{u}}_j\}$ over a period vanishes.

3. Construct the extended system
   - use periodicity to get equations for $x_{1-N}, x_{2-N}, \ldots, x_N$;
   - sum of $\{u_j^{(E)}\}$ vanishes.
4. Find a solution $\hat{u}^{(E)}$ of the extended system.
5. Purify $\hat{u}^{(E)}$ to get $(\hat{u}^{(E)})'$.
6. Obtain the solution u^* of the modified system by letting

$$(u^*)_j = (\hat{u}^{(E)})'_j \quad j = 1, \ldots, N-1. \tag{6.4.35}$$

7. The solution of (6.4.32) is the vector u^* with two end values ϕ_a and ϕ_b.

6.5 The Construction of an Extended System in 2D

The discussion in the one-dimensional case can be extended to a two dimensional case. We consider the following anisotropic problem in two dimensions:

$$\begin{cases} -\alpha\dfrac{\partial^2 u(x,y)}{\partial x^2} - \dfrac{\partial^2 u(x,y)}{\partial y^2} = f(x,y) & (x,y) \in \Omega = (0,1)^2, \\ u = \phi(x,y) & (x,y) \in \partial\Omega \end{cases} \tag{6.5.36}$$

where the constant coefficient $\alpha > 0$. The 5-point finite-difference representation of the above problem can be written in the form

$$\begin{cases} L_h[u_{j,k}] = (2+2\alpha)u_{j,k} - \alpha u_{j-1,k} - \alpha u_{j+1,k} - u_{j,k-1} - u_{j,k+1} \\ \qquad\qquad = h^2 f(x_j, y_k) = b_{j,k}, \\ u_{j,0} = \phi(x_j, y_0) = b_{j,0}, \\ u_{j,N} = \phi(x_j, y_N) = b_{j,N}, \\ u_{0,k} = \phi(x_0, y_k) = b_{0,k}, \\ u_{N,k} = \phi(x_N, y_k) = b_{N,k}, \\ j, k = 1, \ldots, N-1 \end{cases} \tag{6.5.37}$$

The modified system corresponding to (6.5.37) is given by

$$\begin{cases} L_h[\tilde{u}_{j,k}] = \tilde{b}_{j,k}, \\ \tilde{u}_{j,0} = \tilde{b}_{j,0} = 0, \\ \tilde{u}_{j,N} = \tilde{b}_{j,N} = 0, \\ \tilde{u}_{0,k} = \tilde{b}_{0,k} = 0, \\ \tilde{u}_{N,k} = \tilde{b}_{N,k} = 0, \\ j, k = 1, \ldots, N-1 \end{cases} \tag{6.5.38}$$

where

$$\begin{cases} \tilde{b}_{j,k} = h^2 f(x_j, y_k) & j, k = 2, N-2 \\ \tilde{b}_{1,k} = h^2 f(x_1, y_k) + \phi(x_0, y_k) & k = 2, N-2 \\ \tilde{b}_{N-1,k} = h^2 f(x_{N-1}, y_k) + \phi(x_N, y_k) & k = 2, N-2 \\ \tilde{b}_{j,1} = h^2 f(x_j, y_1) + \phi(x_j, y_0) & j = 2, N-2 \\ \tilde{b}_{j,N-1} = h^2 f(x_j, y_{N-1}) + \phi(x_j, y_N) & j = 2, N-2 \end{cases} \tag{6.5.39}$$

and

$$\begin{cases} \tilde{b}_{1,1} = h^2 f(x_1, y_1) + \phi(x_0, y_1) + \phi(x_1, y_0) \\ \tilde{b}_{N-1,1} = h^2 f(x_{N-1}, y_1) + \phi(x_N, y_1) + \phi(x_{N-1}, y_0) \\ \tilde{b}_{1,N-1} = h^2 f(x_1, y_{N-1}) + \phi(x_0, y_{N-1}) + \phi(x_1, y_N) \\ \tilde{b}_{N-1,N-1} = h^2 f(x_{N-1}, y_{N-1}) + \phi(x_N, y_{N-1}) + \phi(x_{N-1}, y_N) \end{cases} \tag{6.5.40}$$

As in the one dimensional case, we construct a vector $\tilde{\tilde{b}}$ on the entire plane by first extending $\tilde{b}$ asymmetrically in both the x- and y-directions from the area $[0,1]^2$ to the area $[-1,1]^2$ and then extending it periodically with period 2 in both directions to the entire plane. The 2D periodically extended system is defined by requiring that $\tilde{\tilde{u}}$ satisfy

1. At every grid point (x_j, y_k) (including $j, k = 0, \pm 1, \pm 2, \ldots$)

$$L_h[\tilde{\tilde{u}}_{j,k}] = b_{j,k} \tag{6.5.41}$$

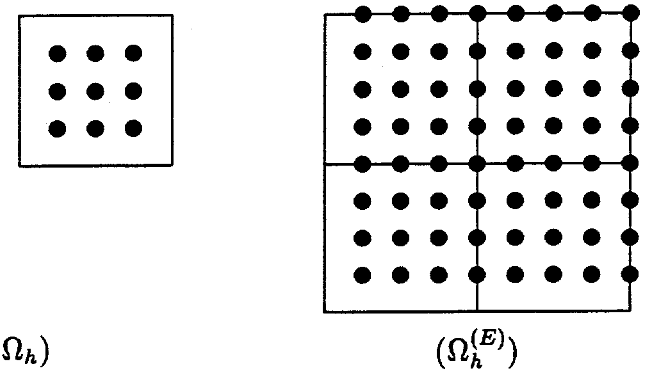
2. $\tilde{\tilde{u}}$ is periodic with period 2 in both directions;

3. The sum of $\{\tilde{\tilde{u}}_{j,k}\}$ over any period of 2 in both direction, excluding the points on any one of the end edges in both directions.

It can be shown, as in the one-dimensional case, that the solution of the 2D periodically extended system is unique. Moreover, the solution of the 2D modified problem can be obtained from the solution of the 2D periodically extended system.

We now consider the following extended system in 2D:

$$\begin{cases} L_h[u^{(E)}_{j,k}] = b^{(E)}_{j,k} & j, k = 1-N, \ldots, N \\ u^{(E)}_{-N,k} = u^{(E)}_{N,k} & k = 1-N, \ldots, N \\ u^{(E)}_{N+1,k} = u^{(E)}_{1-N,k} & k = 1-N, \ldots, N \\ u^{(E)}_{j,-N} = u^{(E)}_{j,N} & j = 1-N, \ldots, N \\ u^{(E)}_{j,N+1} = u^{(E)}_{j,1-N} & j = 1-N, \ldots, N \end{cases} \tag{6.5.42}$$



(Ω_h) ($\Omega_h^{(E)}$)

Figure 6.4: The Original Grid and the Extended Grid in 2D with $h = 1/4$

where

$$b_{j,k}^{(E)} = \begin{cases} \tilde{b}_{j,k} & j,k = 1,\ldots,N-1 \\ -\tilde{b}_{-j,k} & j = 1-N,\ldots,-1;\ k = 1,\ldots,N-1 \\ -\tilde{b}_{j,-k} & j = 1,\ldots,N-1;\ k = 1-N,\ldots,-1 \\ -\tilde{b}_{-j,-k} & j = 1-N,\ldots,-1;\ k = 1-N,\ldots,-1 \\ 0 & j,k = 0,N \end{cases} \tag{6.5.43}$$

or

$$A^{(E)}u^{(E)} = b^{(E)} \tag{6.5.44}$$

This system is defined on the extended grid in the area $[-1,1]^2$. Figure 6.4 illustrates the extended grid Ω_h^E in the case of $h = 1/4$. It can be shown that any $2N \times 2N$ matrix of the form $A^{(E)}$ has rank $2N-1$ and that its null space is spanned by the vector z defined in (6.4.21). Therefore, we can use the same purification procedure in solving the 2D extended system.

Chapter 7

Multiple Coarse Grid Methods in 1D

7.1 Introduction

In this chapter we discuss a class of multigrid methods where, unlike the standard multigrid method, more than one coarse grid is used at each coarse grid level. In our discussion we will refer to this class of methods as multiple coarse grid (MCG) methods. We are concerned with three classes of such methods, namely, multiple coarse grid multigrid methods (MCGMG), frequency decomposition multigrid methods (FDMG) and parallel multigrid methods (PMG). We will use these methods to solve the extended system (6.4.19). For convenience of description, we divide the extended system (6.4.19) by h^2 to obtain an equivalent system which is referred to as

$$A_h u_h = b_h. \tag{7.1.1}$$

7.2 MCGMG Methods in 1D

7.2.1 The Two-Level MCGMG Algorithm in 1D

Let $x_j = jh$ with $h = 1/N$ and

$$\Omega_h = \{x_j \mid j = 1 - N, \ldots, N\}. \tag{7.2.2}$$

be a grid on the interval $(-1, 1]$, where $N = 2^k$ for some positive integer k. We construct two coarse grids in such a way that all the even-numbered grid points belong to one coarse grid and all the odd-numbered grid points belong

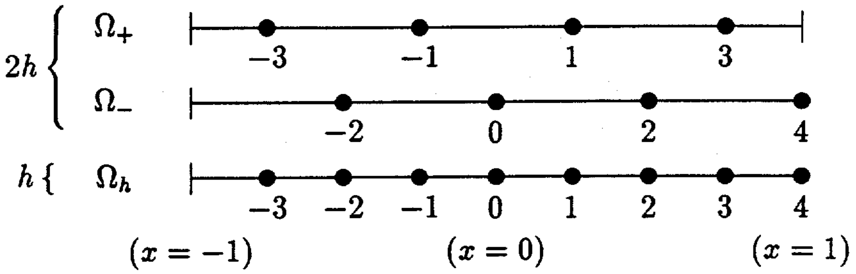


Figure 7.1: Two-Level Grids in 1D with $h = 1/4$

to another. Then, we have

$$\Omega_- = \{x_j \mid x_j \in \Omega_h \text{ and } (j = \text{even})\}, \tag{7.2.3}$$

$$\Omega_+ = \{x_j \mid x_j \in \Omega_h \text{ and } (j = \text{odd})\}. \tag{7.2.4}$$

Figure 7.1 illustrates the grids on two levels, h and $2h$ for the case $N = 4$.

For problem (7.1.1) a two-level MCGMG algorithm is given in Figure 7.2. For the following analysis, we assume that the full weighting restriction of residuals and linear interpolation of corrections are used. The full weighting restriction is defined by

$$(R_h^{(+)} r_h)(x) = \begin{cases} \frac{1}{4}(r_h(x-h) + 2r_h(x) + r_h(x+h)) & x \in \Omega_+ \\ 0 & x \in \Omega_- \end{cases} \tag{7.2.8}$$

$$(R_h^{(-)} r_h)(x) = \begin{cases} 0 & x \in \Omega_+ \\ \frac{1}{4}(r_h(x-h) + 2r_h(x) + r_h(x+h)) & x \in \Omega_- \end{cases} \tag{7.2.9}$$

and the linear interpolation is defined by

$$(P_h^{(+)} \delta_{2h})(x) = \begin{cases} \delta_{2h}(x) & x \in \Omega_+ \\ \frac{1}{2}(\delta_{2h}(x-h) + \delta_{2h}(x+h)) & x \in \Omega_- \end{cases} \tag{7.2.10}$$

Algorithm: MCGMG1D2L($A_h, u_h^{(0)}, b_h$)

1. Do m_1 pre-smoothing iterations using the smoothing iterative method (e.g. damped Jacobi method) to obtain u_h'.

2. Compute the residual $r_h = b_h - A_h u_h'$, restrict the residual onto the coarse grids and perform purification defined in (6.4.24) if necessary to obtain

$$r_{2h}^{(+)} = \mathcal{P}(R_h^{(+)} r_h, z_{2h}^{(+)}), \quad r_{2h}^{(-)} = \mathcal{P}(R_h^{(-)} r_h, z_{2h}^{(-)}) \tag{7.2.5}$$

where $z_{2h}^{(+)}$ and $z_{2h}^{(-)}$ are the eigenvectors in the null spaces of $A_{2h}^{(+)}$ and $A_{2h}^{(-)}$ respectively.

3. Solve the coarse grid systems

$$A_{2h}^{(+)} \delta_{2h}^{(+)} = r_{2h}^{(+)}, \quad A_{2h}^{(-)} \delta_{2h}^{(-)} = r_{2h}^{(-)} \tag{7.2.6}$$

to obtain the purified solutions $\delta_{2h}^{(+)}$ and $\delta_{2h}^{(-)}$.

4. Interpolate $\delta_{2h}^{(+)}$ and $\delta_{2h}^{(-)}$ onto the fine grid to obtain the new approximate solution

$$u_h'' = u_h' + \frac{1}{2}(P_h^{(+)} \delta_{2h}^{(+)} + P_h^{(-)} \delta_{2h}^{(-)}) \tag{7.2.7}$$

5. Do m_2 post-smoothing iterations using the smoothing iterative method and purify the result, if needed, to obtain $u_h^{(1)}$.

Figure 7.2: The 1D Two-Level MCGMG Algorithm

$$(P_h^{(-)}\delta_{2h})(x) = \begin{cases} \frac{1}{2}(\delta_{2h}(x-h)+\delta_{2h}(x+h)) & x \in \Omega_+ \\ \delta_{2h}(x) & x \in \Omega_- \end{cases} \tag{7.2.11}$$

The coarse grid difference operators are defined by the 3-point difference formula, e.g.

$$\begin{aligned} (A_{2h}^{(+)}\delta_{2h}^{(+)})(x) &= (2h)^{-2}[2\delta_{2h}^{(+)}(x) - \delta_{2h}^{(+)}(x-2h) - \delta_{2h}^{(+)}(x+2h)] \\ &\quad x \in \Omega_+ \end{aligned} \tag{7.2.12}$$

$$\begin{aligned} (A_{2h}^{(-)}\delta_{2h}^{(-)})(x) &= (2h)^{-2}[2\delta_{2h}^{(-)}(x) - \delta_{2h}^{(-)}(x-2h) - \delta_{2h}^{(-)}(x+2h)] \\ &\quad x \in \Omega_- \end{aligned} \tag{7.2.13}$$

In general, the restriction of a purified vector on the fine grid may not be a purified vector on coarse grids. However, we have the following lemma.

Lemma 7.1 *Let b_h be a purified vector on the fine grid. If the number of fine grid points is $2N$ for some positive integer N, then the full weighting restriction of b_h on a coarse grid, say b_{2h}, is also a purified vector. In fact the element sum of b_{2h} is one half of the element sum of b_h.*

Proof: It can be directly verified by using the restriction operator definition. In fact, each of the elements of b_h contributes half of its value to b_{2h}.

On the other hand, we have

Lemma 7.2 *Let u_{2h} be a purified vector on a coarse grid. If the number of fine grid points is $2N$ for some integer N, then the linear interpolation of u_{2h} on the fine grid is also a purified vector u_h.*

Proof: It can be directly verified that the sum of the elements in u_h is the same as the sum of the elements of u_{2h}.

Therefore, the purification in step 2 of the MCGMG1D2L algorithm defined in Figure 7.2 is not needed if the full weighting restriction of residuals is used. Moreover, if the Jacobi method is used for smoothing iterations, then the purification in step 6 is also not needed.

7.2.2 Two-Level Convergence Analysis

Let $v^{(p)}$, $p = 1 - N, \ldots, N$, be the Fourier modes (v-basis vectors) defined by

$$(v^{(p)})_j = \exp(-ip\pi jh), \quad j = 1 - N, \ldots, N \tag{7.2.14}$$

The two-color Fourier modes (w-basis vectors) are defined by

$$\begin{aligned} w_h^{(+,p)} &= \tfrac{1}{2}(v_h^{(p)} + v_h^{(p-N)}) \\ w_h^{(-,p)} &= \tfrac{1}{2}(v_h^{(p)} - v_h^{(p-N)}) \end{aligned} \tag{7.2.15}$$

where $p = 1 - N/2, \ldots, N/2$. Note that $v_h^{(p-N)} = v_h^{(p+N)}$. In the case of $N = 4$, we have

$$w_h^{(+,p)} = \begin{bmatrix} \exp(-i3p\pi h) \\ 0 \\ \exp(-ip\pi h) \\ 0 \\ \exp(ip\pi h) \\ 0 \\ \exp(i3p\pi h) \\ 0 \end{bmatrix}, \quad w_h^{(-,p)} = \begin{bmatrix} 0 \\ \exp(-i2p\pi h) \\ 0 \\ 1 \\ 0 \\ \exp(i2p\pi h) \\ 0 \\ \exp(i4p\pi h) \end{bmatrix} \tag{7.2.16}$$

for $p = -1, 0, 1, 2$.

We are interested in the combined effect of the two coarse grids on the coarse grid correction of the MCGMG algorithm (step 2 to step 6 in Figure 7.2). We denote $\bar{u}_h = A_h^{\dagger} b_h$ the purified solution of $A_h u_h = b_h$. Let $e_h' = u_h' - \bar{u}_h$ be the error before the coarse grid correction and $e_h'' = u_h'' - \bar{u}_h$ be the error after the coarse grid correction. We note that both e_h' and e_h'' are in the range of A_h. From (7.2.5) to (7.2.7) we have

$$e_h'' \;=\; e_h' - P_h^{(+)} \delta_{2h}^{(+)} - P_h^{(-)} \delta_{2h}^{(-)}$$

$$\begin{aligned} &= e'_h - \frac{1}{2}(P_h^{(+)}(A_{2h}^{(+)})^\dagger R_h^{(+)} A_h e'_h + P_h^{(-)}(A_{2h}^{(-)})^\dagger R_h^{(-)} A_h e'_h) \\ &= C_h e'_h, \end{aligned} \tag{7.2.17}$$

where

$$C_h = I - \frac{1}{2}(P_h^{(+)}(A_{2h}^{(+)})^\dagger R_h^{(+)} A_h + P_h^{(-)}(A_{2h}^{(-)})^\dagger R_h^{(-)} A_h). \tag{7.2.18}$$

Here the Moore-Penrose inverses $(A_{2h}^{(+)})^\dagger$ and $(A_{2h}^{(-)})^\dagger$ mean that the two coarse grid systems are solved exactly for the purified solutions $\delta_{2h}^{(+)}$ and $\delta_{2h}^{(-)}$ respectively.

We now show that after one MCGMG cycle, the new error vector has no imaginary component. The initial error $e_h^{(0)} = u_h^{(0)} - \bar{u}_h$ can be expressed by a linear combination of the Fourier modes

$$e_h^{(0)} = \sum_{p=1-N}^{N} d_p v_h^{(p)} = \sum_{p=1-N/2}^{N/2} (k_p w_h^{(p,+)} + l_p w_h^{(p,-)}) \tag{7.2.19}$$

where the coefficients d_p, k_p and l_p can be complex values. Since each component in $v_h^{(-p)}$ is the complex conjugate of the corresponding component in $v_h^{(p)}$, $e_h^{(0)}$ is real if and only if

$$d_p = \bar{d}_{-p} \tag{7.2.20}$$

where $\bar{d}_{-p}$ is the complex conjugate of d_{-p}. After one multigrid cycle, the error $e_h^{(1)} = u_h^{(1)} - \bar{u}_h$ is given by

$$e_h^{(1)} = \sum_{p=1-N}^{N} d_p^* v_h^{(p)} = \sum_{p=1-N/2}^{N/2} (k_p^* w_h^{(p,+)} + l_p^* w_h^{(p,-)}). \tag{7.2.21}$$

If we let T_h be the operator of the two-level MCGMG method and let $\hat{T}_{h,v}^{(p)}$ be the v-transform matrix which is defined by

$$T_h(v_h^{(p)}, v_h^{(p-N)}) = (v_h^{(p)}, v_h^{(p-N)})\hat{T}_{h,v}^{(p)} \tag{7.2.22}$$

then (7.2.29) can be written in the form

$$\begin{bmatrix} d_p^* \\ d_{p-N}^* \end{bmatrix} = \hat{T}_{h,v}^{(p)} \begin{bmatrix} d_p \\ d_{p-N} \end{bmatrix}. \tag{7.2.23}$$

If $\hat{T}_{h,v}^{(p)} = \hat{T}_{h,v}^{(-p)}$, which is the case in our analysis, we also have

$$\begin{bmatrix} d_p^* \\ d_{p-N}^* \end{bmatrix} = \hat{T}_{h,v}^{(p)} \begin{bmatrix} d_p \\ d_{p-N} \end{bmatrix} = \hat{T}_{h,v}^{(-p)} \begin{bmatrix} \bar{d}_{-p} \\ \bar{d}_{N-p} \end{bmatrix} = \begin{bmatrix} \bar{d}_{-p}^* \\ \bar{d}_{N-p}^* \end{bmatrix}. \tag{7.2.24}$$

From (7.2.24) one sees that after one MCGMG iteration the error is still real. In the two-color Fourier analysis, we use the w-basis given by (7.2.15). From (4.5.79), we have

$$\begin{bmatrix} k_p \\ l_p \end{bmatrix} = H_1 \begin{bmatrix} d_p \\ d_{p-N} \end{bmatrix} = H_1 \begin{bmatrix} \bar{d}_{-p} \\ \bar{d}_{N-p} \end{bmatrix} = \begin{bmatrix} \bar{k}_{-p} \\ \bar{l}_{-p} \end{bmatrix}. \tag{7.2.25}$$

If the w-transform matrices $\hat{T}_{h,w}^{(p)}$ of T_h are given by

$$T_h(w_h^{(+,p)}, w_h^{(-,p)}) = (w_h^{(+,p)}, w_h^{(-,p)})\hat{T}_{h,w}^{(p)} \tag{7.2.26}$$

and $\hat{T}_{h,w}^{(p)} = \hat{T}_{h,w}^{(-p)}$, which is the case in our analysis, we also have

$$\begin{bmatrix} k_p^* \\ l_p^* \end{bmatrix} = \hat{T}_{h,w}^{(p)} \begin{bmatrix} k_p \\ l_p \end{bmatrix} = \hat{T}_{h,w}^{(-p)} \begin{bmatrix} \bar{d}_{-p} \\ \bar{d}_{N-p} \end{bmatrix} = \begin{bmatrix} \bar{k}_{-p}^* \\ \bar{l}_{-p}^* \end{bmatrix} \tag{7.2.27}$$

Ideally, one would like to have

$$d_p^* = \lambda_p d_p. \tag{7.2.28}$$

This is true, for example, for the Jacobi method itself. However, in general, because of aliasing in the coarse grid correction process, for a normal standard multigrid cycle, we have

$$\begin{aligned} d_p^* &= \lambda_p d_p + \mu_{p-N} d_{p-N}, \\ d_{p-N}^* &= \mu_p d_p + \lambda_{p-N} k_{N-p}. \end{aligned} \tag{7.2.29}$$

If $\mu_{p-N} = 0$ and $\mu_p = 0$ there would be no aliasing. We will show that this is the case for the MCGMG algorithm.

For $p = 1 - \frac{N}{2}, \ldots, \frac{N}{2}$ the w-transform matrices of the difference operator, the interpolation operators, the restriction operators, the coarse grid difference operators and the damped Jacobi operator are given by

$$A_h(w_h^{(+,p)}, w_h^{(-,p)}) = (w_h^{(+,p)}, w_h^{(-,p)})\hat{A}_{h,w}^{(p)}, \tag{7.2.30}$$

$$\begin{aligned} R_h^{(+)}(w_h^{(+,p)}, w_h^{(-,p)}) &= w_h^{(+,p)}\hat{R}_{h,w}^{(+,p)}, \\ R_h^{(-)}(w_h^{(+,p)}, w_h^{(-,p)}) &= w_h^{(-,p)}\hat{R}_{h,w}^{(-,p)}, \end{aligned} \tag{7.2.31}$$

$$\begin{aligned} A_{2h}^{(+)} w_h^{(+,p)} &= w_h^{(+,p)}\hat{A}_{2h,w}^{(+,p)}, \\ A_{2h}^{(-)} w_h^{(-,p)} &= w_h^{(-,p)}\hat{A}_{2h,w}^{(-,p)}, \end{aligned} \tag{7.2.32}$$

$$\begin{aligned} P_h^{(+)} w_h^{(+,p)} &= (w_h^{(+,p)}, w_h^{(-,p)})\hat{P}_{h,w}^{(+,p)}, \\ P_h^{(-)} w_h^{(-,p)} &= (w_h^{(+,p)}, w_h^{(-,p)})\hat{P}_{h,w}^{(-,p)}, \end{aligned} \tag{7.2.33}$$

$$B_\gamma(w_h^{(+,p)}, w_h^{(-,p)}) = (w_h^{(+,p)}, w_h^{(-,p)})\hat{A}_{\gamma,w}^{(p)} \tag{7.2.34}$$

where

$$\hat{A}_{h,w}^{(p)} = \frac{2}{h^2}\begin{bmatrix} 1 & -c_p \\ -c_p & 1 \end{bmatrix}, \tag{7.2.35}$$

$$\hat{R}_{h,w}^{(+,p)} = \frac{1}{2}\begin{bmatrix} 1 & c_p \end{bmatrix}, \tag{7.2.36}$$

$$\hat{R}_{h,w}^{(-,p)} = \frac{1}{2}\begin{bmatrix} c_p & 1 \end{bmatrix}. \tag{7.2.37}$$

$$\hat{A}_{2h,w}^{(+,p)} = \hat{A}_{2h,w}^{(-,p)} = \frac{1}{4h^2}(2 - 2\cos 2p\pi h), \tag{7.2.38}$$

$$\hat{P}^{(+,p)}_{h,w} = \begin{bmatrix} 1 \\ c_p \end{bmatrix}, \tag{7.2.39}$$

$$\hat{P}^{(-,p)}_{h,w} = \begin{bmatrix} c_p \\ 1 \end{bmatrix}. \tag{7.2.40}$$

and

$$\begin{aligned} \hat{B}^{(p)}_{\gamma,w} &= I - \frac{\gamma h^2}{2}\hat{A}^{(p)}_{h,w} \\ &= \begin{bmatrix} 1-\gamma & \gamma c_p \\ \gamma c_p & 1-\gamma \end{bmatrix}. \end{aligned} \tag{7.2.41}$$

The w-transform matrix of the coarse grid correction operator C_h can be written in the form

$$\begin{aligned} \hat{C}^{(p)}_{h,w} &= I - \tfrac{1}{2}(\hat{P}^{(+,p)}_{h,w}(\hat{A}^{(+,p)}_{2h,w})^{-1}\hat{R}^{(+,p)}_{h,w}\hat{A}^{(p)}_{h,w} \\ &\quad +\hat{P}^{(-,p)}_{h,w}(\hat{A}^{(-,p)}_{2h,w})^{-1}\hat{R}^{(-,p)}_{h,w}\hat{A}^{(p)}_{h,w}) \\ &= I - \tfrac{1}{2}\begin{bmatrix} \hat{P}^{(+,p)}_{h,w} & \hat{P}^{(-,p)}_{h,w} \end{bmatrix} \begin{bmatrix} \hat{A}^{(+,p)}_{2h,w} & 0 \\ 0 & \hat{A}^{(-,p)}_{2h,w} \end{bmatrix}^{-1} \begin{bmatrix} \hat{R}^{(+,p)}_{h,w} \\ \hat{R}^{(-,p)}_{h,w} \end{bmatrix} \hat{A}^{(p)}_{h,w} \end{aligned} \tag{7.2.42}$$

Because of the purification process, the coarse grid correction has no effect on the modes $w_h^{(+,0)}$ and $w_h^{(-,0)}$. Therefore, $\hat{C}^{(0)}_{h,w} = I$. From (7.2.36) to (7.2.40), we have

$$\begin{aligned} &\hat{P}^{(+,p)}_{h,w}(\hat{A}^{(+,p)}_{2h,w})^{-1}\hat{R}^{(+,p)}_{h,w}\hat{A}^{(p)}_{h,w} \\ &\quad = \begin{bmatrix} 1 \\ c_p \end{bmatrix} \frac{4h^2}{2-2\cos 2p\pi h}\frac{1}{2}\begin{bmatrix} 1 & c_p \end{bmatrix}\frac{1}{h^2}\begin{bmatrix} 2 & -2c_p \\ -2c_p & 2 \end{bmatrix} \\ &\quad = \begin{bmatrix} 1 & 0 \\ c_p & 0 \end{bmatrix}, \end{aligned} \tag{7.2.43}$$

and

$$\begin{aligned}
&\hat{P}^{(-,p)}_{h,w}(\hat{A}^{(-,p)}_{2h,w})^{-1}\hat{R}^{(-,p)}_{h,w}\hat{A}^{(p)}_{h,w} \\
&= \begin{bmatrix} c_p \\ 1 \end{bmatrix} \frac{4h^2}{2-2\cos 2p\pi h}\frac{1}{2}\begin{bmatrix} c_p & 1 \end{bmatrix}\frac{1}{h^2}\begin{bmatrix} 2 & -2c_p \\ -2c_p & 2 \end{bmatrix} \qquad (7.2.44) \\
&= \begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix}.
\end{aligned}$$

Substituting (7.2.43) and (7.2.44) into (7.2.42), we obtain

$$\begin{aligned}
\hat{C}^{(p)}_{h,w} &= I - \frac{1}{2}\begin{bmatrix} 1 & 0 \\ c_p & 0 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix} 1 & -c_p \\ -c_p & 1 \end{bmatrix}.
\end{aligned} \qquad (7.2.45)$$

By using Lemma 4.1, the corresponding v-transform matrix is given by

$$\begin{aligned}
\hat{C}^{(p)}_{h,v} &= \frac{1}{2}H_1\hat{C}^{(p)}_{h,w}H_1 \\
&= I - \frac{1}{4}\begin{bmatrix} 1+c_p & 1+c_p \\ 1-c_p & 1-c_p \end{bmatrix} - \frac{1}{4}\begin{bmatrix} 1+c_p & -(1+c_p) \\ -(1-c_p) & 1-c_p \end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix} 1-c_p & 0 \\ 0 & 1+c_p \end{bmatrix}. \qquad (7.2.46)
\end{aligned}$$

From (7.2.46) one sees that there is no aliasing in the combined coarse grid correction because the aliasing on even and odd grid points is of opposite sign and therefore cancels.

An upper bound on the two-level convergence factor of the MCGMG algorithm is given by

$$\rho(T_h) = B_\gamma^{m_2}C_hB_\gamma^{m_1}$$

Table 7.1: Two-Level Convergence Factors of 1D MCGMG-Jacobi

γ	(m_1, m_2)			
	(0,1)	(1,1)	(1,2)	(1,3)
0.50	0.2455	0.1480	0.1054	0.0819
0.6	.2083	.1235	.08781	.06819
0.7	.3990	.1593	.07533	.05851
0.8	.5987	.3587	.2149	.1287
0.9	.7985	.6380	.5097	.4072
1.0	.9982	.9971	.9959	.9948

$$= \max_{1-\frac{N}{2}\leq p\leq\frac{N}{2}} \rho((\hat{B}^{(p)}_{\gamma,w})^{m_1}\hat{C}^{(p)}_{h,w}(\hat{B}^{(p)}_{\gamma,w})^{m_2}) \qquad (7.2.47)$$

Table 7.1 lists the two-level convergence factor $\rho(T_h)$ using the Jacobi smoothing iteration with the extrapolation factor γ in the case of $N = 64$. The w-transform matrix of the Jacobi operator is defined in (4.4.67). Comparing the convergence factors of the standard multigrid method in Table 4.1, one sees that the convergence rate of the MCGMG method will generally be faster than the corresponding standard multigrid method for each multigrid cycle. This is because the coarse grid correction process of the MCGMG does not have aliasing errors. Hence, each pair of components in the error of the problem will be damped effectively by the coarse grid correction and by the smoothing iterations without affecting each other. On massively parallel machines the improved convergence rate is attained at no extra computational cost because the coarse grid correction on all coarse grids on each level can be carried out simultaneously.

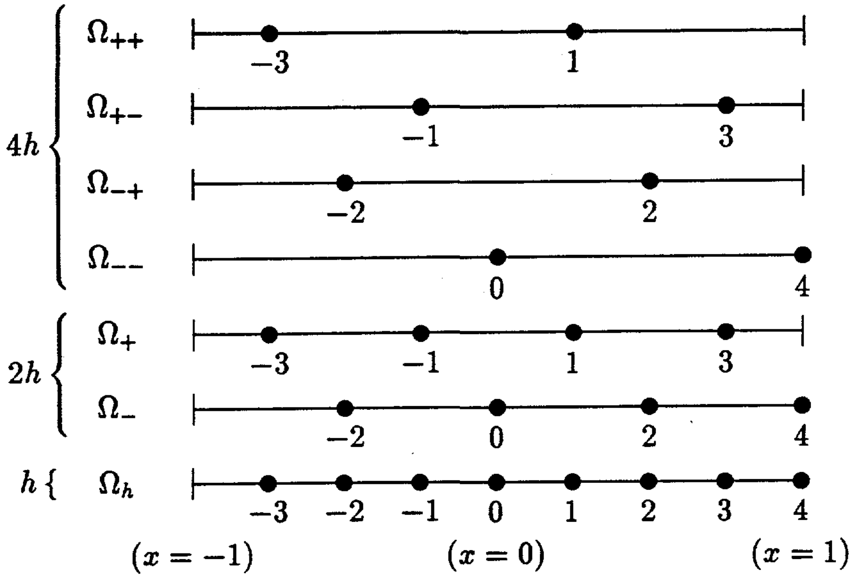


Figure 7.3: Coarse Grids for an Extended Fine Grid: $N = 4$

7.2.3 The Multilevel MCGMG Algorithm in 1D

The $2h$ coarse grids can be divided into even coarser grids in a similar way. Figure 7.3 illustrates all the grids on three levels, h, $2h$ and $4h$ for the case $N = 4$. Figure 7.4 shows the corresponding hierarchical relations among these grids.

A multilevel MCGMG algorithm is similar to the two-level version except the coarse grid problems in step 3 are solved by using the MCGMG1D2L algorithm recursively. For a better understanding of the multilevel MCGMG algorithm, we list a three-level MCGMG algorithm in the following. For convenience of representation, we use the symbol v instead of δ to represent the solutions and b to represent the right-hand side vectors on all levels. The solutions on coarse grids should be thought of as corrections to the solution of the fine grid.

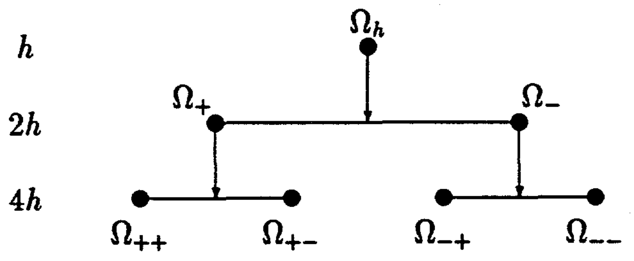


Figure 7.4: Hierarchical Relations Among Grids: $N = 4$

Algorithm: MCGMG1D3L($A_h, u_h^{(0)}, b_h$)

1. Do m_1 smoothing iterations on $A_h u_h = b_h$ with initial guess v_h.

2. Compute

$$b_{2h}^{(+)} = \mathcal{P}(R_h^{(+)} r_h, z_{2h}^{(+)}), \tag{7.2.48}$$

$$b_{2h}^{(-)} = \mathcal{P}(R_h^{(-)} r_h, z_{2h}^{(-)}). \tag{7.2.49}$$

3. Do m_1 smoothing iterations on

$$A_{2h}^{(+)} u_{2h}^{(+)} = b_{2h}^{(+)}, \tag{7.2.50}$$

$$A_{2h}^{(-)} u_{2h}^{(-)} = b_{2h}^{(-)}. \tag{7.2.51}$$

with initial guesses $v_{2h}^{(+)} = 0$ and $v_{2h}^{(-)} = 0$.

4. Compute

$$b_{4h}^{(++)} = \mathcal{P}(R_{2h}^{(++)} r_{2h}^{(+)}, z_{4h}^{(++)}), \tag{7.2.52}$$

$$b_{4h}^{(+-)} = \mathcal{P}(R_{2h}^{(+-)} r_{2h}^{(+)}, z_{4h}^{(+-)}), \tag{7.2.53}$$

$$b_{4h}^{(-+)} = \mathcal{P}(R_{2h}^{(-+)} r_{2h}^{(-)}, z_{4h}^{(-+)}), \tag{7.2.54}$$

$$b_{4h}^{(--)} = \mathcal{P}(R_{2h}^{(--)} r_{2h}^{(-)}, z_{4h}^{(--)}). \tag{7.2.55}$$

5. Solve

$$A_{4h}^{(++)}u_{4h}^{(++)} = b_{4h}^{(++)}, \tag{7.2.56}$$

$$A_{4h}^{(+-)}u_{4h}^{(+-)} = b_{4h}^{(+-)}, \tag{7.2.57}$$

$$A_{4h}^{(-+)}u_{4h}^{(-+)} = b_{4h}^{(-+)}, \tag{7.2.58}$$

$$A_{4h}^{(--)}u_{4h}^{(--)} = b_{4h}^{(--)}. \tag{7.2.59}$$

6. Correct

$$v_{2h}^{(+)} \leftarrow v_{2h}^{(+)} + \frac{1}{2}(P_{2h}^{(++)}v_{4h}^{(++)} + P_{2h}^{(+-)}v_{4h}^{(+-)}) \tag{7.2.60}$$

$$v_{2h}^{(-)} \leftarrow v_{2h}^{(-)} + \frac{1}{2}(P_{2h}^{(-+)}v_{4h}^{(-+)} + P_{2h}^{(--)}v_{4h}^{(--)}). \tag{7.2.61}$$

7. Do m_2 smoothing iterations on

$$A_{2h}^{(+)}u_{2h}^{(+)} = b_{2h}^{(+)}, \tag{7.2.62}$$

$$A_{2h}^{(-)}u_{2h}^{(-)} = b_{2h}^{(-)} \tag{7.2.63}$$

with initial guesses $v_{2h}^{(+)}$ and $v_{2h}^{(-)}$ respectively and purify the results if necessary.

8. Correct

$$v_h \leftarrow v_h + \frac{1}{2}(P_h^{(+)}v_{2h}^{(+)} + P_h^{(-)}v_{2h}^{(-)}) \tag{7.2.64}$$

9. Do m_2 smoothing iterations on $A_h u_h = b_h$ with initial guess v_h and purify the results if necessary.

Here we used the purification notation $\mathcal{P}(v, z)$ defined in (6.4.24). In the case of $N = 4$, there can be three levels. On the second level, the two $2h$ coarse grid systems are given by

$$A_{2h}^{(+)}v_{2h}^{(+)} = \frac{1}{(2h)^2}\begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}\begin{bmatrix} (v_{2h})_{-3} \\ (v_{2h})_{-1} \\ (v_{2h})_1 \\ (v_{2h})_3 \end{bmatrix}$$

$$= \begin{bmatrix} (b_{2h})_{-3} \\ (b_{2h})_{-1} \\ (b_{2h})_{1} \\ (b_{2h})_{3} \end{bmatrix} = \begin{bmatrix} (b_{2h}^{(+)})_{-1} \\ (b_{2h}^{(+)})_{0} \\ (b_{2h}^{(+)})_{1} \\ (b_{2h}^{(+)})_{2} \end{bmatrix} = b_{2h}^{(+)} \tag{7.2.65}$$

and

$$A_{2h}^{(-)}v_{2h}^{(-)} = \frac{1}{(2h)^2}\begin{bmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{bmatrix}\begin{bmatrix} (v_{2h})_{-2} \\ (v_{2h})_{0} \\ (v_{2h})_{2} \\ (v_{2h})_{4} \end{bmatrix}$$

$$= \begin{bmatrix} (b_{2h})_{-2} \\ (b_{2h})_{0} \\ (b_{2h})_{2} \\ (b_{2h})_{4} \end{bmatrix} = \begin{bmatrix} (b_{2h}^{(-)})_{-1} \\ (b_{2h}^{(-)})_{0} \\ (b_{2h}^{(-)})_{1} \\ (b_{2h}^{(-)})_{2} \end{bmatrix} = b_{2h}^{(-)} \tag{7.2.66}$$

Here we use v_{2h} and b_{2h} to represent the fine grid vectors which consist of the coarse grid vectors $v_{2h}^{(+)}$, $v_{2h}^{(-)}$ and $b_{2h}^{(+)}$, $b_{2h}^{(-)}$ respectively.

On the third level, the four $4h$ coarse grid systems are given by

$$A_{4h}^{(++)}v_{4h}^{(++)} = \frac{1}{(4h)^2}\begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}\begin{bmatrix} (v_{4h})_{-3} \\ (v_{4h})_{1} \end{bmatrix}$$

$$= \begin{bmatrix} (b_{4h})_{-3} \\ (b_{4h})_{1} \end{bmatrix} = \begin{bmatrix} (b_{4h}^{(++)})_{0} \\ (b_{4h}^{(++)})_{1} \end{bmatrix} = b_{4h}^{(++)}, \tag{7.2.67}$$

$$A_{4h}^{(+-)}v_{4h}^{(+-)} = \frac{1}{(4h)^2}\begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}\begin{bmatrix} (v_{4h})_{-1} \\ (v_{4h})_{3} \end{bmatrix}$$

$$= \begin{bmatrix} (b_{4h})_{-1} \\ (b_{4h})_{3} \end{bmatrix} = \begin{bmatrix} (b_{4h}^{(+-)})_{0} \\ (b_{4h}^{(+-)})_{1} \end{bmatrix} = b_{4h}^{(+-)}, \tag{7.2.68}$$

$$A_{4h}^{(-+)}v_{4h}^{(-+)} = \frac{1}{(4h)^2}\begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}\begin{bmatrix} (v_{4h})_{-2} \\ (v_{4h})_2 \end{bmatrix}$$

$$= \begin{bmatrix} (b_{4h})_{-2} \\ (b_{4h})_2 \end{bmatrix} = \begin{bmatrix} (b_{4h}^{(-+)})_0 \\ (b_{4h}^{(-+)})_1 \end{bmatrix} = b_{4h}^{(-+)}, \tag{7.2.69}$$

$$A_{4h}^{(--)}v_{4h}^{(--)} = \frac{1}{(4h)^2}\begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}\begin{bmatrix} (v_{4h})_0 \\ (v_{4h})_4 \end{bmatrix}$$

$$= \begin{bmatrix} (b_{4h})_0 \\ (b_{4h})_4 \end{bmatrix} = \begin{bmatrix} (b_{4h}^{(--)})_0 \\ (b_{4h}^{(--)})_1 \end{bmatrix} = b_{4h}^{(--)}. \tag{7.2.70}$$

Here each of the fine grid vectors v_{4h} and b_{4h} consists of four corresponding $4h$ coarse grid vectors. On the third level, the grid points on a coarse grid are not always distributed symmetrically about zero. The systems (7.2.67) and (7.2.68) may not be consistent in general. However, as we showed in Lemma 7.1, for the full weighting of residuals, purification of the right hand sides is not needed as long as the number of grid points can be divided by 2.

7.2.4 Numerical Results

The problem we used for the numerical experiments of a MCG method is the Poisson equation defined in (4.7.100). In solving the extended problem, we use the MCGMG algorithm with linear interpolation of corrections and full weighting restriction of residuals. The damped Jacobi method is used for smoothing.

In our experiments, the grid size is chosen to be $h = 1/64$. Tables 7.2 and 7.3 list the convergence factors of the two-level algorithm and the multilevel scheme algorithm respectively. In this case, we use a six-level scheme which is the maximum number of levels allowed ($2^6 = 64$). γ is the extrapolation factor

Table 7.2: Numerical Convergence Factors of Two-Level 1D MCGMG-J

γ	(m_1, m_2)		
	(0,1)	(1,1)	(1,2)
0.5	.181	.107	.0827
0.6	.149	.0913	.0672
0.7	.192	.0822	.0642
0.8	.294	.163	.0911
0.9	.400	.297	.143
1.0	.506	.471	.448

of the damped Jacobi method, m_1 and m_2 are the number of pre-smoothing and post-smoothing iterations respectively. The convergence factors are the average values of 5 multigrid cycles.

From these two tables, one sees that the numerical 6-level convergence factors are close to the numerical 2-level convergence factors which are bounded by the theoretical upper bounds given in Table 7.1.

To see that the convergence factors of the MCGMG algorithm are independent of the problem size, we used the MCGMG algorithm to solve problem (4.7.100) with different grid sizes. Figure 7.5 plots the convergence histories of the runs with grid sizes $N = 16, 64, 256$, and 512. The maximum number of levels were used (i.e. 4, 6, 8 and 9 levels respectively). The other parameters are $m = 1$ and $\gamma = 0.6$. The figure shows that the convergence factors are almost constant for problem sizes $N = 64, 256$, and 512. Also the convergence factors remain the same at each cycle.

Table 7.3: Numerical Convergence Factors of Six-Level 1D MCGMG-J

γ	(m_1, m_2)		
	(0,1)	(1,1)	(1,2)
0.5	.273	.169	.129
0.6	.210	.145	.116
0.7	.195	.132	.110
0.8	.296	.164	.122
0.9	.401	.298	.226
1.0	.507	.472	.449

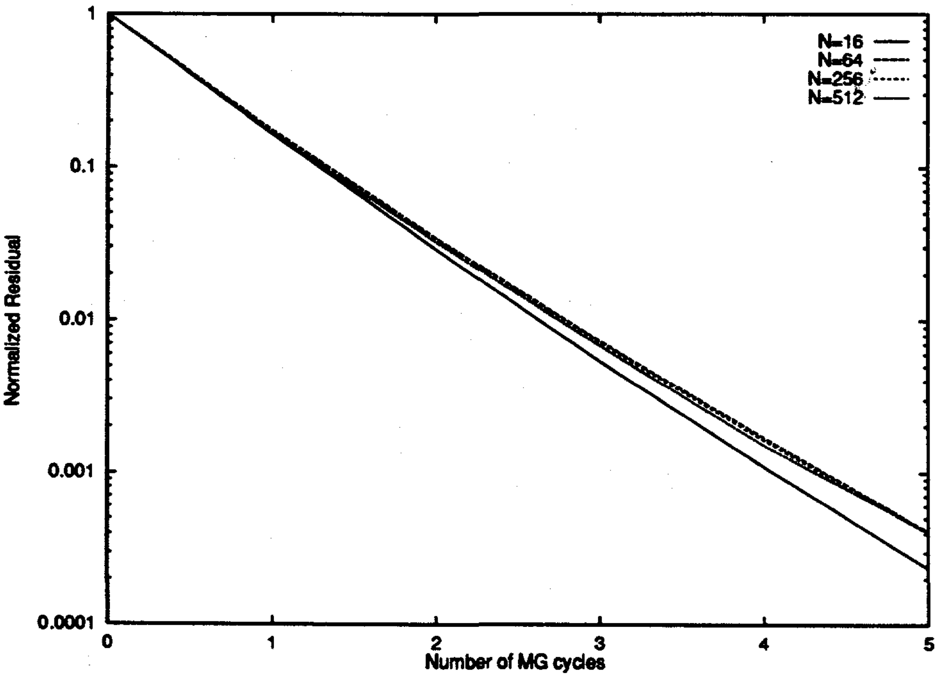


Figure 7.5: Convergence Histories for Different Problem Sizes

7.3 FDMG Methods in 1D

7.3.1 The Two-Level FDMG Algorithm in 1D

We now consider a multigrid method for solving the extended problem (7.1.1), which is similar to the "robust" multigrid method of Hackbusch [34] and which we refer to as the frequency decomposition multigrid method (FDMG method).

The procedure of the FDMG method is the same as the procedure of the MCGMG method defined in Section 7.2.1 except that some of the operators are defined in a different way. In the FDMG method one uses different interpolation operators and different restriction operators on the different grids at each level. The restriction operators are defined by

$$(R_h^{(+)} r_h)(x) = \begin{cases} \frac{1}{4}(r_h(x-h) + 2r_h(x) + r_h(x+h)) & x \in \Omega_+ \\ 0 & x \in \Omega_- \end{cases} \tag{7.3.71}$$

$$(R_h^{(-)} r_h)(x) = \begin{cases} 0 & x \in \Omega_+ \\ \frac{1}{4}(-r_h(x-h) + 2r_h(x) - r_h(x+h)) & x \in \Omega_- \end{cases} \tag{7.3.72}$$

and the interpolation operators are defined by

$$(P_h^{(+)} \delta_{2h})(x) = \begin{cases} \delta_{2h}(x) & x \in \Omega_+ \\ \frac{1}{2}(\delta_{2h}(x-h) + \delta_{2h}(x+h)) & x \in \Omega_- \end{cases} \tag{7.3.73}$$

$$(P_h^{(-)} \delta_{2h})(x) = \begin{cases} -\frac{1}{2}(\delta_{2h}(x-h) + \delta_{2h}(x+h)) & x \in \Omega_+ \\ \delta_{2h}(x) & x \in \Omega_- \end{cases} \tag{7.3.74}$$

The coarse grid difference operators are defined by

$$A_{2h}^{(+)} = P_h^{(+)} A_h R_h^{(+)}, \tag{7.3.75}$$

$$A_{2h}^{(-)} = P_h^{(-)} A_h R_h^{(-)}. \tag{7.3.76}$$

In the case of problem (2.2.4) with $N = 4$, for example, we have

$$R_h^{(+)} = \frac{1}{4}\begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7.3.77}$$

$$R_h^{(-)} = \frac{1}{4}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \tag{7.3.78}$$

The interpolation matrices $P_h^{(+)}$, and $P_h^{(-)}$ are the transposes of the corresponding restriction matrices $R_h^{(+)}$, and $R_h^{(-)}$ respectively multiplied by a factor of 2.

The coarse grid matrices can be written in the form

$$A_{2h}^{(+)} = \frac{1}{(2h)^2}\begin{bmatrix} 2 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 2 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (7.3.79)$$

$$A_{2h}^{(-)} = \frac{1}{(2h)^2}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 3 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 10 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 10 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 3 & 0 & 10 \end{bmatrix} \qquad (7.3.80)$$

It should be noted that the matrix $A_{2h}^{(+)}$ is consistent with the differential equation (2.2.1) while the matrix $A_{2h}^{(-)}$ is not. In some sense, the correction obtained from grid Ω_+ is more important than that obtained from grid Ω_- in this case. As in the MCGMG algorithm, if the coarse grid problems are solved using the same FDMG method recursively, one gets a multilevel version of the FDMG algorithm.

7.3.2 Two-Level Convergence Analysis

We first consider the coarse grid correction operator. It is easy to verify that the w-transform matrices of the the operators $R_h^{(+)}$, $R_h^{(-)}$, $P_h^{(+)}$,

$P_h^{(-)}$, $A_{2h}^{(+)}$, $A_{2h}^{(-)}$ have the following forms:

$$\begin{aligned}
\hat{R}_{h,w}^{(+,p)} &= \frac{1}{2}\begin{bmatrix} 1 & c_p \end{bmatrix}, \\
\hat{R}_{h,w}^{(-,p)} &= \frac{1}{2}\begin{bmatrix} -c_p & 1 \end{bmatrix},
\end{aligned} \tag{7.3.81}$$

$$\begin{aligned}
\hat{P}_{h,w}^{(+,p)} &= \begin{bmatrix} 1 \\ c_p \end{bmatrix}, \\
\hat{P}_{h,w}^{(-,p)} &= \begin{bmatrix} -c_p \\ 1 \end{bmatrix},
\end{aligned} \tag{7.3.82}$$

$$\hat{A}_{2h,w}^{(+,p)} = \hat{R}_{h,w}^{(+,p)} \hat{A}_{h,w} \hat{P}_{h,w}^{(+,p)}$$

$$= \begin{bmatrix} 1 & c_p \end{bmatrix} \frac{2}{h^2} \begin{bmatrix} 1 & -c_p \\ -c_p & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 \\ c_p \end{bmatrix} \tag{7.3.83}$$

$$= \frac{1 - c_p^2}{h^2} \tag{7.3.84}$$

$$\hat{A}_{2h,w}^{(-,p)} = \hat{R}_{h,w}^{(-,p)} \hat{A}_{h,w} \hat{P}_{h,w}^{(-,p)}$$

$$= \begin{bmatrix} 1 & -c_p \end{bmatrix} \frac{2}{h^2} \begin{bmatrix} 1 & -c_p \\ -c_p & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 \\ -c_p \end{bmatrix} \tag{7.3.85}$$

$$= \frac{1 + 3c_p^2}{h^2} \tag{7.3.86}$$

The w-transform matrix of the coarse grid correction operator can then be calculated by

$$\begin{aligned}\hat{C}^{(p)}_{h,w} &= I - \begin{bmatrix} 1 & -c_p \\ c_p & 1 \end{bmatrix} h^2 \begin{bmatrix} 1-c_p^2 & \\ & 1+3c_p^2 \end{bmatrix}^{-1} \frac{1}{2} \begin{bmatrix} 1 & c_p \\ -c_p & 1 \end{bmatrix} \\ &\quad \frac{2}{h^2} \begin{bmatrix} 1-c_p & 0 \\ 0 & 1+c_p \end{bmatrix} \\ &= \frac{c_p}{1+3c_p^2} \begin{bmatrix} -2c_p & 1+c_p^2 \\ 1-3c_p^2 & 2c_p \end{bmatrix}.\end{aligned} \tag{7.3.87}$$

The eigenvalues of this matrix are given by

$$\lambda^{(p)} = \pm\sqrt{\frac{c_p^2(1-c_p^2)}{1+3c_p^2}}. \tag{7.3.88}$$

To study the eigenvalues of $\hat{C}^{(p)}_{h,w}$, we first discuss the function

$$f_m(x) = \frac{x^m(1-x)}{1+3x}. \tag{7.3.89}$$

For the function $f_m(x)$ we have the following lemma.

Lemma 7.3 *For any positive integer m, the function $f_m(x)$ given by* (7.3.89) *has a unique maximum point in the interval (0,1).*

Proof: By (7.3.89) we have

$$\frac{\partial f_m(x)}{\partial x} = \frac{-3mx^2+(2m-4)x+m}{(1+3x)^2} x^{m-1} \tag{7.3.90}$$

The roots of the equation

$$-3mx^2+(2m-4)x+m = 0 \tag{7.3.91}$$

are

$$x = \frac{m - 2 \pm 2\sqrt{(m-\frac{1}{2})^2 + \frac{3}{4}}}{3m} \tag{7.3.92}$$

The larger root satisfies

$$\begin{aligned} x_1 &= \frac{m - 2 + 2\sqrt{(m-\frac{1}{2})^2 + \frac{3}{4}}}{3m} \\ &> \frac{m - 2 + 2(m - \frac{1}{2})}{3m} = 1 - \frac{1}{m} \geq 0 \end{aligned} \tag{7.3.93}$$

and

$$\begin{aligned} x_1 &= \frac{m - 2 + 2\sqrt{(m-\frac{1}{2})^2 + \frac{3}{4}}}{3m} \\ &< \frac{m - 2 + 2m - 1 + \sqrt{3}}{3m} = 1 - \frac{3 - \sqrt{3}}{3m} < 1 \end{aligned} \tag{7.3.94}$$

The other root

$$\begin{aligned} x_2 &= \frac{m - 2 - 2\sqrt{(m-\frac{1}{2})^2 + \frac{3}{4}}}{3m} \\ &< \frac{m - 2 + 2(m - \frac{1}{2})}{3m} = -\frac{m+1}{3m} < 0 \end{aligned} \tag{7.3.95}$$

The conclusion follows from the negative coefficient of the second order term in $\frac{\partial f_m(x)}{\partial x}$. Here we used the relations

$$\sqrt{a+b} < \sqrt{a} + \sqrt{b}$$

for any two positive numbers a and b.

Table 7.4 lists the maximum values of function (7.3.89) with the corresponding position x for $m = 1$ to 4.

By substituting c_p^2 with x in (7.3.88), we can compute the spectral radius of the combination operator of the coarse grid corrections of the two-level 1D FDMG algorithm:

$$\rho(C_{h,w}) = \max_{1 \leq p \leq \frac{N}{2}} |\lambda_p| \leq \max_{0 \leq x \leq 1} \sqrt{f_1(x)} \tag{7.3.96}$$

Thus we have the following result.

Table 7.4: Maximum Values of Function $f_m(x) = \frac{x^m(1-x)}{1+3x}$

m	$x = c_p^2$	$f_m(x) = \rho$
1	0.3333333	0.1111111
2	0.5773502	0.0515668
3	0.6990558	0.0331937
4	0.7675918	0.0244283

Theorem 7.1 *The convergence factor of the two-level 1D FDMG algorithm without any smoothing iterations for the model problem* (2.2.4) *does not exceed* $\frac{1}{3}$.

This convergence factor is consistent with the result obtained by Tuminaro [64]. Figure 7.6 illustrates the relation between the convergence factor $|\lambda_p|$ and the frequency mode index p. It shows that the coarse grid correction operator of the FDMG method eliminates effectively three modes in the error: the highest, the lowest and the middle frequency modes. The largest convergence factor is $\frac{1}{3}$ and this corresponds to the modes with $p \approx \frac{3N}{10}$ and $p \approx \frac{7N}{10}$.

7.3.3 Effect of Smoothing

Now we consider the FDMG algorithm with smoothing iterations. We examine three basic iterative methods defined in Chapter 2, the Jacobi method, the RB-GS method and the RB-SOR method.

Smoothing by the Red/Black Gauss-Seidel (RBGS) Method

We are interested in investigating the behavior of the FDMG algorithm with RBGS smoothing iterations. First we consider the case with one red sub-iteration before the coarse grid correction. From (7.3.87) and (4.6.91),

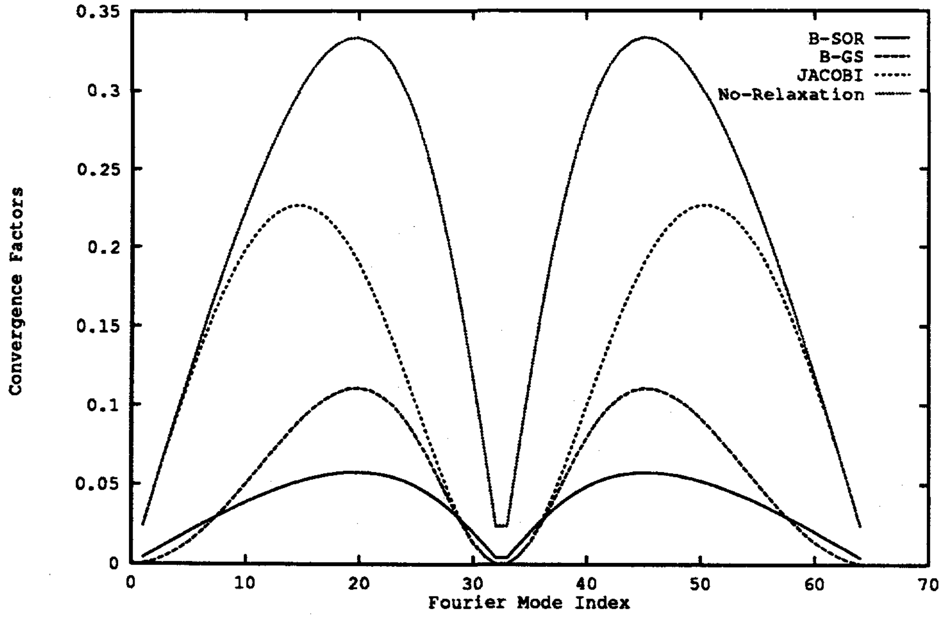


Figure 7.6: Two-Level Convergence Factors of 1D FDMG

we have

$$\begin{aligned}\hat{C}_{h,w}^{(p)}\hat{S}_h^{(+,p)} &= \frac{c_p}{1+3c_p^2}\begin{bmatrix} -2c_p & 1+c_p^2 \\ 1-3c_p^2 & 2c_p \end{bmatrix}\begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix} \\ &= \frac{c_p}{1+3c_p^2}\begin{bmatrix} 0 & 1-c_p^2 \\ 0 & 3c_p(1-c_p^2) \end{bmatrix}. \end{aligned} \tag{7.3.97}$$

The spectral radius of $C_{h,w}S_h^{(+)}$ is given by

$$\rho(C_{h,w}S_h^{(+)}) \leq \max_{1\leq p\leq \frac{N}{2}} \frac{3c_p^2(1-c_p^2)}{1+3c_p^2} = 3f(\frac{1}{3}) = \frac{1}{3}. \tag{7.3.98}$$

This shows that the red sub-iteration does not improve the convergence factor of the FDMG with the restriction and interpolation operators defined in (7.3.71) and (7.3.73). We now consider the use of one black sub-iteration before the coarse grid correction. From (7.3.87) and (4.6.92), we have

$$\begin{aligned}\hat{C}_{h,w}^{(p)}\hat{S}_h^{(-,p)} &= \frac{c_p}{1+3c_p^2}\begin{bmatrix} -2c_p & 1+c_p^2 \\ 1-3c_p^2 & 2c_p \end{bmatrix}\begin{bmatrix} 1 & 0 \\ c_p & 0 \end{bmatrix} \\ &= \frac{c_p}{1+3c_p^2}\begin{bmatrix} -c_p(1-c_p^2) & 0 \\ 1-c_p^2 & 0 \end{bmatrix}. \end{aligned} \tag{7.3.99}$$

The spectral radius of $C_{h,w}S_h^{(-)}$ is given by

$$\rho(C_{h,w}S_h^{(-)}) \leq \max_{1\leq p\leq \frac{N}{2}} \frac{c_p^2(1-c_p^2)}{1+3c_p^2} = f(\frac{1}{3}) = \frac{1}{9}. \tag{7.3.100}$$

The black sub-iteration improves the convergence factor bound from 1/3 to 1/9. We now examine the FDMG algorithm with both red and black smoothing sub-iterations. If we perform one red sub-iteration first followed by one black sub-iteration, from (7.3.87), (4.6.91) and (4.6.92), the w-transform matrix of the FDMG operator is given by

$$\hat{C}_{h,w}^{(p)}\hat{S}_h^{(-,p)}\hat{S}_h^{(+,p)} = \frac{c_p}{1+3c_p^2}\begin{bmatrix} -2c_p & 1+c_p^2 \\ 1-3c_p^2 & 2c_p \end{bmatrix}\begin{bmatrix} 1 & 0 \\ c_p & 0 \end{bmatrix}\begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix}$$

$$= \frac{c_p}{1+3c_p^2}\begin{bmatrix} 0 & -c_p^2(1-c_p^2) \\ 0 & -c_p(1-c_p^2) \end{bmatrix}. \tag{7.3.101}$$

By comparing to (7.3.99) and considering (7.3.100), we know that the spectral radius of $C_{h,w}S_h^{(-)}S_h^{(+)}$ is $\frac{1}{9}$. If we use the black sub-iteration first, then we have

$$\begin{aligned} \hat{C}_{h,w}^{(p)}\hat{S}_h^{(+,p)}\hat{S}_h^{(-,p)} &= \frac{c_p}{1+3c_p^2}\begin{bmatrix} -2c_p & 1+c_p^2 \\ 1-3c_p^2 & 2c_p \end{bmatrix}\begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ c_p & 0 \end{bmatrix} \\ &= \frac{c_p}{1+3c_p^2}\begin{bmatrix} c_p(1-c_p^2) & 0 \\ 3c_p^2(1-c_p^2) & 0 \end{bmatrix}. \end{aligned} \tag{7.3.102}$$

It is obvious that the spectral radius of $C_{h,w}S_h^{(+)}S_h^{(-)}$ and $C_{h,w}S_h^{(-)}S_h^{(+)}$ are the same.

If $m > 0$ steps of RBGS iteration are used, the spectral radius $C_{h,w}(S_h^{(+)}S_h^{(-)})^m$ can also be calculated. From (4.6.92) and (4.6.91), we have

$$\begin{aligned} (\hat{S}_h^{(+,p)}\hat{S}_h^{(-,p)})^m &= \left(\begin{bmatrix} 0 & c_p \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ c_p & 0 \end{bmatrix}\right)^m \\ &= \begin{bmatrix} c_p^2 & 0 \\ c_p & 0 \end{bmatrix}^m = \begin{bmatrix} c_p^{2m} & 0 \\ c_p^{2m-1} & 0 \end{bmatrix}. \end{aligned} \tag{7.3.103}$$

From (7.3.102) and (7.3.103) we have the spectral radius of $C_{h,w}(S_h^{(+)}S_h^{(-)})^m$:

$$\begin{aligned} \rho(C_{h,w}(S_h^{(+)}S_h^{(-)})^m) &= \max_{1\le p\le \frac{N}{2}} \frac{c_p^{2m}(1-c_p^2)}{1+3c_p^2} \\ &\le \max_{0\le x\le 1} f_m(x), \qquad m > 0. \end{aligned} \tag{7.3.104}$$

Similarly it can be shown that $\rho(C_{h,w}(S_h^{(+)}S_h^{(-)})^m)$ has the same expression as (7.3.104). Thus we have the following result.

Theorem 7.2 *The two-level convergence factor of the 1D FDMG algorithm with m RBGS iterations as the smoothing iteration is bounded by (7.3.104).*

Specifically we have

$$\begin{aligned} \rho(C_{h,w}(S_h^{(+)}S_h^{(-)})) &\leq \frac{1}{9} \\ \rho(C_{h,w}(S_h^{(+)}S_h^{(-)})^2) &\leq 0.0515667 \end{aligned} \qquad (7.3.105)$$

Smoothing by the Jacobi Method

Since the coarse grid correction in the FDMG algorithm works particularly well on the modes with p close to 1, $N/2$, and N, the smoothing iteration should be chosen to damp the other modes (e.g. $p = \frac{3N}{2}$). We consider the use of m steps of simple Jacobi iteration. From (4.4.70) with $\gamma = 1$, we have

$$(\hat{B}_w^{(p)})^m = \begin{bmatrix} 0 & c_p \\ c_p & 0 \end{bmatrix}^m = \begin{cases} \begin{bmatrix} 0 & c_p^m \\ c_p^m & 0 \end{bmatrix} & \text{if } m \text{ is odd} \\ \\ \begin{bmatrix} c_p^m & 0 \\ 0 & c_p^m \end{bmatrix} & \text{if } m \text{ is even} \end{cases} \qquad (7.3.106)$$

From (7.3.87), the w-transform matrix of the 1D FDMG operator with m Jacobi iterations can be written in the following forms. For m odd we have

$$\begin{aligned} \hat{C}_{h,w}^{(p)}(\hat{B}_w^{(p)})^m &= \frac{c_p}{1+3c_p^2}\begin{bmatrix} -2c_p & 1+c_p^2 \\ 1-3c_p^2 & 2c_p \end{bmatrix}\begin{bmatrix} 0 & c_p^m \\ c_p^m & 0 \end{bmatrix} \\ &= \frac{c_p^{m+1}}{1+3c_p^2}\begin{bmatrix} 1+c_p^2 & -2c_p \\ 2c_p & 1-3c_p^2 \end{bmatrix}. \end{aligned} \qquad (7.3.107)$$

The eigenvalues of this matrix are complex numbers:

$$\lambda_p = \frac{c_p^{m+1}[(1-c_p^2) \pm 2\sqrt{-(1-c_p^2)c_p^2}]}{1+3c_p^2}. \qquad (7.3.108)$$

The moduli of these two eigenvalues are the same and are given by

$$|\lambda_p| = \sqrt{\frac{c_p^{2m+2}(1-c_p^2)}{1+3c_p^2}}. \tag{7.3.109}$$

When m is even, we have

$$\begin{aligned} \hat{C}_{h,w}^{(p)}(\hat{B}_h^{(p)})^m &= \frac{c_p}{1+3c_p^2}\begin{bmatrix} -2c_p & 1+c_p^2 \\ 1-3c_p^2 & 2c_p \end{bmatrix}\begin{bmatrix} c_p^m & 0 \\ 0 & c_p^m \end{bmatrix} \\ &= \frac{c_p^{m+1}}{1+3c_p^2}\begin{bmatrix} -2c_p & 1+c_p^2 \\ 1-3c_p^2 & 2c_p \end{bmatrix}. \end{aligned} \tag{7.3.110}$$

It is easy to verify that the eigenvalues of this matrix are given by

$$\lambda_p = \pm\sqrt{\frac{c_p^{2m+2}(1-c_p^2)}{1+3c_p^2}}. \tag{7.3.111}$$

In general, we have

$$\begin{aligned} \rho(C_{h,w}B_h^m) &= \max_{1\le p\le\frac{N}{2}}\sqrt{\frac{c_p^{2m+2}(1-c_p^2)}{1+3c_p^2}} \\ &= \max_{0\le x\le 1}\sqrt{f_{m+1}(x)} \qquad m>0. \end{aligned} \tag{7.3.112}$$

Thus we have the following result.

Theorem 7.3 *The two-level convergence factors of the 1D FDMG algorithm with m Jacobi relaxations are bounded by the values specified in (7.3.112).*

Specifically from Table 7.4, the convergence factors of FDMG with one or two Jacobi relaxations are respectively

$$\begin{aligned} \rho(C_{h,w}B_h) &\le 0.227083 \\ \rho(C_{h,w}(B_h)^2) &\le 0.1821913. \end{aligned} \tag{7.3.113}$$

Smoothing by the Red/Black SOR Method (RBSOR)

In this section, we are concerned with using the red/black SOR method (RBSOR) as the smoothing method in the FDMG algorithm. Like the RBGS method, the RBSOR procedure defined in (3.5.42) can also be partitioned into two sub-iterations defined by

$$(L_h^{(+)}u_h)_j = \begin{cases} \frac{\omega}{2}((u_h)_{j+1} + (u_h)_{j-1}) + (1-\omega)(u_h)_j & j = \text{odd}, \\ (u_h)_j & j = \text{even}, \end{cases} \tag{7.3.114}$$

$$(L_h^{(-)}u_h)_j = \begin{cases} \frac{\omega}{2}((u_h)_{j+1} + (u_h)_{j-1}) + (1-\omega)(u_h)_j & j = \text{even}, \\ (u_h)_j & j = \text{odd}. \end{cases} \tag{7.3.115}$$

for $j = 1, \ldots, N-1$. Here the odd-numbered points are red and the even-numbered points are black. The corresponding w-transform matrices are given by

$$\hat{L}^{(+,p,\omega)} = \begin{bmatrix} 1-\omega & \omega c_p \\ 0 & 1 \end{bmatrix} \tag{7.3.116}$$

and

$$\hat{L}^{(-,p,\omega)} = \begin{bmatrix} 1 & 0 \\ \omega c_p & 1-\omega \end{bmatrix}. \tag{7.3.117}$$

From (7.3.87) and (7.3.117), the w-transform matrix of the FDMG with one black SOR sub-iteration can be written in the form

$$\begin{aligned} \hat{C}_{h,w}^{(p)}\hat{L}^{(-,p,\omega)} &= \frac{c_p}{1+3c_p^2}\begin{bmatrix} -2c_p & 1+c_p^2 \\ 1-3c_p^2 & 2c_p \end{bmatrix}\begin{bmatrix} 1 & 0 \\ \omega c_p & 1-\omega \end{bmatrix} \\ &= \frac{c_p}{1+3c_p^2}\begin{bmatrix} (\omega-2)c_p + \omega c_p^3 & (1-\omega)(1+c_p^2) \\ 1+(2\omega-3)c_p^2 & 2(1-\omega)c_p \end{bmatrix}. \end{aligned} \tag{7.3.118}$$

Table 7.5: Convergence Factors of 1D FDMG SOR vs. ω

ω	ρ
1.00	0.1111
1.01	0.0880
1.02	0.0784
1.03	0.0577
1.04	0.0665
1.05	0.0745
1.06	0.0816

If we let

$$\hat{C}^{(p)}_{h,w}\hat{L}^{(-,p,\omega)} = \begin{pmatrix} a(p,\omega) & b(p,\omega) \\ c(p,\omega) & d(p,\omega) \end{pmatrix} \tag{7.3.119}$$

then the spectral radius of this matrix is

$$\rho(\hat{C}^{(p)}_{h,w}\hat{L}^{(-,p,\omega)}) = \begin{cases} \dfrac{|a(p,\omega)+d(p,\omega)|+\sqrt{\Delta}}{2} & \text{if } \Delta > 0 \\ \dfrac{(|a(p,\omega)+d(p,\omega)|^2+|\Delta|)^{\frac{1}{2}}}{2} & \text{if } \Delta \leq 0. \end{cases} \tag{7.3.120}$$

where $\Delta = (a(p,\omega)-d(p,\omega))^2 + 4b(p,\omega)c(p,\omega)$. We are looking for a ω^* such that

$$\begin{aligned} \rho(C_{h,w}L^{(-,\omega^*)}) &= \min_{0<\omega<2}\rho(C_{h,w}L^{(-,\omega)}) \\ &= \min_{0<\omega<2}\max_{1\leq p\leq\frac{N}{2}}\rho(\hat{C}^{(p)}_{h,w}\hat{L}^{(-,p,\omega)}) \end{aligned} \tag{7.3.121}$$

We solve this optimization problem numerically. The relationship among the spectrum of $\rho(\hat{C}^{(p)}_{h,w}\hat{L}^{(-,p,\omega)})$, the mode index p and the iteration

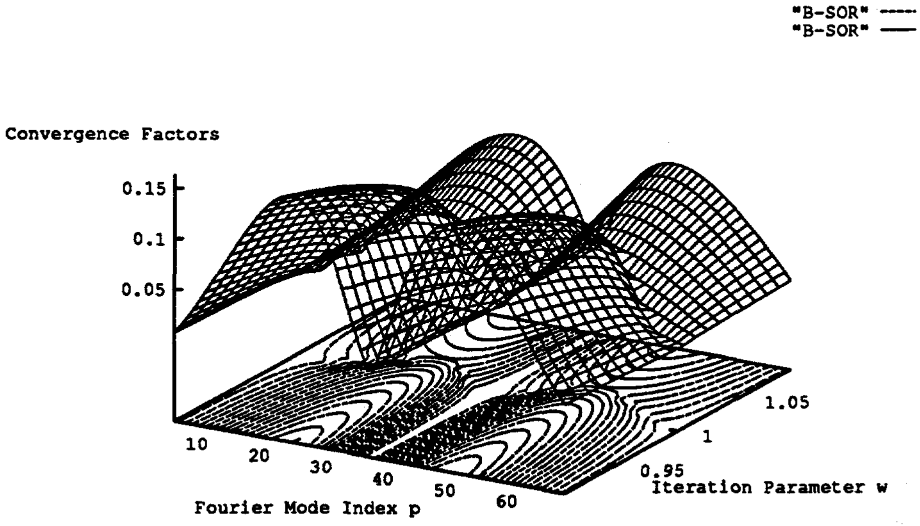


Figure 7.7: Two-Level Convergence Factors of 1D FDMG with B-SOR

parameter ω is plotted in Figure 7.7. Table 7.5 lists the two-level convergence factors of 1D FDMG with one black SOR sub-iteration (ρ) vs. the corresponding iteration parameter ω. Like the RBGS, the red sub-iteration has little effect on the performance of the 1D FDMG algorithm.

7.4 PMG Methods in 1D

7.4.1 The Two-Level PMG Algorithm in 1D

In this section we consider a class of parallel multigrid algorithms (PMG) for solving the extended problem (7.1.1). In the MCGMG method one averages the two coarse grid corrections. This is equivalent to what one would get by using a single grid at each level but with different *scale* or grid spacing $h_l = 2^{-l}$. The combined restriction and interpolation operators can be regarded as smoothing operators on the single grid. In fact, one could consider more general operators with the PMG methods.

The two-level PMG algorithm for the solution of the extended matrix problem $A_h u_h = b_h$, starting with an initial guess $u_h^{(0)}$, is described in Figure 7.8. As in the MCGMG algorithm, the coarse level problem in step 4 can be solved by transfering to an even coarser level. This process can be repeated down to the coarsest level where the problem is solved directly. If more than two levels are involved, one has a multilevel PMG algorithm.

To show that the MCGMG method with averaging is equivalent to a PMG method in the two-level case, it is enough to show that the combination of the restriction operators, the combination of the interpolation operators and the combination of the coarse grid difference operators in the MCGMG method are equivalent to the corresponding operators in the PMG method. This is because that the two coarse grid problems in the MCGMG method are solved independently.

For solving the 1D extended system (7.1.1) using the PMG method,

Two-Level Algorithm PMG(A_h, $u_h^{(0)}$, b_h):

1. Carry out m_1 pre smoothing interations to get u_h'.
2. Compute the residual: $r_h = b_h - A_h u_h'$.
3. Carry out one restriction-like smoothing operation, possibly full weighting:

   $r_{2h}(x) = (R_h r_h)(x) = \frac{1}{4}(r_h(x-h) + 2r_h(x) + r_h(x+h))$,

   or injection $r_{2h}(x) = (R_h r_h)(x) = r_h(x)$

   and then purify r_{2h} if needed.
4. Solve the correction equation for the $2h$ scale

   $$A_{2h}\delta_{2h} = r_{2h}$$
5. Carry out one interpolation-like smoothing operation, possibly linear:

   $(\delta_h)(x) = (P_h \delta_{2h})(x) = \frac{1}{4}[\delta_{2h}(x-h) + 2\delta_{2h}(x) + \delta_{2h}(x+h)]$,

   or injection: $(\delta_h)(x) = (P_h \delta_{2h})(x) = (\delta_{2h})(x)$

   and update the solution $u_h'' = u_h' + \delta_h$
6. Carry out m_2 post smoothing iterations and purify the result, if needed, to obtain the new solution $u_h^{(1)}$.

Figure 7.8: The PMG Algorithm

the coarse level difference matrix A_{2h} is defined by

$$(A_{2h}u_{2h})(x) = \frac{1}{(2h)^2}[2u_{2h}(x) - u_{2h}(x-2h) - u_{2h}(x+2h)] \;\; x \in \Omega_h \qquad (7.4.122)$$

which can be written in the form

$$(A_{2h}^{(+)}u_{2h})(x) = \frac{1}{(2h)^2}[2u_{2h}(x) - u_{2h}(x-2h) - u_{2h}(x+2h)] \;\; x \in \Omega_+ \qquad (7.4.123)$$

and

$$(A_{2h}^{(-)}u_{2h})(x) = \frac{1}{(2h)^2}[2u_{2h}(x) - u_{2h}(x-2h) - u_{2h}(x+2h)] \;\; x \in \Omega_- \qquad (7.4.124)$$

From (7.2.12), (7.2.13), (7.4.123) and (7.4.124), one sees that the coarse grid difference operator of the PMG method is the same as the combination of the two coarse grid difference operators of the MCGMG method.

Similarly, the full weighting operator R_h of the PMG method defined in Figure 7.8 is the combination of of the two restriction operators $R_h^{(+)}$ and $R_h^{(-)}$ of the MCGMG method defined in (7.2.8) and (7.2.9). The linear interpolation operator P_h of the PMG method is the combination of the two coarse grid interpolation operators $P_h^{(+)}$ and $P_h^{(-)}$ of the MCGMG method defined in (7.2.10) and (7.2.11), multiplied by 0.5. As in the MCGMG methods, the purification of the residual is not needed if the full weighting operator is used for the restriction-like smoothing. The system on the coarse level can also be solved by using the two-level PMG algorithm on the coarse level in which case one has a three-level PMG algorithm. If this process is carried out recursively, one gets a multilevel PMG algorithm.

Although the PMG method we discussed here seems to be similar to the MCGMG method, they are two different classes of methods in general. For instance, with the MCGMG methods one could use different operators on different coarse grids on a given level, while with the PMG methods one would normally use the same operator on different coarse grids on a given level. On the other hand, in the PMG methods, the coarse level operators can use any neighboring grid points; while in the MCGMG methods the coarse grid operators can only use the grid points of the corresponding finer grids. For example, in the MCGMG methods the restriction of residuals on the $4h$ grids one only uses points at the related $2h$ grids, while in the PMG methods one could use any points on the $2h$ level.

7.4.2 Two-Level Convergence Analysis

The w-transform matrices corresponding to the operators $A_{2h}^{(+)}$, $A_{2h}^{(-)}$, $P_h^{(+)}$ and $P_h^{(-)}$ are given by

$$\begin{aligned}
\hat{A}_{2h,w}^{(+,p)} &= \frac{1}{4h^2}(2 - 2\cos 2p\pi h), \\
\hat{A}_{2h,w}^{(-,p)} &= \frac{1}{4h^2}(2 - 2\cos 2p\pi h), \qquad (7.4.125)
\end{aligned}$$

$$\begin{aligned}
\hat{P}_{h,w}^{(+,p)} &= \begin{bmatrix} 1 \\ c_p \end{bmatrix}, \\
\hat{P}_{h,w}^{(-,p)} &= \begin{bmatrix} c_p \\ 1 \end{bmatrix}. \qquad (7.4.126)
\end{aligned}$$

For full weighting restriction operators $R_h^{(+)}$ and $R_h^{(-)}$, we have

$$\begin{aligned}
\hat{R}_{h,w}^{(+,p)} &= \frac{1}{2}\begin{bmatrix} 1 & c_p \end{bmatrix}, \\
\hat{R}_{h,w}^{(-,p)} &= \frac{1}{2}\begin{bmatrix} c_p & 1 \end{bmatrix}. \qquad (7.4.127)
\end{aligned}$$

The $\hat{A}_{h,w}^{(p)}$ is defined in (4.4.63). For the coarse grid correction operator C_h we have

$$\begin{aligned}
\hat{C}_{h,w}^{(p)} &= I - P_{h,w}^{(p)}(\hat{A}_{h,w}^{(p)})^{-1} R_{h,w}^{(p)} \hat{A}_{h,w}^{(p)} \\
&= I - \frac{1}{2}\begin{bmatrix} \hat{P}_{h,w}^{(+,p)} & \hat{P}_{h,w}^{(-,p)} \end{bmatrix} \begin{bmatrix} \hat{A}_{2h,w}^{(+,p)} & 0 \\ 0 & \hat{A}_{2h,w}^{(-,p)} \end{bmatrix}^{-1} \begin{bmatrix} \hat{R}_{h,w}^{(+,p)} \\ \hat{R}_{h,w}^{(-,p)} \end{bmatrix} \hat{A}_{h,w}^{(p)} \\
&= I - \begin{bmatrix} 1 & c_p \\ c_p & 1 \end{bmatrix} \left(\frac{1 - c_p^2}{h^2} I\right)^{-1} \begin{bmatrix} 1 & c_p \\ c_p & 1 \end{bmatrix} \frac{2}{h^2} \begin{bmatrix} 1 & -c_p \\ -c_p & 1 \end{bmatrix} \qquad (7.4.128)
\end{aligned}$$

Comparing (7.4.128) to (7.2.42), one sees, that in this case, the PMG method is equivalent to the MCGMG method. If the trivial injection is used instead of the full weighting operator R_h, we have

$$\begin{aligned} \hat{C}^{(p)}_{h,w} &= I - P^{(p)}_{h,w}(\hat{A}^{(p)}_{h,w})_{-1}\hat{A}^{(p)}_{h,w} \\ &= I - \frac{1}{2}\begin{bmatrix} 1 & c_p \\ c_p & 1 \end{bmatrix}(\frac{1-c_p^2}{h^2}I)^{-1}\frac{2}{h^2}\begin{bmatrix} 1 & -c_p \\ -c_p & 1 \end{bmatrix} \\ &= 0 \end{aligned} \qquad (7.4.129)$$

This means that for the 1D model problem (7.1.1), the coarse grid correction of the two-level PMG method is exact. This two-level convergence result can be extended to the multilevel case.

Theorem 7.4 *The multilevel PMG method with the injection restriction of residuals and the linear interpolation of corrections is exact for the 1D model problem* (7.1.1), *if the solution on the coarsest level is exact.*

Proof: By induction, the result follows (7.4.129).

Chapter 8

Multiple Coarse Grid Multigrid Methods in 2D

8.1 Introduction

In this chapter, we extend the discussion of the multiple coarse grid methods (MCG) to two-dimensional cases. The three classes of MCG methods described in the previous chapter, namely MCGMG methods, PMG methods and FDMG methods, are considered here for the two-dimensional extended system (6.5.41). Again, for convenience, we will divide (6.5.41) by h^2 to get an equivalent system

$$A_h u_h = b_h. \tag{8.1.1}$$

8.2 MCGMG Methods in 2D

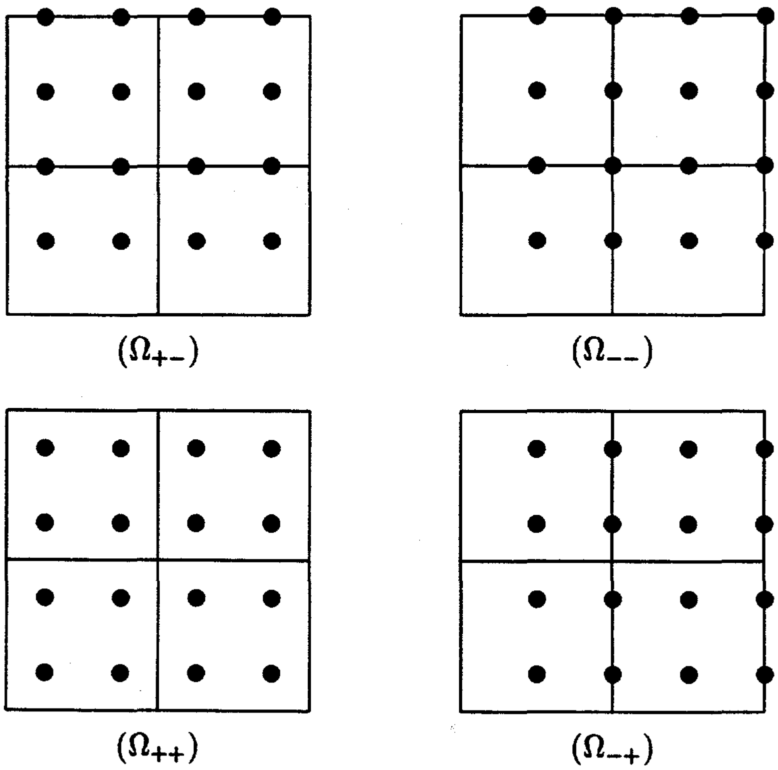
8.2.1 The Two-Level MCGMG Algorithm in 2D

If we let $x_j = jh$ and $y_k = kh$ with $h = 1/N$, the fine grid on the area $(-1, 1]^2$ is defined by

$$\Omega_h = \{(x_j, y_k) \mid j, k = 1 - N, \ldots, N\}. \tag{8.2.2}$$

On this fine grid, the four coarse grids can be defined by (5.2.2) which are illustrated in Figure 8.1 in the case of $N = 4$.

A two-level MCGMG algorithm in 2D is a straightforward extension of the corresponding two-level MCGMG algorithm in 1D defined in Figure 7.2.



(Ω_{+-}) (Ω_{--})

(Ω_{++}) (Ω_{-+})

Figure 8.1: Coarse Grid Points for a 2D Problem with $h = 1/4$

Algorithm: MCGMG2D2L($A_h, u_h^{(0)}, b_h$)

1. Do m_1 pre-smoothing iterations using the smoothing iterative method (e.g. damped Jacobi method) to obtain u_h'.

2. Compute the residual $r_h = b_h - A_h u_h'$, restrict the residual onto each of the four coarse grids and perform purification defined in (6.4.24) if necessary to obtain

   $$r_{2h}^{(s)} = \mathcal{P}(R_h^{(s)} r_h, z_{2h}^{(s)}), \quad s = ++, -+, +-, --, \tag{8.2.3}$$

   where $z_{2h}^{(s)}$ is the eigenvector in the null space of $A_{2h}^{(s)}$.

3. Solve the coarse grid systems

   $$A_{2h}^{(s)} \tilde{\delta}_{2h}^{(s)} = r_{2h}^{(s)}, \quad s = ++, -+, +-, --, \tag{8.2.4}$$

   for $\widetilde{\delta_{2h}}^{(s)}$.

4. Purify $\tilde{\delta}_{2h}^{(s)}$ and interpolate the purified corrections $\delta_{2h}^{(s)}$ onto the fine grid to obtain the new approximate solution

   $$\delta_{2h}^{(s)} = \mathcal{P}(\tilde{\delta}_{2h}^{(+)}, z_{2h}^{(s)}), \quad s = ++, -+, +-, --, \tag{8.2.5}$$

   $$u_h'' = u_h' + \frac{1}{4} \sum_s P_h^{(s)} \delta_{2h}^{(s)}. \tag{8.2.6}$$

5. Do m_2 post-smoothing iterations using the smoothing iterative method to obtain and return $u_h^{(1)}$.

Figure 8.2: The 2D Two-Level MCGMG Algorithm

For a problem $A_h u_h = b_h$ with a given initial guess $u_h^{(0)}$, a two-level MCGMG algorithm in 2D is given in Figure 8.2.

A specific version of the MCGMG algorithm is determined by the selection of operators $R_h^{(s)}$, $P_h^{(s)}$ and $A_{2h}^{(s)}$. In the following analysis, we assume that the full weighting restriction of residuals defined in (5.2.3) and a simple injection mapping of corrections $\delta_h(x,y) = \delta_{2h}(x,y)$ are used. This choice of the restriction operators and the interpolation operators is equivalent to that using the injection restriction operators and the linear interpolation operators with an averaging factor of 1/4. The coarse grid difference operators are defined by the 5-point difference formula on the corresponding coarse grids

$$\begin{aligned}(A_{2h}^{(s)}\delta_{2h}^{(s)})(x,y) \quad &= \quad \frac{1}{(2h)^2}\Big[(2+2\alpha)\delta_{2h}^{(s)}(x,y) - \alpha\delta_{2h}^{(s)}(x-2h,y) \\ &\quad -\alpha\delta_{2h}^{(s)}(x+2h,y) - \delta_{2h}^{(s)}(x,y-2h) - \delta_{2h}^{(s)}(x,y+2h)\Big] \\ &\quad (x,y) \in \Omega_s \qquad (8.2.7) \\ &\quad s = ++, -+, +-, --\end{aligned}$$

For smoothing iteration, we use the Jacobi method or the SOR method in red/black ordering.

As in the one dimensional case, if the coarse grid linear systems themselves (8.2.4) are solved using the two-level MCGMG algorithm

$$\delta_{2h}^{(s)} = \text{MCGMG2D}(A_{2h}^{(s)}, 0, r_{2h}^{(s)}), \qquad (8.2.8)$$

one gets a three-level two-dimensional MCGMG algorithm. This process can be done recursively and one gets a multilevel 2D MCGMG algorithm.

8.2.2 Two-Level Convergence Analysis

For $p, q = 1-N, \ldots, N$, let $v^{(p,q)}$ be the two dimensional Fourier modes (v-basis vectors) defined by

$$(v_h^{(p,q)})_{j,k} = \exp(i\pi h(pj+qk)), \quad j,k = 1-N, \ldots, N. \qquad (8.2.9)$$

The four-color Fourier modes (the w-basis vectors) $w_h^{(s,p,q)}$, corresponding to the four coarse grids $s = ++, -+, +-, --$, are defined by

$$(w_h^{(s,p,q)})_{j,k} \;=\; \begin{cases} (v_h^{(p,q)})_{j,k} & \text{if}\,(x_j, y_k) \in \Omega_s \\ 0 & \text{otherwise} \end{cases} \tag{8.2.10}$$

$$j, k = 1 - N, \ldots, N \tag{8.2.11}$$

for $p, q = 1 - N/2, \ldots, N/2$.

We first consider the coarse grid correction operator C_h. As in the discussion of the 1D case, the coarse grid correction operator can be written in the form (referring to (7.2.18))

$$C_h = I - \sum_s P_h^{(s)} (A_{2h}^{(s)})^\dagger R_h^{(s)} A_h. \tag{8.2.12}$$

We note that there is no factor 1/4 before the combination of the coarse grid corrections because the simple injection mapping of corrections is used. For $p, q = 1 - N/2, \ldots, N/2$ and $(p, q) \neq (0, 0)$, the w-transform matrices of the operators on the basis

$$E_w^{(p,q)} = (w_h^{(++,p,q)}, w_h^{(-+,p,q)}, w_h^{(+-,p,q)}, w_h^{(--,p,q)}) \tag{8.2.13}$$

are given by

$$A_h E_w^{(p,q)} = E_w^{(p,q)} \hat{A}_{h,w}^{(p,q)}, \tag{8.2.14}$$

$$R_h^{(s)} E_w^{(p,q)} = w_h^{(s,p,q)} \hat{R}_{h,w}^{(s,p,q)}, \tag{8.2.15}$$

$$A_{2h}^{(s)} w_h^{(s,p,q)} = w_h^{(s,p,q)} \hat{A}_{2h,w}^{(s,p,q)}, \tag{8.2.16}$$

$$P_h^{(s)} w_{2h}^{(s,p,q)} = E_w^{(p,q)} \hat{P}_{h,w}^{(s,p,q)}, \tag{8.2.17}$$

where $s = ++, -+, +-, --$. The w-transform matrix of the coarse grid correction operator can be written in the form

$$\begin{aligned}\hat{C}_{h,w}^{(p,q)} &= I - \textstyle\sum_s \hat{P}_{h,w}^{(s,p,q)}(\hat{A}_{2h,w}^{(s,p,q)})^{-1}\hat{R}_{h,w}^{(s,p,q)}\hat{A}_{h,w}^{(p,q)} \\ &= I - \hat{P}_{h,w}^{(p,q)}(\hat{A}_{2h,w}^{(p,q)})^{-1}\hat{R}_{h,w}^{(p,q)}\hat{A}_{h,w}^{(p,q)}\end{aligned} \tag{8.2.18}$$

where for the selected operators we have

$$\hat{R}_{h,w}^{(p,q)} = \begin{bmatrix} \hat{R}_{h,w}^{(++,p,q)} \\ \hat{R}_{h,w}^{(-+,p,q)} \\ \hat{R}_{h,w}^{(+-,p,q)} \\ \hat{R}_{h,w}^{(--,p,q)} \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & c_p & c_q & c_pc_q \\ c_p & 1 & c_pc_q & c_q \\ c_q & c_pc_q & 1 & c_p \\ c_pc_q & c_q & c_p & 1 \end{bmatrix}, \tag{8.2.19}$$

$$\begin{aligned}\hat{A}_{2h,w}^{(p,q)} &= \begin{bmatrix} \hat{A}_{2h,w}^{(++,p,q)} & & & \\ & \hat{A}_{2h,w}^{(-+,p,q)} & & \\ & & \hat{A}_{2h,w}^{(+-,p,q)} & \\ & & & \hat{A}_{2h,w}^{(--,p,q)} \end{bmatrix} \\ &= \frac{1}{h^2}(1 + \alpha - \alpha c_p^2 - c_q^2)I,\end{aligned} \tag{8.2.20}$$

$$\hat{P}_{h,w}^{(p,q)} = \begin{bmatrix} \hat{P}_{h,w}^{(++,p,q)} & \hat{P}_{h,w}^{(-+,p,q)} & \hat{P}_{h,w}^{(+-,p,q)} & \hat{P}_{h,w}^{(--,p,q)} \end{bmatrix} = I \tag{8.2.21}$$

where c_p and c_q are $\cos p\pi h$ and $\cos q\pi h$ respectively. $\hat{A}_{h,w}^{(p,q)}$ is given in (5.4.35). The w-transform matrix of the coarse grid correction operator can then be calculated using (8.2.18) to (8.2.21) and (5.4.35):

$$\begin{aligned}
\hat{C}_{h,w}^{(p,q)} &= I - \frac{h^2}{1+\alpha-\alpha c_p^2 - c_q^2}\frac{1}{4}\begin{bmatrix} 1 & c_p & c_q & c_p c_q \\ c_p & 1 & c_p c_q & c_q \\ c_q & c_p c_q & 1 & c_p \\ c_p c_q & c_q & c_p & 1 \end{bmatrix} \\
&\qquad \frac{2}{h^2}\begin{bmatrix} 1+\alpha & -\alpha c_p & -c_q & 0 \\ -\alpha c_p & 1+\alpha & 0 & -c_q \\ -c_q & 0 & 1+\alpha & -\alpha c_p \\ 0 & -c_q & -\alpha c_p & 1+\alpha \end{bmatrix} \\
&= \begin{bmatrix} \eta_1 & \eta_2 & \eta_3 & 0 \\ \eta_2 & \eta_1 & 0 & \eta_3 \\ \eta_3 & 0 & \eta_1 & \eta_2 \\ 0 & \eta_3 & \eta_2 & \eta_1 \end{bmatrix} \qquad (8.2.22)
\end{aligned}$$

where

$$\begin{cases} \eta_1 = \frac{1}{2}, \\ \eta_2 = -\frac{c_p(1-c_q^2)}{2(\alpha+1-\alpha c_p^2 - c_q^2)} \\ \eta_3 = -\frac{\alpha c_q(1-c_p^2)}{2(\alpha+1-\alpha c_p^2 - c_q^2)} r \end{cases} \qquad (8.2.23)$$

From (8.2.2), (5.4.49) and (5.4.50), the corresponding v-transform matrix is

$$C_{h,v}^{(p,q)} = \begin{bmatrix} \eta_1+\eta_2+\eta_3 & 0 & 0 & 0 \\ 0 & \eta_1-\eta_2+\eta_3 & 0 & 0 \\ 0 & 0 & \eta_1+\eta_2-\eta_3 & 0 \\ 0 & 0 & 0 & \eta_1-\eta_2-\eta_3 \end{bmatrix} \qquad (8.2.24)$$

Here we see that the *aliasing errors* caused by each of the coarse grids are eliminated.

We now consider the two-level multigrid operator given by

$$T_h = G_h^{m_2} C_h G_h^{m_1} \tag{8.2.25}$$

where G_h is the smoothing operator. If we let $\hat{G}_w^{(p,q)}$ be the w-transform matrix of the smoothing iteration operator, the two-level convergence factor of the MCGMG algorithm can be calculated by

$$\begin{aligned} \rho(T_h) &= \max_{1-N/2 \le p,q, \le N/2} \rho(\hat{T}_{h,w}^{(p,q)}) \\ &= \max_{1-N/2 \le p,q, \le N/2} \rho[(\hat{G}_{h,w}^{(p,q)})^{m_2} \hat{C}_{h,w}^{(p,q)} (\hat{G}_{h,w}^{(p,q)})^{m_1}]. \end{aligned} \tag{8.2.26}$$

Suppose the damped Jacobi method is used for smoothing iterations. The damped Jacobi operator is defined by

$$B_\gamma = I - \frac{\gamma h^2}{2(1+\alpha)} A_h \tag{8.2.27}$$

and the corresponding w-transform matrix is given by

$$\hat{B}_{\gamma,w}^{(p,q)} = I - \frac{\gamma h^2}{2(1+\alpha)} \hat{A}_{h,w}^{(p,q)} \tag{8.2.28}$$

From (5.4.35), we have

$$(\hat{B}_{\gamma,w}^{(p,q)})^m = \begin{bmatrix} \zeta_1^{(m)} & \zeta_2^{(m)} & \zeta_3^{(m)} & \zeta_4^{(m)} \\ \zeta_2^{(m)} & \zeta_1^{(m)} & \zeta_4^{(m)} & \zeta_3^{(m)} \\ \zeta_3^{(m)} & \zeta_4^{(m)} & \zeta_1^{(m)} & \zeta_2^{(m)} \\ \zeta_4^{(m)} & \zeta_3^{(m)} & \zeta_2^{(m)} & \zeta_1^{(m)} \end{bmatrix}. \tag{8.2.29}$$

where

$$\begin{cases} \zeta_1^{(m)} = \frac{1}{4}[(\mu^{(p,q)})^m + (\mu^{(p-N,q)})^m + (\mu^{(p,q-N)})^m + (\mu^{(p-N,q-N)})^m], \\ \zeta_2^{(m)} = \frac{1}{4}[(\mu^{(p,q)})^m - (\mu^{(p-N,q)})^m + (\mu^{(p,q-N)})^m - (\mu^{(p-N,q-N)})^m], \\ \zeta_3^{(m)} = \frac{1}{4}[(\mu^{(p,q)})^m + (\mu^{(p-N,q)})^m - (\mu^{(p,q-N)})^m - (\mu^{(p-N,q-N)})^m], \\ \zeta_4^{(m)} = \frac{1}{4}[(\mu^{(p,q)})^m - (\mu^{(p-N,q)})^m - (\mu^{(p,q-N)})^m + (\mu^{(p-N,q-N)})^m] \end{cases} \tag{8.2.30}$$

and

$$\mu^{(p,q)} = 1 - \gamma + \gamma \frac{\alpha c_p + c_q}{1+\alpha}. \tag{8.2.31}$$

If we let $m = m_1 + m_2$, it can be verified by calculation that

$$\begin{aligned}
(\hat{B}^{(p,q)}_{\gamma,w})^{m_2} \hat{C}^{(p,q)}_{h,w} (\hat{B}^{(p,q)}_{\gamma,w})^{m_1} &= \hat{C}^{(p,q)}_{h,w} (\hat{B}^{(p,q)}_{\gamma,w})^{m} \\
&= \begin{bmatrix} \kappa_1^{(m)} & \kappa_2^{(m)} & \kappa_3^{(m)} & \kappa_4^{(m)} \\ \kappa_2^{(m)} & \kappa_1^{(m)} & \kappa_4^{(m)} & \kappa_3^{(m)} \\ \kappa_3^{(m)} & \kappa_4^{(m)} & \kappa_1^{(m)} & \kappa_2^{(m)} \\ \kappa_4^{(m)} & \kappa_3^{(m)} & \kappa_2^{(m)} & \kappa_1^{(m)} \end{bmatrix}.
\end{aligned} \tag{8.2.32}$$

where

$$\begin{cases}
\kappa_1^{(m)} = \eta_1 \zeta_1^{(m)} + \eta_2 \zeta_2^{(m)} + \eta_3 \zeta_3^{(m)}, \\
\kappa_2^{(m)} = \eta_2 \zeta_1^{(m)} + \eta_1 \zeta_2^{(m)} + \eta_3 \zeta_4^{(m)}, \\
\kappa_3^{(m)} = \eta_3 \zeta_1^{(m)} + \eta_1 \zeta_3^{(m)} + \eta_2 \zeta_4^{(m)}, \\
\kappa_4^{(m)} = \eta_3 \zeta_2^{(m)} + \eta_2 \zeta_3^{(m)} + \eta_1 \zeta_4^{(m)}.
\end{cases} \tag{8.2.33}$$

The convergence factor of the two-level MCGMG algorithm with Jacobi smoothing iterations can be then calculated by

$$\rho(T_h) = \max_{1-N/2 \leq p,q, \leq N/2} \rho[\hat{C}^{(p,q)}_{h,w} (\hat{B}^{(p,q)}_{\gamma,w})^{m}] \tag{8.2.34}$$

Table 8.1 lists values of $\rho(T_h)$. Here the number of smoothing iterations $m = 1$ and the number of grid points $N = 64$. It shows that at extrapolation factor $\gamma = 0.66$, $\rho(T_L)$ reaches a minimum of 0.319.

Table 8.2 shows the relationship between the convergence factor and the coefficient α. One sees the deterioration of the convergence in anisotropic cases. We will discuss this issue in the next chapter.

In the case of using the Red/Black SOR iterative method defined in (3.5.42) for smoothing, the w-transform matrices corresponding to the red

Table 8.1: Convergence Factors of MCGMG-Jacobi vs. γ ($\alpha = 1$)

γ	ρ
0.50	0.4763
0.60	0.3771
0.65	0.3276
0.66	0.3190
0.67	0.3390
0.70	0.3989
0.75	0.4988
1.00	0.9982

Table 8.2: Convergence Factor of MCGMG-Jacobi vs. α ($\gamma_* = 0.66$)

α	ρ
0.00001 or 100000	0.998623
0.00010 or 10000.	0.998414
0.00100 or 1000.0	0.996334
0.01000 or 100.00	0.980861
0.10000 or 10.000	0.863659
1	0.319042

sub-iteration (the first formula in (3.5.42)) and to the black sub-iteration (the second formula in (3.5.42)) are given by

$$\hat{S}_{\omega}^{(r,p,q)} = \begin{bmatrix} 1-\omega & \xi_2 & \xi_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \xi_3 & \xi_2 & 1-\omega \end{bmatrix} \tag{8.2.35}$$

and

$$\hat{S}_{\omega}^{(b,p,q)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \xi_2 & 1-\omega & 0 & \xi_3 \\ \xi_3 & 0 & 1-\omega & \xi_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.2.36}$$

respectively, where

$$\xi_2 = \frac{\omega\alpha c_p}{1+\alpha} \tag{8.2.37}$$

and

$$\xi_3 = \frac{\omega c_q}{1+\alpha}. \tag{8.2.38}$$

If the black unknowns are updated first followed by the red ones, we have

$$\hat{S}_{\omega}^{(rb,p,q)} = \hat{S}_{\omega}^{(r,p,q)}\hat{S}_{\omega}^{(b,p,q)} = \begin{bmatrix} \xi_1 & (1-\omega)\xi_2 & (1-\omega)\xi_3 & \xi_2\xi_3 \\ \xi_2 & 1-\omega & 0 & \xi_3 \\ \xi_3 & 0 & 1-\omega & \xi_2 \\ \xi_2\xi_3 & (1-\omega)\xi_3 & (1-\omega)\xi_2 & \xi_1 \end{bmatrix} \tag{8.2.39}$$

Otherwise, we have

$$\hat{S}_{\omega}^{(br,p,q)} = \hat{S}_{\omega}^{(b,p,q)}\hat{S}_{\omega}^{(r,p,q)} = \begin{bmatrix} 1-\omega & \xi_2 & \xi_3 & 0 \\ (1-\omega)\xi_2 & \xi_1 & \xi_2\xi_3 & (1-\omega)\xi_3 \\ 0 & \xi_3 & \xi_2 & 1-\omega \\ (1-\omega)\xi_3 & \xi_2\xi_3 & \xi_1 & (1-\omega)\xi_2 \end{bmatrix} \tag{8.2.40}$$

where

$$\xi_1 = 1 - \omega + \xi_2^2 + \xi_3^2. \tag{8.2.41}$$

From (8.2.2), (8.2.39) and (8.2.40), one obtains the w-transform matrix of the two-level MCGMG algorithm using SOR smoothing with red/black or black/red ordering

$$\hat{T}_{h,w}^{(rb,p,q)} = \hat{C}_{h,w}^{(p,q)} \hat{S}_{\omega}^{(rb,p,q)} = \begin{bmatrix} \beta_1 & \gamma_2 & \gamma_3 & \beta_4 \\ \beta_2 & \gamma_1 & \gamma_4 & \beta_3 \\ \beta_3 & \gamma_4 & \gamma_1 & \beta_2 \\ \beta_4 & \gamma_3 & \gamma_2 & \beta_1 \end{bmatrix} \tag{8.2.42}$$

and

$$\hat{T}_{h,w}^{(br,p,q)} = \hat{C}_{h,w}^{(p,q)} \hat{S}_{\omega}^{(br,p,q)} = \begin{bmatrix} \gamma_1 & \beta_2 & \beta_3 & \gamma_4 \\ \gamma_2 & \beta_1 & \beta_4 & \gamma_3 \\ \gamma_3 & \beta_4 & \beta_1 & \gamma_2 \\ \gamma_4 & \beta_3 & \beta_2 & \gamma_1 \end{bmatrix} \tag{8.2.43}$$

where

$$\left\{ \begin{array}{lcl} \beta_1 & = & \eta_1\xi_1 + \eta_2\xi_2 + \eta_3\xi_3 \\ \beta_2 & = & \eta_1\xi_2 + \eta_2\xi_1 + \eta_3\xi_2\xi_3 \\ \beta_3 & = & \eta_1\xi_3 + \eta_2\xi_2\xi_3 + \eta_3\xi_1 \\ \beta_4 & = & \eta_1\xi_2\xi_3 + \eta_2\xi_3 + \eta_3\xi_2 \\ \gamma_1 & = & (1-\omega)(\eta_1 + \eta_2\xi_2 + \eta_3\xi_3) \\ \gamma_2 & = & (1-\omega)(\eta_1\xi_2 + \eta_2) \\ \gamma_3 & = & (1-\omega)(\eta_1\xi_3 + \eta_3) \\ \gamma_4 & = & (1-\omega)(\eta_2\xi_3 + \eta_3\xi_2) \end{array} \right. \tag{8.2.44}$$

and η_1, η_2, and η_3 are defined in (8.2.23). Therefore we have the following lemma.

Table 8.3: Convergence Factor of MCGMG-SOR vs. ω ($\alpha = 1$)

ω	ρ
0.50	0.4795
0.70	0.2828
0.90	0.0900
0.92	0.0712
0.94	0.0795
1.00	0.1240
1.20	0.2788
1.50	0.5288

Lemma 8.1 *The matrices $\hat{T}_{h,w}^{(rb,p,q)}$ (8.2.42) and $\hat{T}_{h,w}^{(br,p,q)}$ (8.2.43) have the same eigenvalue set.*

Proof: It is easy to see that there is a permutation relation between the matrices $\hat{T}_{h,w}^{(rb,p,q)}$ and $\hat{T}_{h,w}^{(br,p,q)}$. This means that the matrix $\hat{T}_{h,w}^{(rb,p,q)}$ is similar to the matrix $\hat{T}_{h,w}^{(br,p,q)}$.

We notice that the nonzero eigenvalues of the matrix $\hat{T}_{h,w}^{(rb,p,q)}$, when $\omega = 1$, can be written as

$$\lambda = \beta_1 \pm \beta_4. \tag{8.2.45}$$

In the case of $N = 64$, we use a numerical procedure to compute the convergence factor defined by

$$\rho(T_h) = \max_{1-N/2 \leq p,q, \leq N/2} \rho(\hat{T}_{h,w}^{(br,p,q)})a \tag{8.2.46}$$

Table 8.3 lists the convergence factors of the MCGMG with the SOR smoothing iteration with red/black ordering for different values of ω. The convergence factor is about 0.0712, when ω is 0.92. Table 8.4 lists the convergence

Table 8.4: Convergence Factor of MCGMG-SOR vs. α ($\omega_* = 0.92$)

α	ρ
0.001	0.9942
0.010	0.9644
0.100	0.7492
1.000	0.0712
10.00	0.7492
100.0	0.9644
1000.	0.9942

factors in anisotropic cases. Here again, the performance of the algorithm deteriorates in the cases where $\alpha \neq 1$.

8.2.3 Numerical Results

We used the multilevel 2D MCGMG algorithm corresponding to the two-level algorithm defined in Figure 8.2 to solve the extended system of the test problem (5.5.51). In the algorithm we used linear interpolation of corrections, full weighting restriction of residuals and the damped Jacobi smoothing iterations.

Table 8.5 shows the convergence factors of the MCGMG algorithm for the case with the grid size $N = 64$. We use a six-level scheme which is the maximum number of levels allowed ($2^6 = 64$) in this case; ($2^6 = 64$) in this case. γ are the extrapolation factors of the damped Jacobi method, m_1 and m_2 are the number of pre-smoothing and post-smoothing iterations respectively. The convergence factors are the average values of 5 multigrid cycles. Comparing to the theoretical results in Table 8.1, one sees that the numerical convergence factors are below the theoretical upper bounds. From

Table 8.5: Numerical Convergence Factors of 6-level 2D MCGMG-J

γ	(m_1, m_2)		
	(0,1)	(1,1)	(1,2)
0.5	.244	.141	.114
0.6	.190	.125	.102
0.7	.154	.116	.093
0.8	.146	.108	.085
0.9	.188	.129	.094
1.0	.241	.206	.186

Table 5.3, one also sees that the MCGMG methods converges much faster than the correponding standard multigrid methods.

8.3 FDMG Methods in 2D

8.3.1 The Two-Level FDMG Algorithm in 2D

As in the 1D cases, the procedure of the 2D FDMG algorithm is the same as that of the 2D MCGMG algorithm defined in Figure 8.2. In the FDMG algorithm, different restriction and interplolation operators are used on different coarse grids. The restriction operators, corresponding to the four coarse grids Ω_{++}, $\Omega_{++}^{(E)}$, $\Omega_{++}^{(E)}$ and $\Omega_{++}^{(E)}$ respectively, are defined by

$$\begin{aligned} r_{2h}(x,y) &= (R_h^{(++)} r_h)(x,y) \\ &= \tfrac{1}{4}(r_h(x-h,y+h) + 2r_h(x,y+h) + r_h(x+h,y+h) \\ &+ 2r_h(x-h,y) + 4r_h(x,y) + 2r_h(x+h,y) \qquad (8.3.47) \\ &+ r_h(x-h,y-h) + 2r_h(x,y-h) + r_h(x+h,y-h)) \\ &\quad (x,y) \in \Omega_{++}. \end{aligned}$$

$$\begin{aligned}
r_{2h}(x,y) &= (R_h^{(-+)} r_h)(x,y) \\
&= \tfrac{1}{4}(-r_h(x-h,y+h) + 2r_h(x,y+h) - r_h(x+h,y+h) \\
&- 2r_h(x-h,y) + 4r_h(x,y) - 2r_h(x+h,y) \\
&- r_h(x-h,y-h) + 2r_h(x,y-h) - r_h(x+h,y-h)) \\
&\quad (x,y) \in \Omega_{-+}.
\end{aligned} \tag{8.3.48}$$

$$\begin{aligned}
r_{2h}(x,y) &= (R_h^{(+-)} r_h)(x,y) \\
&= \tfrac{1}{4}(-r_h(x-h,y+h) - 2r_h(x,y+h) - r_h(x+h,y+h) \\
&+ 2r_h(x-h,y) + 4r_h(x,y) + 2r_h(x+h,y) \\
&- r_h(x-h,y-h) - 2r_h(x,y-h) - r_h(x+h,y-h)) \\
&\quad (x,y) \in \Omega_{+-}.
\end{aligned} \tag{8.3.49}$$

$$\begin{aligned}
r_{2h}(x,y) &= (R_h^{(--)} r_h)(x,y) \\
&= \tfrac{1}{4}(r_h(x-h,y+h) - 2r_h(x,y+h) + r_h(x+h,y+h) \\
&- 2r_h(x-h,y) + 4r_h(x,y) - 2r_h(x+h,y) \\
&+ r_h(x-h,y-h) - 2r_h(x,y-h) + r_h(x+h,y-h)) \\
&\quad (x,y) \in \Omega_{--}.
\end{aligned} \tag{8.3.50}$$

The four corresponding interpolation operators are defined by

$$\begin{aligned}
&\delta_h(x,y) = (P_h^{(++)} \delta_{2h})(x,y) \\
&= \begin{cases}
\delta_{2h}(x,y) & (x,y) \in \Omega_{++} \\
\dfrac{1}{2}(\delta_{2h}(x-h,y) + \delta_{2h}(x+h,y)) & (x,y) \in \Omega_{-+} \\
\dfrac{1}{2}(\delta_{2h}(x,y-h) + \delta_{2h}(x,y+h)) & (x,y) \in \Omega_{+-} \\
\dfrac{1}{4}(\delta_{2h}(x-h,y-h) + \delta_{2h}(x-h,y+h) & \\
+\delta_{2h}(x+h,y-h) + \delta_{2h}(x+h,y+h)) & (x,y) \in \Omega_{--}.
\end{cases}
\end{aligned} \tag{8.3.51}$$

$$\delta_h(x,y) = (P_h^{(-+)}\delta_{2h})(x,y)$$
$$= \begin{cases} \delta_{2h}(x,y) & (x,y) \in \Omega_{-+} \\ \frac{-1}{2}(\delta_{2h}(x-h,y) + \delta_{2h}(x+h,y)) & (x,y) \in \Omega_{++} \\ \frac{1}{2}(\delta_{2h}(x,y-h) + \delta_{2h}(x,y+h)) & (x,y) \in \Omega_{--} \\ \frac{-1}{4}(\delta_{2h}(x-h,y-h) + \delta_{2h}(x-h,y+h) & \\ +\delta_{2h}(x+h,y-h) + \delta_{2h}(x+h,y+h)) & (x,y) \in \Omega_{+-}. \end{cases} \tag{8.3.52}$$

$$\delta_h(x,y) = (P_h^{(+-)}\delta_{2h})(x,y)$$
$$= \begin{cases} \delta_{2h}(x,y) & (x,y) \in \Omega_{+-} \\ \frac{1}{2}(\delta_{2h}(x-h,y) + \delta_{2h}(x+h,y)) & (x,y) \in \Omega_{--} \\ \frac{-1}{2}(\delta_{2h}(x,y-h) + \delta_{2h}(x,y+h)) & (x,y) \in \Omega_{++} \\ \frac{-1}{4}(\delta_{2h}(x-h,y-h) + \delta_{2h}(x-h,y+h) & \\ +\delta_{2h}(x+h,y-h) + \delta_{2h}(x+h,y+h)) & (x,y) \in \Omega_{-+}. \end{cases} \tag{8.3.53}$$

$$\delta_h(x,y) = (P_h^{(--)}\delta_{2h})(x,y)$$
$$= \begin{cases} \delta_{2h}(x,y) & (x,y) \in \Omega_{--} \\ \frac{-1}{2}(\delta_{2h}(x-h,y) + \delta_{2h}(x+h,y)) & (x,y) \in \Omega_{+-} \\ \frac{-1}{2}(\delta_{2h}(x,y-h) + \delta_{2h}(x,y+h)) & (x,y) \in \Omega_{-+} \\ \frac{1}{4}(\delta_{2h}(x-h,y-h) + \delta_{2h}(x-h,y+h) & \\ +\delta_{2h}(x+h,y-h) + \delta_{2h}(x+h,y+h)) & (x,y) \in \Omega_{++}. \end{cases} \tag{8.3.54}$$

If the operators are written in matrix form, it can be shown that

$$P_h^{(s)} = (R_h^{(s)})^T \qquad s = ++, -+, +-, --\,. \tag{8.3.55}$$

The operators of the coarse grid systems are constructed by

$$A_{2h}^{(s)} = R_h^{(s)} A_h P_h^{(s)} \qquad s = ++, -+, +-, -- . \tag{8.3.56}$$

This completes the description of the 2D FDMG algorithm.

If more than two levels are involved, each $2h$ coarse grid can be considered a fine grid for the $4h$ level and thus has four $4h$ coarse grids related to it. The $2h$ restriction operators can then be defined similarly with h replaced by $2h$.

8.3.2 Two-Level Convergence Analysis

For simplicity, we use the index numbers $1, 2, 3$, and 4 to represent $++, -+, +-$, and $--$ respectively. The w-transform matrices of the restriction operators $R_h^{(s)}$, $P_h^{(s)}$ and $A_{2h}^{(s)}$ can be written in the forms of (8.2.19), (8.2.21), and (5.4.35) respectively. Thus for the restriction operator we have

$$\begin{aligned}\hat{R}_{h,w}^{(p,q)} &= \begin{bmatrix} \hat{R}_{h,w}^{(1,p,q)} \\ \hat{R}_{h,w}^{(2,p,q)} \\ \hat{R}_{h,w}^{(3,p,q)} \\ \hat{R}_{h,w}^{(4,p,q)} \end{bmatrix} = \begin{bmatrix} \hat{R}_{h,w}^{(++,p,q)} \\ \hat{R}_{h,w}^{(-+,p,q)} \\ \hat{R}_{h,w}^{(+-,p,q)} \\ \hat{R}_{h,w}^{(--,p,q)} \end{bmatrix} \\ &= \begin{bmatrix} 1 & c_p & c_q & c_p c_q \\ -c_p & 1 & -c_p c_q & c_q \\ -c_q & -c_p c_q & 1 & c_p \\ c_p c_q & -c_q & -c_p & 1 \end{bmatrix}\end{aligned} \tag{8.3.57}$$

where $c_p = \cos p\pi h$ and $c_q = \cos q\pi h$. The w-transform matrix of the interpolation operator is given by

$$\hat{P}_{h,w}^{(p,q)} = \begin{bmatrix} \hat{P}_{h,w}^{(1,p,q)} & \hat{P}_{h,w}^{(2,p,q)} & \hat{P}_{h,w}^{(3,p,q)} & \hat{P}_{h,w}^{(4,p,q)} \end{bmatrix} = (\hat{R}_{h,w}^{(p,q)})^T \tag{8.3.58}$$

The w-transform matrix of the coarse grid operator is given by

$$\hat{A}_{2h,w}^{(p,q)} = \begin{bmatrix} \tilde{a}_{11} & & & \\ & \tilde{a}_{22} & & \\ & & \tilde{a}_{33} & \\ & & & \tilde{a}_{44} \end{bmatrix} \tag{8.3.59}$$

where

$$\begin{aligned} \tilde{a}_{11} &= \hat{A}_{2h,w}^{(1,p,q)} = \hat{R}_{h,w}^{(1,p,q)} \hat{A}_{h,w}^{(p,q)} \hat{P}_{h,w}^{(1,p,q)} \\ &= \tfrac{2}{h^2}(1 + \alpha + (1-\alpha)c_p^2 + (\alpha - 1)c_q^2 \\ &\quad -(1+\alpha)c_p^2 c_q^2) \\ \tilde{a}_{22} &= \hat{A}_{2h,w}^{(2,p,q)} = \hat{R}_{h,w}^{(2,p,q)} \hat{A}_{h,w}^{(p,q)} \hat{P}_{h,w}^{(2,p,q)} \\ &= \tfrac{2}{h^2}(1 + \alpha + (1+3\alpha)c_p^2 + (\alpha - 1)c_q^2 \\ &\quad +(3\alpha - 1)c_p^2 c_q^2) \\ \tilde{a}_{33} &= \hat{A}_{2h,w}^{(3,p,q)} = \hat{R}_{h,w}^{(3,p,q)} \hat{A}_{h,w}^{(p,q)} \hat{P}_{h,w}^{(3,p,q)} \\ &= \tfrac{2}{h^2}(1 + \alpha + (1-\alpha)c_p^2 + (3 + \alpha)c_q^2 \\ &\quad +(3\alpha - 1)c_p^2 c_q^2) \\ \tilde{a}_{44} &= \hat{A}_{2h,w}^{(4,p,q)} = \hat{R}_{h,w}^{(4,p,q)} \hat{A}_{h,w}^{(p,q)} \hat{P}_{h,w}^{(4,p,q)} \\ &= \tfrac{2}{h^2}(1 + \alpha + (1+3\alpha)c_p^2 + (3 + \alpha)c_q^2 \\ &\quad +3(1+\alpha)c_p^2 c_q^2). \end{aligned} \tag{8.3.60}$$

The w-transform matrix of the coarse grid correction operator is then given by

$$\hat{C}_{h,w}^{(p,q)} = I - \hat{P}_{h,w}^{(p,q)}(\hat{A}_{2h,w}^{(p,q)})^{-1}\hat{R}_{h,w}^{(p,q)}\hat{A}_{h,w}^{(p,q)}. \tag{8.3.61}$$

We can make a similarity transformation by multiplying $\hat{P}_{h,w}^{(p,q)}$ to the right side of the $\hat{C}_{h,w}^{(p,q)}$ and $(\hat{P}_{h,w}^{(p,q)})^{-1}$ to the left side, and we have

$$\tilde{C}_{h,w}^{(p,q)} = (\hat{P}_{h,w}^{(p,q)})^{-1}\hat{C}_{h,w}^{(p,q)}\hat{P}_{h,w}^{(p,q)} = I - (\hat{A}_{2h,w}^{(p,q)})^{-1}\hat{R}_{h,w}^{(p,q)}\hat{A}_{h,w}^{(p,q)}\hat{P}_{h,w}^{(p,q)}. \tag{8.3.62}$$

If we let

$$\tilde{A}^{(p,q)}_{h,w} = \hat{R}^{(p,q)}_{h,w}\hat{A}^{(p,q)}_{h,w}\hat{P}^{(p,q)}_{h,w} \tag{8.3.63}$$

then from (8.3.59), the matrix $\hat{A}^{(p,q)}_{2h,w}$ is the main diagonal part of the matrix $\tilde{A}^{(p,q)}_{h,w}$. Therefore we have the following lemma.

Lemma 8.2 *The coarse grid correction of the FDMG algorithm is equivalent to a block Jacobi iteration applied to the matrix* $\tilde{A}_h = R_h A_h P_h$.

Proof: From (8.3.62), we know that $\hat{C}^{(p,q)}_{h,w}$ is similar to applying a Jacobi iteration to $\tilde{A}^{(p,q)}_{h,w}$. Since all subspaces $E^{(p,q)}$ are the invariant subspace w.r.t. C_h, the conclusion follows.

If matrix $\tilde{A}^{(p,q)}_{h,w}$ is a diagonal matrix, the coarse grid correction is exact. In order to obtain the spectral number of $\hat{C}^{(p,q)}_{h,w}$, we rewrite (8.3.61) in the form

$$\begin{aligned}
\tilde{C}^{(p,q)}_{h,w} &= I - Diag(\tilde{A}^{(p,q)}_h)^{-1}\tilde{A}^{(p,q)}_h \\
&= \begin{bmatrix} \tilde{a}_{11}^{-1} & 0 & 0 & 0 \\ 0 & \tilde{a}_{22}^{-1} & 0 & 0 \\ 0 & 0 & \tilde{a}_{33}^{-1} & 0 \\ 0 & 0 & 0 & \tilde{a}_{44}^{-1} \end{bmatrix} \begin{bmatrix} 0 & -\tilde{a}_{12} & -\tilde{a}_{13} & -\tilde{a}_{14} \\ -\tilde{a}_{21} & 0 & -\tilde{a}_{23} & -\tilde{a}_{24} \\ -\tilde{a}_{31} & -\tilde{a}_{32} & 0 & -\tilde{a}_{34} \\ -\tilde{a}_{41} & -\tilde{a}_{42} & -\tilde{a}_{43} & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & -\tilde{a}_{21}\tilde{a}_{11}^{-1} & -\tilde{a}_{31}\tilde{a}_{11}^{-1} & 0 \\ -\tilde{a}_{21}\tilde{a}_{22}^{-1} & 0 & 0 & -\tilde{a}_{31}\tilde{a}_{22}^{-1} \\ -\tilde{a}_{31}\tilde{a}_{33}^{-1} & 0 & 0 & -\tilde{a}_{21}\tilde{a}_{33}^{-1} \\ 0 & -\tilde{a}_{31}\tilde{a}_{44}^{-1} & -\tilde{a}_{21}\tilde{a}_{44}^{-1} & 0 \end{bmatrix}
\end{aligned} \tag{8.3.64}$$

where $\tilde{a}_{jk} = \hat{R}^{(j,p,q)}_{h,w}\hat{A}^{(p,q)}_{h,w}\hat{P}^{(k,p,q)}_{h,w}$ for $j,k = 1,2,3,4$. The spectral radius of the matrix $\tilde{C}^{(p,q)}_{h,w}$ is given by

$$\rho(\tilde{C}^{(p,q)}_{h,w}) = \left|\frac{b + \sqrt{b^2 - 4c}}{2}\right|^{\frac{1}{2}} \tag{8.3.65}$$

Table 8.6: Convergence Factor of FDMG Without Smoothing vs. h

h	ρ
$\frac{1}{8}$	0.329036
$\frac{1}{16}$	0.332694
$\frac{1}{32}$	0.333326
$\frac{1}{64}$	0.333247
$\frac{1}{128}$	0.333313

where

$$b = \tilde{a}_{21}^2 \left(\frac{1}{\tilde{a}_{11}\tilde{a}_{22}} + \frac{1}{\tilde{a}_{22}\tilde{a}_{44}} \right) + \tilde{a}_{31}^2 \left(\frac{1}{\tilde{a}_{11}\tilde{a}_{33}} + \frac{1}{\tilde{a}_{33}\tilde{a}_{44}} \right) > 0 \tag{8.3.66}$$

and

$$c = \frac{(\tilde{a}_{21}^2 - \tilde{a}_{31}^2)^2}{\tilde{a}_{11}\tilde{a}_{22}\tilde{a}_{33}\tilde{a}_{44}}. \tag{8.3.67}$$

The convergence factor of the coarse grid correction of the FDMG is bounded by

$$\rho(C_h) \leq \max_{1-\frac{N}{2} \leq p,q \leq \frac{N}{2}} \rho(\tilde{C}_{h,w}^{(p,q)}). \tag{8.3.68}$$

Table 8.6 gives the upper bound of the convergence factor (ρ) for different mesh sizes (h). Table 8.7 shows that this bound remains valid in anisotropic cases. The convergence factor as a function of the Fourier modes is plotted in Figure 8.3.

We first consider using the damped Jacobi method for smoothing iterations. The w-transform matrix of the two level FDMG operator is given by

$$\hat{T}_{h,w}^{(p,q)} = \hat{C}_{h,w}^{(p,q)} \hat{B}_{\gamma}^{(p,q)} \tag{8.3.69}$$

Table 8.7: Convergence Factor of FDMG Without Smoothing vs. α

α	ρ
10^{-4}	0.333326
10^{-3}	0.333326
10^{-2}	0.333326
10^{-1}	0.333326
10^{0}	0.333326
10^{+1}	0.333326
10^{+2}	0.333326
10^{+3}	0.333326
10^{+4}	0.333326

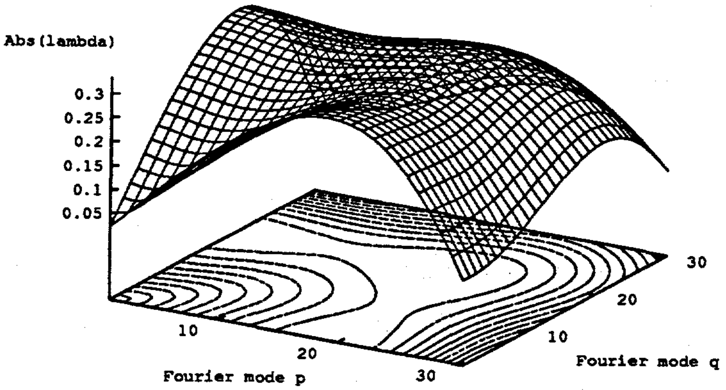


Figure 8.3: Convergence Factor of FDMG with no Smoothing vs. Fourier Mode

where $\hat{C}^{(p,q)}_{h,w}$ is given in (8.3.61) and $\hat{B}^{(p,q)}_{\gamma,w}$ is given by

$$\hat{B}^{(p,q)}_{\gamma} = I - \gamma D^{-1}_{\hat{A}^{(p,q)}_h} \hat{A}^{(p,q)}_{h,w} = \begin{bmatrix} b_1 & b_2 & b_3 & 0 \\ b_2 & b_1 & 0 & b_3 \\ b_3 & 0 & b_1 & b_2 \\ 0 & b_3 & b_2 & b_1 \end{bmatrix} \tag{8.3.70}$$

where

$$\begin{aligned} b_1 &= 1-\gamma, \\ b_2 &= \gamma\frac{\alpha\cos\theta_p}{1+\alpha}, \\ b_3 &= \gamma\frac{\cos\theta_q}{1+\alpha}. \end{aligned} \tag{8.3.71}$$

From (5.4.42) and (8.3.69), the two-level convergence factor can be calculated. Table 8.8 lists the convergence factors with the different values of γ. Table 8.9 lists the convergence factors with different values of anisotropy parameter α. Figure 8.4 illustrates the convergence factor vs. the Fourier modes with the optimal γ for the Poisson equation ($\alpha = 1$).

We now consider using the red/black SOR method for smoothing iterations. The w-transform matrix of the two-level operator is given by

$$\hat{T}^{(p,q)}_{h,w} = \hat{C}^{(p,q)}_{h,w}\hat{S}^{(rb,p,q)}_{\omega} \tag{8.3.72}$$

where $\hat{C}^{(p,q)}_{h,w}$ is defined in (8.3.61) and $\hat{S}^{(rb,p,q)}_{\omega}$ is defined in (8.2.39).

Table 8.10 lists the convergence factors of the two-level FDMG vs. ω. Table 8.11 lists the convergence factors vs. the anisotropic parameter α. The optimal value of ω (to make $\rho(T_h)$ minimum) is around 1.2. The distribution of the convergence factor of the two-level FDMG algorithm on the Fourier mode domain is shown in Figure 8.5. With the grid size 64 × 64, the two-grid convergence rate of the FDMG with one RB-SOR iteration as a smoother is about 0.1126 at $\alpha = 1$.

Table 8.8: Convergence Factors for FDMG-Jacobi vs. γ ($\alpha = 1$)

γ	ρ
0.50	0.2741
0.70	0.2481
1.00	0.2047
1.20	0.1716
1.23	0.1666
1.30	0.1910
1.50	0.2713

Table 8.9: Convergence Factor of FDMG-Jacobi vs. α ($\gamma_* = 1.23$)

α	ρ
0.0001	0.2706
0.0010	0.2704
0.0100	0.2881
0.1000	0.3029
1.0000	0.1666
10.000	0.3029
100.00	0.2881
1000.0	0.2704
10000.	0.2706

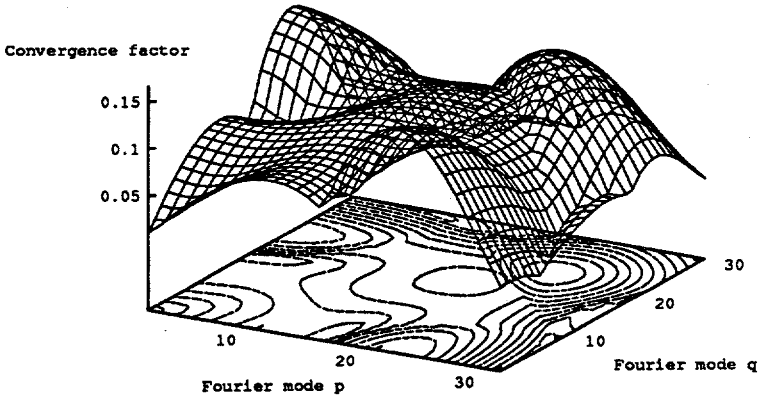


Figure 8.4: Convergence Factor of FDMG-Jacobi vs. Fourier Mode

Table 8.10: Convergence Factor of FDMG-SOR vs. ω ($\alpha = 1$)

ω	ρ
0.50	0.2670
0.70	0.2336
0.90	0.1940
1.00	0.1704
1.10	0.1437
1.20	0.1130
1.21	0.1126
1.30	0.1308
1.50	0.1772

Table 8.11: Convergence Factor of FDMG-SOR vs. α ($\omega_* = 1.21$)

α	ρ
0.0001	0.1410
0.0010	0.1431
0.0100	0.2867
0.1000	0.2884
1.0000	0.1126
10.000	0.2884
100.00	0.2867
1000.0	0.1431
10000.	0.1410

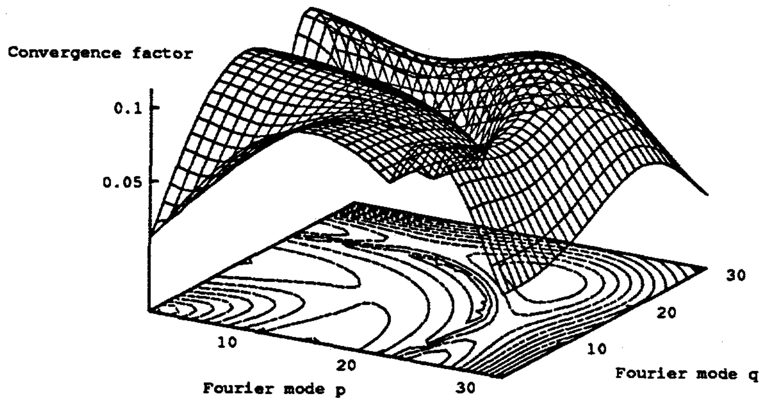


Figure 8.5: Convergence Factor of FDMG-SOR vs. Fourier Mode

Comparing the convergence factors of FDMG method with and without smoothing, one sees that both the Jacobi iteration and the SOR iteration provide much improvement for the convergence rate in some value ranges of α but have little effect in others.

8.4 PMG Methods in 2D

8.4.1 The Two-Level PMG Algorithm in 2D

A two-dimensional PMG algorithm is a straighforward extension of the one-dimensional PMG algorithm defined in Figure 7.8. The only difference is that all the operators in a two dimensional PMG method are defined on a single two-dimensional grid. Therefore, the 1D PMG algorithm defined in Figure 7.8 can also represent a 2D PMG algorithm with the operators R_h, P_h and A_{2h} replaced by corresponding 2D operators.

For the following analysis, we assume that the restriction-like smoothing operator is defined by

$$\begin{aligned}(R_h\delta_h)(x,y) &= \frac{1}{16}(\delta_h(x-h,y+h)+2\delta_h(x,y+h)+\delta_h(x+h,y+h)\\ &\quad +2\delta_h(x-h,y)+4\delta_h(x,y)+2\delta_h(x+h,y)\\ &\quad +\delta_h(x-h,y-h)+2\delta_h(x,y-h)+\delta_h(x+h,y-h))\\ &\quad (x,y)\in\Omega_h \end{aligned} \tag{8.4.73}$$

and the interpolation-like smoothing operator is defined by

$$(P_h\delta_{2h})(x,y)=\delta_{2h}(x,y). \tag{8.4.74}$$

For problem (8.1.1), coarse level finite-difference operator A_{2h} is defined by (8.2.7) but on the same grid Ω_h. We will use the 2D Jacobi method defined in (8.2.27) for smoothing iterations.

8.4.2 Two-Level Convergence Analysis

All the operators in the PMG algorithm we selected are defined on a single grid. Each of these operators, when being applied to a Fourier mode, will reduce to a scalar function multiplied by the Fourier mode. Therefore, the standard Fourier analysis of the PMG methods becomes particularly convenient.

We denote by $\lambda_{p,q}(K)$ the scalar function of the operator K associated with the Fourier mode $v_h^{(p,q)}$ (8.2.9). It can be shown that

$$\lambda_{p,q}(A_h) = 2h^{-2}(\alpha + 1 - \alpha c_p - c_q) \tag{8.4.75}$$

$$\begin{aligned} \lambda_{p,q}(A_{2h}) &= 2(2h)^{-2}(\alpha + 1 - \alpha c_{2p} - c_{2q}) \\ &= h^{-2}(\alpha + 1 - \alpha c_p^2 - c_q^2) \end{aligned} \tag{8.4.76}$$

and

$$\lambda_{p,q}(R_h) = \frac{1}{4}(1 + c_p + c_q + c_p c_q) \tag{8.4.77}$$

where $c_p = \cos p\pi h$. Since the operator R_h is the identity operator, the two-level coarse grid correction operator can be written in the form

$$C_h = I - R_h A_{2h}^{\dagger} A_h \tag{8.4.78}$$

and we have

$$\lambda_{p,q}(C_h) = 1 - \lambda_{p,q}(R_h)\frac{\lambda_{p,q}(A_h)}{\lambda_{p,q}(A_{2h})}a. \tag{8.4.79}$$

Here $A_{2h}^{\dagger}$ denotes the Moore-Penrose inverse of A_{2h}. We note that $\lambda_{0,0}(C_h)$ is not defined. However, $\lambda_{p,q}(C_h)$ can be treated as a function $\lambda_{C_h}(a,b)$ of $a = c_p$ and $b = c_q$. Since Fourier mode $v_h^{(0,0)}$ has been removed from the purified solutions, one can redefine $\lambda_{0,0}(C_h)$ by

$$\lambda_{0,0}(C_h) = \lim_{\substack{a \to 1 \\ b \to 1}} \lambda_{C_h}(a,b). \tag{8.4.80}$$

For the Jacobi iteration operator (8.2.27) we have

$$\lambda_{p,q}(B_\gamma) = 1 - \frac{\gamma}{2+2\alpha}\lambda_{p,q}(A_h). \tag{8.4.81}$$

Since the two-level PMG operator is given by

$$T_h = (B_\gamma)^m C_h \tag{8.4.82}$$

where $m = m_1 + m_2$, the two-level convergence factor of the PMG algorithm is bounded by

$$\begin{aligned} \rho(T_h) &= \max_{1-N\le p,q\le N} |\lambda^m_{p,q}(B_\gamma)\lambda_{p,q}(C_h)| \\ &= \max_{1-N\le p,q\le N} |\lambda^m_{p,q}(B_\gamma)(1-\lambda_{p,q}(P_h)\frac{\lambda_{p,q}(A_h)}{\lambda_{p,q}(A_{2h})})|. \end{aligned} \tag{8.4.83}$$

By numerical calculation, the convergence factors $\rho(T_h)$ of the 2D PMG method are the same as those listed in Tables 7.1 and 7.2. This is because the configuration of our two-level PMG algorithm is equivalent to the corresponding two-level MCGMG algorithm.

8.4.3 Multilevel Convergence Analysis

We now consider the multilevel PMG algorithm in 2D. We recall that the two-level PMG operator on level L is given by

$$\begin{aligned} T_L &= S_L C_L \\ &= S_L(I - R_L A^\dagger_{L-1} A_L), \end{aligned} \tag{8.4.84}$$

where S_L is a smoothing operator. In the multilevel PMG algorithm, the system on level $L-1$ is solved by applying the multilevel PMG algorithm on level $L-1$. Let M_L and M_{L-1} be operators corresponding to a multilevel PMG cycle on levels L and $L-1$ respectively. On level $L-1$, if the system

$$A_{L-1}u_{L-1} = b_{L-1} \tag{8.4.85}$$

is solved by applying one multilevel PMG cycle with the initial guess $u_{L-1}^{(0)} = 0$, then we have

$$\begin{aligned} u_{L-1}^{(1)} - \bar{u}_{L-1} &= e_{L-1}^{(1)} \\ &= M_{L-1}e_{L-1}^{(1)} \\ &= M_{L-1}(u_{L-1}^{(0)} - \bar{u}_{L-1}) \\ &= -M_{L-1}\bar{u}_{L-1} \\ &= -M_{L-1}A_{L-1}^{\dagger}b_{L-1}. \end{aligned} \tag{8.4.86}$$

In other words, applying one multilevel PMG cycle is equivalent to using $(I - M_{L-1}A_{L-1}^{\dagger})$ to approximate $A_{L-1}^{\dagger}$. Here we assume (8.4.85) is consistent. Therefore, replacing $A_{L-1}^{\dagger}$ by $(I - M_{L-1}A_{L-1}^{\dagger})$ in (8.4.84) we have

$$M_L = S_L(I - R_L(I - M_{L-1})A_{L-1}^{\dagger}A_L). \tag{8.4.87}$$

If we assume that all the operators in (8.4.87) are commutative, then we have the following recursive relations:

$$\begin{aligned} M_L &= S_L(I - R_L(I - M_{L-1})A_{L-1}^{\dagger}A_L) \\ &= S_L(I - R_LA_{L-1}^{\dagger}A_L(I - M_{L-1})) \\ &= S_L(I - R_LA_{L-1}^{\dagger}A_L) + S_LR_LA_{L-1}^{\dagger}A_LM_{L-1} \\ &= T_L + (S_L - T_L)M_{L-1} \end{aligned} \tag{8.4.88}$$

In general, we have

$$M_l = T_l + (S_l - T_l)M_{l-1}. \tag{8.4.89}$$

In the case of $h = 2^{-L}$, there are at most $L+1$ levels. Suppose we use $L+1$ levels in the PMG algorithm. We denote by $A_lu_l = b_l$ a linear system on level l with grid spacing or scale $h_l = 2^{-l}$, $l = 0, \ldots, L$.

For the class of model problems considered, the use of the Moore-Penrose solution is equivalent to defining

$$\lambda_{0,0}(M_l) = 1 \tag{8.4.90}$$

(see Frederickson and McBryan [31]). Here $\lambda_{0,0}(M_l)$ is the scalar function of the operator M_l associated with the Fourier mode $v_h^{(0,0)}$ defined in (8.2.9). Thus we have

Lemma 8.3 *Let*

$$\rho(M_l) = \max_{1-N\leq p,q\leq N} |\lambda_{p,q}(M_l)|. \tag{8.4.91}$$

For any $l \in \{1, \ldots, L\}$, the multilevel PMG operator M_l satisfies

$$\rho(M_{l-1}) \leq \rho(M_l) \tag{8.4.92}$$

Proof: First, we show that

$$\lambda_{2p,2q}(M_l) = \lambda_{p,q}(M_{l-1}). \tag{8.4.93}$$

We note that

$$\begin{aligned} \cos(2p\pi h) &= \cos(p\pi 2h), && (8.4.94)\\ \cos(2q\pi h) &= \cos(q\pi 2h). && (8.4.95) \end{aligned}$$

For $l = 2$ we have, from (8.4.89),

$$\lambda_{2p,2q}(M_1) = \lambda_{2p,2q}(T_1) + [\lambda_{2p,2q}(S_1) - \lambda_{2p,2q}(T_1)]\lambda_{2p,2q}(M_0) \tag{8.4.96}$$

Since

$$\lambda_{2p,2q}(M_0) = \lambda_{0,0}(M_0) \equiv 1 \tag{8.4.97}$$

the relation (8.4.96) reduces to

$$\lambda_{2p,2q}(M_1) = \lambda_{2p,2q}(S_1) = \lambda_{p,q}(S_0) = \lambda_{p,q}(M_0) \tag{8.4.98}$$

Now assume that relation (8.4.93) holds for M_{l-1}. Then we have

$$\begin{aligned}\lambda_{2p,2q}(M_l) &= \lambda_{2p,2q}(T_l) + [\lambda_{2p,2q}(S_l) - \lambda_{2p,2q}(T_l)]\lambda_{2p,2q}(M_{l-1}) \\ &= \lambda_{p,q}(T_{l-1}) + [\lambda_{p,q}(S_{l-1}) - \lambda_{p,q}(T_{l-1})]\lambda_{p,q}(M_{l-2}) \\ &= \lambda_{p,q}(M_{l-1}). \qquad (8.4.99)\end{aligned}$$

Thus, we have

$$|\lambda_{p,q}(M_{l-1})| = |\lambda_{2p,2q}(M_l)| \leq \rho(M_l). \qquad (8.4.100)$$

for $p, q = 1 - N, \ldots, N$ and therefore, relation (8.4.92) is proved.

We now give an upper bound of the convergence factor of the multilevel PMG algorithm.

Theorem 8.1 *Let*

$$\sigma_1 = \max_{1-N\leq p,q\leq N} \frac{|\lambda_{p,q}(T_L)|}{1 - |\lambda_{p,q}(S_L) - \lambda_{p,q}(T_L)|}. \qquad (8.4.101)$$

If

$$\rho(M_l) = \max_{1-N\leq p,q\leq N} |\lambda_{p,q}(M_l)| \leq \sigma_1 \qquad (8.4.102)$$

holds on level l, then we have

$$\rho(M_L) \leq \sigma_1. \qquad (8.4.103)$$

Proof: The proof is based on the fact that (8.4.101) is valid for all the levels. In fact, since

$$\begin{aligned}\lambda_{p,q}(T_{l-1}) &= \lambda_{2p,2q}(T_l), \qquad (8.4.104)\\ \lambda_{p,q}(S_{l-1}) &= \lambda_{2p,2q}(S_l), \qquad (8.4.105)\end{aligned}$$

we have

$$\begin{aligned}\sigma_1 &= \max_{1-N\le p,q\le N} \frac{|\lambda_{p,q}(T_L)|}{1-|\lambda_{p,q}(S_L)-\lambda_{p,q}(T_L)|}\\ &\ge \max_{1-N\le p,q\le N} \frac{|\lambda_{p,q}(T_l)|}{1-|\lambda_{p,q}(S_l)-\lambda_{p,q}(T_l)|}. \qquad (8.4.106)\end{aligned}$$

Now assume that $\rho(M_{l-1}) \le \sigma_1$ holds. For level l we have

$$\begin{aligned}|\lambda_{p,q}(M_l)| &= |\lambda_{p,q}(T_l) + (\lambda_{p,q}(S_l - \lambda_{p,q}(T_l))\lambda_{p,q}(M_{l-1})|\\ &\le |\lambda_{p,q}(T_l)| + |\lambda_{p,q}(S_l) - \lambda^{p,q}(T_l)|\rho(M_{l-1})\\ &\le (1-|\lambda_{p,q}(S_l)-\lambda_{p,q}(T_l)|)\sigma_1 + |\lambda_{p,q}(S_l)-\lambda_{p,q}(T_l)|\sigma_1\\ &= \sigma_1. \qquad (8.4.107)\end{aligned}$$

A sharper upper bound on the convergence factor of the multilevel PMG algorithms can be obtained by using information on more than two levels. Let

$$\lambda_{p,q}(F_l) = \lambda_{p,q}(S_l) - \lambda_{p,q}(T_l). \qquad (8.4.108)$$

We define

$$\lambda_{p,q}^{(\nu+1)}(F_l) = \lambda_{p,q}(F_l)\lambda_{p,q}(F_{l-1})\ldots\lambda_{p,q}(F_{l-\nu}) = \prod_{j=l-\nu}^{l} \lambda_{p,q}(F_j) \qquad (8.4.109)$$

and

$$\begin{aligned}\lambda_{p,q}^{(\nu+1)}(T_l) &= \lambda_{p,q}(T_l) + \lambda_{p,q}(F_l)\lambda_{p,q}(T_{l-1}) + \lambda_{p,q}(F_l)\lambda_{p,q}(F_{l-1})\lambda_{p,q}(T_{l-2})\\ &\quad + \ldots + \lambda_{p,q}(F_l)\lambda_{p,q}(F_{l-1})\ldots\lambda_{p,q}(F_{l-\nu+1})\lambda_{p,q}(T_{l-\nu})\\ &= \lambda_{p,q}(T_l) + \sum_{j=1}^{\nu} \lambda_{p,q}^{(j)}(F_l)\lambda_{p,q}(T_{l-j}) \qquad (8.4.110)\end{aligned}$$

We now have the following theorem.

Theorem 8.2 *Let*

$$\sigma_\nu = \max_{1-N\le p,q\le N} \frac{|\lambda_{p,q}^{(\nu)}(T_L)|}{1-|\lambda_{p,q}^{(\nu)}(F_L)|} \qquad (8.4.111)$$

where $\nu \geq 1$. If

$$\rho(M_l) \leq \sigma_\nu \tag{8.4.112}$$

holds on one of the levels, then we have

$$\rho(M_L) \leq \sigma_\nu \tag{8.4.113}$$

Proof: Suppose that $\rho(M_{l-k}) \leq \sigma_\nu$. For level l we have

$$\begin{aligned} |\lambda_{p,q}(M_l)| &= |\lambda^{(\nu)}_{p,q}(T_l) + \lambda^{(\nu)}_{p,q}(F_l)\lambda^{(\nu)}_{p,q}(M_{l-\nu})| \\ &\leq |\lambda^{(\nu)}_{p,q}(T_l)| + |\lambda^{(\nu)}_{p,q}(F_l)|\rho(\lambda^{(\nu)}_{p,q}(M_{l-k}) \\ &\leq (1 - |\lambda^{(\nu)}_{p,q}(F_l)|)\sigma_\nu + |\lambda^{(\nu)}_{p,q}(F_l)|\sigma_\nu \\ &= \sigma_\nu \end{aligned} \tag{8.4.114}$$

Here we use the property that the inequality (8.4.111) is valid for any $l = \nu, \ldots, L$. From Lemma 8.3 and repeatedly applying the result of the proof to level $l + \nu$ and so on, we obtain that $\rho(M_L) \leq \sigma_\nu$.

Theorem 8.1 is a special case of Theorem 8.2 with $\nu = 1$. When $\nu > 1$, the upper convergence bound σ_ν will be sharper than σ_1. The assumption $\rho(\bar{M}_{l-k}) \leq \sigma_\nu$ is not a major restriction, since an PMG algorithm usually works more effectively on a coarse level than on a fine level.

Table 8.12 lists the upper bound of the multilevel convergence factors σ_ν defined in (8.4.111) for the model problem (2.3.8) with $\alpha = 1$. The values are obtained by using grid size $N = 64$, and performing one Jacobi post smoothing iteration and no pre-smoothing. The result shows that the multilevel convergence bounds σ_ν are very close to the two-level convergence bound ρ. In other words, the two-level convergence bound is a good estimate for the multilevel convergence bound.

Table 8.12: Multilevel Convergence Factors for PMG-Jacobi vs. γ ($\alpha = 1$)

γ	σ_1	σ_2	σ_3	σ_4	σ_5	ρ
.50	.4992	.4980	.4977	.4924	.4798	.4762
.60	.4536	.4524	.4519	.4455	.4193	.3771
.70	.4156	.4145	.4137	.4064	.3990	.3989
.80	.5989	.5989	.5989	.5989	.5989	.5987
.90	.7989	.7989	.7988	.7988	.7988	.7985
1.00	.9988	.9988	.9988	.9988	.9986	.9982

Chapter 9

Semicoarsening and Line Smoothing

9.1 Introduction

The convergence rate of the standard multigrid methods usually deteriorates for the problem

$$\begin{cases} -\alpha\dfrac{\partial^2 u(x,y)}{\partial x^2} - \dfrac{\partial^2 u(x,y)}{\partial y^2} = f(x,y) & (x,y) \in \Omega = (0,1)^2, \\ u = \phi(x,y) & (x,y) \in \partial\Omega \end{cases} \tag{9.1.1}$$

where the coefficient $\alpha > 0$. If $\alpha \gg 1$ or $\alpha \ll 1$, we will refer to such problems as *anisotropic problems.*

Two techniques, namely block-wise smoothing and semicoarsening, are commonly used to restore the efficiency of the standard multigrid methods. In this chapter we consider applying these two techniques to the parallel multigrid methods (PMG) and give multilevel convergence analysis of the resulting algorithm.

9.2 Convergence of SMG in an Anisotropic Case

Let

$$A_h u = b \tag{9.2.2}$$

be the linear equation system defined in (2.3.10) which corresponds to the 5-point difference representation of (9.1.1) on a two-dimensional grid Ω_h with

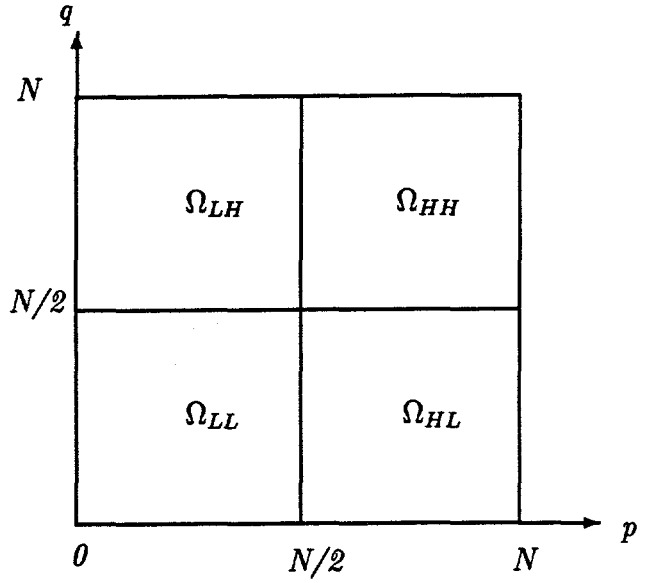


Figure 9.1: Four Subsets of the Fourier Modes

grid spacing $h = 1/N$. An initial error vector can be represented by a linear combination of $(N-1)^2$ Fourier modes $v^{(p,q)}$, where p and q represent the number of waves in the x- and y-directions respectively. We can divide the set of the Fourier modes into four subsets Ω_{LH}, Ω_{HH}, Ω_{LL} and Ω_{HL} as illustrated in Figure 9.1. Here the subscripts L and H indicate that the modes in the subset belong to low and high frequency respectively. For instance, a mode in Ω_{LH} has low frequency in the x-direction and high frequency in the y-direction.

In the standard multigrid method, the standard coarsening scheme (i.e. the coarsening process being carried out in two directions) is used. The Fourier modes in subset Ω_{LL} can be reduced by the coarse grid correction efficiently and the Fourier modes in the other subsets are supposed to be damped by the smoothing iteration operator. In an isotropic case ($\alpha = 1$), a point-wise smoothing scheme (e.g. the damped Jacobi method) works well. However, in an anisotropic case, most point-wise smoothing operators are generally unable to efficiently damp the Fourier modes in Ω_{HL} when $\alpha << 1$ or in Ω_{LH}

when $\alpha >> 1$. This difficulty also exists when a PMG method is used for an anisotropic problem.

9.3 Line Jacobi Method

For the grid point (x_j, y_k), the 5-point discrete anisotropic operator A_h defined in (2.3.10) can be written in the form

$$(A_h u)_{j,k} = \alpha \frac{2u_{j,k} - u_{j-1,k} - u_{j+1,k}}{h^2} + \frac{2u_{j,k} - u_{j,k-1} - u_{j,k+1}}{h^2}. \tag{9.3.3}$$

The damped x-line Jacobi method for the problem $A_h u = b_h$ is defined by

$$\begin{cases} \frac{1}{h^2}\{\alpha(2v_{j,k} - v_{j-1,k} - v_{j+1,k}) + (2v_{j,k} - u^n_{j,k-1} - u^n_{j,k+1})\} = b_{j,k}, \\ u^{n+1} = \gamma v + (1-\gamma)u^n. \end{cases} \tag{9.3.4}$$

From (3.2.3), the matrix for the damped x-line Jacobi method is given by

$$B^{(x)}_\gamma = I - \gamma Q^{(x)} A_h \tag{9.3.5}$$

where the splitting matrix $Q^{(x)}$ is given by

$$(Q^{(x)}u)_{j,k} = \frac{1}{h_y^2}\left[\tilde{\alpha}(2u_{j,k} - u_{j-1,k} - u_{j+1,k}) + 2u_{j,k}\right]. \tag{9.3.6}$$

The eigenvalues of the matrix $B^{(x)}_\gamma$ are given by

$$\nu(B^{(x)}_\gamma)^{(p,q)} = 1 - \gamma \frac{\tilde{\alpha}(1-c_p) + 1 - c_q}{\alpha(1-c_p) + 1} \tag{9.3.7}$$

where $c_p = \cos p\pi h_x$ and $c_q = \cos q\pi h_y$. Choosing an appropriate value of γ, e.g. $\gamma = 2/3$, the absolute values of the eigenvalues $\nu(B^{(x)}_\gamma)^{(p,q)}$ are less than $1/3$ in the subsets Ω_{LH} and Ω_{HH} for any value of α.

Similarly, it can be shown that the absolute values of the eigenvalues of the y-line Jacobi method are small in the subsets Ω_{HL} and Ω_{HH}. In the case of $\alpha << 1$, where a multigrid algorithm loses efficiency in Ω_{HL}, one uses the x-line Jacobi smoothing iteration; while in the case of $\alpha >> 1$, where a multigrid algorithm loses efficiency in Ω_{LH}, one uses the y-line smoothing iteration. If one uses both x-line and y-line Jacobi smoothing together in the standard multigrid method, one can damp all the Fourier modes efficiently.

9.4 Semicoarsening Scheme

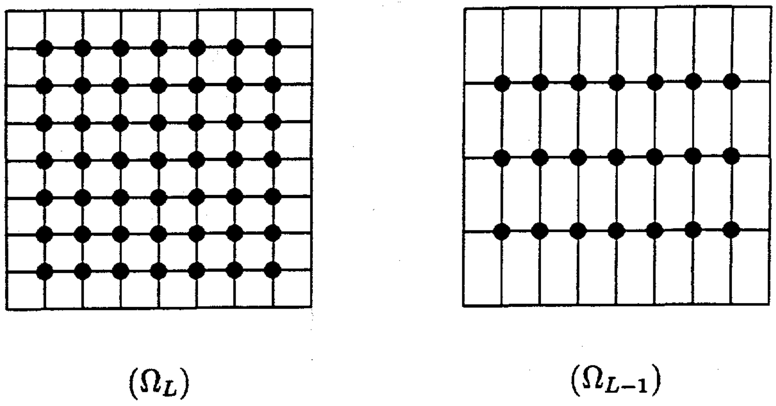
In the standard coarsening scheme, the coarse grids are constructed by coarsening the fine grid in all axis directions (e.g. the x- and y-directions in 2D). In the semicoarsening scheme, however, the coarse grids are constructed by coarsening the fine grid only in one direction.

We assume that the coarsening process is carried out in the y-direction. We use the subsript l to represent the level of the grids ($l = 1$ denotes the coarsest level and $l = L$ denotes the finest level). On grid Ω_l the grid spacing the y-direction (the coarsening direction) is $h_l = 2^{l-L}h$ and the grid spacing in the x-direction is h. The grid Ω_l is given by

$$\Omega_l = \{(x_j, y_k) \mid j = 1, \ldots, N-1;\ k = 1, \ldots, 2^{l-L}N - 1\} \qquad (9.4.8)$$

where $x_j = jh$ and $y_k = kh_l$. For the model problem (2.3.8) with $h = h_L = 1/8$, the coarse grid Ω_{L-1} is shown in Figure 9.2.

Since no coarsening is carried out in the x-direction, the modes with high-frequency waves in the x-direction can be represented on the coarse grids. The coarse grid correction procedure can damp the modes in Ω_{HL} as well as Ω_{LL} (see Figure 9.1). Therefore, the smoothing iteration procedure only needs to damp the modes in Ω_{HH} and Ω_{LH}. In a general case, the semicoarsening scheme can be used with the line smoothing iteration. For example, we can use the y-direction coarsening with the x-line smoothing iteration.



(Ω_L) (Ω_{L-1})

Figure 9.2: 2D Semicoarsening Coarse Grids: $h = 1/8$

9.5 Semicoarsening SMG

For the model problem (2.3.8), the difference equations on the grid Ω_l are given by

$$\begin{aligned}(A_l u_l)(x,y) &= h_l^{-2}((2+2\alpha_l)u_l(x,y) - \alpha_l u_l(x-h,y) \\ &\quad -\alpha_l u_l(x+h,y) - u_l(x,y-h_l) - u_l(x,y+h_l)) \\ &= b_l(x,y), \\ &\quad (x,y) \in \Omega_l\end{aligned} \tag{9.5.9}$$

where

$$\alpha_l = \frac{h_l^2}{h^2}\alpha = 4^{L-l}\alpha. \tag{9.5.10}$$

The value α_l is the anisotropic coefficient on level l which is four times as large as α_{l+1}. Therefore, a strong coupling in the y-direction on the fine level weakens on coarse levels.

The standard multigrid method using the semicoarsening scheme can be described as follows.

Algorithm SCSMG(A_l, b_l, u_l):

1. Carry out m_1 pre-smoothing iterations using the damped x-line Jacobi iterative method to obtain u_l'.

2. Compute the residual $r_l = b_l - A_l u_l'$.

3. Restrict the residual to the coarse grid to obtain

$$\begin{aligned} r_{l-1}(x,y) &= \frac{1}{4}(r_l(x,y+h_l) + 2r_l(x,y) + r_l(x,y-h_l)) \\ & \quad (x,y) \in \Omega_{l-1} \end{aligned} \tag{9.5.11}$$

4. Solve the correction equation on the coarse grid

$$A_{l-1}\delta_{l-1} = r_{l-1} \tag{9.5.12}$$

for δ_{l-1}. In the case of more than two levels

$$\delta_{l-1} = \text{SCSMG}(A_{l-1}, r_{l-1}, 0), \tag{9.5.13}$$

and at the coarsest grid, solve the problem directly.

5. Interpolate the coarse grid correction δ_{l-1} onto the fine grid

$$\delta_l(x,y) = \begin{cases} \delta_{l-1}(x,y) & (x,y) \in \Omega_{l-1} \\ \frac{1}{2}(\delta_{l-1}(x,y-h_l) + \delta_{l-1}(x,y+h_l)) & (x,y) \in \Omega_l/\Omega_{l-1} \end{cases} \tag{9.5.14}$$

and

$$u_l'' = u_l' + \delta_l. \tag{9.5.15}$$

6. Carry out m_2 post-smoothing iterations using the damped x-line Jacobi iterative method and return the new solution.

The 2D standard multigrid method using the semicoarsening scheme is analogous to a 1D standard multigrid method in the sense that both restriction of residuals and prolongation of corrections are performed in one direction.

Now we give a brief two-level convergence analysis for this algorithm. Since the coarsening process is carried out only in the y-direction, we consider eigenvectors $v_h^{(p,q)}$ and $v_h^{(p,N-q)}$ defined in (2.3.14) as a pair of aliasing vectors. If we let

$$E_v^{(p,q)} = (v_h^{(p,q)}, v_h^{(p,N-q)}) \tag{9.5.16}$$

and apply the operators on it, for $p = 1, \ldots, N-1$ and $q = 1, \ldots, N/2$, we have

$$A_h E_v^{(p,q)} = E_v^{(p,q)} \hat{A}_{h,v}^{(p,q)}, \tag{9.5.17}$$

$$R_h E_v^{(p,q)} = v_{2h}^{(p,q)} \hat{R}_{h,v}^{(p,q)}, \tag{9.5.18}$$

$$A_{2h} v_{2h}^{(p,q)} = v_{2h}^{(p,q)} \hat{A}_{2h,v}^{(p,q)}, \tag{9.5.19}$$

$$P_h v_{2h}^{(p,q)} = E_v^{(p,q)} \hat{P}_{h,v}^{(p,q)}, \tag{9.5.20}$$

$$B_\gamma E_v^{(p,q)} = E_v^{(p,q)} \hat{B}_{\gamma,v}^{(pq)} \tag{9.5.21}$$

where

$$\hat{A}_{h,v}^{(p,q)} = \frac{2}{h^2} \begin{bmatrix} \alpha(1-c_p)+1-c_q & 0 \\ 0 & \alpha(1-c_p)+1+c_q \end{bmatrix}, \tag{9.5.22}$$

$$\hat{R}_{h,v}^{(p,q)} = \frac{1}{2} \begin{bmatrix} 1+c_q & c_q-1 \end{bmatrix}, \tag{9.5.23}$$

$$\hat{A}_{2h,v}^{(p,q)} = \frac{1}{h^2}(2\alpha(1-c_p)+(1-c_q^2), \tag{9.5.24}$$

Table 9.1: Two-Level Convergence Factors for SCSMG

	γ			
α	0.5	0.6	0.7	0.8
0.00001	0.00001	0.0017	0.0035	0.0050
0.0001	0.0001	0.0145	0.0291	0.0437
0.001	0.0010	0.0623	0.1253	0.1883
0.01	0.0096	0.0883	0.1817	0.2751
0.1	0.0714	0.0883	0.1821	0.2766
1	0.1999	0.1600	0.1820	0.2767
10	0.2439	0.1951	0.1791	0.2744
100	0.2494	0.1995	0.1611	0.2547
1000	0.2499	0.1999	0.1500	0.1000
10000	0.2500	0.2000	0.1500	0.1000
100000	0.2500	0.2000	0.1500	0.1000

$$\hat{P}^{(p,q)}_{h,v} = \frac{1}{2}\begin{bmatrix} 1+c_p \\ c_p-1 \end{bmatrix}, \tag{9.5.25}$$

$$\hat{B}^{(p,q)}_{\gamma,v} = I - \frac{\gamma h^2}{2(1+\alpha)}\hat{A}^{(pq)}_{h,v}. \tag{9.5.26}$$

Here $c_p = \cos(p\pi h)$ and $c_q = \cos(q\pi h)$. The v-transform matrix of the coarse grid correction operator can be computed by

$$\hat{C}^{(p,q)}_{h,v} = I - \hat{P}^{(p,q)}_{h,v}(\hat{A}^{(p,q)}_{2h,v})^{-1}\hat{R}^{(p,q)}_{h,v}\hat{A}^{(p,q)}_{h,v}. \tag{9.5.27}$$

The two-level convergence factor is given by

$$\rho(T_h) = \max_{\substack{1 \leq p \leq N-1 \\ 1 \leq q \leq \frac{N}{2}}} \rho(\hat{C}_{h,v}^{(p,q)} \hat{B}_{\gamma,v}^{(p,q)}). \tag{9.5.28}$$

The numerical values of the two-level convergence factors of the SCSMG method for different values of the coefficient α and the extrapolation factor γ are listed in Table 9.1. It shows that the SMG method using the semicoarsening scheme together with line smoothing iterations works well in cases when α is either very small or very large.

9.6 Semicoarsening PMG Algorithm

We now consider applying semicoarsening and line smoothing techniques to the PMG methods. Since the coarsening process is only carried out in one direction, the system only needs to be extended in that direction. For the model problem (2.3.8) with $h = 1/8$, the extended grid and the original grid are shown in Figure 9.3.

The procedure of the semicoarsening PMG method (SCPMG) is described in Figure 9.4. Comparing to Figure 7.8 one sees that the restriction-like smoothing and the interpolation-like smoothing of the semicoarsening PMG methods is analogous to those used in 1D PMG methods. In the following analysis we assume that the full weighting operator is used for restriction-like smoothing and the trivial injection is used for interpolation-like smoothing.

The difference operator on level l is given by

$$\begin{aligned} (A_l u_l)(x,y) &= h_l^{-2}((2+2\alpha_l)u_l(x,y) - \alpha_l u_l(x-h,y) \\ &\quad -\alpha_l u_l(x+h,y) - u_l(x,y-h_l) - u_l(x,y+h_l)) \\ &= b_l(x,y), \\ &\quad (x,y) \in \Omega_h \end{aligned} \tag{9.6.29}$$

where α_l is defined in (9.5.10).

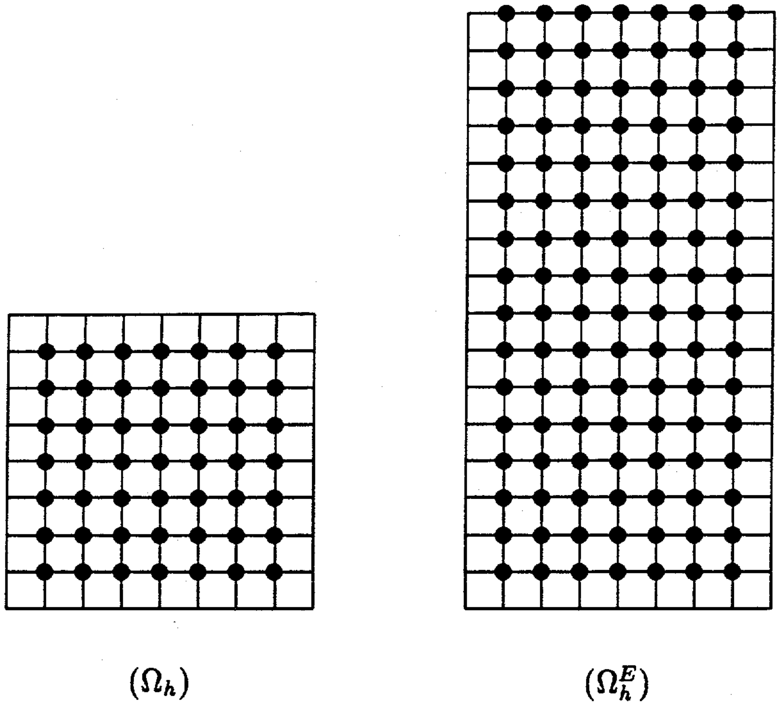


Figure 9.3: 2D Semicoarsening Extended Grids: $h = 1/8$

Algorithm SCPMG(A_h, $u_h^{(0)}$, b_h):

1. Carry out m_1 pre-smoothing interations to get u_h'.
2. Compute the residual: $r_h = b_h - A_h u_h'$.
3. Carry out one restriction-like smoothing operation, possibly full weighting:

   $r_{2h}(x,y) = (R_h r_h)(x,y)$

   $\quad = \frac{1}{4}(r_h(x,y-h) + 2r_h(x,y) + r_h(x,y+h))$,

   or injection $r_{2h}(x,y) = (R_h r_h)(x,y) = r_h(x,y)$
4. Solve the correction equation for the $2h$ scale

   $$A_{2h}\delta_{2h} = r_{2h}$$

   In the case of more than two levels, this can be done by using the procedures:

   $\delta_{2h} = \mathrm{PMG}(A_{2h}, 0, r_{2h})$.

   At the coarsest level, the problem is solved directly.
5. Carry out one interpolation-like smoothing operation, possibly linear:

   $(\delta_h)(x,y) = (P_h\delta_{2h})(x,y)$

   $\quad = \frac{1}{4}[\delta_{2h}(x,y-h) + 2\delta_{2h}(x,y) + \delta_{2h}(x,y+h)]$,

   or injection: $(\delta_h)(x,y) = (P_h\delta_{2h})(x,y) = (\delta_{2h})(x,y)$

   and update the solution $u_h'' = u_h' + \delta_h$
6. Carry out m_2 post-smoothing iterations to get the new solution $u_h^{(1)}$.

Figure 9.4: The SCPMG Algorithm

9.6.1 Two-Level Convergence Analysis

If the damped x-line Jacobi iterative method is used for smoothing iteration, the matrices of all the operators used in the SCPMG algorithm have the same eigenvector set defined by

$$\begin{cases} (v_h^{(p,q)})_{j,k} = \sin(p\pi jh)\exp(iq\pi kh), \\ \quad j = 1, \ldots, N-1 \\ \quad k = 1-N, \ldots, N \end{cases} \tag{9.6.30}$$

for $p = 1, \ldots, N-1$ and $q = 1-N, \ldots, N$. In this case the standard Fourier analysis on the two-level SCPMG algorithm is very simple because the v-transform matrix of any of the operators degrades to a single eigenvalue.

For the eigenvector $v_h^{(p,q)}$, the corresponding eigenvalue of the difference operator A_l is given by

$$\hat{A}_l^{(p,q)} = \frac{2}{h_l^2}\left[\alpha_l(1-c_p) + 1 - c_{q_l}\right] \tag{9.6.31}$$

where $c_p = \cos p\pi h$ and $c_{q_l} = \cos q\pi h_l = \cos 2^{L-l}q\pi h$. From (9.6.31), one sees that $\hat{A}_l^{(p,q)}$ is always positive and therefore, the systems on each of the levels is nonsingular. At level $l-1$, we have

$$\begin{aligned} \hat{A}_{l-1}^{(p,q)} &= \frac{2}{h_{l-1}^2}\left[\alpha_{l-1}(1-c_p) + 1 - c_{q_{l-1}}\right] \\ &= \frac{1}{h_l^2}\left[2\alpha_l(1-c_p) + 1 - c_{q_l}^2\right] \end{aligned} \tag{9.6.32}$$

where we use the relation $c_{q_{l-1}} = 2c_{q_l}^2 - 1$. The corresponding eigenvalue of the restriction-like operator R_l is given by

$$\hat{R}_l^{(p,q)} = \frac{1+c_{q_l}}{2}. \tag{9.6.33}$$

The corresponding eigenvalue of the damped x-line Jacobi iteration matrix is given by

$$\hat{B}_l^{(p,q)} = 1 - \gamma + \gamma\frac{c_{q_l}}{1+\alpha_l(1-c_p)}. \tag{9.6.34}$$

Table 9.2: Two-Level Convergence Factors for SCPMG-LJ

α	γ			
	0.5	0.6	0.7	0.8
0.00001	0.00001	0.0034	0.0067	0.0050
0.0001	0.0001	0.0291	0.0583	0.0101
0.001	0.0010	0.1250	0.2510	0.0875
0.01	0.0098	0.1771	0.3640	0.3770
0.1	0.0832	0.1771	0.3649	0.5509
1	0.3331	0.1999	0.3646	0.5539
10	0.4759	0.3712	0.3588	0.5494
100	0.4972	0.3968	0.3227	0.5101
1000	0.4995	0.3995	0.2995	0.1995
10000	0.4887	0.3997	0.2998	0.1998
100000	0.4997	0.3997	0.2998	0.1999

From (9.6.31), (9.6.32) and (9.6.33) and noticing that the interpolation-like smoothing is the trivial injection, the corresponding eigenvalue of the coarse grid correction matrix $C_{y,l}$ is given by

$$\begin{aligned}
\hat{C}_{y,l}^{(p,q)} &= \hat{I} - \hat{R}_l^{(p,q)}(\hat{A}_{l-1}^{(p,q)})^{-1}\hat{A}_l^{(p,q)} \\
&= 1 - (1 + c_{q_l})\frac{\alpha_l(1-c_p) + 1 - c_{q_l}}{2\alpha_l(1-c_p) + 1 - c_{q_l}^2} \\
&= \frac{\alpha_l(1-c_p)(1-c_{q_l})}{2\alpha_l(1-c_p) + 1 - c_{q_l}^2}. \qquad (9.6.35)
\end{aligned}$$

Therefore, from (9.6.34) and (9.6.35), the eigenvalue of the matrix of the two-level SCPMG operator with m damped line Jacobi smoothing iteration is given by

$$\begin{aligned}
\hat{T}_l^{(p,q)}(m) &= (\hat{B}_l^{(pq)})^m \hat{C}_{y,l}^{(pq)} \\
&= \left[1 - \gamma + \gamma\frac{c_{q_l}}{1 + \alpha_l(1-c_p)}\right]^m \frac{\alpha_l(1-c_p)(1-c_{q_l})}{2\alpha_l(1-c_p) + 1 - c_{q_l}^2}. \qquad (9.6.36)
\end{aligned}$$

The convergence factor of the two-level SCPMG algorithm can then be calculated by

$$\rho(T_L)(m) = \max_{1 \le p,q \le N-1} \rho(\hat{T}_L^{(p,q)}(m)). \qquad (9.6.37)$$

Table 9.2 lists the convergence factors of the two-level SCPMG with damped line Jacobi smoothing iteration (SCPMG-LJ) with different values of the extrapolation factor γ for problem (2.3.8) with different values of α. The result is obtained numerically with the grid size $N = 64$.

9.6.2 Modified Line Jacobi Method

Using the line Jacobi smoothing iteration the convergence speed of the standard multigrid method cannot be improved very much by using the PMG method. However, a modified Jacobi method can improve the performance of the PMG in an anisotropic case [64] [66]. Now we show that the modified Jacobi method can also improve the performance of the SCPMG.

Table 9.3: Two-Level Convergence Factors for SCPMG-MLJ

	γ			
α	0.8	0.9	1.0	1.1
0.00001	0.0037	0.0017	0.00001	0.0017
0.0001	0.0293	0.0147	0.0001	0.0145
0.001	0.1265	0.0634	0.0010	0.0627
0.01	0.1893	0.0949	0.0095	0.0938
0.1	0.1991	0.0998	0.0649	0.0988
1	0.2000	0.1401	0.1110	0.0996
10	0.2000	0.1400	0.1109	0.0999
100	0.2000	0.1391	0.1110	0.0999
1000	0.2000	0.1398	0.1104	0.0999
10000	0.1999	0.1004	0.0350	0.0999
100000	0.1999	0.0999	0.0042	0.0999

The modified damped x-line Jacobi method for the problem $A_l u = b_l$ is defined by

$$\begin{aligned}&\tfrac{1}{h_l^2}\{\alpha_l(2v_{j,k} - v_{j-1,k} - v_{j+1,k}) + (4v_{j,k} - 2u^n_{j,k} - u^n_{j,k-1} - u^n_{j,k+1})\} = b_{j,k},\\ &u^{n+1} = \gamma v + (1-\gamma)u^n.\end{aligned} \tag{9.6.38}$$

The eigenvalues of the corresponding iterative operator are given by

$$\hat{B}_l^{(p,q)} = 1 - \gamma + \gamma\frac{1 - c_{q_l}}{2 + \alpha_l(1 - c_p)}. \tag{9.6.39}$$

Table 9.3 gives the convergence factors of the two-level SCPMG with the modified line Jacobi smoothing iteration (SCPMG-MLJ). The case is the same as that used in creating Table 9.2. We note that a good choice of the extrapolation factor γ is around 1.1 in this case.

9.6.3 Red/Black Line SOR Method

We now consider using the red/black line SOR method for smoothing iteration. One red/black line SOR iteration can be considered as two sub-iterations, the black sub-iteration and the red sub-iteration which are defined respectively by

$$\begin{cases} u_{jk}^{n+\frac{1}{2}} = u_{jk}^n, & \text{if } k \text{ is odd,}\\ (2+2\alpha)u_{jk}^{n+\frac{1}{2}} - \alpha(u_{j-1k}^{n+\frac{1}{2}} + u_{j+1k}^{n+\frac{1}{2}}) & \\ \quad = \omega(u^n_{jk+1} + u^n_{jk-1} + h^2 b_{jk}) & \\ \quad +(1-\omega)(2+2\alpha)u^n_{jk} - \alpha(u^n_{j-1k} + u^n_{j+1k}), & \text{if } k \text{ is even} \end{cases} \tag{9.6.40}$$

and

$$\begin{cases} (2+2\alpha)u_{jk}^{n+1} - \alpha(u_{j-1k}^{n+1} + u_{j+1k}^{n+1}) & \\ \quad = \omega(u_{jk+1}^{n+\frac{1}{2}} + u_{jk-1}^{n+\frac{1}{2}} + h^2 b_{jk}) & \\ \quad +(1-\omega)(2+2\alpha)u_{jk}^{n+\frac{1}{2}} - \alpha(u_{j-1k}^{n+\frac{1}{2}} + u_{j+1k}^{n+\frac{1}{2}}), & \text{if } k \text{ is odd,}\\ u_{jk}^{n+1} = u_{jk}^{n+\frac{1}{2}}, & \text{if } k \text{ is even.} \end{cases} \tag{9.6.41}$$

We will use $S_l^{(-)}$ to represent the black sub-iteration operator and $S_l^{(+)}$ to represent the red sub-iteration operator.

It will be easier to perform the two-color Fourier convergence analysis for the SCPMG with a red/black ordering smoothing iteration. In the semicoarsening scheme, the fine grid is divided into two coarse grids and the corresponding w-basis vectors are given by

$$
\begin{aligned}
(w_h^{(s,p,q)})_{j,k} &= \begin{cases} (v_h^{(p,q)})_{j,k} & \text{if}\,(x_j, y_k) \in \Omega_s \\ 0 & \text{otherwise} \end{cases} \\
&\quad j,k,p = 1,\ldots,N-1 \\
&\quad q = 1,\ldots,\frac{N}{2} \\
&\quad s = +,\, -
\end{aligned}
\tag{9.6.42}
$$

where $v_h^{(p,q)}$ are defined in (8.2.9), and Ω_+ and Ω_- are defined by

$$
\begin{cases}
\Omega_+ = \{(x_j, y_k) \mid (k = \text{odd})\}, \\
\Omega_- = \{(x_j, y_k) \mid (k = \text{even})\}.
\end{cases}
\tag{9.6.43}
$$

If we let

$$
E_w^{(p,q)} = (w_h^{(+,p,q)}, w_h^{(-,p,q)}),
\tag{9.6.44}
$$

the w-transform matrices of the operators A_L and A_{L-1} defined in (9.6.29), P_L defined in (9.5.14), $S_L^{(+)}$ defined in (9.6.41) and $S_L^{(-)}$ defined in (9.6.40) can be written in the form

$$
\hat{A}_L^{(p,q)} = \frac{2}{h^2}\begin{bmatrix} \alpha(1-c_p)+1 & -c_q \\ -c_q & \alpha(1-c_p)+1 \end{bmatrix},
\tag{9.6.45}
$$

$$
\hat{A}_{L-1}^{(p,q)} = \frac{1}{h^2}(2\alpha(1-c_p)+1-c_q^2)\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},
\tag{9.6.46}
$$

$$\hat{P}_L^{(p,q)} = \frac{1}{2}\begin{bmatrix} 1 & c_q \\ c_q & 1 \end{bmatrix}, \tag{9.6.47}$$

$$\hat{S}_L^{(-,p,q)} = \begin{bmatrix} 1 & 0 \\ \eta & 1-\omega \end{bmatrix}, \tag{9.6.48}$$

$$\hat{S}_L^{(+,p,q)} = \begin{bmatrix} 1-\omega & \eta \\ 0 & 1 \end{bmatrix} \tag{9.6.49}$$

where

$$\eta = \frac{\omega c_{q_l}}{1+\alpha_l(1-c_p)}. \tag{9.6.50}$$

So the representation of the whole red/black line SOR operator is given by

$$\hat{S}_L^{(p,q)} = \hat{S}_L^{(+,p,q)}\hat{S}_L^{(-,p,q)} = \begin{bmatrix} 1-\omega+\eta^2 & (1-\omega)\eta \\ \eta & 1-\omega \end{bmatrix}. \tag{9.6.51}$$

From (9.6.45), (9.6.46) and (9.6.47), the w-transform matrix of the coarse grid correction operator is given by

$$\hat{C}_L^{(p,q)} = I - \hat{A}_L^{(p,q)}\hat{P}_L^{(p,q)}(\hat{A}_{L-1}^{(p,q)})^{-1} = \begin{bmatrix} \xi & -c_q\xi \\ -c_q\xi & \xi \end{bmatrix} \tag{9.6.52}$$

where

$$\xi = \frac{\alpha(1-c_p)}{2\alpha(1-c_p)+1-c_q^2}. \tag{9.6.53}$$

The w-transform matrix of the two-level SCPMG-SOR operator is given by

$$\hat{T}_L^{(p,q)} = \hat{S}_L^{(p,q)}\hat{C}_L^{(p,q)} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \tag{9.6.54}$$

where

$$\begin{aligned} t_{11} &= \xi((1-\omega)(1-c_q\eta+\eta^2), && (9.6.55)\\ t_{12} &= \xi((1-\omega)(\eta-c_q)-c_q\eta^2), && (9.6.56)\\ t_{21} &= \xi(\eta-(1-\omega)c_q), && (9.6.57)\\ t_{22} &= \xi(1-\omega-c_q\eta). && (9.6.58)\end{aligned}$$

Here η is defined in (9.6.50) and ξ is defined in (9.6.53). The eigenvalues of the matrix $\hat{T}_L^{(p,q)}$ are given by

$$\lambda = \frac{1}{2}(t_{11}+t_{22}) \pm \sqrt{(t_{11}-t_{22})^2+4t_{12}t_{21}}). \qquad (9.6.59)$$

Hence the convergence factors of the two-level SCPMG-SOR algorithm can be calculated by

$$\rho(T_L) = \max_{\substack{1 \le p \le N-1 \\ 1 \le q \le \frac{N}{2}}} \rho(\hat{T}_L^{(p,q)}). \qquad (9.6.60)$$

Table 9.4 lists two-level convergence factors of the SCPMG with the red/black line SOR smoothing iteration for problem (2.3.8) with different values of α. The result is obtained by numerical procedure with the grid size $N = 64$. The iteration parameter ω is 0.89.

9.7 Multilevel Convergence Analysis

The multilevel convergence analysis of the SCPMG methods can be carried out in a similar way to the analysis of the PMG methods discussed in Chapter 8. Let M_l be the multilevel SCPMG operator on level l defined by

$$M_l = S_l(I - R_l(I - M_{l-1})A_{l-1}^{-1}A_l) \qquad (9.7.61)$$

for $l = 1, \ldots, N$ and

$$M_0 = S_0. \qquad (9.7.62)$$

Table 9.4: Two-Level Convergence Factor of SCPMG-SOR $\omega = 0.89$

α	$\rho(T_0)$
0.00001	0.000197
0.0001	0.001706
0.001	0.007008
0.01	0.008256
0.1	0.025835
1	0.043987
10	0.087332
100	0.107490
1000	0.109689
10000	0.109911
100000	0.109933

All the operators in (8.4.87) corresponding to the SCPMG algorithm are commutative, it can be shown that M_l and M_{l-1} have the recursive relation (8.4.89).

The following theorem gives an upper bound on the convergence factor of the multilevel SCPMG algorithm.

Theorem 9.1 *Let*

$$\sigma_1 = \sup_{\alpha>0} \max_{\substack{1 \le p \le N-1 \\ 1-N \le q \le N}} \frac{|\lambda_{p,q}(T_L(\alpha))|}{1 - |\lambda_{p,q}(S_L(\alpha)) - \lambda_{p,q}(T_L(\alpha))|} \tag{9.7.63}$$

If

$$\rho(M_l) = \sup_{\alpha>0} \max_{\substack{1 \le p \le N-1 \\ 1-N \le q \le N}} |\lambda_{p,q}(M_l(\alpha))| \le \sigma_1 \tag{9.7.64}$$

holds on level l, then we have

$$\rho(M_L) \le \sigma_1 \tag{9.7.65}$$

Proof: The proof is based on the fact that (9.7.63) is valid for $l = 1, \ldots, L$. In fact, because

$$\lambda_{p,q}(T_{l-1}(\alpha)) = \lambda_{p,2q}(T_l(\alpha)) \tag{9.7.66}$$

$$\lambda_{p,q}(S_{l-1}(\alpha)) = \lambda_{p,2q}(S_l(\alpha)), \tag{9.7.67}$$

we have

$$\begin{aligned} \sigma_1 &= \sup_{\alpha>0} \max_{\substack{1 \le p \le N-1 \\ 1-N \le q \le N}} \frac{|\lambda_{p,q}(T_L(\alpha))|}{1 - |\lambda_{p,q}(S_L(\alpha)) - \lambda_{p,q}(T_L(\alpha))|} \\ &\ge \sup_{\alpha>0} \max_{\substack{1 \le p \le N-1 \\ 1-N \le q \le N}} \frac{|\lambda_{p,q}(T_l(\alpha))|}{1 - |\lambda_{p,q}(S_l(\alpha)) - \lambda_{p,q}(T_l(\alpha))|} \end{aligned} \tag{9.7.68}$$

Table 9.5: Multilevel Convergence Factors for SCPMG-MLJ

	γ			
α	0.8	0.9	1.0	1.1
0.00001	0.0203	0.0181	0.0163	0.0148
0.0001	0.1349	0.1218	0.1109	0.1019
0.001	0.2381	0.2174	0.2000	0.1851
0.01	0.2381	0.2173	0.1999	0.1851
0.1	0.2379	0.2169	0.1995	0.1847
1	0.2380	0.2174	0.2000	0.1851
10	0.2375	0.2166	0.1991	0.1841
100	0.2370	0.2143	0.1951	0.1788
1000	0.2245	0.1828	0.1492	0.1219
10000	0.2000	0.1017	0.0465	0.0999
100000	0.1999	0.0996	0.0042	0.0999

Now assume that $\rho(M_{l-1}) \leq \sigma_1$ holds. For level l we have

$$\begin{aligned}
|\lambda_{p,q}(M_l(\alpha))| &= |\lambda_{p,q}(T_l(\alpha)) + (\lambda_{p,q}(S_l(\alpha)) - \lambda_{p,q}(T_l(\alpha)))\lambda_{p,q}(M_{l-1}(\alpha))| \\
&\leq |\lambda_{p,q}(T_l(\alpha))| + |\lambda_{p,q}(S_l(\alpha)) - \lambda_{p,q}(T_l(\alpha))|\rho(M_{l-1}) \\
&\leq (1 - |\lambda_{p,q}(S_l(\alpha)) - \lambda_{p,q}(T_l(\alpha))|)\sigma_1 \\
&\quad + |\lambda_{p,q}(S_l(\alpha)) - \lambda_{p,q}(T_l(\alpha))|\sigma_1 \\
&= \sigma_1. \qquad (9.7.69)
\end{aligned}$$

Table 9.5 gives the multilevel convergence factors of the SCPMG method with the modified line Jacobi smoothing iteration (SCPMG-MLJ). The case is the same as that used in creating Table 9.2. We note that a good choice of the extrapolation factor γ is around 1.1 in this case.

Chapter 10

Matrix-Dependent Interpolation and Restriction

10.1 Introduction

The coefficients of many differential equations of real problems (e.g. petroleum reservoir simulations) are discontinuous. They might have jumps of several magnitudes across some inner boundaries. In these cases, the linear interpolation and full weighting restriction operators are not accurate and the multigrid methods will show a slow convergence rate. To overcome this difficulty, the interpolation and restriction operators should include some information about the coefficients. In this chapter we discuss several matrix-dependent interpolation and restriction operators.

10.2 Discontinuous Diffusion Coefficients

We are concerned with solving the differential equation

$$\begin{cases} -\dfrac{\partial}{\partial x}\alpha^{(1)}\dfrac{\partial u}{\partial x} - \dfrac{\partial}{\partial y}\alpha^{(2)}\dfrac{\partial u}{\partial y} + \sigma u = f & (x,y) \in \Omega, \\ u = \phi(x,y) & (x,y) \in \partial\Omega \end{cases} \tag{10.2.1}$$

on a bounded region Ω in R^2. The coefficients $\alpha^{(i)}$ are positive and σ is nonnegative. α_i, σ and f are allowed to be discontinuous across internal boundaries in Ω.

The finite difference representation of problem (10.2.1) can be derived by the integral approximation approach (see e.g. Young and Gregory [71] or

Varga [65]):

$$\begin{cases} a_{j,k}^{(c)}u_{j,k} + a_{j,k}^{(e)}u_{j+1,k} + a_{j,k}^{(w)}u_{j-1,k} + a_{j,k}^{(n)}u_{j,k+1} + a_{j,k}^{(s)}u_{j,k-1} = h^2 f_{j,k}, \\ u_{j,0} = \phi(x_j, y_0), \\ u_{j,N} = \phi(x_j, y_N), \\ u_{0,k} = \phi(x_0, y_k), \\ u_{N,k} = \phi(x_N, y_k), \\ j, k = 1, \ldots, N-1 \end{cases} \qquad (10.2.2)$$

where

$$\begin{cases} a_{j,k}^{(c)} = \alpha_{j-\frac{1}{2},k}^{(1)} + \alpha_{j+\frac{1}{2},k}^{(1)} + \alpha_{j,k-\frac{1}{2}}^{(2)} + \alpha_{j,k+\frac{1}{2}}^{(2)} + h^2\sigma_{j,k}, \\ a_{j,k}^{(w)} = -\alpha_{j-\frac{1}{2},k}^{(1)}, \\ a_{j,k}^{(e)} = -\alpha_{j+\frac{1}{2},k}^{(1)}, \\ a_{j,k}^{(s)} = -\alpha_{j,k-\frac{1}{2}}^{(2)}, \\ a_{j,k}^{(n)} = -\alpha_{j,k+\frac{1}{2}}^{(2)}. \end{cases} \qquad (10.2.3)$$

The corresponding matrix format can be represented by

$$A_h u_h = b_h. \qquad (10.2.4)$$

In the case of $\alpha^{(i)}$ having jumps at some places in the domain, ∇u will also have jumps at these places. The approximation accuracy of linear interpolation depends on the continuity of ∇u, and only smooth functions can be accurately interpolated onto the fine grid by linear interpolation. Therefore, a multigrid algorithm with a linear interpolation operator does not provide the usual fast convergence rate.

Noticing that the product of $\alpha^{(i)}$ and ∇u are continuous, a natural way to overcome this difficulty is to construct an interpolation operator in such a way that the continuity of $\alpha^{(i)}\nabla u$ is reserved. Obviously, the interpolation operator must include some information about $\alpha^{(i)}$. For linear system (10.2.2), this is equivalent to using the matrix A_h. In the remainder of this chapter, we will discuss the construction of the matrix-dependent interpolation and restriction operators in standard coarsening cases as well as in semicoarsening cases.

10.3 Standard Coarsening Cases

10.3.1 Matrix-Dependent Interpolation

For convenience, we represent A_h in the difference stencil form at point (x_j, y_k) as

$$\begin{bmatrix} a_{j,k}^{(nw)} & a_{j,k}^{(n)} & a_{j,k}^{(ne)} \\ a_{j,k}^{(w)} & a_{j,k}^{(c)} & a_{j,k}^{(e)} \\ a_{j,k}^{(sw)} & a_{j,k}^{(s)} & a_{j,k}^{(se)} \end{bmatrix} \tag{10.3.5}$$

where

$$a_{j,k}^{(sw)} = a_{j,k}^{(nw)} = a_{j,k}^{(se)} = a_{j,k}^{(ne)} = 0 \tag{10.3.6}$$

and the other five elements are defined in (10.2.3). For the near boundary grid points, the corresponding undefined elements are set to zero. Since we treat each grid point in the same way, the subscript j, k is often dropped if there is no ambiguity. We generally consider the 9-point difference formula cases, so that the discussion can be applied to the matrix problem arising from the 9-point discretization scheme.

Let the fine grid be divided into four subsets Ω_{++}, Ω_{-+}, Ω_{+-}, and Ω_{--} defined in (5.2.2) and let subset Ω_{--} be used as the coarse grid Ω_{2h}.

The interpolation operator P_h is constructed in such a way that after the coarse grid correction, the elements of the new residual vector will not get larger on the points which do not belong to the coarse grid Ω_{2h}. This can be achieved by requiring

$$(A_h P_h \delta_{2h})(x, y) \approx 0 \quad (x, y) \notin \Omega_{2h} \tag{10.3.7}$$

If we let

$$\begin{cases} A^{(W)} &= a^{(nw)} + a^{(w)} + a^{(sw)}, \\ A^{(E)} &= a^{(ne)} + a^{(e)} + a^{(se)}, \\ A^{(N)} &= a^{(ne)} + a^{(n)} + a^{(nw)}, \\ A^{(S)} &= a^{(se)} + a^{(s)} + a^{(sw)}, \\ A^{(C)} &= a^{(n)} + a^{(c)} + a^{(s)}, \\ A^{(L)} &= a^{(e)} + a^{(c)} + a^{(w)}, \end{cases} \tag{10.3.8}$$

we require that

$$\begin{cases} A^{(W)}\delta_h)(x-h,y) + A^{(C)}\delta_h)(x,y) + A^{(E)}\delta_h)(x+h,y) \\ \qquad = 0 \qquad (x,y) \in \Omega_{+-}, \\ A^{(S)}\delta_h)(x,y-h) + A^{(L)}\delta_h)(x,y) + A^{(N)}\delta_h)(x,y+h) \\ \qquad = 0 \qquad (x,y) \in \Omega_{-+}, \\ a^{(sw)}\delta_h)(x-h,y-h) + a^{(s)}\delta_h)(x,y-h) + a^{(se)}\delta_h)(x+h,y-h) \\ +a^{(w)}\delta_h)(x-h,y) + a^{(c)}\delta_h)(x,y) + a^{(e)}\delta_h)(x+h,y) \\ +a^{(nw)}\delta_h)(x-h,y+h) + a^{(n)}\delta_h)(x,y+h) + a^{(ne)}\delta_h)(x+h,y+h) \\ \qquad = 0 \qquad (x,y) \in \Omega_{++}. \end{cases} \tag{10.3.9}$$

Based on (10.3.9), the interpolation $\delta_h = P_h\delta_{2h}$ can be defined by

$$\delta_h(x,y) = \begin{cases} \delta_{2h}(x,y) & (x,y) \in \Omega_{--} \\ p^{(e)}(x-h,y)\delta_{2h}(x-h,y) & \\ \quad +p^{(w)}(x+h,y)\delta_{2h}(x+h,y) & (x,y) \in \Omega_{+-} \\ p^{(n)}(x,y-h)\delta_{2h}(x,y-h) & \\ \quad +p^{(s)}(x,y+h)\delta_{2h}(x,y+h) & (x,y) \in \Omega_{-+} \\ p^{(ne)}(x-h,y-h)\delta_{2h}(x-h,y-h) & \\ \quad +p^{(se)}(x-h,y+h)\delta_{2h}(x-h,y+h) & \\ \quad +p^{(nw)}(x+h,y-h)\delta_{2h}(x+h,y-h) & \\ \quad +p^{(sw)}(x+h,y+h)\delta_{2h}(x+h,y+h) & (x,y) \in \Omega_{++} \end{cases} \tag{10.3.10}$$

where for $(x,y) \in \Omega_{--}$

$$\begin{cases} p^{(e)}(x,y) &= -(A^{(W)}(x+h,y))/(A^{(C)}(x+h,y)) \\ p^{(w)}(x,y) &= -(A^{(E)}(x-h,y))/(A^{(C)}(x-h,y)) \\ p^{(n)}(x,y) &= -(A^{(S)}(x,y+h))/(A^{(L)}(x,y+h)) \\ p^{(s)}(x,y) &= -(A^{(N)}(x,y-h))/(A^{(L)}(x,y-h)) \end{cases} \tag{10.3.11}$$

and

$$\begin{cases} p^{(sw)}(x,y) &= -[a^{(ne)}(x-h,y-h) + a^{(e)}(x-h,y-h)p^{(s)}(x,y) \\ &\quad +a^{(n)}(x-h,y-h)p^{(w)}(x,y)]/(a^{(c)}(x-h,y-h)) \\ p^{(nw)}(x,y) &= -[a^{(se)}(x-h,y+h) + a^{(e)}(x-h,y+h)p^{(n)}(x,y) \\ &\quad +a^{(s)}(x-h,y+h)p^{(w)}(x,y)]/(a^{(c)}(x-h,y+h)) \\ p^{(se)}(x,y) &= -[a^{(nw)}(x+h,y-h) + a^{(w)}(x+h,y-h)p^{(s)}(x,y) \\ &\quad +a^{(n)}(x+h,y-h)p^{(e)}(x,y)]/(a^{(c)}(x+h,y-h)) \\ p^{(ne)}(x,y) &= -[a^{(sw)}(x+h,y+h) + a^{(w)}(x+h,y+h)p^{(n)}(x,y) \\ &\quad +a^{(s)}(x-h,y+h)p^{(w)}(x,y)]/(a^{(c)}(x+h,y+h)) \end{cases} \tag{10.3.12}$$

The interpolated corrections generated by the interpolation operator (10.3.10) will not cause a large residual on the fine grid. In fact $A_h\delta^{(h)}$ is zero at any point $(x,y) \in \Omega_{++}$ and is approximately zero at any point $(x,y) \in \Omega_{-+}$ or $(x,y) \in \Omega_{+-}$.

The transpose of the interpolation matrix can be used as the restriction matrix. Specifically for any point $(x,y) \in \Omega_{--}$, the restriction $r_{2h} = R_h r_h$ is defined by

$$\begin{aligned} (R_h r_h)(x,y) = r_h(x,y) & \\ &+p^{(e)}(x,y)r_h(x+h,y) + p^{(w)}(x,y)r_h(x-h,y) \\ &+p^{(n)}(x,y)r_h(x,y+h) + p^{(s)}(x,y)r_h(x,y-h) \\ &+p^{(ne)}(x,y)r_h(x+h,y+h) + p^{(se)}(x,y)r_h(x+h,y-h) \\ &+p^{(nw)}(x,y)r_h(x-h,y+h) + p^{(sw)}(x,y)r_h(x-h,y-h) \end{aligned} \tag{10.3.13}$$

where $p^{(e)}$, $p^{(w)}$,$p^{(n)}$,$p^{(s)}$,$p^{(ne)}$, $p^{(se)}$,$p^{(nw)}$ and $p^{(sw)}$ are defined in (10.3.11) and (10.3.12).

After the interpolation and the restriction operators have been constructed, the coarse grid operator A_{2h} can be defined by

$$A_{2h} = R_h A_h P_h. \tag{10.3.14}$$

It can be shown that the A_{2h} is a 9-point stencil operator if A_h is also a 9-point stencil operator.

10.3.2 Coarse Grid Matrix

We use z_h to denote the vector on the grid Ω_h with all its elements unity. We have the following lemmas.

Lemma 10.1 *If the row sums of the matrix A_h are zero and the values $A^{(C)}$, $S^{(L)}$ and $a^{(c)}$ are not zero, then the interpolation operator P_h defined in (10.3.10) has the property*

$$P_h z_{2h} = z_h. \tag{10.3.15}$$

Proof: We use $A^{(R)}$ denote the row sum of the matrix A_h. By direct calculation from (10.3.11) and (10.3.12), we have

$$\begin{aligned} p^{(e)}(x,y) + p^{(w)}(x+2h,y) \\ = 1 - A^{(R)}(x+h,y)/A^{(C)}(x+h,y) = 1, \end{aligned} \tag{10.3.16}$$

$$\begin{aligned} p^{(n)}(x,y) + p^{(s)}(x,y+2h) \\ = 1 - A^{(R)}(x,y+h)/A^{(L)}(x,y+h) = 1, \end{aligned} \tag{10.3.17}$$

and

$$\begin{aligned} &p^{(ne)}(x,y)+p^{(nw)}(x+2h,y)+p^{(se)}(x,y+2h)\\ &\quad+p^{(sw)}(x+2h,y+2h)\\ &\qquad=1-A^{(R)}(x+h,y+h)/a^{(c)}(x+h,y+h)=1. \end{aligned} \tag{10.3.18}$$

Since the restriction matrix is the transpose of the interpolation matrix, we have

$$(z_{2h})^T R_h = (z_h)^T. \tag{10.3.19}$$

Lemma 10.2 *If the conditions in Lemma 10.1 are valid and the coarse grid problem is defined by $A_{2h}\delta_{2h} = r_{2h}$ with $A_{2h} = R_h A_h P_h$, and $r_{2h} = R_h r_h$, then*

1. *The sum of elements of r_{2h} is equal to the sum of elements of r_h;*
2. *Every row sum of matrix A_{2h} equals zero if every row sum of matrix A_h equals zero;*
3. *Every column sum of matrix A_{2h} equals zero if every column sum of matrix A_h equals zero;*
4. *A_{2h} is a nine-diagonal matrix with a nine-point difference stencil.*

Proof: All four parts can be verified by direct calculation. For part (1) to part (3), we have

$$\begin{aligned} (z_{2h})^T r_{2h} &= (z_{2h})^T R_h r_h \\ &= (z^{(h)})^T r_h, \end{aligned} \tag{10.3.20}$$

$$\begin{aligned} A_{2h} z_{2h} &= R_h A_h P_h z_{2h} \\ &= R_h A_h z_h, \end{aligned} \tag{10.3.21}$$

$$\begin{aligned} (z_{2h})^T A_{2h} &= (z_{2h})^T R_h A_h P_h \\ &= (z_h)^T A_h P_h. \end{aligned} \tag{10.3.22}$$

Part (4) is based on the fact that for any row of the matrix R_h, there are at most nine columns of the matrix $A_h P_h$ such that their inner product is not zero.

For the Poisson problem with homogeneous Neumann boundary conditions, the linear system $A_h u_h = b_h$, arising from a conservative discretization scheme, is singular but solvable. The row sums, the column sums and the sum of elements of $B^{(h)}$ are all zeros. Because of Lemmas 10.1 and 10.2, the linear systems constructed on all coarse levels are also singular and solvable.

10.4 Semicoarsening Interpolation

If a semicoarsening scheme is used in multigrid methods, the interpolation and restriction are applied only in the coarsening direction. We consider two-dimensional cases with the y direction as the coarsening direction. The extension of the discussion to higher dimensional cases is straightforward. The coarse grid is defined in (9.4.8).

10.4.1 Pointwise Scheme

In this scheme, the elements of the interpolation matrix P_h at point (x,y) depend only on the elements of A_h at this point and its neighbor points. The interpolation $\delta^{(h)} = P_h \delta^{(2h)}$ is defined by requiring

$$\begin{aligned} &A^{(S)}(x,y)\delta_h(x,y-h) + A^{(L)}(x,y)\delta_h(x,y) \\ &\quad + A^{(N)}(x,y)\delta_h(x,y+h) = 0 \qquad (x,y) \in \Omega_+. \end{aligned} \tag{10.4.23}$$

Therefore, we have

$$\left\{ \begin{array}{rcll} \delta_h(x,y) & = & \delta_{2h}(x,y) & (x,y) \in \Omega_-, \\ \delta_h(x,y) & = & p^{(n)}(x,y-h)\delta_{2h}(x,y-h) & \\ & & +p^{(s)}(x,y+h)\delta_{2h}(x,y+h) & (x,y) \in \Omega_+, \end{array} \right. \tag{10.4.24}$$

where for $(x, y) \in \Omega_-$

$$\begin{cases} p^{(n)}(x,y) &= -(A^{(S)}(x,y+h))/(A^{(L)}(x,y+h)), \\ p^{(s)}(x,y) &= -(A^{(N)}(x,y-h))/(A^{(L)}(x,y-h)). \end{cases} \tag{10.4.25}$$

The transpose of the interpolation matrix is used as the restriction matrix and $r_{2h} = R_h r_h$ is defined by

$$\begin{aligned}(R_h r_h)(x,y) = r_h(x,y) + p^{(n)}(x,y) r_h(x,y+h) \\ + p^{(s)}(x,y) r_h(x,y-h)\end{aligned} \tag{10.4.26}$$

where $p^{(n)}$ and $p^{(s)}$ are defined in (10.4.25).

The interpolation and restriction operators defined here still have properties (10.3.15) and (10.3.19). If the conditions in Lemma 10.1 are true, then the coarse grid matrix $A_{2h} = R_h A_h P_h$ still has the properties described in Lemma 10.2.

10.4.2 Blockwise Scheme

In the interpolation (restriction) process of the semicoarsening multigrid methods, we treat a two-dimensional case as a one-dimensional case and each unit is a line in the x-direction. Therefore it will be more accurate if we consider all the points on the whole x-line together rather than individual points in the interpolation process.

The matrices A_h can be written in the block form with each block row corresponding to points on one x-line. In the case of a five x-lines grid, for example, the block form of the matrix A_h is given by

$$A_h = \begin{bmatrix} A_{1,1} & A_{1,2} & & & \\ A_{2,1} & A_{2,2} & A_{2,3} & & \\ & A_{3,2} & A_{3,3} & A_{3,4} & \\ & & A_{4,3} & A_{4,4} & A_{4,5} \\ & & & A_{5,4} & A_{5,5} \end{bmatrix}. \tag{10.4.27}$$

If we let $\vec{\delta}_h(y_k)$ denote the sub–vector of δ_h corresponding to the points on the x-line with $y = y_k$, then we require

$$\begin{aligned} &A_{k,k-1}\vec{\delta}_h(y_{k-1}) + A_{k,k}\vec{\delta}_h(y_k) + A_{k,k+1}\vec{\delta}_h(y_{k+1}) = \vec{0}, \\ &\qquad (x, y_k) \in \Omega_+. \end{aligned} \tag{10.4.28}$$

Based on (10.4.28), the interpolation $\delta_h = P_h\delta_{2h}$ is given by

$$\vec{\delta}_h(y_k) = \begin{cases} \vec{\delta}_{2h}(y_k) & (x, y_k) \in \Omega_- \\ P^{(n)}(y_{k-1})\vec{\delta}_{2h}(y_k - h) & \\ \quad +P^{(s)}(y_{k+1})\vec{\delta}_{2h}(y_k + h) & (x, y_k) \in \Omega_+ \end{cases} \tag{10.4.29}$$

where the matrices are given by

$$\begin{cases} P^{(n)}(y_{k-1}) &= -A^{-1}_{k-1,k-1}A_{k-1,k} \\ P^{(s)}(y_{k+1}) &= -A^{-1}_{k+1,k+1}A_{k+1,k}. \end{cases} \tag{10.4.30}$$

Unfortunately, this definition would lead to a nonsparse interpolation and therefore lead to nonsparse coarse grid operators. In practice, $p^{(n)}(y_{k-1})$ and $p^{(s)}(y_{k+1})$ can be defined as diagonal matrices which are approximations to $-A^{-1}_{k-1,k-1}A_{k-1,k}$ and $-A^{-1}_{k+1,k+1}A_{k+1,k}$ in the following sense

$$\begin{cases} P^{(n)}(y_{k-1})\vec{z} &= -A^{-1}_{k-1,k-1}A_{k-1,k}\vec{z} \\ P^{(s)}(y_{k+1})\vec{z} &= -A^{-1}_{k+1,k+1}A_{k+1,k}\vec{z} \end{cases} \tag{10.4.31}$$

Here, $\vec{z}$ is the vector with all unit elements. The interpolation (10.4.29) can now be written in the form

$$\delta_h(x,y) = \begin{cases} \delta_{2h}(y) & (x, y_k) \in \Omega_- \\ p^{(n)}(x, y_{k-1})\delta_{2h}(x, y_k - h) & \\ \quad +p^{(s)}(x, y_{k+1})\delta_{2h}(y_k + h) & (x, y_k) \in \Omega_+ \end{cases} \tag{10.4.32}$$

where $p^{(n)}(x, y_{k-1})$ and $p^{(s)}(x, y_{k+1})$ are the diagonal elements corresponding to position x in the matrices $P^{(n)}(y_{k-1})$ and $P^{(s)}(y_{k+1})$ respectively.

As in the pointwise scheme, the transpose of the interpolation matrix is used as the restriction matrix and $r_{2h} = R_h r_h$ is defined by (10.4.26).

The calculation of the diagonal matrices $p^{(n)}(y_{k-1})$ and $p^{(s)}(y_{k+1})$ involves solving small three-diagonal systems. For example, to calculate $P^{(s)}(y_{k+1})$, we need to solve the equation

$$A_{k+1,k+1}\vec{v} = A_{k+1,k}\vec{z} \tag{10.4.33}$$

for vector $\vec{v}$ which is the the main diagonal of $P^{(s)}(y_{k+1})$:

$$P^{(s)}(y_{k+1}) = I\vec{v}. \tag{10.4.34}$$

In order to make the whole algorithm effective, (10.4.33) need not be solved exactly. Usually one cycle of a low dimensional multigrid iteration is good enough (See Smith [60]).

Since the interpolation operator defined in (10.4.31) still holds the property of Lemma 10.1, the coarse grid matrix $A_{2h} = R_h A_h P_h$ still has the properties described in Lemma 10.2.

10.4.3 Schur Complement Scheme

It is well known that the Schur complement scheme for solving a linear equation system can be considered as a two-level multigrid method ([17], [2]). Based on the Schur complement scheme, we can construct the interpolation and the restriction operators.

We use the partition of unknowns based on the so-called *zebra ordering.* We mark all the odd numbered x-lines red and all the even numbered x-lines black. We then reorder the position of unknowns in such a way that the red unknowns are counted first. The linear system can then be expressed in the form

$$Au = \begin{bmatrix} A_r & A_{rb} \\ A_{br} & A_b \end{bmatrix} \begin{bmatrix} u_r \\ u_b \end{bmatrix} = \begin{bmatrix} b_r \\ b_b \end{bmatrix} = b. \tag{10.4.35}$$

The solution procedure of the Schur complement method may be described as follows. First we solve a smaller Schur complement linear system

$$A_s u_b = b_s \tag{10.4.36}$$

for u_b, where

$$A_s = A_b - A_{br} A_r^{-1} A_{rb} \tag{10.4.37}$$

and

$$b_s = b_b - A_{br} A_r^{-1} b_r. \tag{10.4.38}$$

After u_b becomes known, we solve

$$A_r u_r = b_r - A_{rb} u_b \tag{10.4.39}$$

for u_r.

If we consider the black nodes as the coarse grid, then the Schur complement method can be viewed as a two-level multigrid method where (10.4.36) is the coarse grid linear system. The matrix A_s in (10.4.36) can be written in the form

$$A_s = \begin{bmatrix} T_{br} & I_b \end{bmatrix} \begin{bmatrix} A_r & A_{rb} \\ A_{br} & A_b \end{bmatrix} \begin{bmatrix} T_{rb} \\ I_b \end{bmatrix} = RAP \tag{10.4.40}$$

where I_b is the identity matrix on the black node set and T_{br} and T_{rb} must be determined.

Equation (10.4.38) can be considered as the restriction operation $b_s = Rb$ and the restriction operator R is given by

$$R = \begin{bmatrix} T_{br} & I_b \end{bmatrix} = \begin{bmatrix} -A_{br} A_r^{-1} & I_b \end{bmatrix} \tag{10.4.41}$$

By comparing (10.4.40) to (10.4.37), one can see that if T_{br} is defined by (10.4.41), the coarse grid matrix A_s defined by (10.4.40) is independent of T_{rb}. In other words, there is no restriction for choosing the interpolation operator as far as tthe coarse grid matrix A_s is concerned.

Equation (10.4.39) can be considered as the interpolation operation $u = Pu_b$. If we let $b_r = 0$, the interpolation operator P is given by

$$P = \begin{bmatrix} T_{rb} \\ I_b \end{bmatrix} = \begin{bmatrix} -A_r^{-1}A_{rb} \\ I_b \end{bmatrix}. \tag{10.4.42}$$

In the case of five x-lines, the nine-diagonal matrix A, arising from the discretization of differential equation (10.2.1) can be written in the form

$$A = \begin{bmatrix} A_{1,1} & & & A_{1,2} & \\ & A_{3,3} & & A_{3,2} & A_{3,4} \\ & & A_{5,5} & & A_{5,4} \\ A_{2,1} & A_{2,3} & & A_{2,2} & \\ & A_{4,3} & A_{4,5} & & A_{4,4} \end{bmatrix} \tag{10.4.43}$$

where each $A_{i,j}$ is a tridiagonal sub–matrix. The corresponding T_{rb} and T_{br} will be

$$\begin{aligned} T_{rb} &= -A_r^{-1}A_{rb} \\ &= -\begin{bmatrix} A_{1,1}^{-1} & & \\ & A_{3,3}^{-1} & \\ & & A_{5,5}^{-1} \end{bmatrix} \begin{bmatrix} A_{1,2} & \\ A_{3,2} & A_{3,4} \\ & A_{5,4} \end{bmatrix} \\ &= -\begin{bmatrix} A_{1,1}^{-1}A_{1,2} & \\ A_{3,3}^{-1}A_{3,2} & A_{3,3}^{-1}A_{3,4} \\ & A_{5,5}^{-1}A_{5,4} \end{bmatrix} \end{aligned} \tag{10.4.44}$$

and

$$\begin{aligned} T_{br} &= -A_{br}A_r^{-1} \\ &= -\begin{bmatrix} A_{2,1} & A_{2,3} & \\ & A_{4,3} & A_{4,5} \end{bmatrix} \begin{bmatrix} A_{1,1}^{-1} & & \\ & A_{3,3}^{-1} & \\ & & A_{5,5}^{-1} \end{bmatrix} \\ &= -\begin{bmatrix} A_{2,1}A_{1,1}^{-1} & A_{2,3}A_{3,3}^{-1} & \\ & A_{4,3}A_{3,3}^{-1} & A_{4,5}A_{5,5}^{-1} \end{bmatrix}. \end{aligned} \tag{10.4.45}$$

The Schur complement linear system (10.4.36) is generally a dense system even if the original matrix A is sparse. However, we can use the diagonal approximation of the sub-matrices in T_{rb} and T_{br} to construct the 3-point interpolation operator and the 3-point restriction operator and to keep the coarse grid matrix A_s a nine-diagonal matrix.

From (10.4.44) and (10.4.42), the interpolation $\delta_h = P_h\delta_{2h}$ is defined by

$$\vec{\delta}_h(y_k) = \begin{cases} \vec{\delta}_{2h}(y_k) & (x,y_k) \in \Omega_- \\ P^{(n)}(y_{k-1})\vec{\delta}_{2h}(y_k - h) & \\ \quad +P^{(s)}(y_{k+1})\vec{\delta}_{2h}(y_k + h) & (x,y_k) \in \Omega_+ \end{cases} \tag{10.4.46}$$

where the diagonal matrices $P^{(n)}(y_{k-1})$ and $P^{(s)}(y_{k+1})$ are given by

$$\begin{cases} P^{(n)}(y_{k-1})\vec{z} = -A_{k-1,k-1}^{-1}A_{k-1,k}\vec{z} \\ P^{(s)}(y_{k+1})\vec{z} = -A_{k+1,k+1}^{-1}A_{k+1,k}\vec{z}. \end{cases} \tag{10.4.47}$$

From (10.4.45) and (10.4.41), the restriction $r_{2h} = R_h r_h$ is defined by

$$\begin{aligned} \vec{r}_{2h}(y_k) &= \vec{r}_h(y_k) + R^{(n)}(y_k)\vec{r}_h(y_{k+1}) \\ &\quad +R^{(s)}(y_k)\vec{r}_h(y_{k-1}) \qquad (x,y_k) \in \Omega_- \end{aligned} \tag{10.4.48}$$

where the diagonal matrices $R^{(n)}(y_k)$ and $R^{(s)}(y_k)$ are given by

$$\left\{ \begin{array}{rcl} \vec{z}^T R^{(n)}(y_k) & = & -\vec{z}^T A_{k,k+1} A_{k+1,k+1}^{-1} \\ \vec{z}^T R^{(s)}(y_k) & = & -\vec{z}^T A_{k,k-1} A_{k-1,k-1}^{-1}. \end{array} \right. \tag{10.4.49}$$

Here $\vec{z}$ is a vector with all unit elements.

Comparing the interpolation operator and the restriction operator derived from the Schur complement scheme to those derived from the blockwise scheme, we find that the interpolation operators are actually the same although they come from different points of view. The restriction operator from the Schur complement scheme uses the information of the transpose of the matrix A. In the case of nonsymmetric problem, the matrix R is in general not equal to the transpose of the matrix P. Since the Schur complement scheme does not require a symmetric problem, the interpolation operator and the restriction operator constructed from the Schur complement scheme are also suitable in nonsymmtric cases.

Chapter 11

A Compositional Reservoir Simulator

11.1 Introduction

One of the target application fields of multigrid methods is in reservoir simulation. Numerical reservoir simulation refers to the development and usage of a mathematical model or simulator which describes the flow of fluids in a permeable medium. In general, the reservoir simulator requires the numerical solution of a set of coupled, nonlinear partial differential equations describing complex physical processes in a three-dimensional domain.

The most complex simulators are those used for the simulation of enhanced oil recovery processes. Enhanced oil recovery processes involve the injection of chemicals, solvents or heat into the underground reservoirs in order to supplement the natural energy and to increase the recovery of the trapped oil from the reservoir rock.

Those methods using solvents such as carbon dioxide or enriched gas are called miscible gas flooding. Miscible gas flooding processes typically involve the multiphase flow of a large number of components. This process can be represented by a mathematical formulation involving a set of coupled, highly nonlinear, time-dependent partial differential equations [59].

Two basic methods for solving multiphase coupled equations are the implicit pressure explicit saturation method (IMPES) and the fully implicit method. In the IMPES method, the gridblock pressure is solved for implicitly using explicit dating of saturation dependent terms. After the pressure solution is obtained, the saturations are explicitly updated by substituting the results in the material balance equations. In the fully implicit method, the coupled equations are solved simultaneously. Since the number of unknowns is the number of grid points multiplied by the number of phases, the method becomes expensive per time step for multidimensional problems. However, the fully implicit method has much better stability properties. This means that one is able to take much larger time steps in general. Therefore, the fully implicit method may solve the simulation problem faster due to its ability of taking larger time steps in comparision to IMPES.

The governing partial differential equations, due to their nonlinear nature, must be solved by using numerical methods such as finite-difference or finite-element methods. In the petroleum industry, the most popular methods are those involving finite-difference techniques. Thus the partial differential equations are discretized using finite-difference approximations of time and spatial derivatives. This leads to a system of algebraic equations that must be solved at each time level numerically.

Because the performance of a miscible process depends heavily on the accuracy of the solution of the governing nonlinear miscible flooding equations, both a fine mesh and very small time steps are needed. Thus, a very large amount of computer time, as well as a large amount of storage, is required for field simulation. The solution of the governing equations takes a large part of the total computer time and usually determines the storage needed. Therefore it is especially important to use a highly efficient numerical method to solve the governing equations.

In this chapter we discuss the mathematical formulation of UTCOMP, a three-dimensional, multicomponent, multiphase miscible-flooding simulator developed at The University of Texas at Austin [12], [13].

11.2 Description of the Simulator

UTCOMP is an isothermal, three-dimensional, miscible-flooding compositional simulator. In a compositional model, the principles of mass conservation and phase equilibria are employed to compute phase pressures, saturations and phase compositions at each grid block in the reservoir. Compositional models are usually required when the fluid properties are dependent on composition and pressure in petroleum reservoir simulation. Such reservoir processes include

1. miscible flooding by carbon dioxide or enriched gas injection; and

2. depletion of volatile oil reservoirs or gas condensate reservoirs.

The model permits a maximum of four phases to flow simultaneously: (1) an aqueous phase, (2) an oil phase, (3) a gas phase and (4) an additional nonaqueous liquid phase. Water is only allowed in the aqueous phase. Water is slightly compressible and water viscosity is constant.

The model also assumes that the reservoir is surrounded by impermeable zones so that no-flow boundaries exist. The permeability tensor is orthogonal and aligned with the coordinate system. The adsorption of the rock is negligible. The fluid flow in the reservoir is characterized by Darcy's law for multiphase flow. The injection and production of fluids can be treated as source or sink terms.

The solution scheme is analogous to IMPES. In this procedure, we first solve the pressure equation implicitly for the grid block pressure by using explicit dating of saturation dependent terms. Then we solve the material balance equations explicitly for the total concentration of each component in moles. Finally, we obtain phase compositions for each component by flash calculations.

11.3 Mass Conservation Equations

Multicomponent, multiphase flow in porous media occurs as a transport of chemical species in multiple homogeneous phases under the influence of four forces: viscous, gravity, dispersion (or diffusion) and capillary forces.

The mass conservation of each component should hold at each point in the reservoir.

The law of mass conservation for component κ can be written in the following form:

$$\frac{\partial W_\kappa}{\partial t} + \vec{\nabla} \cdot \vec{F}_\kappa - R_\kappa = 0, \tag{11.3.1}$$

where W_κ, $\vec{F}_\kappa$ and R_κ are the mass accumulation, flux and source terms respectively. If equation (11.3.1) is expressed in terms of moles per unit bulk volume per unit time, the accumulation term can be expressed in terms of the sum of the moles in each phase. Since hydrocarbon is not permitted in the aqueous phase nor water in the hydrocarbon phases, the accumulation terms can be written in the form

$$W_\kappa = \phi \sum_{l=1}^{n_p} \xi_l S_l x_{\kappa l} = \begin{cases} \phi \sum\limits_{l=2}^{n_p} \xi_l S_l x_{\kappa l} & \kappa = 1, \ldots, n_c \\ \phi \xi_1 S_1 & \kappa = n_c + 1 \end{cases} \tag{11.3.2}$$

where ϕ is porosity, ξ_l is the molar density of phase l, S_l (saturation) is the fraction of the pore space occupied by phase l, and $x_{\kappa l}$ is the mole fraction of component κ in phase l. Here we assume that there are $n_p = 4$ phases and the phase index is in the following order: (1) aqueous phase, (2) oil phase, (3) gas phase and (4) an additional nonaqueous phase. We also assume that there are n_c hydrocarbon components and a water component (the index number is $n_c + 1$).

The flux of component κ can be represented as a sum of convective flux and dispersive flux:

$$\vec{F}_\kappa = \sum_{l=1}^{n_p} \xi_l x_{\kappa l} \vec{u}_l - \sum_{l=1}^{n_p} \phi \xi_l S_l \vec{\vec{K}}_{\kappa l} \nabla x_{\kappa l} \tag{11.3.3}$$

where $\vec{u}_l$ is the superficial velocity (flux) of phase j and $\overset{\rightrightarrows}{K}_{\kappa l}$ is the dispersion tensor. Physical dispersion in the simulator is modeled using the full dispersion tensor:

$$\overset{\rightrightarrows}{K}_{\kappa l} = \begin{bmatrix} K_{\kappa l}^{(xx)} & K_{\kappa l}^{(xy)} & K_{\kappa l}^{(xz)} \\ K_{\kappa l}^{(yx)} & K_{\kappa l}^{(yy)} & K_{\kappa l}^{(yz)} \\ K_{\kappa l}^{(zx)} & K_{\kappa l}^{(zy)} & K_{\kappa l}^{(zz)} \end{bmatrix} \tag{11.3.4}$$

Each element of $\overset{\rightrightarrows}{K}_{\kappa l}$ is a function of the flux $\vec{u}_l$ and the saturation S_l.

The relationship between the flux and the pressure gradient in each phase is described by the multiphase version of Darcy's law for fluid flow in porous media:

$$\vec{u}_l = -\overset{\rightrightarrows}{k}\lambda_{rl}(\nabla P_l - \gamma_l \nabla D) \tag{11.3.5}$$

where $\overset{\rightrightarrows}{k}$ is the absolute permeability tensor, λ_{rl} is the relative mobility, γ_l is the specific weight of phase l and D is depth. The relative mobility is defined as

$$\lambda_{rl} = \frac{k_{rl}}{\mu_l} \tag{11.3.6}$$

where k_{rl} is the relative permeability of phase l and μ_l is the viscosity.

In most cases the absolute permeability tensor $\overset{\rightrightarrows}{k}$ is assumed to be a diagonal tensor given by:

$$\overset{\rightrightarrows}{k} = \begin{bmatrix} k_x & & \\ & k_y & \\ & & k_z \end{bmatrix} \tag{11.3.7}$$

where k_x, k_y and k_z are the components of the permeability tensor in the x, y and z directions respectively. If these three values are the same, the medium is called *isotropic*, otherwise it is *anisotropic*. An anisotropic medium typically occurs under most reservoir conditions.

The source terms are determined by the well conditions. Thus

$$R_\kappa = \frac{q_\kappa}{V_b} \qquad \kappa = 1, \ldots, n_c + 1 \tag{11.3.8}$$

where V_b is the bulk volume of a grid block and q_κ is the molar flow rate of component κ which is positive for injection. For grid blocks which do not include a well, q_κ is set to zero.

Substituting (11.3.2), (11.3.3) and (11.3.8) into equation (11.3.1), one obtains the mass conservation equations in moles:

$$\frac{\partial}{\partial t}(\phi \sum_{l=1}^{n_p} \xi_l S_l x_{\kappa l}) + \vec{\nabla} \cdot \sum_{l=1}^{n_p}(\xi_l x_{ij} \vec{u}_l - \phi \xi_l S_l \vec{\vec{K}}_{\kappa l} \nabla x_{\kappa l}) - \frac{q_\kappa}{V_b} = 0$$
$$\kappa = 1, \ldots, n_c + 1 \tag{11.3.9}$$

Equation (11.3.9) is a set of coupled, nonlinear differential equations with $n_c n_p + 6n_p + 2$ variables. There exist $n_c n_p - n_c + 6n_p + 1$ other independent functional relationships among these variables (see Chang [13]). Therefore the system is solvable.

Since the solution scheme is analogous to IMPES, only the pressures are solved implicitly. In fact, each of the phase pressures is related to a reference phase pressure P_J and the capillary pressure which is a function of phase saturations and compositions (see Chang [13]). Thus only one variable, the phase pressure P_J needs to be solved implicitly.

11.4 The Pressure Equation

The pressure equation in UTCOMP is derived on the assumption that the pore volume $V_p(P)$, which is a function of the pressure P only, is filled completely by the total fluid volume $V_t(P, \vec{N})$ which is a function of pressure P and the total number of moles of each component N_κ:

$$V_t(P, \vec{N}) = V_p(P) \tag{11.4.10}$$

Differentiating both sides of equation (11.4.10) with respect to time using the chain rule gives

$$\frac{\partial V_t}{\partial P}\frac{\partial P}{\partial t}+\sum_{\kappa=1}^{n_c+1}\frac{\partial V_t}{\partial N_\kappa}\frac{\partial N_\kappa}{\partial t}=\frac{\partial V_p}{\partial P}\frac{\partial P}{\partial t} \tag{11.4.11}$$

Since the formation is assumed to be slightly compressible, the pore volume V_p is approximately a linear function of the pressure P

$$V_p=V_p^o[1+c_f(P-P^o)] \tag{11.4.12}$$

where V_p^o is the pore volume at some constant pressure P^o and c_f is a constant coefficient.

The accumulation term, in units of moles of component κ per volume V_b is given by

$$N_\kappa=V_bW_\kappa=V_b\phi\sum_{l=1}^{n_p}\xi_lS_lx_{\kappa l} \tag{11.4.13}$$

The mass conservation equation (11.3.9) can then be written in the form of the net change for component κ in moles

$$\frac{\partial(N_\kappa)}{\partial t}+V_b\vec{\nabla}\cdot\sum_{l=1}^{n_p}(\xi_lx_{\kappa l}\vec{u}_l-\phi\xi_lS_l\vec{\vec{K}}_{\kappa l}\nabla x_{\kappa l})-q_\kappa=0$$

$$\kappa=1,\ldots,n_c+1 \tag{11.4.14}$$

Substituting equations (11.4.12) and (11.4.14) into equation (11.4.11), we have

$$\frac{\partial V_t}{\partial P}\frac{\partial P}{\partial t}-V_b\sum_{\kappa=1}^{n_c+1}\bar{V}_{t\kappa}\vec{\nabla}\cdot\sum_{l=1}^{n_p}(\xi_lx_{\kappa l}\vec{u}_l-\phi\xi_lS_l\vec{\vec{K}}_{\kappa l}\nabla x_{\kappa l})$$

$$+\sum_{\kappa=1}^{n_c+1}\bar{V}_{t\kappa}q_\kappa=V_p^oc_f\frac{\partial P}{\partial t} \tag{11.4.15}$$

where

$$\bar{V}_{t\kappa}=\frac{\partial V_t}{\partial N_\kappa} \tag{11.4.16}$$

If the pressure of phase 2, the oil phase, is chosen to be solved as the reference phase pressure, $P_J = P_2$, then other pressures can be expressed by the relations

$$P_l = P_J - P_{c2l} \tag{11.4.17}$$

where P_{c2l} is the capillary pressure between phase 2 (oil phase) and phase l. P_{c2l} is assumed to be a known function of saturation.

Substituting (11.4.14) and (11.4.17) into (11.4.15), and using the multiphase fluid flow version of Darcy's law (11.3.5), we have the final expression for the pressure equation

$$\begin{aligned}
(V_p^o c_f - \frac{\partial V_t}{\partial P})\frac{\partial P}{\partial t} - V_b \sum_{\kappa=1}^{n_c+1} \bar{V}_{t\kappa} \vec{\nabla} \cdot \sum_{l=1}^{n_p} \vec{\vec{k}} \lambda_{rl} \xi_l x_{\kappa l} \nabla P & \\
= V_b \sum_{\kappa=1}^{n_c+1} \bar{V}_{t\kappa} \vec{\nabla} \cdot \sum_{l=1}^{n_p} \vec{\vec{k}} \lambda_{rl} \xi_l x_{\kappa l} (\nabla P_{c2l} - \gamma_l \nabla D) & \\
+ V_b \sum_{\kappa=1}^{n_c+1} \bar{V}_{t\kappa} \vec{\nabla} \cdot \sum_{l=1}^{n_p} \phi \xi_l S_l \vec{\vec{K}}_{\kappa l} \nabla x_{\kappa l} + \sum_{\kappa=1}^{n_c+1} \bar{V}_{t\kappa} q_\kappa &
\end{aligned} \tag{11.4.18}$$

11.5 The Finite-Difference Form of the Pressure Equation

The discretized form of the pressure equation (11.4.18) is obtained using the central difference scheme in space and the backward difference scheme in time. The transmissibilities for phase l are given by

$$(T_l)^n_{i\pm\frac{1}{2}jk} = \frac{2\Delta y_j \Delta z_k (\lambda_{rl}\xi_l)^n_{i\pm\frac{1}{2}jk}}{\dfrac{\Delta x_i}{(k_x)_{ijk}} + \dfrac{\Delta x_{i\pm1}}{(k_x)_{i\pm1jk}}} \tag{11.5.19}$$

$$(T_l)^n_{ij\pm\frac{1}{2}k} = \frac{2\Delta x_i \Delta z_k (\lambda_{rl}\xi_l)^n_{ij\pm\frac{1}{2}k}}{\dfrac{\Delta y_j}{(k_y)_{ijk}} + \dfrac{\Delta y_{j\pm1}}{(k_y)_{ij\pm1k}}} \tag{11.5.20}$$

$$(T_l)^n_{ijk\pm\frac{1}{2}} = \frac{2\Delta x_i \Delta y_j (\lambda_{rl}\xi_l)^n_{ijk\pm\frac{1}{2}}}{\dfrac{\Delta z_k}{(k_z)_{ijk}} + \dfrac{\Delta z_{k\pm 1}}{(k_z)_{ijk\pm 1}}} \tag{11.5.21}$$

We denote the central difference operator as Δ, e.g.

$$\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}} \tag{11.5.22}$$

and define

$$\begin{aligned}(\Delta A^n \Delta P)_{ijk} = {} & A^n_{i+\frac{1}{2}jk}(P_{i+1jk} - P_{ijk}) - A^n_{i-\frac{1}{2}jk}(P_{ijk} - P_{i-1jk}) \\ & + A^n_{ij+\frac{1}{2}k}(P_{ij+1k} - P_{ijk}) - A^n_{ij-\frac{1}{2}k}(P_{ijk} - P_{ij-1k}) \\ & + A^n_{ijk+\frac{1}{2}}(P_{ijk+1} - P_{ijk}) - A^n_{ijk-\frac{1}{2}}(P_{ijk} - P_{ijk-1})\end{aligned} \tag{11.5.23}$$

where the superscript n indicates the time level and the subscript ijk is the index for spatial block $V_b = V_{ijk}$ and

$$A^n_{i\pm\frac{1}{2}jk} = \Delta t \sum_{\kappa=1}^{n_c+1} (\bar{V}_{t\kappa})^n_{ijk} \sum_{l=1}^{n_p} (x_{\kappa l} T_l)^n_{i\pm\frac{1}{2}jk} \tag{11.5.24}$$

$$A^n_{ij\pm\frac{1}{2}k} = \Delta t \sum_{\kappa=1}^{n_c+1} (\bar{V}_{t\kappa})^n_{ijk} \sum_{l=1}^{n_p} (x_{\kappa l} T_l)^n_{ij\pm\frac{1}{2}k} \tag{11.5.25}$$

$$A^n_{ijk\pm\frac{1}{2}} = \Delta t \sum_{\kappa=1}^{n_c+1} (\bar{V}_{t\kappa})^n_{ijk} \sum_{l=1}^{n_p} (x_{\kappa l} T_l)^n_{ijk\pm\frac{1}{2}} \tag{11.5.26}$$

The finite difference form of equation (11.4.18) can be written in the form

$$\begin{aligned}(V_p^o c_f - \frac{\partial V_t}{\partial P})^n_{ijk} P^{n+1}_{ijk} - (\Delta A^n \Delta P^{n+1})_{ijk} = {} & (V_t - V_p)^n_{ijk} \\ & + (V_p^o c_f - \frac{\partial V_t}{\partial P})^n_{ijk} P^n_{ijk} + \Delta t \sum_{\kappa=1}^{n_c+1} (\bar{V}_{t\kappa})^n_{ijk} q_\kappa \\ & + \Delta t (B^n_c - B^n_g + B^n_d)\end{aligned} \tag{11.5.27}$$

where B_c^n, B_g^n and B_d^n are the collection of terms corresponding to capillary pressure, the gravity and the physical dispersion respectively. The term related to capillary pressure is given by

$$B_c^n = \sum_{\kappa=1}^{n_c+1} (\bar{V}_{t\kappa})_{ijk}^n \sum_{l=1}^{n_p} (\Delta(x_{\kappa l}T_l)^n \Delta P_{c2l}^n)_{ijk} \tag{11.5.28}$$

The gravity term is given by

$$B_g^n = \sum_{\kappa=1}^{n_c+1} (\bar{V}_{t\kappa})_{ijk}^n \sum_{l=1}^{n_p} (\Delta(x_{\kappa l}T_l\gamma_l)^n \Delta D)_{ijk} \tag{11.5.29}$$

where

$$(\gamma_l)_{i\pm\frac{1}{2}jk}^n = \frac{(V_p\gamma_l)_{ijk}^n + (V_p\gamma_l)_{i\pm1jk}^n}{(V_p)_{ijk}^n + (V_p)_{i\pm1jk}^n} \tag{11.5.30}$$

$(\gamma_l)_{ij\pm\frac{1}{2}k}^n$ and $(\gamma_l)_{ijk\pm\frac{1}{2}}^n$ are expressed similarly but with differencing being carried out in the y and z directions respectively. The contribution due to physical dispersion is given by

$$B_d^n = (V_p)_{ijk} \sum_{\kappa=1}^{n_c+1} (\bar{V}_{t\kappa})_{ijk}^n \sum_{l=1}^{n_p} (\Delta_x(J_{\kappa l}^n)_{ijk} + \Delta_y(J_{\kappa l}^n)_{ijk} + \Delta_z(J_{\kappa l}^n)_{ijk}) \tag{11.5.31}$$

where

$$\begin{aligned}
\Delta_x(J_{\kappa l}^n)_{ijk} &= (J_{\kappa l}^n)_{i+\frac{1}{2}jk} - (J_{\kappa l}^n)_{i-\frac{1}{2}jk} \\
&= \frac{1}{\Delta x_i}\left[2(\xi_l\phi S_l K_{\kappa l}^{(xx)})_{i+\frac{1}{2}jk}^n \frac{(x_{\kappa l})_{i+1jk}^n - (x_{\kappa l})_{ijk}^n}{\Delta x_i + \Delta x_{i+1}}\right. \\
&\quad +(\xi_l\phi S_l K_{\kappa l}^{(xy)})_{i+\frac{1}{2}jk}^n \frac{(x_{\kappa l})_{i+\frac{1}{2}j+1k}^n - (x_{\kappa l})_{i+\frac{1}{2}j-1k}^n}{\Delta y_j + 0.5(\Delta y_{j+1} + \Delta y_{j-1})} \\
&\quad \left. +(\xi_l\phi S_l K_{\kappa l}^{(xz)})_{i+\frac{1}{2}jk}^n \frac{(x_{\kappa l})_{i+\frac{1}{2}jk+1}^n - (x_{\kappa l})_{i+\frac{1}{2}jk-1}^n}{\Delta z_k + 0.5(\Delta z_{k+1} + \Delta z_{k-1})}\right] \\
&- \frac{1}{\Delta x_i}\left[2(\xi_l\phi S_l K_{\kappa l}^{(xx)})_{i-\frac{1}{2}jk}^n \frac{(x_{\kappa l})_{ijk}^n - (x_{\kappa l})_{i-1jk}^n}{\Delta x_i + \Delta x_{i-1}}\right.
\end{aligned}$$

$$
\begin{aligned}
&+(\xi_l \phi S_l K_{\kappa l}^{(xy)})_{i-\frac{1}{2}jk}^{n} \frac{(x_{\kappa l})_{i-\frac{1}{2}j+1k}^{n} - (x_{\kappa l})_{i-\frac{1}{2}j-1k}^{n}}{\Delta y_j + 0.5(\Delta y_{j+1} + \Delta y_{j-1})} \\
&+(\xi_l \phi S_l K_{\kappa l}^{(xz)})_{i-\frac{1}{2}jk}^{n} \frac{(x_{\kappa l})_{i-\frac{1}{2}jk+1}^{n} - (x_{\kappa l})_{i-\frac{1}{2}jk-1}^{n}}{\Delta z_k + 0.5(\Delta z_{k+1} + \Delta z_{k-1})} \Bigg] ,(11.5.32)
\end{aligned}
$$

$\Delta_y(J_{\kappa l}^n)_{ijk}$ and $\Delta_z(J_{\kappa l}^n)_{ijk}$ can be expressed in a similar way but with the differencing being carried out in the y and z directions respectively.

A "volumetric error" term, $(V_t - V_p)$, is added in equation (11.5.27) to account for the discrepancy in pore volume and fluid volume at the previous time step. It reflects the fact that the pressure obtained at the previous time step may not have been computed exactly.

The coefficient matrix of the resulting linear system is a nonsymmetric, seven-diagonal matrix.

Chapter 12

Numerical Results

12.1 Introduction

In this chapter we present the numerical results of multigrid methods for solving both model problems and real reservoir simulation problems on parallel machines. The problems may be strongly anisotropic. The coefficients are often nonconstant with jumps across some inner boundaries.

The motivation of our experiments is twofold. First we want to examine the performance of the multigrid methods in solving real problems (here we use reservoir simulation problems). Second we want to test the performance of the multigrid methods on currently available parallel machines.

12.2 The Multigrid Algorithm

We are concerned with a multigrid algorithm which should be robust for solving a relatively broad range of problems in the sense that the algorithm should obtain an expected convergence rate without special treatment. The multigrid algorithm we used in our numerical experiments is a semicoarsening multigrid algorithm (SCMG).

To handle an anisotropic problem, plane smoothing iterations in the coarsening direction are needed. The plane smoothing iterations are required to solve many 2D matrix problems. If a 2D direct method or an iterative method is used to solve these 2D matrix problems, the effect of the whole 3D multigrid method may not be attractive [5]. One way to overcome this difficulty is to use a 2D multigrid method to solve these 2D problems [24]. Since the solution of the 2D problems is part of the smoothing iterations, they need not to be solved exactly. In practice, one multigrid V-cycle is adequate for solving the 2D problems. Thus the 3D multigrid method can be very effective in solving anisotropic problems.

To handle problems with discontinuous coefficients, matrix dependent interpolation operators and restriction operators should be used. The interpolation and restriction matrices we used are obtained using the Schur complement approximation method. This is because the reservoir simulation problems we used are nonsymmetric.

The coarse grid matrices are constructed using (10.3.14). Since the interpolation operation and the restriction operation only involve three points in the coarsening direction, the coarse grid operators have 15-diagonal structure.

For smoothing iterations, the *zebra* line Jacobi method is used in 2D cases. In 3D cases, the *zebra* plane Jacobi method is used.

12.3 Implementation

12.3.1 Coarsening Direction

Although the choice of the coarsening direction does not affect the performance of the multigrid algorithms on a serial scalar machine, this may not be the case on a parallel machine. Without loss of generality, we assume that $n_y > n_x$. If we choose the y-direction as the coarsening direction, the parallel smoothing work of each V-cycle will be

$$W(x)\log(n_y) \tag{12.3.1}$$

where $W(x)$ is the number of parallel operations for an x-line of nodes per level.

In a 2D multigrid algorithm for solving anisotropic problems, a line smoothing iteration method is often used. In this case, $W(x) \approx d(n_x)$, where d is some constant. It is obvious that $dn_x \log(n_y) < dn_y \log(n_x)$. Therefore, the coordinate direction in which the number of points is largest should be chosen as the coarsening direction for a semicoarsening multigrid method on parallel machines. Figure 12.1 illustrates the semicoarsening grids in three levels.

12.3.2 Data Partition

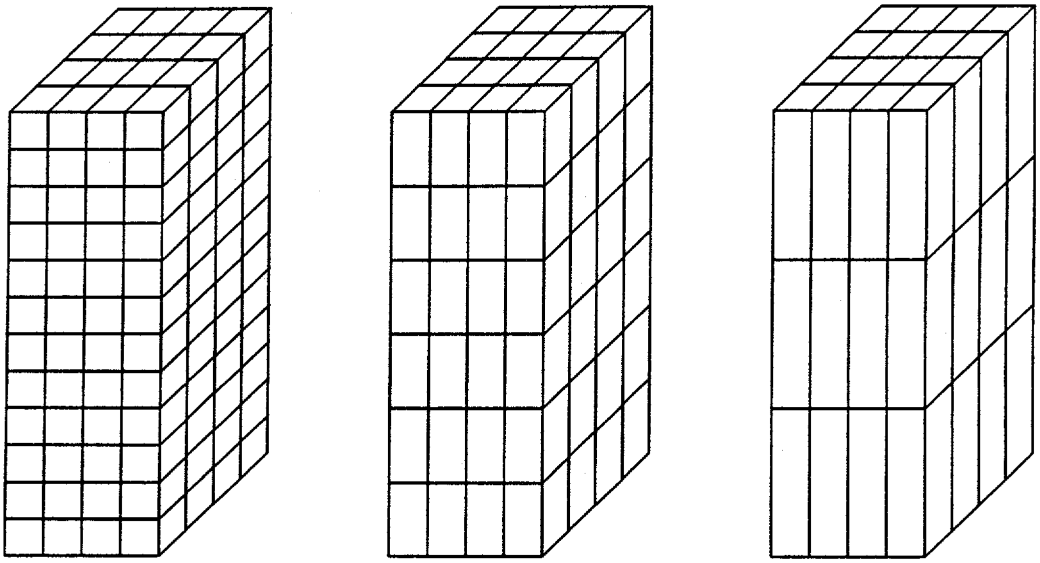
In our current code, we use a one-dimensional parallel scheme. Each of the grids is partitioned into subgrids in one direction. Each of the processor nodes holds a subproblem defined on one of the subgrids. One of the reasons for choosing this partition is to reduce communication costs since each of the nodes has at most two neighbors. For a parallel system with high start-up communication costs, having fewer neighbors can reduce the total communication costs.

The parallel direction is chosen as the same as the coarsening direction. Since the number of points in this coordinate direction is the largest, the number of levels which can utilize all of the processor nodes will also be largest. Thus the algorithm will have the maximum parallel efficiency.

As mentioned earlier, in a 3D multigrid algorithm with plane smoothing iterations, the 2D subproblems can be effectively solved by using 2D multigrid methods. By using the one-direction parallel scheme, the 2D multigrid algorithm can be carried out in one node. Thus parallel computing can be done on a large granularity scale.

12.3.3 Below C-Level

One of the major parallel implementation concerns of the semicoarsening multigrid method is the "below C-level" problem (see Briggs et al. [11]). Below a certain level, the number of the partitions is smaller than the number of processors. The communication between the nodes becomes complicated because of two reasons: some nodes are idle and neighbor partitions do not reside on the physical neighbor nodes.



Level L Level $L-1$ Level $L-2$

Figure 12.1: Semicoarsening Grids in Three Levels

There are several ways to overcome this difficulty. One way to do that is to avoid this situation by not going below the C-level, which is called the "U-cycle" in Briggs et al. [11]. However, if the number of nodes is quite large, this approach will degrade performance.

Another way to do this is to have one or a group of nodes obtain all data from the others, complete the below C-level part of a *V-cycle*, and then pass the data to the other nodes. In a semi-coarsening case, the global copies of the current problem are distributed to all nodes in the semi-coarsening direction. This approach does not scale well. For a moderate number of nodes, however, parallel performance is good (Smith [60]).

A more natural way is to have all processors do their portion of the job. Those nodes that are allocated no unknowns are set idle and then reawakened upon returning to high levels. This is the approach used by both Hempel and Schuller [37] and Briggs et al. [11]. In a semi-coarsening case, the implementation can be simpler.

Since we do coarsening in only one direction, each processor has at most two neighbors. Each processor keeps a record of its neighbors' identification numbers for all levels and its own "sleep-level." Below its "sleep-level" the processor goes to sleep and above this level it awakens. Communication includes the inner boundary data exchange and global partial summary data. If a red/black smoothing scheme is used, another flag is needed for each processor to indicate whether the inner boundary is red or black.

12.3.4 Storage Space

In the SCMG algorithm, the matrices at each level, such as the restriction matrices R_l, the interpolation matrices P_l, and the problem matrices A_l, should be constructed and saved at the so-called preprocessing stage.

Since the coarse grid matrices A_l for 3D problems generally have 15 nonzero diagonals, the space for storing all the A_l should be larger than

$7N + 15N$. For matrices P_l and R_l, only two diagonals need to be stored, so the storage space for these matrices is $4N$.

If the 2D SCMG algorithm is used for plane smoothing iterations, we have to consider storing the 2D matrices at coarse levels. There are two strategies for generating the 2D coarse grid matrices. We can compute all of the 2D matrices for all levels and save them for later use. This approach requires a very large amount of storage space but can save CPU time. We can also calculate the 2D matrices whenever the 2D SCMG routine is called. This approach requires a very small amount of storage space but increases the computation time.

12.4 Parallel Machines

12.4.1 Intel iPSC/860

The Intel iPSC/860 system is a MIMD parallel machine. The iPSC/860 system consists of up to 128 compute nodes, each with an Intel i860 processor and up to 64 MByte of memory. Each processor is capable of performing up to 80 million single-precision or 60 million double-precision floating point operations per second (MFLOPS).

The hardware for message passing that resides on each node board links the processors together in a physical hypercube configuration. Each message initiating from a node takes about 65 microseconds of CPU time to establish its path through the cube. The message is then passed along at the peak hardware bandwidth of 2.8 MByte/sec. Because each node has a direct connect routing module which is separate from the processor, message passing through intermediate nodes do not interrupt those processors, so there is no performance penalty for communicating between nonadjacent nodes.

To run an executable program, the user must allocate a cube consisting of all or part of the nodes in the system. If the user successfully obtains a cube, he is the owner the cube and no one else can obtain the processors in the cube until the user releases the cube.

12.4.2 Connection Machine 5

A CM-5 system is a massively parallel MIMD computer. It may contain thousands of computational processing nodes and one or more control processors. Every node is connected to two scalable interprocessor communication networks, the Data Network and the Control Network. Each processing node is a general-purpose computer with its own local memory. Nodes can execute the same (SIMD-style) instruction or independent (MIMD-style) instructions. A whole CM-5 system can be divided into groups, known as partitions, with a separate control processor, known as a partition manager. Each partition is viewed as an independent machine by users.

The operating system on a CM-5 uses a time-sharing scheme. Each user gets the whole partition during his time slice. The CM-5 timer functions take this into account automatically.

There are two parallel programming models supported by the CM-5, the data parallel model and the message passing model.

To use the data parallel model, codes should be written in a high-level parallel language (e.g., CM Fortran which is similar to Fortran 90, C*, or *Lisp). Some of the most important features of CM Fortran are the array operations. In these advanced parallel languages, an array is treated as a "parallel variable" and can be used as a single operand of an arithmetic operator or a single argument of an intrinsic function. By using this model, users can concentrate on the logical design of their applications and let the advanced compilers, assemblers, and other system software deal with many details associated with parallel processing, such as layout across nodes, and synchronization of operations.

Although many underlying detail problems related to parallelism no longer need to be taken care of in the data parallel model, the array shape conversion problem must be dealt with. The current version of CM Fortran does not allow passing a front-end array (a Fortran 77 array) to a subroutine that expects a CM array, or passing a CM array to a subroutine that expects a front-end array. It also does not allow resizing or reshaping a CM array across subroutine boundaries.

In our multigrid codes, one section of a one-dimensional array in the calling routine is interpreted as a multidimensional array in the called subroutine. This is not trivial in CM Fortran although it can be easily done in Fortran 77.

One way to solve this problem is to declare the storage space as a one-dimensional CM array in the calling routine and to pass the proper section of the array to the called subroutine. Inside the called subroutine, a intrinsic function RESHAPE is used to create a multidimensional CM array from elements of the one-dimensional array.

The other way is to declare the space as a one-dimensional FE (Front-End) array in the calling routine and to pass the relavant part of it to the subroutine as in Fortran 77. Inside the called subroutine, two transfer subroutines, CMF_FE_ARRAY_TO_CM and CMF_FE_ARRAY_FROM_CM, are used to transfer the multidimensional FE array to and from the multidimensional CM array. We have chosen this approach in our codes for convenience and efficiency.

In the message passing model, users take care of communication at the procedure level. Users must specify explicitly the locations where communication is involved. This model gives users great flexibility. When operation of arrays includes procedure calls, the data parallel model usually cannot help and the message passing model has to be used.

On the CM-5, the message passing model is supported by the CM message-passing library (CMMD). CMMD provides common message passing subroutines, global operation subroutines, parallel input/output modes and node timing functions. The CMMD timers measure three values: busy time (cpu time), idle time, and elapsed time with microsecond precision.

12.5 Parallel Virtual Machine

PVM is a software package developed jointly at Oak Ridge National Laboratory and The University of Tennessee, which allows concurrent computing on heterogeneous networked computers via sockets. Users run an application program using PVM just like on a real parallel machine. PVM takes care of data format transformations between different types of machines. The communication subroutines are provided in a library (libpvm3.a).

Although a group of different types of machines can be used under PVM, the execution time is limited by the slowest one if each of the machines accepts the same amount of work. In this case, the work load should be carefully distributed among the networked machines based on their processing speeds to obtain the maximum speedup. In our case, we used 16 SUN4 workstations which are networked with Ethernet-10 with a speed 10 Mb/sec.

The performance of PVM depends on the performance of the local computer system, especially on the speed and reliability of the local network. In order to reduce communication contention and minimize network traffic, each node keeps the whole set of data and only exchanges the updated inner boundary data.

The nodes in a PVM system are real computers, and usually have more processing power than the corresponding processor nodes of a real parallel machine. However, the communication speed might be slow compared to those of real parallel machines since the speed of the networks is usually slower than the data channels in a real parallel computer. Subtasks of an application program should, therefore, have a moderately large level of granularity in order to get a reasonable speedup result.

12.6 Numerical Results

12.6.1 Anisotropic Problems

The first problem is the 2D anisotropic problem defined by (2.3.8) with the finest grid size 100 × 100.

The second problem is a 3D anisotropic problem defined by

$$\begin{cases} -\alpha u_{xx}(x,y,z) - \beta u_{yy}(x,y,z) - \gamma u_{zz}(x,y,z) = 1.0 \\ \qquad\qquad (x,y,z) \in \Omega = (0,1) \times (0,1) \times (0,1) \\ u(x,y,z) = 0 \qquad \text{on the boundary } \partial\Omega \end{cases} \tag{12.6.2}$$

with the finest grid size 20 × 20 × 20.

The initial guess $u_L^{(0)}$ is set to zero. The average residual reduction factor is obtained by averaging the residual reduction factors over 5 *V-cycles*

$$\rho_A = \left(\frac{\| r_L^{(5)} \|}{\| b_L \|} \right)^{\frac{1}{5}} \tag{12.6.3}$$

where $r_L^{(n)} = b_L - A_L u_L^{(n)}$. The convergence results for the 2D and the 3D problems are presented in Tables 12.1 and 12.2, respectively. In these two tables, the average convergence factors are listed vs. different anisotropic coefficients situations. For both problems, the convergence factors for the anisotropic cases are smaller than those for the isotropic cases.

A standard measure of the speedup of an algorithm is defined by

$$S^{(N)} = \frac{T^{(N)}}{T^{(1)}} \tag{12.6.4}$$

where $T^{(N)}$ is the wall clock time for solving a problem on N computational processors and $T^{(1)}$ is the wall clock time for solving the same problem with the best serial code on one computational processor. In our experiments, $T^{(1)}$ is

Table 12.1: Convergence Factor of SCMG in 2D Anisotropic Cases (NX=100, NY=100)

Coefficients α	r_0	r_5	Average Convergence Factor
1000	7.166×10^{-1}	6.777×10^{-12}	6.240×10^{-3}
100	1.210	4.279×10^{-7}	5.125×10^{-2}
10	1.348	1.513×10^{-5}	1.023×10^{-1}
1	1.338	1.798×10^{-5}	1.061×10^{-1}
0.1	1.398	4.454×10^{-6}	7.955×10^{-2}
0.01	1.400	1.589×10^{-6}	6.471×10^{-2}
0.001	1.400	1.885×10^{-7}	4.225×10^{-2}

approximated by the time for solving the problem with the same parallel code on one processor. A measure of the efficiency is defined by

$$E^{(N)} = \frac{S^{(N)}}{N}. \tag{12.6.5}$$

The efficiency is used to measure processor ultilization.

Tables 12.3, 12.4 and 12.5 list the speedup and efficiency of the SCMG algorithm for the 2D and 3D anisotropic problems defined in (2.3.8) and (12.6.2) on the CM-5 and iPSC/860 parallel systems and the networked SUN4 workstations under PVM, respectively. The numerical results are obtained for the problems with different sizes of grids and on different numbers of processors. Here we use T_{comp}, T_{comm} and T_{sol} to represent the time for computation, communication and solution, respectively. In a multi-user system such as the CM-5, we use the CPU time instead of the wall clock time because the wall clock time depends on the number of user processes running on the system.

Table 12.2: Convergence Factor of SCMG in 3D Anisotropic Cases (NX=20, NY=20, NZ=20)

Coefficients	r_0	r_5	Average
α, β, γ			Convergence Factor
1, 1, 1	1.225	1.967×10^{-6}	6.936×10^{-2}
100, 1, 1	0.4224	2.542×10^{-13}	3.597×10^{-3}
1, 100, 1	0.4225	1.058×10^{-12}	4.783×10^{-3}
1, 1, 100	1.341	1.233×10^{-7}	3.915×10^{-2}
100, 100, 1	0.2470	4.463×10^{-13}	4.481×10^{-3}
1, 100, 100	1.281	1.577×10^{-6}	6.577×10^{-2}
100, 1, 100	1.281	1.575×10^{-6}	6.577×10^{-2}
10000, 100, 1	5.291^{-3}	2.475×10^{-13}	8.590×10^{-3}
10000, 1, 100	0.4325	2.486×10^{-13}	3.564×10^{-3}
1, 10000, 100	0.4323	3.064×10^{-13}	3.716×10^{-3}
100, 10000, 1	0.7479	3.766×10^{-13}	3.470×10^{-3}
100, 1, 10000	1.341	8.755×10^{-8}	3.656×10^{-2}
1, 100, 10000	1.342	8.755×10^{-8}	3.656×10^{-2}

Table 12.3: Speedup and Efficiency of SCMG on the CM-5

Size	Nodes	T_{comp}	T_{comm}	T_{sol}	Speedup	Efficiency
8*8*128	1	21.1	0	21.6	1	1
	2	10.7	0.19	11.1	1.95	0.98
	4	5.43	0.39	5.94	3.64	0.91
	8	2.84	0.61	3.55	6.08	0.76
	16	1.57	0.98	2.67	8.09	0.51
	32	0.98	1.69	2.78	7.77	0.24
4*4*128	1	7.67	0	7.74	1	1
	2	3.86	0.13	4.05	1.91	0.96
	4	1.97	0.25	2.28	3.39	0.85
	8	1.04	0.45	1.54	5.03	0.63
	16	0.59	0.70	1.33	5.81	0.36
	32	0.37	1.10	1.52	5.09	0.16
8*8*32	1	5.26	0	5.31	1	1
	2	2.66	0.16	2.86	1.86	0.93
	4	1.39	0.29	1.73	3.07	0.77
	8	0.82	0.51	1.36	3.90	0.49
	16	0.56	0.69	1.29	4.12	0.26
	32	0.47	1.10	1.66	3.20	0.10
16*512	1	1.33	0	1.46	1	1
	2	0.62	0.12	0.85	1.72	0.86
	4	0.31	0.24	0.65	2.25	0.56
	8	0.17	0.43	0.69	2.12	0.27
	16	0.10	0.86	1.06	1.38	0.09
	32	0.07	1.56	1.73	0.84	0.03

Table 12.4: Speedup and Efficiency of SCMG on the iPSC/860

Size	Nodes	T_{comp}	T_{comm}	T_{sol}	Speedup	Efficiency
4*4*128	1	5.51	0	5.77	1	1
	2	2.87	0.07	3.00	1.92	0.96
	4	1.48	0.13	1.64	3.52	0.88
	8	0.76	0.18	0.96	6.01	0.75
	16	0.39	0.24	0.65	8.88	0.56
	32	0.19	0.30	0.51	11.31	0.35
8*8*32	1	3.32	0	3.37	1	1
	2	1.74	0.05	1.81	1.86	0.93
	4	0.98	0.09	1.10	3.06	0.77
	8	0.61	0.14	0.78	4.32	0.54
	16	0.43	0.18	0.65	5.18	0.32
	32	0.40	0.22	0.66	5.11	0.16
16*512	1	0.411	0	0.420	1	1
	2	0.223	0.071	0.301	1.40	0.7
	4	0.122	0.101	0.230	1.83	0.46
	8	0.071	0.117	0.192	2.19	0.27
	16	0.044	0.129	0.176	2.39	0.15
	32	0.025	0.138	0.167	2.51	0.08

Table 12.5: Speedup and Efficiency of SCMG on a Cluster of SUN4 Using PVM

Size	Nodes	T_{comp}	Speedup	Efficiency
8*8*128	1	18.4	1	1
	2	9.56	1.92	0.96
	4	5.17	3.56	0.89
4*4*128	1	7.42	1	1
	2	3.96	1.87	0.94
	4	2.11	3.52	0.88
	8	1.18	6.29	0.79
	16	0.81	9.16	0.57
8*8*32	1	4.13	1	1
	2	2.32	1.78	0.89
	4	1.38	2.99	0.75
	8	0.88	4.69	0.59
	16	0.65	6.35	0.40
16*512	1	1.21	1	1
	2	0.77	1.57	0.79
	4	0.69	1.75	0.44

The timing results on the CM-5 system are obtained using the timing subroutines in the CMMD library. These timers can be used to measure busy time and idle time on each node with microsecond precision. Timing on the iPSC/860 is performed using the dclock() function which can measure the cpu time with 100 ns precision. Timers on the networked workstations under PVM depend on the timing utilities available on the workstations. Here we use the UNIX system call getrusage() on the SUN4 workstations. In Table 12.5, only computation times are listed because low-speed communication channels and repeatedly resending lost messages makes the communication time very large.

The computational time decreases as the number of node processes increases. The communication time increases as the number of node processes increases. If the increase in the communication time is larger than the decrease in the computation time, the solution time will increase.

Speedup depends on the problem size, domain shape and the ratio of the computational work to the communication work. Speedup becomes larger when the problem size is larger. This is because the cost of the serial portion of the process becomes negligible compared to the parallel portion. The domain shape affects the ratio of the computational work to the communication work. The longer the shape is in the parallel direction the larger the speedup that can be obtained.

12.6.2 Reservoir Simulation Problems

We have implemented a semicoarsening multigrid (SCMG) method and have used the code in the multicomponent, multiphase miscible flooding simulator UTCOMP as a solver for the numerical solution of the linear systems of equations arising from the discretization of the governing pressure partial differential equations. We give some numerical results which show that the SCMG solver is very efficient and stable.

The first three cases are two-dimensional carbon dioxide flooding simulations with multiphase flow and heterogeneous field-scale conditions. In these cases, the effects of gravity, physical dispersion, capillary pressure, phase behavior, and heterogeneity are combined and simulated for carbon dioxide flooding on a field scale using stochastic permeability fields. A detailed description of these cases can be found in [48]. The grid size of these three cases are 80×10, 80×20 and 80×80 corresponding to the cases M2DCLL671, M2DCLL672 and M2DCLL679 in [48] respectively. The simulations contain up to three phases and three components.

We run these three cases on a CRAY Y-MP supercomputer. The resulting linear systems are highly nonsymmetric. For comparison, we have also run the cases with some other numerical solvers, such as the biconjugate gradient method (BCG), the biconjugate gradient square method (BCGS) [40], the ORTHOMIN method (OMIN) [72] and the banded Gaussian elimination method (DIR). The incomplete LU decomposition (ILU) or the modified incomplete LU decomposition (MILU) [53] is used as a preconditioner for the CG-like methods. For the modified incomplete LU decomposition preconditioner, the modification factor ω is set to 0.9. (The discussion of the modification factor ω can be found in [56].) The simulation is carried out for 200 simulation days for the first two cases and 50 simulation days for the third case. The timing results are listed in Tables 12.6, 12.7 and 12.8 where $T_{\rm sol}$ is the CPU time for solvers, $T_{\rm tot}$ is CPU time for the whole simulation runs and $R_{\rm sol}$ is the ratio of $T_{\rm sol}$ based on the SCMG method.

For all three cases, the performance of the multigrid method is very good. It is about 17 times faster than the OMIN(ILU) method which is the second fastest method for case 3. For case 1, the size of the linear system is the smallest one (80×10) in our experiments and the banded direct solver is faster than all the other CG-like methods except the SCMG method which, by contrast, is still about 3 times faster than the direct method.

In our experiments, we also consider two three-dimensional waterflooding simulation cases which use the reservoir and fluid properties of the Monahans Clearfork reservoir, located in West Texas, operated by Shell Oil Co [47]. Up to three nonaqueous phases and six components are considered in

Table 12.6: Timing Results for Case 1 (80 × 10)

Solver	$T_{\text{sol}}(sec.)$	$T_{\text{tot}}(sec.)$	R_{sol}
SCMG	10	213	1
OMIN(MILU)	127	332	13
BCGS(MILU)	121	324	12
BCG(MILU)	132	336	13
DIR	27	231	3

Table 12.7: Timing Results for Case 2 (80 × 20)

Solver	$T_{\text{sol}}(sec.)$	$T_{\text{tot}}(sec.)$	R_{sol}
SCMG	19	556	1
OMIN(MILU)	406	951	21
BCGS(MILU)	390	927	20
BCG(MILU)	407	949	21
DIR	143	685	8

Table 12.8: Timing Results for Case 3 (80 × 80)

Solver	$T_{\text{sol}}(sec.)$	$T_{\text{tot}}(sec.)$	R_{sol}
SCMG	113	928	1
OMIN(ILU)	1922	2734	17
BCGS(ILU)	2406	3237	21
BCG(ILU)	3197	4030	28
DIR	7036	7842	62

these two simulation cases. We refer to these two cases as Cases 4 and 5 corresponding to Runs SIM21HW1C and SIM46DW1 in [48], respectively. In Case 4, a three-dimensional unconditioned stochastic permeability field is generated and a grid of $10 \times 10 \times 8$ is used for a real range of $1980 \times 1980 \times 20ft^3$. In case 5, a three-dimensional stochastic permeability field conditioned with core data are used with a grid of $10 \times 10 \times 16$.

We run these two cases on a DECalpha workstation. The simulations are carried out until the number of pore volumes injected is equal to 0.1. All these runs are successful. For most time steps it takes only one or two cycles for convergence of the SCMG method. For these two three-dimensional cases, the CG-like iterative methods are faster than the SCMG method. One reason is that the current implementation of SCMG needs an initializing process which takes a large amount of time if only a few multigrid cycles are performed. Another reason is that the numbers of grid points in the coarsening direction used in these two cases are too small to obtain usual multigrid efficiency.

Chapter 13

Conclusions

13.1 Review of Dissertation

In this dissertation we studied multigrid methods for the numerical solution of elliptic partial differential equations. The primary focus of our study was on parallel multigrid methods and the application of multigrid methods to reservoir simulation.

In Chapter 2, we described the model problems in 1D and 2D and the corresponding linear systems which will be used in later chapters. In Chapter 3, we briefly discussed some basic iterative methods, including the Jacobi method. We also discussed polynomial acceleration procedures such as Chebyshev acceleration and conjugate gradient acceleration.

In Chapters 4 and 5, we discussed the standard multigrid methods. We gave a convergence analysis of the standard multigrid methods for a class of model problems in 1D and 2D using both standard Fourier analysis and a multicolor Fourier analysis. The new multicolor Fourier analysis is equivalent to the standard Fourier analysis since there is a similarity transformation relationship between the two bases. However, the multicolor Fourier analysis can be more conveniently used in the analysis of the standard multigrid methods using red/black smoothing iteration methods.

In Chapters 6 to 9, we considered three types of multiple coarse grid methods (MCG). In the multiple coarse grid multigrid methods (MCGMG) more than one coarse grid is used and the interpolation and restriction operators on each of the coarse grids on one level are the same. In the frequency decomposition multigrid methods (FDMG) more than one coarse grid is used but the interpolation and restriction operators on each of the coarse grids on one level are different. In the parallel multigrid methods (PMG) one averages all the coarse grid corrections on one level.

In Chapter 6, we considered an extended problem corresponding to a given problem with Dirichlet boundary condition. A similar discussion can be applied to other boundary conditions such as the Neumann-type boundary condition. The MCG methods can be more effectively applied to the extended system. In the MCG procedure to solve the extended system, we used a purification process to obtain Moore-Penrose solutions of singular systems.

In Chapters 7 and 8, we described and analyzed three types of MCG methods including MCGMG, FDMG and PMG in one and two dimensions. Multicolor Fourier analysis was used in our analysis. We derived convergence factors of the two-level procedures for MCG methods with various smoothing iteration methods including the damped Jacobi method and the red/black SOR method. We showed that with an MCGMG procedure or a PMG procedure with a certain type of smoothing iteration (e.g. the damped Jacobi method) the "aliasing error" caused by the coarse grid correction process on each of the coarse grids was eliminated and thus the convergence was improved. The analysis showed that using the red/black SOR smoothing iteration method led to a better MCGMG convergence rate than using the damped Jacobi method. We showed that the convergence factors of the FDMG method without any smoothing iterations were bounded from above by 1/3. The coarse-grid correction of the FDMG method is in fact equivalent to a block Jacobi iteration applied to a similarity transformation of the given system.

In Chapter 9, we considered a variant of PMG methods using the semicoarsening and line smoothing techniques to solve anisotropic problems. A multilevel convergence analysis of the semicoarsening PMG method (SCPMG) for an anisotropic problem was given. The results of the analysis show that the multilevel convergence factor of the SCPMG method with one line Jacobi smoothing iteration is bounded from above by 0.5 for the anisotropic model problem with the coefficient ratio being from 10^{-5} to 10^{5}. If the modified line Jacobi smoothing iteration proposed by Tuminaro [64] is used then the convergence factor can be bounded by 0.2. The bound of the convergence factor can be reduced further to 0.11 if the red/black SOR smoothing iteration is used.

In Chapters 10 to 12, we considered the application of multigrid methods to petroleum reservoir simulation. We developed a multigrid code using the semicoarsening and the line (plane) smoothing techniques for UTCOMP, a three-dimensional, multiphase, multicomponent, compositional reservoir simulator developed at The University of Texas at Austin. The governing equation in the reservoir simulator is an anisotropic differential equation which may have discontinuous coefficients. The matrix problem arising from the discretization of the governing equation is nonsymmetric. Special matrix-dependent interpolation and restriction operators were used to handle the discontinuous coefficients. The numerical results showed that the multigrid procedure we used were competitive with the Gaussian elimination method and with standard iterative methods. The multigrid algorithm has also been implemented on a variety of parallel systems such as the CM-5, iPSC/860 and networked workstations under PVM. From the results of the numerical experiments, we observed that the speedup depends on the size of the problem to be solved and the computing environment.

13.2 Summary of Contributions

In this dissertation we introduced a new multicolor Fourier analysis, described and analyzed the multiple coarse grid methods including a new one based on the use of semicoarsening and line smoothing techniques, and applied multigrid methods to reservoir simulation.

In our analysis of two-level multigrid methods we used a multicolor Fourier analysis based on the partitioning of coarse grids. It can be conveniently used in cases where more than one coarse grid is used and/or a multicolor smoothing iteration is used. The connection between standard Fourier analysis and multicolor Fourier analysis is derived.

We studied the coarse grid methods (MCG), where at each coarse grid level more than one coarse grid is used to improve convergence performance. Three types of MCG methods were considered, including multiple coarse grid multigrid methods (MCGMG), frequency decomposition multigrid methods (FDMG) and parallel multigrid methods (PMG).

We constructed an extended system corresponding to a given problem with Dirichlet boundary conditions. The extended system is periodic and thus can be conveniently handled by MCG methods. We used a purification process to obtain the Moore–Penrose solution of the singular systems which were encountered.

We considered a new variant of PMG methods using semicoarsening and line smoothing techniques to handle anisotropic problems. A multilevel convergence analysis was carried out.

We applied multigrid methods to petroleum reservoir simulation. We developed a multigrid code using semicoarsening and line (plane) smoothing techniques for solving the governing pressure equation of a three-dimensional reservoir simulator. We used special matrix-dependent interpolation and restriction operators in the code which can be used for solving the pressure equation with discontinuous coefficients.

We systematically performed numerical experiments of the multigrid method on a variety of parallel systems such as the CM-5, iPSC/860 and networked workstations under PVM.

13.3 Future Research

The performance of an MCG method largely depends on the choice of the interpolation and restriction operators. Since the operators can be chosen differently on different coarse grids, there are a large number of possible choices. A further study should be carried out on the choice of operators for the MCG methods.

The analysis of the MCG methods should be extended to three-dimensional cases to see if the performance declines. A broader class of problems should be considered. Such problems include elliptic problems with mixed boundary conditions. The use of MCG methods for solving equations with discontinuous coefficients should also be considered.

Semicoarsening techniques can be used with the FDMG methods. A semicoarsening FDMG method has fewer coarse grids and therefore the algorithm has a simpler structure. With the semicoarsening scheme, the interpolation and restriction operators are also much simpler. In a higher dimensional case, this advantage is greater.

In reservoir simulation, the solution can be very smooth in some regions and oscillatory in others. For such cases it might be more efficient to use a grid with more resolution in some regions. Using local mesh refinement techniques, the efficiency of a multigrid method can be further improved.

The governing pressure equation in UTCOMP is a nonlinear equation. It may be more efficient to apply the multigrid method directly to the original nonlinear problem instead of to the linearized systems.

For time dependent problems where the low-frequency modes in the solution on the finest grid do not change very much with time, the coarse grid corrections do not have to be sent back to the finest level at each time step. Instead, one can run a simulation by solving the coarse grid system for several time steps and then solving the fine grid system once. Procedures based on this idea should be developed.

Appendix A

Moore-Penrose Solution of a Symmetric Linear System

Let A be a real symmetric $N \times N$ singular matrix. The linear system

$$Au = b \tag{A.0.1}$$

may either have an infinite number of solutions or no solution depending on whether the vector b lies in the range of A. However, there always exists a unique solution to the problem of finding w such that $\|b - Aw\|$ is minimized and such that $\|w\|$ is minimized. The solution u^* of the modified problem is referred to as the *Moore-Penrose solution* (see Moore [54] and Penrose [58]) and is denoted by

$$u^* = A^\dagger b. \tag{A.0.2}$$

The Moore-Penrose solution can be determined as follows. Since A is symmetric there exists a orthogonal matrix V such that

$$V^{-1}AV = \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p, 0, \ldots, 0) \tag{A.0.3}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_p$ are the nonzero eigenvalues of A. The *Moore-Penrose inverse* $A^\dagger$ of A is defined by

$$A^\dagger = V\mathrm{diag}(\lambda_1^{-1}, \lambda_2^{-1}, \ldots, \lambda_p^{-1}, 0, \ldots, 0)V^{-1} \tag{A.0.4}$$

Note that

$$V^{-1} = V^T \tag{A.0.5}$$

The Moore-Penrose solution is given by

$$u^* = A^\dagger b \tag{A.0.6}$$

We note that if A is nonsingular, then $u^* = A^{-1}b$. Moreover if A is singular and b is in the range of A then

$$\|b - Au^*\| = 0 \tag{A.0.7}$$

Let $v^{(1)}, v^{(2)}, \ldots, v^{(p)}$ denote the eigenvectors of A associated with nonzero eigenvalues and let $v^{(p+1)}, \ldots, v^{(N)}$ denote the eigenvectors associated with the zero eigenvalue. Evidently b can be represented in the form

$$b = d_1 v^{(1)} + \cdots + d_p v^{(p)} + d_{p+1} v^{(p+1)} + \cdots + d_N v^{(N)} \tag{A.0.8}$$

The Moore-Penrose solution $u^* = A^\dagger b$ can be written in the form

$$u^* = d_1 \lambda_1^{-1} v^{(1)} + \cdots + d_p \lambda_p^{-1} v^{(p)} \tag{A.0.9}$$

We note that u^* is a solution of the system

$$Au = b' \tag{A.0.10}$$

where

$$b' = b - d_{p+1} \frac{(v^{(p+1)}, b)}{(v^{(p+1)}, v^{(p+1)})} v^{(p+1)} + \cdots + d_N \frac{(v^{(N)}, b)}{(v^{(N)}, v^{(N)})} v^{(N)} \tag{A.0.11}$$

We say that b' is a "purified" vector corresponding to b. Moreover, the general solution of the modified system $Au = b'$ is given by

$$\hat{u} = d_1 \lambda_1^{-1} v^{(1)} + \cdots + d_p \lambda_p^{-1} v^{(p)} + e_{p+1} v^{(p+1)} + \cdots + e_N v^{(N)} \tag{A.0.12}$$

Therefore the Moore-Penrose solution u^* can be obtained from $\hat{u}$ by "purification".

BIBLIOGRAPHY

[1] R. E. Alcouffe, A. Brandt, J. E. Dendy and J. W. Painter, "The Multi-Grid Method for the Diffusion Equation with Strongly Discontinuous Coefficients," *SIAM Journal on Scientific and Statistical Computing*, Vol. 2, 1981, pp. 430–454.

[2] O. Axelsson, "A General Incomplete Block-Matrix Factoriztion Method," *Linear Algebra and its Applications*, Vol. 74, 1986, pp. 179–190.

[3] K. Aziz and A. Settari, *Petroleum Reservoir Simulation*, Elsevier Applied Science Publishers, London, 1979.

[4] P. Bastian and C. Horton, "Parallelization of Robust Multigrid Methods: ILU Fctorization and Frequency Decomposition Method," *SIAM Journal on Scientific and Statistical Computing*, Vol. 12, No. 6, November 1991, pp. 1457–1470.

[5] A. Behie and P. A. Forsyth, "Multi-Grid Solution of Three-Dimensional Problems With Discontinuous Coefficients," *Applied Mathematics and Computation*, Vol. 13, 1983, pp. 103–120.

[6] A. Behie and P. A. Forsyth, "Multigrid Solution of the Pressure Equation in Reservoir Simulation," *Society of Petroleum Engineers Journal*, August 1983, pp. 623–632.

[7] C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T. Krogh, "Basic Linear Algebra Subprograms for Fortran Usage," *ACM Transactions on Mathematical Software*, Vol. 5, No. 3, September 1979, pp. 308–323.

[8] A. Brandt, "Multi-level Adaptive Solutions to Boundary-Value Problems," *Mathematics of Computation*, Vol. 31, No. 138, 1977, pp. 333–390.

[9] A. Brandt, "Multigrid Techniques: 1984 Guide, with Applications to Fluid Dynamics," Nr. 85, GMD-AIW, Postfach 1240, D5205, GMD Studien, St. Augustin 1, Germany, 1984.

[10] W. L. Briggs, *A Multigrid Tutorial*, Philadelphia: Society for Industrial and Applied Mathematics, 1987.

[11] B. Briggs, L. Hart, S. McCormick and D. Quinlan, "Multigrid Methods on a Hypercube," Theory, Applications, and Supercomputing, S. F. McCormick, ed., Marcel Dekker, Inc., 1988.

[12] Y. Chang, G. Pope and K. Sepehrnoori, "A Higher-order Finite-difference Compositional Simulator," *Journal of Petroleum Science and Engineering,* 5, 1990, pp. 35–50.

[13] Y. Chang, "Development and Application of an Equation of State Compositional Simulator," Doctoral Dissertation, The University of Texas at Austin, 1990.

[14] T. F. Chan and H. C. Elman, "Fourier Analysis of Iterative Methods for Elliptic Problems," *SIAM Review*, Vol. 31, No. 1, March 1989, pp. 20–49.

[15] T. F. Chan, C.-C. J. Kuo and C. Tong, "Parallel Elliptic Preconditioners: Fourier Analysis and Performance on the Connection Machine," *Computer Physics Communications*, Vol. 53, 1989, pp. 237–252.

[16] L. Cowsar, A. Weiser and M. F. Wheeler, "Parallel Multigrid and Domain Decomposition Algorithms for Elliptic Equations," Report CRPC-TR91154, Center for Research on Parallel Computation, Rice University, June, 1991.

[17] W. Dahmen and L. Elsner, "Algebraic Multigrid Methods and the Schur Complement," *Numerische Mathematik*, Vol. 56, 1989, pp. 58–68.

[18] N. Decker, "On the Parallel Efficiency of the Frederickson-McBryan Multigrid," ICASE Report No. 90-17, Institute for Computer Applications in Science and Engineering, Hampton, VA, February 1990.

[19] N. Decker, "A Note on the Parallel Efficiency of the Frederickson-McBryan Multigrid Algorithm," *SIAM Journal on Scientific and Statistical Computing*, Vol. 12, 1991, pp 208–220.

[20] N. H. Naik (Decker) and J. V. Rosendale, "The Improved Robustness of Multigrid Elliptic Solvers Based on Multiple Semicoarsened Grids," ICASE Report No. 91-70, Institute for Computer Applications in Science and Engineering, Hampton, VA, September 1991.

[21] J. E. Dendy Jr., "Black Box Multigrid," *Journal of Computational Physics*, Vol. 48, 1982, pp. 366–386.

[22] J. E. Dendy Jr., "Black Box Multigrid for Nonsymmetric Problems," *Applied Mathematics and Computation*, Vol. 13, 1983, pp. 261–283.

[23] J. E. Dendy Jr., "Black Box Multigrid for Systems," *Applied Mathematics and Computation*, Vol. 19, 1986, pp. 57–74.

[24] J. E. Dendy, Jr., S. F. McCormick, J. W. Ruge and T. F. Russell, "Multigrid Methods for Three-Dimensional Petroleum Reservoir Simulation," paper SPE 18409 presented at the 1989 SPE Symposium on Reservoir Simulation, Houston, TX, February 6–8.

[25] J. E. Dendy, Jr., M. P. Ida, and J. M. Rutledge, "A Semicoarsening Multigrid Algorithm for SIMD Machines," *SIAM Journal on Scientific and Statistical Computing*, Vol. 13, No. 6, November 1992, pp. 1460–1469.

[26] J. E. Dendy, Jr., "Multigrid Methods for Petroleum Reservoir Simulation on SIMD Machines," paper SPE 25243 presented at the 1993 SPE Symposium on Reservoir Simulation, New Orlean, LA, February 28 – March 3.

[27] T. W. Fogwell and F. Brakhagen, "Nonlinear Hyperbolic Equations – Theory, Computation Methods, and Applications," *Notes on Numerical Fluid Mechanics*, Vol. 24, 1989, pp. 139-148.

[28] P. O. Frederickson and O. McBryan, "Parallel Superconvergent Multigrid," *Multigrid Methods* (ed. S. McCormick). New York: Marcel Dekker, 1988.

[29] P. O. Frederickson and O. McBryan, "Recent Developments for Parallel Multigrid," *Proc. Third European Conference on Multigrid Methods*, October 1990 (eds. U. Trottenberg and W. Hackbusch), Birkh:auser Verlag International Series of Numerical Mathematics, Basel, to appear. Also published as University of Colorado Report CU-CS-524-90, March 1991.

[30] P. O. Frederickson and O. McBryan, "Normalized Convergence Rates for the PSMG Method," *SIAM Journal on Scientific and Statistical Computing*, Vol. 12, 1991, pp. 208–220.

[31] P. O. Frederickson and O. McBryan, "Parallel Multigrid Revisited," University of Colorado CS Dept. Technical Report, 1991.

[32] W. Hackbusch, *Multi-grid Methods and Applications*, New York: Springer Series in Computational Mathematics, Springer-Verlag, 1985.

[33] W. Hackbusch, "A New Approach to Robust Multi-grid Methods," *First International Conference on Industrial and Applied Mathematics*, Paris: 1987.

[34] W. Hackbusch, "The Frequency Decomposition Multigrid Method, Part I: Application to Anisotropic Equations," *Numerische Mathematik*, Vol. 56, 1989, pp. 229–245.

[35] L. A. Hageman and D. M. Young, *Applied Iterative Methods*, New York: Academic Press, 1981.

[36] R. W. Hockney and C. R. Jesshope, *Parallel Computers*, Bristol: Adam Higer Ltd, 1984.

[37] R. Hempel and A. Schuller, "Experiments with Parallel Multigrid Algorithms Using the SUPREN UM Communications Subroutine Library," GMD Studie 141, St. Augustin, April 1984.

[38] K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*, New York: Mc Graw-Hill, 1984.

[39] *Math/Library User's Manual, version 1.0 of edition 10.0*, Houston: IMSL, Inc., 1987.

[40] P. Joly and R. Eymard, "Preconditioned Biconjugate Gradient Methods for Numerical Reservoir Simulation," Paris: Universite Pierre et Marie Curie, 1987.

[41] R. Kettler and K. Shell and P. Wesseling, "Aspects of Multigrid Methods for Problems in Three Dimensions," *Applied Mathematics and Computation*, Vol. 19, 1986, pp. 159–168.

[42] J. E. Killough and R. Bhogeswara, "Simulation of Computational Reservoir Phenomena on a Parallel Computer," paper SPE 21208 presented at the 1991 SPE Symposium on Reservoir Simulation, Anaheim, CA, February 17–20.

[43] C.-C. Jay Kuo and T. F. Chan, "Two-Color Fourier Analysis of Iterative Algorithms for Elliptic Problems with Red-Black Ordering," *SIAM Journal on Scientific and Statistical Computing*, Vol. 11, No. 4, July 1990, pp. 767–793.

[44] C.-C. Jay Kuo and B. C. Levy, "Two-Color Fourier Analysis of the Multigrid Method with Red-Black Guass-Seidel Smoothing," *Applied Mathematics and Computation*, Vol. 29, 1989, pp. 69–87.

[45] C.-C. Jay Kuo and B. C. Levy, "A Two-Level Four-Color SOR Method," *SIAM Journal of Numerical Analysis*, Vol. 26, No. 1, February 1989, pp. 129–151.

[46] C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T. Krogh, "Basic Linear Algebra Subprograms for Fortran Usage," *ACM Transactions on Mathematical Software*, Vol. 5, No. 3, September 1979, pp. 308–323.

[47] M.T. Lim, S.A. Khan, K. Sepehrnoori, and G.A. Pope, "Simulation of Carbon Dioxide Flooding Using Horizontal Wells," paper SPE 24929 presented at the 1992 SPE Annual Tech. Conf., Washington, DC, October 4–7.

[48] M.T. Lim, "Field-Scale Numerical Modeling of Multiple Contact Miscible Processes Using Horizontal Wells in Heterogeneous Reservoirs," Doctoral Dissertation, The University of Texas at Austin, 1993.

[49] S. F. McCormick, *Multigrid Methods*, Society for Industrial and Applied Mathematics, Philadephia, PA, 1987.

[50] O. McBryan and E. Van de Velde, "The Multigrid Method on Parallel Processors," *Proceeding of 2nd European Multigrid Conference*, Cologne, October 1985, GMD Studie Nr. 110, J. Linden, ed., GMD, July 1986.

[51] O. McBryan, "The connection Machine: PDE Solution on 65536 Processors," *Parallell Computing*, Vol. 9, pp. 1-24, North-Holland, 1988.

[52] O. McBryan, "New Architectures: Performance Highlights and New Algorithms," *Parallell Computing*, Vol. 7, 1988, pp. 477–499.

[53] J, A, Meijerink and H. A. van der Vorst, "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix," *Mathematics if Computation*, Vol. 31, No. 137, January 1977, pp. 148–162.

[54] E. H. Moore, "On the Reciprocal of the General Algebraic Matrix," *Bull. Amer. Math. Soc.*, Vol. 26, 1919, pp. 394–395.

[55] Wim A. Mulder, "A New Multigrid Approach to Convection Problems," *Journal of Computational Physics*, Vol. 83, 1989, pp. 303–323.

[56] T. C. Oppe, W. D. Joubert and D. R. Kincaid, "NSPCG User's Guide Version 1.0," Report CNA-216, Center for Numerical Analysis, The University of Texas at Austin, April 1988.

[57] S. W. Parter, "Remarks on Multigrid Convergence Theorems," *Journal of Applied Mathematics and Computation.* Vol. 23, 1987, pp. 103–120.

[58] R. Penrose, "A Generalized Inverse for Matrices," *Proc. Cambridge Philological Society,* Vol. 51, 1955, pp. 406–413.

[59] G. Shiles, G.A. Pope, K. Sepehrnoori, "Petroleum reservoir Simulation on the CRAY Y-MP," *Science and Engineering on Supercomputers Proceeding of Fifth International Conference,* October 22-24, 1990, E.J. Pitcher (ed.), Springer-Verlag.

[60] R. A. Smith and A. Weiser, "Semicoarsening Multigrid on a Hypercube," *SIAM Journal on Scientific and Statistical Computing,* Vol. 13, No. 6, November 1992, pp. 1314–1329.

[61] H. S. Stone, "An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations," *Journal of the Association fot Computing Machinery,* Vol. 20, No. 1, January 1973, pp. 27–38.

[62] H. S. Stone, "Parallel Tridiagonal Equation Solvers," *ACM Transactions on Mathematical Software,* Vol. 1, No. 4, Decenber 1981, pp. 289–307.

[63] K. Stüben and U. Trottenberg, "Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications," *Multigrid Methods,* W. Hackbusch and U. Trottenberg, eds., Berlin, 1981, Springer-Verlag, pp. 1–176. Lecture Notes in Mathematics 960.

[64] R. S. Tuminaro, *Multigrid Algorithms on Parallel Processing Systems,* Doctoral Dissertation, Stanford University, 1990.

[65] R. S. Varga, *Matrix Iterative Analysis,* Englewood Cliffs, NJ: Prentice-Hall, 1962.

[66] Bi R. Vona, "Parallel Multilevel Iterative Methods," Doctoral Dissertation, Report CNA-261, Center for Numerical Analysis, The University of Texas at Austin, March 1990.

[67] H. H. Wang "A Parallel Method for Tridiagonal Equations," *ACM Transactions on Mathematical Software.* Vol. 7, No. 2, June 1981, pp. 170–183.

[68] A. Weiser, "Semicoarsening Multigrid with Collapsed Coarse Grid Stencils," Report Rice CRPC 1991.

[69] P. Wesseling *An Introduction to Multigrid Methods*, John Wiley & Sons Ltd., 1992.

[70] D. M. Young, *Iterative Solution of Large Linear Systems*, New York: Academic Press, 1971.

[71] D. M. Young and R. T. Gregory, *A Survey of Numerical Mathematics Volumn II*, New York: Dover Publications, Inc., 1988.

[72] D. M. Young and K. C. Jea, "Generalized Conjugate Gradient Acceleration of Non-symmetrizable iterative methods," *Linear Algebra and Applications*, Vol. 34, 1980, pp. 159–194.

[73] D. M. Young and Bi R. Vona, "Parallel Multilevel Methods," Report CNA-243, Center for Numerical Analysis, The University of Texas at Austin, March 1990.

[74] D. M. Young, *Notes for Course CS/M 393N – Numerical Solution of Elliptic Partial Differential Equations*, The University of Texas as Austin, December 1992.