

NUMERICAL MULTIGRID ALGORITHM FOR SOLVING  
INTEGRAL EQUATIONS

A THESIS

SUBMITTED TO THE GRADUATE SCHOOL  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE  
MASTER OF SCIENCE

BY

SUBRATA PAUL

DR. IRENE LIVSHITS - ADVISOR

BALL STATE UNIVERSITY

MUNCIE, INDIANA

MAY, 2014

# Acknowledgements

First and foremost I offer my sincerest gratitude to my supervisor, Dr. Irene Livshits, who has supported me throughout my thesis with her patience and knowledge whilst allowing me the room to work in my own way. Her encouragement was the source of my enthusiasm. I have never felt alone because of her presence to discuss whenever I had a question. It is a great experience and remarkable achievement for me to be able to work with her. One simply could not wish for a better or friendlier supervisor.

I am thankful to the Department of Mathematical sciences. Friendliness of the professors creates an wonderful working environment. Especial thanks to the Graduate School for the Graduate Merit Fellowship.

# Contents

Acknowledgements . . . . .	ii
<b>1 Introduction</b>	<b>1</b>
1.1 Literature Review . . . . .	3
<b>2 Multigrid Methods</b>	<b>5</b>
2.0.1 From Continuous to Discrete . . . . .	6
2.1 Error and Residual . . . . .	8
2.2 Relaxation Methods . . . . .	10
2.2.1 Jacobi Iteration . . . . .	11
2.2.2 Gauss-Seidel Iteration . . . . .	12
2.2.3 Damped Jacobi . . . . .	13
2.2.4 Kaczmarz Relaxation . . . . .	14
2.2.5 Distributive Relaxation . . . . .	16
2.2.6 Convergence of Iterative Methods . . . . .	17
2.3 Smoothing Property of the Relaxation Techniques . . . . .	18
2.4 Two-Grid Correction . . . . .	28
2.5 Elements of Multigrid . . . . .	29
2.5.1 Interpolation . . . . .	30
2.5.2 Restriction . . . . .	32
2.6 Numerical Example for $-u_{xx} = 0$ . . . . .	34

2.7	V-Cycle Scheme . . . . .	38
<b>3</b>	<b>Integral Equations</b>	<b>42</b>
3.1	Introduction . . . . .	42
3.2	Discretization . . . . .	43
3.2.1	Discretization Using Trapezoidal Rule . . . . .	44
3.2.2	Discretization using Gaussian Quadrature . . . . .	45
3.3	Picard's Iteration . . . . .	46
3.3.1	Example . . . . .	48
3.4	Picard's Method in a Discrete Setting . . . . .	48
3.4.1	Numerical Example . . . . .	49
3.5	Nyström Method . . . . .	49
3.5.1	Example . . . . .	51
<b>4</b>	<b>Multigrid Methods for Integral Equations</b>	<b>54</b>
4.1	Introduction . . . . .	54
4.2	Two Level Method . . . . .	55
4.2.1	Application of Two Level Method . . . . .	57
4.3	Hemker-Schippers Modification . . . . .	58
4.4	Numerical Experiments . . . . .	60
4.4.1	Using Kaczmarz iteration . . . . .	61
4.4.2	Using Distributive relaxation . . . . .	64
<b>5</b>	<b>Conclusion</b>	<b>67</b>
	<b>Bibliography</b>	<b>69</b>

# Chapter 1

## Introduction

In 1825 Abel, an Italian mathematician, first produced integral equations in connection with the famous tautochrone problem. Nowadays integral equations and their applications is an important subject in applied mathematics. They are used to model various physical phenomena such as radiative transfer, signal processing and many others. Mathematical physics models, such as diffraction and scattering in quantum mechanics, conformal mapping and wave propagation also give a rise to the creation of integral equations. The Chandrasekhar H equation, spectral concentration problem are notable examples of integral equations.

Mathematical models of some problems appear directly in terms of integral equations. Other problems whose direct representation is in terms of differential equations and their auxiliary conditions, may also be reduced to integral equations. The advantage of reducing a differential equation, especially a boundary value problem, to an integral equation is that the auxiliary conditions are automatically satisfied as they are incorporated in the process of formulating the corresponding integral equation.

There are many analytical approaches to solve integral equations. For their overview see, for example, [13],[18]. These include method of successive approximation, Adomian

decomposition method[18], Picard's method [11] etc. The analytical approaches have limited applicability either because the analytical solutions of some of the problems those arise in application do not exist or because they are more time consuming. Therefore, many integral equations arising in mathematical modeling of physical world problem demands efficient numerical techniques. A number of numerical techniques such as Galerkin's method [13], collocation method [13], Nyström interpolation [4] etc. have been proposed by various authors. Among others, Atkinson [4] provides firm theoretical discussion of the numerical techniques to solve integral equations.

In numerical context, solving of integral equations is reduced to solving a system of linear equations. Iterative techniques are available to solve a system of linear equations such as Jacobi and, Gauss-Seidel. Multigrid methods also solve system of linear equations with faster convergence and less computation cost.

Kaczmarz relaxation technique, that always converges, does not need the system to be symmetric positive definite which is necessary condition for Jacobi like pointwise iteration techniques. The multigrid methods available to solve integral equations [12],[3] uses the Picard's iteration which converges for a specific type of integral equations. In this thesis we proposed a multigrid methods using Kaczmarz and distributive relaxation techniques. We compared the multigrid setup using Kaczmarz or distributive relaxation with that using Picard's iteration.

The remainder of the thesis is organized as follows. Chapter 1 contains the basic ideas of multigrid methods along with the description of relaxation techniques. In this chapter differential equations are used as a model to illustrate ideas of multigrids and its components. Chapter 2 contains a brief discussion of integral equations and their classification along with some analytical and numerical techniques to solve an integral equation. Chapter 3 consists of

the application of multigrid techniques to integral equations. It starts with the discussion of available multigrid schemes and then presents the results of using Kaczmarz and distributive relaxation techniques.

## 1.1 Literature Review

Hackbusch [12],[11] give description of the multigrid techniques and the integral equations from both theoretical and computational points of views. Starting with preliminary ideas for analysis [12], discusses theoretical development of multigrid techniques. Chapter 3 of this book contains description and analysis of components of multigrid. Computational complexity is described for a general multigrid scheme. Smoothing property of the Jacobi, Damped Jacobi, Gauss-Seidel relaxation are discussed.

Chapter 16, named, The Multi-Grid Method of the Second Kind[12], introduces the application of multigrid technique to integral equations. Three variants of the multigrid algorithms are discussed to solve integral equations. First is the Atkinson-Brakhage iteration, the second is the the Hemker-Schippers modification and the third one is designed to reduce computational complexity. This chapter also includes ideas of solving partial differential equations using integral equation methods.

Atkinson proposed the two grid method [3], for integral equations. This is one of the pioneer paper that applies multigrid schemes to solve integral equations. His book [4], gives theoretical background of the integral equations and numerical techniques to solve them. Chapter 6 of this book involves two grid iterative methods using Nyström interpolation for solving integral equations. He also introduced conjugate gradient method and multigrid iteration for collocation methods. The book contains brief discussion on non-symmetric integral equations and a chapter is dedicated to the boundary integral equations on a

smooth planar boundary.

Brakhage's [6] two grid algorithm using is similar to the Atkinson's two grid. Among other research works, Kelley's paper [15], give a fast multigrid algorithm based on the Atkinson-Brakhage two grid methods. This article shows how the discretization of integral equations by composite Gauss rules can be related to approximations of integral operators that converges in the operator norm, rather than strongly converge. The proposed multigrid algorithm is applicable to both linear and nonlinear equations and is roughly half as costly as the Atkinson-Brakhage iteration.

Multigrid Tutorial [8], is helpful to get the basic ideas of multigrid techniques. This book discusses multigrid in computational aspect. It contains geometric multigrid, algebraic multigrid and multilevel adaptive methods.

# Chapter 2

## Multigrid Methods

Multigrid methods were initially developed to solve partial differential equations more efficiently than other existing numerical techniques. Fedorenko [10] first formulated a multigrid algorithm to solve the Poisson equation in a rectangular domain in 1964. In 1966 Bachvalov [5] generalized the technique for general elliptic partial differential equations. Up to this time, the approach was not yet practical. The first practical results were published in a pioneering paper by Brandt [7] in 1973. He outlined the purpose and practical utility of multigrid methods. Hackbusch [12] independently discovered multigrid methods and provided some theoretical foundation in 1976.

We start by introducing ideas of multigrid applied to a system of linear equations

$$Av = f, \tag{2.1}$$

a result of discretization of a partial differential equation or an integral equation, is based on the following observations.

## 2.0.1 From Continuous to Discrete

We consider the following second order boundary value problem as a model:

$$-u''(x) + \sigma u(x) = f(x), \quad 0 < x < 1, \quad \sigma \geq 0, \quad (2.2)$$

$$u(0) = u_0, \quad u(1) = u_1. \quad (2.3)$$

While there are many analytical and numerical methods to solve this problem in 1D similar equations in higher dimension are not always solvable analytically. To solve (2.2) numerically, first it is reformulated from a continuous to a discrete formulation using finite difference method.

The process involves discretization of equation (2.2) on the mesh created by dividing the domain  $\{x : 0 \leq x \leq 1\}$  into  $n$  equal subintervals. The size of each subinterval is  $h = \frac{1}{n}$ . Points of partition of the interval  $[0, 1]$  are called the grid points and they are given by  $x_j = jh$ ,  $j = 0, 1, \dots, n$ . Such a partition, represented by  $\Omega^h$ , is called a grid.

Suppose the grid  $\Omega^h$  of the domain  $[a, b]$  consists of  $n$  subintervals with mesh-size  $h$ . The grid that consists of  $\frac{n}{2}$  subintervals with mesh size  $2h$  is represented by  $\Omega^{2h}$  with the grid points are  $x_j = 2hj$ ,  $j = 0, 1, \dots, \frac{n}{2}$ .  $\Omega^{2h}$  is called the coarser than  $\Omega^h$  because the number of subintervals in  $\Omega^{2h}$  is half of that in  $\Omega^h$ . Conversely  $\Omega^h$  is finer than  $\Omega^{2h}$ .

At the interior mesh points  $x_j$ ,  $j = 1, 2, \dots, n-1$ , the differential equation (2.2) is given by,

$$-u''(x_j) + \sigma u(x_j) = f(x_j). \quad (2.4)$$

Expanding  $u$  in a third Taylor polynomial about  $x_j$  evaluated at  $x_{j-1}$  and  $x_{j+1}$ , we have, assuming that  $u \in C^4([x_{j-1}, x_{j+1}])$ ,

$$u(x_{j+1}) = u(x_j + h) = u(x_j) + hu'(x_j) + \frac{h^2}{2}u''(x_j) + \frac{h^3}{3}u'''(x_j) + \frac{h^4}{24}u^{(4)}(\xi_j^+), \quad (2.5)$$

for some  $\xi_j^+$  in  $(x_j, x_j + 1)$

$$u(x_{j-1}) = u(x_j - h) = u(x_j) - hu'(x_j) + \frac{h^2}{2}u''(x_j) - \frac{h^3}{3}u'''(x_j) + \frac{h^4}{24}u^{(4)}(\xi_j^-), \quad (2.6)$$

for some  $\xi_j^-$  in  $(x_{j-1}, x_j)$

Adding equations (2.4), (2.5) and (2.6) yields,

$$u''(x_j) = \frac{1}{h^2} [u(x_{j-1}) - 2u(x_j) + u(x_{j+1})] - \frac{h^2}{24} [u^{(4)}(\xi_j^+) + u^{(4)}(\xi_j^-)].$$

The error term can be simplified using the Intermediate Value Theorem, to gives

$$u''(x_j) = \frac{1}{h^2} [u(x_{j-1}) - 2u(x_j) + u(x_{j+1})] - \frac{h^2}{12}u^{(4)}(\xi_j), \quad (2.7)$$

for some  $\xi_j$  in  $(x_{j-1}, x_{j+1})$ . The equation (2.7) is called the second order central-difference formula for  $u''(x_j)$ .

Suppose now that  $v_j$  is an approximation to the exact solution  $u(x_j)$ . So the components of approximate solution vector  $v = [v_0, v_2, \dots, v_n]^T$  satisfy the  $n + 1$  linear equations,

$$\begin{aligned} v_0 &= u_0, \\ \frac{-v_{j-1} + 2v_j - v_{j+1}}{h^2} + \sigma v_j &= f(x_j), \quad 1 \leq j \leq n, \\ v_n &= u_n, \end{aligned} \quad (2.8)$$

or equivalently in matrix form,

$$Av = b. \tag{2.9}$$

Here,  $A$  is an  $(n - 1) \times (n - 1)$  matrix,

$$A = \frac{1}{h^2} \begin{bmatrix} 2 + \sigma h^2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 + \sigma h^2 & -1 & \dots & \dots & 0 \\ 0 & -1 & 2 + \sigma h^2 & -1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 + \sigma h^2 \end{bmatrix}$$

and

$$\begin{aligned} b &= \left[ f(x_1) + \frac{1}{h^2}u_0, f(x_2), \dots, f(x_{n-1}) + \frac{1}{h^2}u_1 \right]^T \\ &= \left[ f_1 + \frac{1}{h^2}u_0, f_2, \dots, f_{n-1} + \frac{1}{h^2}u_1 \right]^T. \end{aligned}$$

The solution of the system of linear equations approximates the solution of the boundary value problem (2.2),(2.3): i.e.  $u(x_j) \approx v_j$ .

## 2.1 Error and Residual

Consider a system of linear equations,

$$Av = f. \tag{2.10}$$

Suppose that the matrix equation (2.10) has a unique solution  $v$  and that  $\tilde{v}$  approximates the solution  $v$ . The error associated with  $\tilde{v}$  is given by,

$$e = v - \tilde{v}.$$

Practically, the error is inaccessible as the exact solution itself. A measure of how well  $\tilde{v}$  approximates the exact solution  $v$ , that we can actually compute, is the residual  $r$ ; given by,

$$r = f - A\tilde{v}.$$

That is, the residual is a measure of the amount by which the approximation fails to satisfy the original equation. As the approximation  $\tilde{v} \rightarrow v$  that is when  $e \rightarrow 0$ , we see that  $r \rightarrow 0$ . The converse is not always true. That is a small residual not necessarily implies a small error. To see this let us consider a linear system  $Av = b$  given by

$$\begin{bmatrix} 1 & 2 \\ 1.0001 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 3.0001 \end{bmatrix}. \quad (2.11)$$

The system (2.11) has exact solution  $v = [1, 1]^T$ . Assume that somehow we come up with a poor approximation  $\tilde{v} = [3, -0.0001]^T$ . The residual corresponding to the approximation is,

$$r = b - A\tilde{v} = \begin{bmatrix} 0.0002 \\ 0 \end{bmatrix}.$$

Although the norm of the residual vector is very small, the approximation  $\tilde{v} = [3, -0.0001]^T$  is obviously quite poor; in fact  $\|e\| = \|v - \tilde{v}\| = 2$ . This phenomena can be understood and a bound for error can be obtained using condition number.

The condition number of a nonsingular matrix  $A$  relative to a norm  $\|\cdot\|$  is

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|.$$

A matrix is said to be well-conditioned if  $\kappa(A)$  is close to 1, and ill-conditioned when  $\kappa(A)$  is significantly greater than 1. The error bound is given by the following theorem.

**Theorem 1.** *Suppose that  $\tilde{v}$  is an approximation to the solution of  $Av = b$ ,  $A$  is a nonsingular matrix, and  $r$  is the residual vector for  $\tilde{v}$ . Then for any natural norm,*

$$\|v - \tilde{v}\| \leq \kappa(A) \frac{\|r\|}{\|b\|}.$$

According to the theorem 1, if  $\kappa(A)$  is small i.e. if the system is well conditioned then small residual will imply small error. In case of ill-conditioned matrix it is not reasonable to say that small residual implies small error.

There is a relation between the error and residual given by,

$$r = f - A\tilde{v} = Av - A\tilde{v} = A(v - \tilde{v}) = Ae.$$

Shortly we will discover that this relation plays an important role in the Multigrid algorithms.

## 2.2 Relaxation Methods

Relaxation is an iterative strategy of solving a system of linear equations. It is a cost-saving alternative to direct methods that are equivalent to finding  $v$  as  $v = A^{-1}f$ . The commonly used relaxation methods are, for example, Jacobi, Gauss Seidel, weighted Jacobi method.

Generally, an iterative technique is formulated by choosing a matrix  $B$  and rewriting system (2.10) as,

$$Bv = (B - A)v + f.$$

Here matrix  $B$  is chosen in a way that it is nonsingular and its easy to find the inverse. The following recurrence relation generates a sequence of approximate solutions,

$$\tilde{v}^{(n+1)} = B^{-1}(B - A)\tilde{v}^{(n)} + B^{-1}f = (I - B^{-1}A)\tilde{v}^{(n)} + B^{-1}f,$$

where  $\tilde{v}^{(n)}$  and  $\tilde{v}^{(n+1)}$  represent the previous and the current approximations, respectively, and  $\tilde{v}^{(0)}$  is an arbitrary initial vector. The strategy produces a sequence of approximate solution vectors  $\tilde{v}^{(0)}, \tilde{v}^{(1)}, \dots$  for the system so that the sequence converges to the actual solution  $v$ .

The followings are standard iterative techniques.

### 2.2.1 Jacobi Iteration

Considers the matrix  $A$  in the form,

$$A = D - L - U,$$

where  $D, L$  and,  $U$  are the diagonal, the lower diagonal and the upper diagonal parts of  $A$ , respectively. Then  $Av = f$  is equivalent to

$$(D - L - U)v = f. \tag{2.12}$$

For the Jacobi method matrix  $B$  is chosen to be the diagonal matrix  $D$ . The equation (2.12) yields the iterative equation as,

$$\begin{aligned}
D\tilde{v}^{(n+1)} &= (L + U)\tilde{v}^{(n)} + f \\
\Rightarrow \tilde{v}^{(n+1)} &= D^{-1}(L + U)\tilde{v}^{(n)} + D^{-1}f \\
\Rightarrow \tilde{v}^{(n+1)} &= D^{-1}(D - A)\tilde{v}^{(n)} + D^{-1}f \\
\Rightarrow \tilde{v}^{(n+1)} &= (I - D^{-1}A)\tilde{v}^{(n)} + D^{-1}f.
\end{aligned}$$

Convergence of Jacobi method depends on the properties of the matrix  $R_J = I - D^{-1}A$ . The matrix  $R_J$  is known as the Jacobi iteration matrix and the method converges if the matrix norm  $\|R_J\| < 1$ .

Jacobi method, in a component form, is formulated to solve the  $j$ th equation for the  $j$ th unknown using the previous approximation of the other unknowns. Thus, one full sweep of Jacobi method can be expressed as,

$$\tilde{v}_j^{(n+1)} = \frac{1}{2}(\tilde{v}_{j-1}^{(n)} + \tilde{v}_{j+1}^{(n)} + h^2 f_j), \quad 1 \leq j \leq n - 1.$$

### 2.2.2 Gauss-Seidel Iteration

For Gauss-Seidel the matrix  $B$  is chosen to be the lower diagonal matrix  $D - L$  and so the iteration defined recursively as,

$$\tilde{v}^{(n+1)} = (D - L)^{-1}U\tilde{v}^{(n)} + (D - L)^{-1}f.$$

The matrix  $R_G = (D - L)^{-1}U$ , known as Gauss-Seidel iteration matrix, determines the convergence of this scheme.

In this method components of the new approximation are used as soon as they are available. The component form of Gauss Seidel can be expressed as,

$$\tilde{v}_j^{(n+1)} = \frac{1}{2}(\tilde{v}_{j-1}^{(n+1)} + \tilde{v}_{j+1}^{(n)} + h^2 f_j), \quad 1 \leq j \leq n-1,$$

where  $\tilde{v}_j$  is computed in the lexicographic order.

### 2.2.3 Damped Jacobi

For Damped Jacobi method matrix  $B$  is chosen to be  $\omega^{-1}D$  for some  $\omega$  in  $(0, 1)$ . In matrix form, the weighted Jacobi method is given by,

$$\tilde{v}^{(n+1)} = [(1 - \omega)I + \omega R_J] \tilde{v}^{(n)} + \omega D^{-1} f.$$

where  $I$  is the identity matrix and  $R_J$  is the Jacobi iteration matrix. So, weighted Jacobi iteration matrix is given by,

$$R_\omega = (1 - \omega)I + \omega R_J.$$

To see the computation component-wise we compute an intermediate value  $\tilde{v}_j^*$  by the Jacobi iteration,

$$\tilde{v}_j^* = \frac{1}{2} \left( v_{j-1}^{(n)} + v_{j+1}^{(n)} + h^2 f_j \right), \quad 1 \leq j \leq n-1.$$

The iteration of weighted Jacobi is given by,

$$\tilde{v}_j^{(n+1)} = (1 - \omega)\tilde{v}_j^{(n)} + \omega\tilde{v}_j^*, \quad 1 \leq j \leq n-1.$$

where the weighting factor  $\omega$  is a real number in  $(0, 1)$ . Note that  $\omega = 1$  yields the Jacobi iteration.

Another form of weighted Jacobi method can be obtained using the residual vector;

$$\begin{aligned}
\tilde{v}^{(n+1)} &= [(1 - \omega)I + \omega R_J] \tilde{v}^{(n)} + \omega D^{-1} f \\
&= \tilde{v}^{(n)} - \omega D^{-1} D v^{(n)} \tilde{v}^{(n)} + \omega D^{-1} (L + U) \tilde{v}^{(n)} + \omega D^{-1} f \\
&= \tilde{v}^{(n)} - \omega D^{-1} (D - L - U) \tilde{v}^{(n)} + \omega D^{-1} f \\
&= \tilde{v}^{(n)} - \omega D^{-1} A \tilde{v}^{(n)} + \omega D^{-1} f \\
&= \tilde{v}^{(n)} + \omega D^{-1} [f - A \tilde{v}^{(n)}] \\
&= \tilde{v}^{(n)} + \omega D^{-1} r^{(n)},
\end{aligned}$$

where  $r^{(n)}$  is the residual associated with the approximation  $\tilde{v}^{(n)}$ . This representation of weighted Jacobi iteration shows that a new approximation is obtained from the previous approximation by adding an appropriate weighted residual.

## 2.2.4 Kaczmarz Relaxation

All schemes discussed above are pointwise. They also have a limited applicability i.e. restricted to nearly diagonally dominant or symmetric positive definite matrices. Integral equations often yields matrices that are not diagonally dominant or symmetric positive definite. Here we discuss the Kaczmarz [14] iterative method, proposed by the Polish mathematician Stefan Kaczmarz, that does not need the condition of diagonal dominance. Very important feature of this method is that it is always convergent.

To discuss Kaczmarz relaxation scheme let us consider the system of linear equations,

$$Ax = b, \tag{2.13}$$

where  $A$  is an  $n \times n$  matrix,  $b$  is a column vector in  $\mathbb{R}^n$  and  $x \in \mathbb{R}^n$  is the unknown. The  $i^{th}$  equation of the system (2.13) is,

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{ii}x_i + \dots + a_{in}x_n = b_i, \quad (2.14)$$

where  $a_{ij}$  is the entry of matrix  $A$  corresponding to  $i^{th}$  row and  $j^{th}$  column.

Unlike the pointwise relaxation techniques such as Jacobi iteration that solves the  $i^{th}$  equation to get formulation for  $i^{th}$  component, Kaczmarz iteration involves changing,

$$x_j \leftarrow x_j + a_{ij}\delta, \dots, j = 1, \dots, n, \quad (2.15)$$

so that the  $i^{th}$  row of matrix equation (2.13) contribute to update of all the components of the unknown vector. Our goal is to find  $\delta$  so that the  $i^{th}$  component of the residual  $r = Ax - b$  becomes zero, that is,

$$a_{i1}(x_1 + a_{i1}\delta) + a_{i2}(x_2 + a_{i2}\delta) + \dots + a_{ii}(x_i + a_{ii}\delta) + \dots + a_{in}(x_n + a_{in}\delta) = b_i.$$

Which implies,

$$\begin{aligned} (a_{i1}^2 + a_{i2}^2 + \dots + a_{in}^2)\delta &= b_i - a_{i1}x_1 - a_{i2}x_2 - \dots - a_{in}x_n \\ \Rightarrow \sum_{j=1}^n a_{ij}^2 \delta &= r_i \\ \Rightarrow \delta &= \frac{r_i}{\sum_{j=1}^n a_{ij}^2}. \end{aligned}$$

Substituting in equation (2.15), we get the iterative equation,

$$x_j \leftarrow x_j + a_{ij} \frac{r_i}{\sum_{j=1}^n a_{ij}^2}. \quad (2.16)$$

To describe the algorithm, Kaczmarz starts with an arbitrary first approximation  $[x_1^0, x_2^0, \dots, x_m^0]^T$  and splits the following approximations into groups, each of which consist of  $n$  approximations. The  $p^{th}$  group are,

$$\begin{aligned}
 & [x_1^{(p,1)}, x_2^{(p,1)}, \dots, x_n^{(p,1)}], \\
 & \dots \quad \dots \quad \dots \quad \dots \\
 & [x_1^{(p,2)}, x_2^{(p,2)}, \dots, x_n^{(p,2)}], \\
 & \dots \quad \dots \quad \dots \quad \dots \\
 & [x_1^{(p,n)}, x_2^{(p,n)}, \dots, x_n^{(p,n)}],
 \end{aligned}$$

and then define the  $(p + 1)^{th}$  group as,

$$\begin{aligned}
 x_i^{(p+1,1)} &= x_i^{(p,n)} + a_{1i} \frac{b_1 - \sum_{j=1}^n a_{1j} x_j^{(p,n)}}{\sum_{j=1}^n a_{1j}^2}, \\
 x_i^{p+1,2} &= x_i^{(p+1,1)} + a_{2i} \frac{b_2 - \sum_{j=1}^n a_{2j} x_j^{(p+1,1)}}{\sum_{j=1}^n a_{2j}^2}, \\
 & \dots \quad \dots \quad \dots \quad \dots \\
 x_i^{p+1,n} &= x_i^{(p+1,n-1)} + a_{ni} \frac{b_n - \sum_{j=1}^n a_{nj} x_j^{(p+1,n-1)}}{\sum_{j=1}^n a_{nj}^2}.
 \end{aligned}$$

## 2.2.5 Distributive Relaxation

Though Kaczmarz is always convergent it has higher computational costs than pointwise relaxation schemes. The goal of distributive relaxation techniques are to find balance between the Kaczmarz and the pointwise relaxation techniques. They are applicable to wider range of matrices than the pointwise relaxation and also not as expensive as the Kaczmarz iteration.

In a distributive iteration not all unknowns participating in the  $i^{th}$  equation are updated but only those with relatively large coefficients. If a matrix  $A$  has the property that  $A_{i,j} \rightarrow 0$

with the growth of  $|i - j|$ , then the unknowns  $x_j$  such that  $j$  is close to  $i$  are updated. For that a neighborhood of  $A_{ij}$  is selected those contribute to update  $x_j$ . Let  $\nu_j$  represents the set of neighboring entries. For a selected integer  $p$  the neighborhood is chosen as,

$$\nu_j := \{a_{i,j} : j - p \leq i \leq j + p\} \text{ if } p < j < n - p.$$

The neighborhood are chosen accordingly for the entries close to the boundary i.e. when  $j < p$  and  $n - j < p$ . The iterative equation is then given by

$$x_j \leftarrow x_j + a_{ij} \frac{r_j}{\sum_{\nu_j} a_{ij}^2}$$

## 2.2.6 Convergence of Iterative Methods

A general iterative method can be written as,

$$\tilde{v}^{(n+1)} = R\tilde{v}^{(n)} + c,$$

where  $R$  is the iteration matrix. Convergence of the iterative method depends on the iteration matrix  $R$  and is given by the following theorem,

**Theorem 2.** *For any  $\tilde{v}^{(0)} \in \mathbb{R}^n$ , the sequence  $\{\tilde{v}^{(n)}\}_{n=0}^{\infty}$  defined by,*

$$\tilde{v}^{(n+1)} = R\tilde{v}^{(n)} + c,$$

*converges to the unique solution of  $\tilde{v} = R\tilde{v} + c$  if and only if  $\rho(R) < 1$ .*

There exist a special class of matrices for which it is possible to state that the methods are sure to converge. The following theorems states some of the cases.

**Theorem 3.** *If  $A$  is a strictly diagonally dominant matrix i.e.  $|a_{ii}| > \sum_{j=1}^m |a_{i,j}|$ , for  $j \neq i$  and  $i = 1, \dots, \dots, m$ , the Jacobi and Gauss-Seidel methods are convergent.*

**Theorem 4.** *If  $A$  is symmetric positive definite, then the damped Jacobi method is convergent iff  $0 < \omega < \frac{2}{\rho(D^{-1}A)}$ .*

A detailed discussion on convergence of such iterative methods and the stopping criteria can be found in [9], [19].

## 2.3 Smoothing Property of the Relaxation Techniques

The idea of multigrid method comes from the observation that the relaxation techniques discussed earlier have the so-called smoothing property. In case of very oscillatory error the relaxation methods are quicker to reduce the error but for smooth error the process of reducing the amplitude of error become slower. The idea of multigrid is to relax error in the fine grid, make it smooth and transfer the corresponding residual to the coarser grid where it is more oscillatory than the finer grid error vector. Due to the oscillation in the coarser grid the relaxation can reduce the error more efficiently than corresponding smooth fine grid error. In this section we will discuss the smoothing property of the relaxation techniques.

We consider the homogeneous system,

$$Au = 0, \tag{2.17}$$

to study generic relaxation techniques. We assume that  $\det(A) \neq 0$ . One of the reasons of selecting the homogeneous system is that the exact solution is known which to be  $u = 0$ .

Brandt [7] uses the Fourier Local Analysis to show the smoothing property of the relaxation techniques. The general form of a Fourier mode is given by  $v$  with the  $j^{th}$  component,

$$v_j = \sin \frac{jk\pi}{n}, \quad 0 \leq j \leq n, 0 \leq k < n,$$

where the integer  $k$  represents the wave number or frequency and indicates the number of half sine waves. The larger  $k$  represents more oscillations in the vector  $v_j$ .

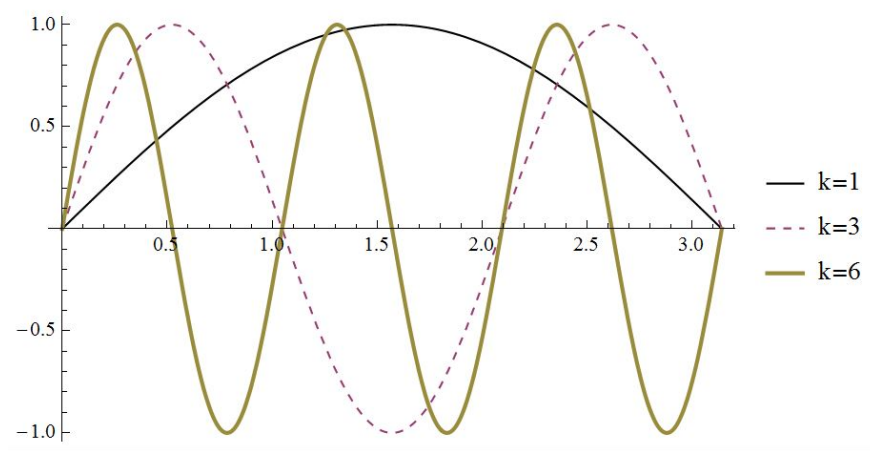


Figure 2.1: Fourier nodes with wavenumbers  $k = 1, 3, 6$ .

A generic relaxation technique can be represented by,

$$v^{(1)} = Rv^{(0)} + g. \quad (2.18)$$

For example,  $R = D^{-1}(L + D)$  for the Jacobi method discussed earlier. For all the relaxation techniques the exact solution is the fixed point of the iteration equation. That is

$$u = Ru + g, \quad (2.19)$$

where  $u$  represents the exact solution. Subtracting (2.18) from equation (2.19) we get

$$e^{(1)} = Re^{(0)}.$$

Here we see that the change in error with each iteration does not depend on the right hand side function  $g(x)$  of (2.18) and this is the another reason of considering homogeneous system

$Au = 0$ . By induction after  $m$  iterations

$$e^{(m)} = R^m e^{(0)},$$

$$\Rightarrow \|e^{(m)}\| \leq \|R^m\| \|e^{(0)}\|.$$

The above equation asserts very clearly that if  $\lim_{m \rightarrow \infty} \|R^m\| = 0$  then the iteration converges and we know that  $\|R^m\| \rightarrow 0$  if and only if the spectral radius  $\rho(R) < 1$ .  $\rho(R)$  is also known as the asymptomatic convergence factor of the iterative method that associates  $R$ . For simplicity we take, for example, a sparse Toeplitz matrix  $A$  resulted from discretization of the boundary value problem given by (2.2),(2.3) with  $\sigma = 0$  and  $f = 0$ , using finite difference.

$$A = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix}$$

Note that the diagonal, strictly lower triangular and upper triangular parts of the matrix  $A$  are,

$$D = \begin{bmatrix} 2 & 0 & & & \\ 0 & 2 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 0 \\ & & & 0 & 2 \end{bmatrix}, L = \begin{bmatrix} 0 & 0 & & & \\ 1 & 0 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & \ddots & 0 \\ & & & 1 & 0 \end{bmatrix}, U = \begin{bmatrix} 0 & 1 & & & \\ 0 & 2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 0 & 2 \end{bmatrix}.$$

So,  $D^{-1} = \frac{1}{2}I$  where  $I$  is the identity matrix. As relaxation technique we consider the weighted Jacobi method with weight  $\omega$ . The iteration matrix for weighted Jacobi,

$$\begin{aligned}
R_\omega &= (1 - \omega)I + \omega D^{-1}(L + U) \\
&= I - \omega D^{-1}D + \omega D^{-1}(L + U) \\
&= I - \omega D^{-1}(D - L - U) \\
&= I - \omega \frac{1}{2}IA \\
&= I - \frac{1}{2}\omega A.
\end{aligned}$$

Eigenvalues plays an important role to understand how the relaxation techniques work. An relaxation technique is determined and can be represented by its iteration matrix and that is why we are interested in the eigenvalues and eigenvectors of the iteration matrix  $R_\omega$  of weighted Jacobi method.

$$\lambda(R_\omega) = 1 - \frac{\omega}{2}\lambda(A).$$

The notation  $\lambda(R_\omega)$  means the eigenvalues of the matrix  $R$ . The eigenvalues of the original matrix,  $A$  are

$$\lambda_k(A) = 4 \sin^2 \left( \frac{k\pi}{2n} \right), \quad 1 \leq k \leq n - 1,$$

where symbolically  $\lambda_k(A)$  represents the  $k^{th}$  eigenvalue of  $A$  and the eigenvectors are given by,

$$w_{k,j} = \sin \left( \frac{jk\pi}{n} \right),$$

where  $w_{j,k}$  is the  $j^{th}$  component of the  $k^{th}$  eigenvector. The reason of choosing such system is that the eigenvectors of  $A$  are simply the Fourier modes. The eigenvalues of  $R$  are,

$$\lambda_k(R_\omega) = 1 - 2\omega \sin^2 \left( \frac{k\pi}{2n} \right),$$

and the eigenvectors of  $R_\omega$  are same as the eigenvectors of  $A$ . The importance of eigenvector is that any vector can be represented as linear combination of the eigenvectors. Let us represent  $e^{(0)}$ , the initial error associated with the initial guess in the weighted Jacobi method, using eigenvectors, as,

$$e^{(0)} = \sum_{k=1}^{n-1} c_k w_k.$$

The coefficients  $c_k \in \mathbb{R}$  gives the amount of each mode in the error. After  $m$  iteration sweeps of weighted Jacobi method,

$$\begin{aligned} e^{(m)} &= R_\omega^m e^{(0)} \\ &= R_\omega^m \sum_{k=1}^{n-1} c_k w_k \\ &= \sum_{k=1}^{n-1} c_k R_\omega^m w_k \\ &= \sum_{k=1}^{n-1} c_k \lambda_k(R_\omega) w_k \end{aligned}$$

Since the Fourier modes are the eigenvectors of  $R$ , multiplication by the matrix  $R$  does not convert a mode to another mode, rather it can only change the amplitude (given by the coefficients of the linear combination) of the mode. That is, the weighted Jacobi method, as well as, other relaxation methods does not convert a single mode to another mode. So, comparing with modes (the eigenvectors) in the expansion of  $e^{(m)}$  with that of  $e^{(0)}$  we conclude that after  $m$  iteration sweeps, the  $k^{th}$  mode of the initial error has been reduced by a factor of corresponding eigenvalue of the iteration matrix,  $\lambda_k(R)$ . Now observe that for  $0 < \omega < 1$  we must have  $|\lambda_k(R)| < 1$  and that concludes the convergence of the weighed Jacobi method. For faster convergence we have to find  $\omega$  that makes  $|\lambda_k(R)|$  as small as possible for all  $1 \leq k \leq n - 1$ .

Let us consider the smoothest mode. The eigenvalue associated with smoothest mode is  $\lambda_1$ ,

which is given by,

$$\begin{aligned}\lambda_1 &= 1 - 2\omega \sin^2\left(\frac{\pi}{2n}\right) \\ &= 1 - 2\omega \sin^2\left(\frac{\pi h}{2}\right) \\ &\approx 1 - \frac{\omega\pi^2 h^2}{2}\end{aligned}$$

Considering  $h$  is small, the above expression of  $\lambda_1$  shows roughly that  $\lambda_1$  will be close to 1 for any choice of  $\omega$ . So, no value of  $\omega$  will reduce the smooth components of the error efficiently. A general sense of increasing the convergence rate of such iteration technique is by decreasing the grid spacing  $h$  that is by increasing the the number of grid points  $n$ . But with decreasing of  $h$ , by the above expression,  $\lambda_1$  will be closer to 1 and that will worsen the convergence of smooth components.

We take the components of the initial vector with  $\frac{n}{2} \leq k \leq n-1$  as the oscillatory. Smoothing factor represents the amount of damping experienced by high frequency Fourier modes. The weighted Jacobi method give high damping factor with  $\omega = \frac{2}{3}$ . To find the damping factor of weighted Jacobi method with weight,  $\omega = \frac{2}{3}$  we have to find the maximum value of the following function.

$$\left|\lambda_k(R_{\frac{2}{3}})\right| = \left|1 - \frac{4}{3} \sin^2\left(\frac{k\pi}{2n}\right)\right|, \quad \frac{n}{2} \leq k \leq n-1. \quad (2.20)$$

The inequality  $\frac{n}{2} \leq k \leq n-1$  in the equation (2.20) leads to  $\frac{\pi}{4} \leq \frac{k\pi}{2n} \leq \frac{\pi}{2} - \frac{\pi}{2n}$ . As  $n \rightarrow \infty$  the inequality become  $\frac{\pi}{4} \leq \frac{k\pi}{2n} \leq \frac{\pi}{2}$ . For notational simplicity let  $f(\theta) = 1 - \frac{4}{3} \sin^2 \theta$  with  $\frac{\pi}{4} \leq \theta \leq \frac{\pi}{2}$ . To find maximum value of  $f(\theta)$  we set  $f'(\theta) = 0$  and that leads to,

$$\begin{aligned}\sin(2\theta) &= 0 \\ \Rightarrow 2\theta &= n\pi, \quad n = 0, \pm 1, \pm 2, \dots \\ \Rightarrow \theta &= \frac{\pi}{2}n. \quad n = 0, \pm 1, \pm 2, \dots\end{aligned}$$

Thus  $|f(\theta)|$  is bounded by  $|f(\frac{\pi}{4})|$  or  $|f(\frac{\pi}{2})|$

$$\begin{aligned}
\left| \lambda_k \left( R_{\frac{2}{3}} \right) \right| &\leq \max \left( \left| f\left(\frac{\pi}{4}\right) \right|, \left| f\left(\frac{\pi}{2}\right) \right| \right) \\
&\leq \max \left( \left| 1 - \frac{4}{3} \times \frac{1}{2} \right|, \left| 1 - \frac{4}{3} \right| \right) \\
&\leq \max \left( \frac{1}{3}, \frac{1}{3} \right) \\
&= \frac{1}{3}.
\end{aligned}$$

We conclude from here that the oscillatory components are reduced at least by a factor  $\frac{1}{3}$  with each relaxation sweep. The damping factor of any relaxation technique is also called the smoothing factor of the technique. Another observation from this discussion is that the damping factor does not depend on the mesh size  $h$  of a grid. So the relaxation techniques have same strength in all the grids. So, the attempt of relaxing in comparatively coarser grid does not put us in a risk of experiencing less damping by each relaxation sweep.

What happens if we choose  $\omega$  in a way to damp the comparatively smooth modes is now an important question. Let us find the value of  $\omega$  for which the smoothest component will be reduced by a factor of  $\lambda \sim \frac{1}{2}$ .

$$\begin{aligned}
\lambda_1 &\approx 1 - \frac{\omega\pi^2h^2}{2} \\
\Rightarrow 1 - \frac{\omega\pi^2h^2}{2} &= \frac{1}{2} \\
\Rightarrow \frac{\omega\pi^2h^2}{2} &= \frac{1}{2} \\
\Rightarrow \omega &= \frac{1}{\pi^2h^2}.
\end{aligned}$$

With this value of  $\omega$  the  $k^{th}$  eigenvalue corresponding to the component with wavenumber  $k$  become,

$$\lambda_k(R_\omega) = 1 - 2 \left( \frac{1}{\pi^2 h^2} \sin^2 \left( \frac{k\pi}{2n} \right) \right) = \frac{2n^2}{\pi^2} \sin^2 \left( \frac{k\pi}{2n} \right).$$

So, for the high frequency components when with  $k = n - 1$ ,

$$\lambda_{n-1}(R_\omega) = 1 - \frac{2n^2}{\pi^2} \sin^2 \left( \frac{\pi}{2} - \frac{\pi}{2n} \right) \approx 1 - \frac{2n^2}{\pi^2}.$$

When  $n > \pi$  we get,

$$\lambda_{n-1}(R_\omega) < -1.$$

Thus,  $|\lambda_{n-1}(R_\omega)| > 1$  when  $n > \pi$ . That is high frequency component will grow if we choose  $\omega$  to reduce the smooth components.

Let us see the fact with an example. The weighted Jacobi method is applied to the model problem (2.17) on a grid with  $n = 64$  points. The initial guess are single modes with wavenumbers  $k = 3, k = 16$

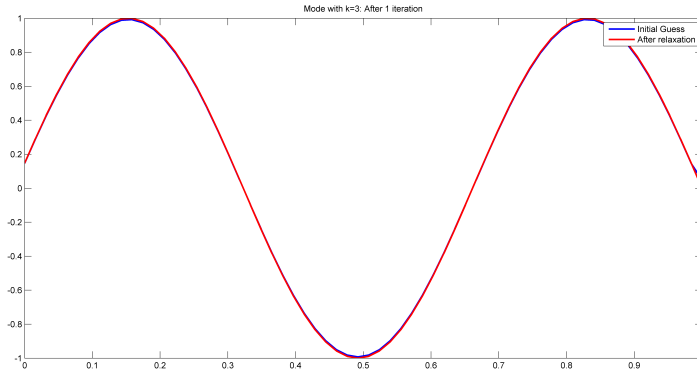


Figure 2.2: Single smooth mode after 1 iteration.

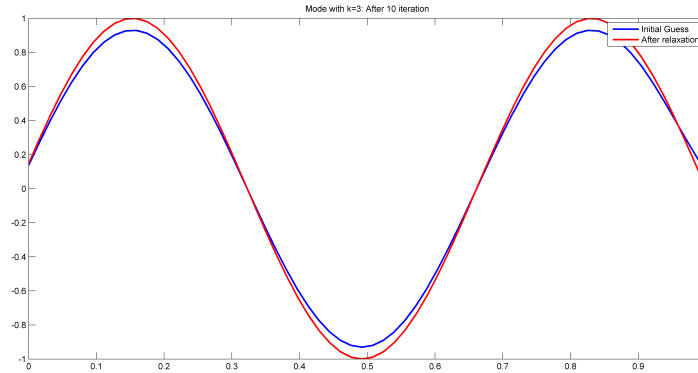


Figure 2.3: Single smooth mode after 10 iteration.

Figure 2.2 and Figure 2.3 show the effect of weighted Jacobi relaxation with  $\omega = \frac{2}{3}$  to the initial guess consist of the Fourier mode of wavelength  $k = 3$ . Figure 2.2 shows the effect of one iteration and Figure 2.3 shows the effect ten iterations. Since the initial guess is smooth it was damped very slowly.

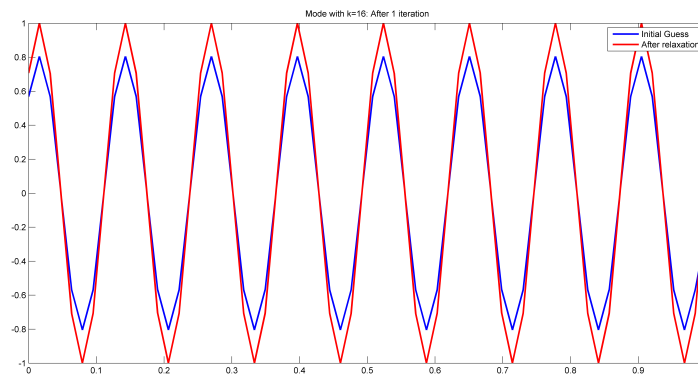


Figure 2.4: Oscillatory mode after 1 iteration.

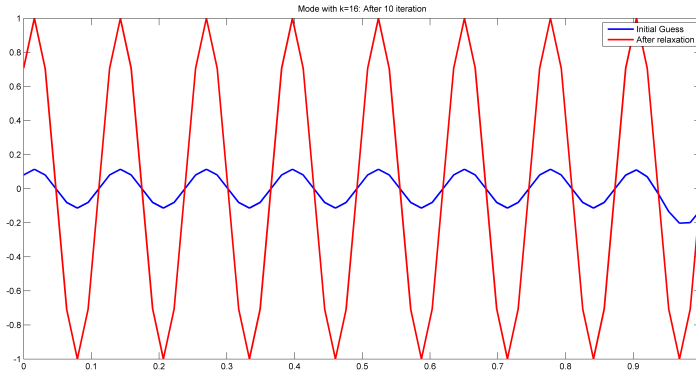


Figure 2.5: Oscillatory mode after 10 iterations.

Figure 2.4 and Figure 2.5 show the effect of weighted Jacobi relaxation with  $\omega = \frac{2}{3}$  to the initial guess consist of the Fourier mode of wavelength  $k = 16$  after one iteration (Fig. 2.4) and after ten iterations (Fig. 2.5). Damping process is faster here with relatively oscillatory initial guess.

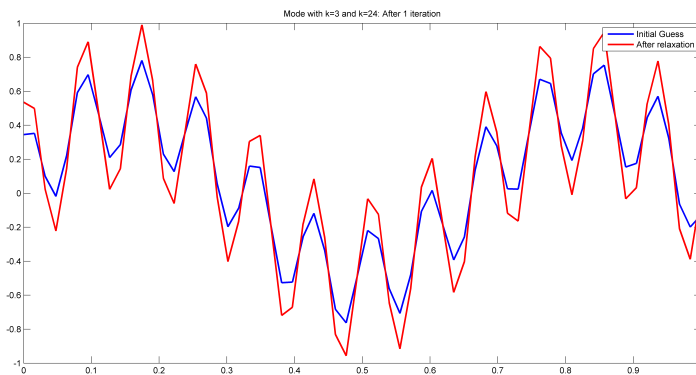


Figure 2.6: Mixed Fourier modes after 1 iteration.

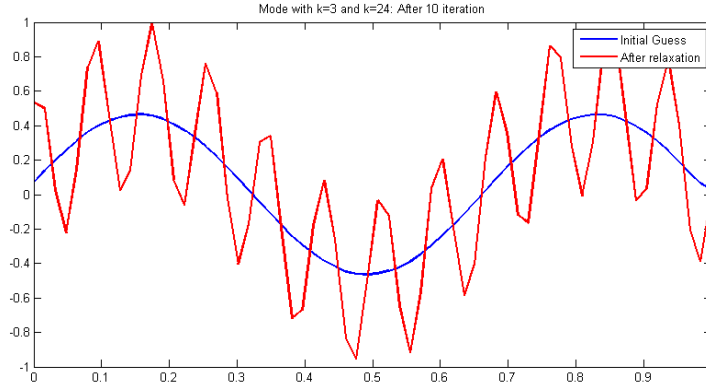


Figure 2.7: Mixed Fourier modes after 10 iterations.

Figure 2.6 and Figure 2.7 show the effect of weighted Jacobi relaxation with  $\omega = \frac{2}{3}$  to the initial guess consist of the Fourier modes of wavelength  $k = 3$  and  $k = 24$ . Figure 2.6 shows the effect of one iteration and Figure 2.7 illustrates the effect of ten iterations. The oscillatory component with  $k = 24$  is damped very fast while the smooth component with  $k = 3$  is damped slowly.

## 2.4 Two-Grid Correction

We start our discussion with the two grid correction scheme. Suppose  $u$  is an exact solution and  $v$  is an approximation of the exact solution  $u$  of the system,

$$Au = f. \tag{2.21}$$

Then  $u = v + e$  where  $e$  is the error  $e = u - v$  associated with the approximation  $v$ .

Recall the relation between the error and the residual given by  $Ae = r$ . So, we can relax directly on the error by using the residual equation and then correct the approximation to get better approximation and continue this process until we are satisfied if the process converges. The multigrid come to play in this position. Instead of relaxing or solving the

residual equation on the fine grid, the residual is transferred to the coarser grid, relaxed or solved there and then transferred back to the finer grid and then make the correction. This process gives the Two-grid correction scheme. The standard two grid algorithm is as follows,

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

- Relax  $\nu_1$  times on  $A^h u^h = f^h$  with initial guess  $v^h$ .
- Compute the fine-grid residual  $r^h = f^h - A^h v^h$  and send the residual to the coarse grid.
- Solve  $A^{2h} e^{2h} = r^{2h}$  on  $\Omega^{2h}$ .
- Transfer the coarse-grid error to the fine grid and correct the fine grid approximation by  $v^h \leftarrow v^h + e^h$ .
- Relax  $\nu_2$  times on  $A^h u^h = f^h$  on  $\Omega^h$  with initial guess  $v^h$ .

We have discussed the relaxation schemes and now we need to define the inter-grid transfer operators which are very important part of the multigrid schemes.

## 2.5 Elements of Multigrid

Our goal is to solve a system of linear equations. Direct methods such as Gaussian Elimination or LU factorization solve a system resulting in the exact solution. The computational complexity to solve an  $n \times n$  system in direct methods is  $O(n^3)$ . In case of band matrices with bandwidth  $k$  the computational cost is  $O(n^2 k)$ . In practice we have to solve large matrices i.e.  $n \gg 1$ . Computational cost for direct solve is very high for this kind of large matrices. The iterative methods such as Jacobi and Gauss-Seidel, approximate the solution with computational complexity of  $O(n)$  per single iteration. As we discussed in section 2.3, the oscillatory component in the error vector is eliminated by first few iteration of these

relaxation schemes leaving the smooth modes. Due to the so-called smoothing property these methods have the limitation of getting less effective in reducing the remaining smooth components. Multigrid methods are the remedy to overcome this limitation of these iterative techniques.

Assume that after applying few relaxation sweeps we have smooth components in the error vector on the fine grid. The idea of multigrid is to send the residual to the coarser grid. Having solved the residual equation in the coarser grid it is then again transferred to the fine grid to update previous approximation. So we need tools for the inter grid transfer operators. In this section we will talk about these operators namely **restriction** that transfers a vector from a fine grid to a coarser grid and **interpolation** or **prolongation** that transfers a vector from a coarse grid to finer grid. Note that  $h$  and  $2h$  in the notations  $u^h$  or  $u^{2h}$  indicate the grid on which the particular vector or matrix is defined. That is,  $u^h$  represents the approximation on the finer grid  $\Omega^h$  with mesh size  $h$  and  $u^{2h}$  represents the approximation on the coarser grid  $\Omega^{2h}$  with mesh size  $2h$ .

### 2.5.1 Interpolation

A coarse grid vector has  $\frac{n}{2}$  components and we need to transfer it to the finer grid with  $n$  grid points. Interpolation is the numerical technique of constructing new data points within the range of a discrete set of known data points. In this work, two types of interpolation operators are employed. In this section we introduce the linear interpolation. For most multigrid processes the linear interpolation is quite effective.

A linear interpolation operator, denoted by  $I_{2h}^h$ , symbolically means a transformation that takes a vector from a coarse grid  $\Omega^{2h}$  with mesh size  $2h$  and produces a vector on the finer grid  $\Omega^h$  with mesh size  $h$ . The transformation is defined as,

$$I_{2h}^h v^{2h} = v^h,$$

where,

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

$$v_{2j+1}^h = \frac{1}{2}(v_j^{2h} + v_{j+1}^{2h}), \quad 0 \leq j \leq \frac{n}{2} - 1.$$

That is at even numbered fine grid points, the values of the vector are transferred directly from  $\Omega^{2h}$  to  $\Omega^h$ . At odd numbered fine grid points, the value of  $v^h$  is the average of the adjacent coarse grid values. The following figure shows a linear interpolation of a vector from a coarse ( $n = 6$ ) grid  $\Omega^{2h}$  to the finer ( $N = 12$ ) grid  $\Omega^h$ .

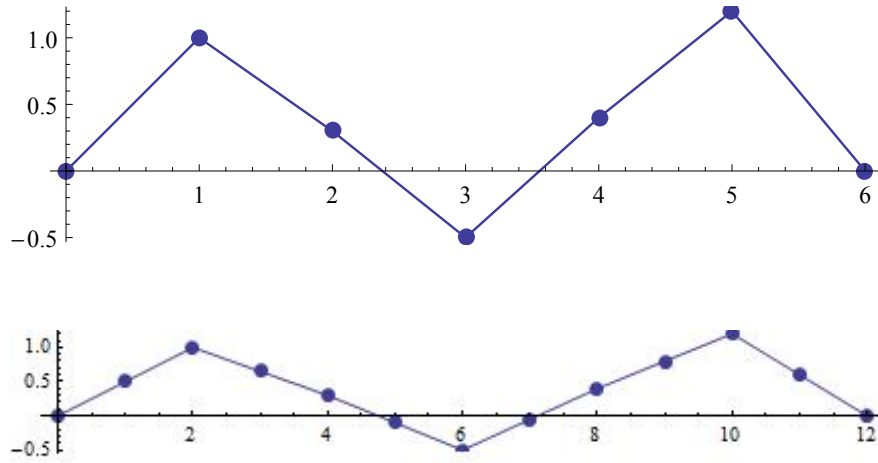


Figure 2.8: Linear Interpolation.

Consider the coarse grid  $\Omega^{2h}$  consists of four subintervals. We want to interpolate the vector  $[v_0^{2h}, v_1^{2h}, v_2^{2h}, v_3^{2h}, v_4^{2h}]^T$  in the coarse grid  $\Omega^{2h}$  to the fine grid  $\Omega^h$  with eight subintervals. The end points are interpolated by injection. In matrix notation the transformation can be written as,

$$I_{2h}^h v^{2h} = \frac{1}{2} \begin{bmatrix} 1 \\ 2 \\ 1 & 1 \\ 2 \\ 1 & 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} v_1^{2h} \\ v_2^{2h} \\ v_3^{2h} \end{bmatrix}_{2h} = \begin{bmatrix} v_1^h \\ v_2^h \\ v_3^h \\ v_4^h \\ v_5^h \\ v_6^h \\ v_7^h \end{bmatrix}_h$$

where  $[v_0^h, v_1^h, \dots, v_8^h]^T$  is the vector in the fine grid  $\Omega^h$ .

## 2.5.2 Restriction

Injection and full weighting are commonly used restriction operators. Injection is the process of assigning the fine grid values to the corresponding coarse grid point. The value of the  $(2j)^{th}$  component of the fine grid vector is assigned to the  $j^{th}$  component of the coarse grid vector. Assume that  $n$  is even and the fine grid vector is  $[v_0^h, v_1^h, v_2^h, \dots, v_n^h]$ . The injected vector  $[v_0^{2h}, v_{12h}, \dots, v_{\frac{n}{2}^{2h}}]$  on the coarse grid is then given by,  $v_0^{2h} = v_0^h, v_1^{2h} = v_2^h, \dots, v_j^{2h} = v_{2j}^h, \dots, v_{\frac{n}{2}^{2h}} = v_n^h$ .

Consider a fine grid error vector that consist of Fourier mode  $\omega_{k,j}^h \sin\left(\frac{jk\pi}{n}\right)$  with wavenumber  $1 \leq k \leq \frac{n}{2}$  i.e. the smooth component. The  $j^{th}$  grid point in the coarse grid  $\Omega^{2h}$  is the  $2j^{th}$  component of the fine grid  $\Omega^h$ . So,

$$\omega_{k,2j}^h = \sin\left(\frac{2jk\pi}{n}\right) = \sin\left(\frac{jk\pi}{n/2}\right) = \omega_{k,j}^{2h}, \quad 1 \leq k \leq \frac{n}{2}.$$

The Fourier mode  $\omega_{k,j}^{2h}$  is more oscillatory than  $\omega_{k,2j}^h$  provided that  $1 \leq k \leq \frac{n}{2}$ . We conclude that the smooth components in the error on the fine grid become more oscillatory in the



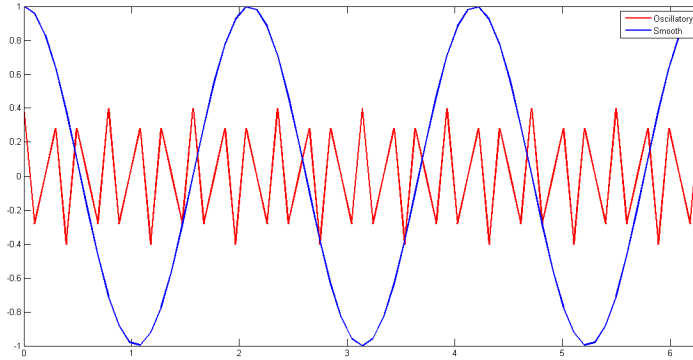


Figure 2.9: Fine grid  $\Omega^h$ .

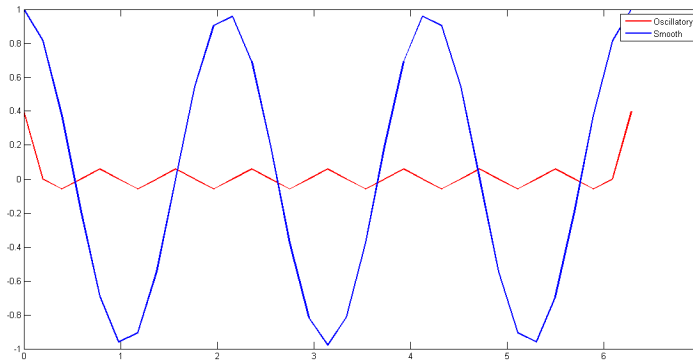


Figure 2.10: Coarse grid  $\Omega^{2h}$ .

Full weighting restriction transfers a smooth curve in the fine grid to comparatively oscillatory curve in the coarse grid but does not have significant effect on the amplitude. But the operator significantly change amplitude of a oscillatory curve.

## 2.6 Numerical Example for $-u_{xx} = 0$

To visualize the effects of the multigrid components along with the smoothing property of the relaxation techniques we consider a model problem,  $Av = 0$  where  $A$  is a  $63 \times 63$  matrix

resulted from discretization of the model problem (2.2) with  $\sigma = 0$  and  $f = 0$ ,

$$A = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & 0 & 0 & \dots & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & \dots & \dots & -1 & 2 \end{bmatrix}$$

resulted from discretization of (2.2) with the boundary conditions,  $u(0) = u(1) = 0$  in  $n = 64$  grid  $\Omega^h$  with mesh size  $h = \frac{1}{64}$ .

As an initial guess, we take a vector that consist of two Fourier modes with wavenumber  $k = 16$  and  $k = 40$ .

$$v_j^h = \frac{1}{2} \left[ \sin \left( \frac{16j\pi}{n} \right) + \sin \left( \frac{40j\pi}{n} \right) \right].$$

In this example we use the Weighted Jacobi method with weight,  $\omega = \frac{2}{3}$  as relaxation and linear interpolation and full weighting restriction as intergrid transfer operations. According to the two grid algorithm we compute the following steps

- Relax three times on  $A^h v^h = 0$  on  $\Omega^h$  with initial guess  $\tilde{v}^h$ .
- Compute residual in the fine grid and restrict it to the coarser grid,  $r^{2h} = I_{2h}^h r^h$
- In the coarse grid solve the residual equation  $A^{2h} e^{2h} = r^{2h}$ .
- Interpolate the error to the fine grid and correct the fine-grid approximation:  $\tilde{v}^h \leftarrow \tilde{v}^h + I_{2h}^h e^{2h}$ .

We repeat the process once. The results of this calculation are given in the following figures.

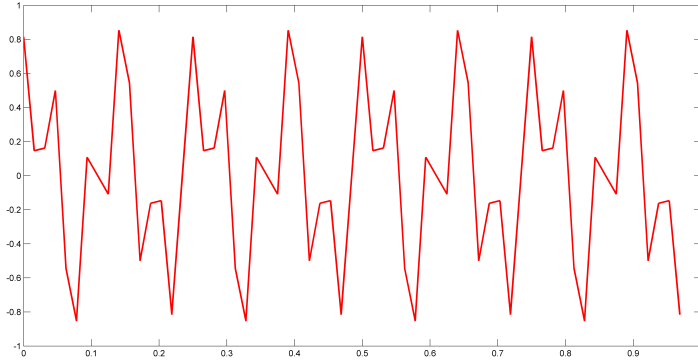


Figure 2.11: Initial Guess.

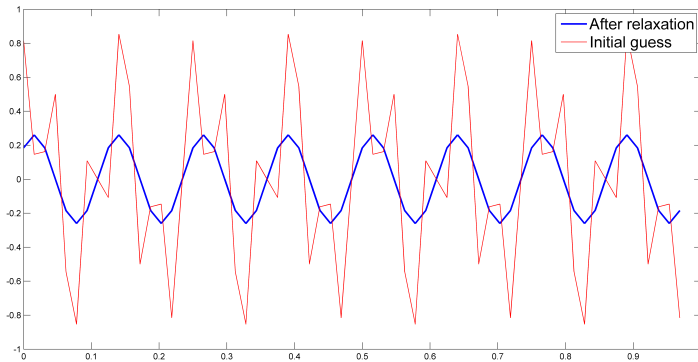


Figure 2.12: After relaxation.

Figure 2.11 shows the initial guess that consists of Fourier modes of wavenumbers  $k = 12$  and  $k = 40$ . Figure 2.12 shows approximation  $\tilde{v}^h$  after three relaxation sweeps. Much of the oscillatory component of the initial guess has already been removed. Also it is smooth enough to apply the restriction operator to send the residual vector to the coarser grid  $\Omega^{2h}$ .

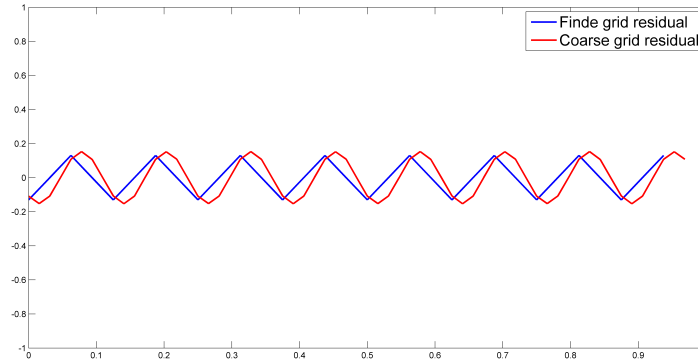


Figure 2.13: Residual on the fine grid and the coarse grid.

Figure 2.13 shows the residual on the fine grid  $\Omega^h$  and on the coarse grid  $\Omega^{2h}$ . The residual is more oscillatory on the coarse grid than that on fine grid though the wavenumber is same. The relaxation works better in the coarser grid residual than the fine grid residual having more oscillatory component.

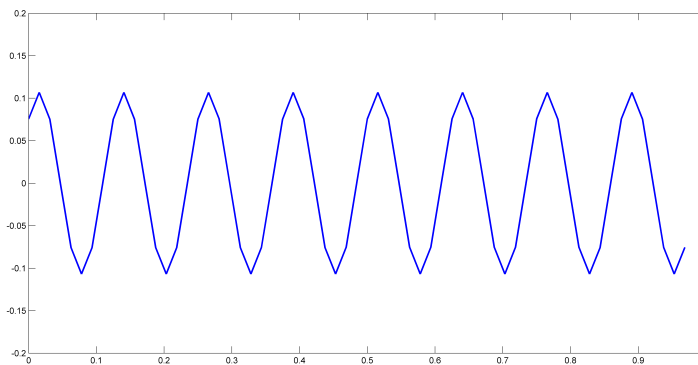


Figure 2.14: Error after one V-Cycle.

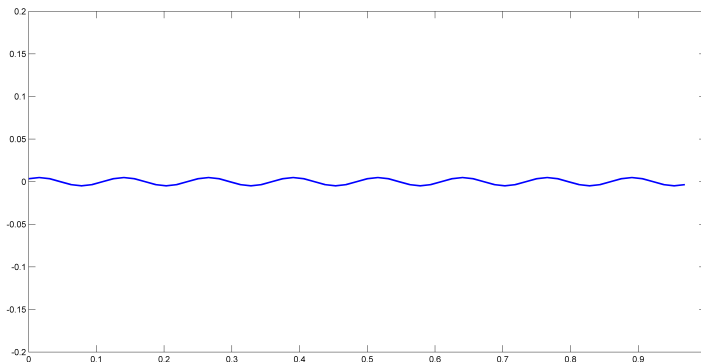


Figure 2.15: Error after two V-Cycles.

If we keep applying the weighted Jacobi in fine grid after some relaxation sweeps much of the oscillations will be reduced after some sweeps and the smoother component will slow down the smoothing process and takes more relaxation sweeps.

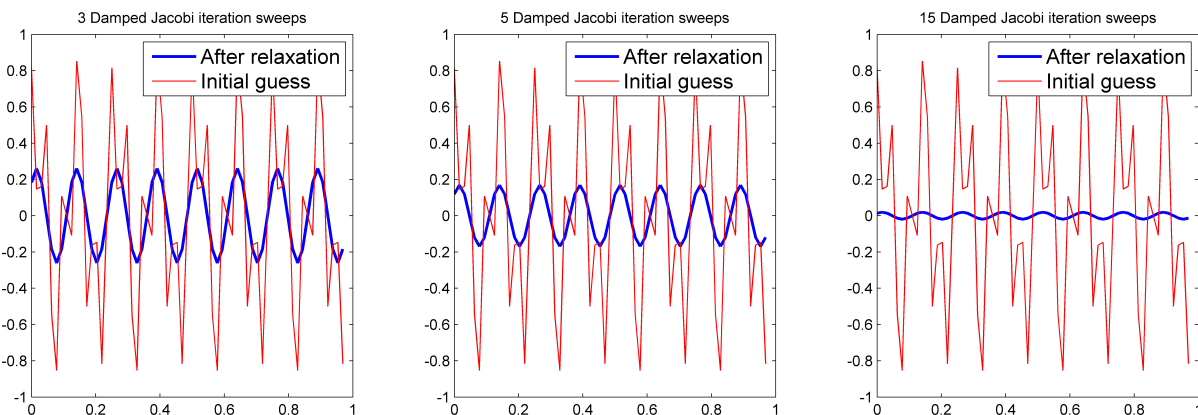


Figure 2.16: From left to right the figures shows the effect of the weighted Jacobi method with weight  $\omega = \frac{2}{3}$  after three, five and fifteen relaxation sweeps.

## 2.7 V-Cycle Scheme

One of the limitations of two grid method is that, for very large finest grid the coarsest grid is still large since it is just one half of the finest grid. Direct solve of the equation  $A^{2h}e^{2h} = r^{2h}$  in the coarse grid  $\Omega^{2h}$  is still very costly. To get a significant reduction in computation we need to make the coarsest grid as small as possible. One way to reach to a

very coarse grid is by embedding a two grid correction scheme in to itself. The corresponding algorithm is known as  $V$  cycle.

For notational simplicity some changes is necessary. The equation to be solve in the coarse grid  $\Omega^{2h}$  is  $A^{2h}e^{2h} = r^{2h}$  which is notationally similar to the original problem  $A^h\tilde{v}^h = f^h$  in the fine grid. Instead of  $r^{2h}$  and  $e^{2h}$  in the coarse grid iteration we use  $f^{2h}$  and  $\tilde{v}^{2h}$  respectively. So, in the coarse grid equation  $f^{2h}$  means the residual and  $\tilde{v}^{2h}$  means the error vector. This notation is very helpful for computer implementation of the following algorithms.

### V-Cycle Scheme

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

- Relax on  $A^h u^h = f^h$ ,  $\nu_1$  times with initial guess  $v^h$
- Compute the residual in the fine gird and restrict it to the coarser grid, i.e.  $f^{2h} = I_h^{2h} r^h$ 
  - Relax on  $A^{2h} u^{2h} = f^{2h}$   $\nu_1$  times with the initial guess  $v^{2h} = 0$
  - Compute  $f^{4h} = I_{2h}^{4h} r^{2h}$ 
    - \* Relax on  $A^{4h} u^{4h} = f^{4h}$ ,  $\nu_1$  times with initial guess  $v^{4h} = 0$
    - \* Compute  $f^{8h} = I_{4h}^{8h} r^{4h}$
    - .
    - .
    - Solve  $A^{2^L h} u^{2^L h} = f^{2^L h}$
    - .
    - .
    - \* Correct  $v^{4h} \leftarrow v^{4h} + I_{8h}^{4h} v^{8h}$

- \* Relax on  $A^{4h}u^{4h} = f^{4h}$ ,  $\nu_2$  times with initial guess  $v^{4h}$ .
- Correct  $v^{2h} \leftarrow v^{2h} + I_{4h}^{2h}v^{4h}$
- Relax on  $A^{2h}u^{2h} = f^{2h}$ ,  $\nu_2$  times with initial guess  $v^{2h}$ .
- Correct  $v^h \leftarrow v^h + I_{2h}^h v^{2h}$
- Relax on  $A^h u^h = f^h$ ,  $\nu_2$  times with initial guess  $v^h$ .

This algorithm telescopes down to the coarsest grid. Such a coarsest grid can consist of one or a few interior grid points. Direct solve in such a small grid is very cost effective. Getting the exact solution in the coarsest grid the algorithm then works its way back to the fine grid. The following figure shows the computational path of the algorithm and for the pattern in this diagram it is known as the  $V - cycle$ .

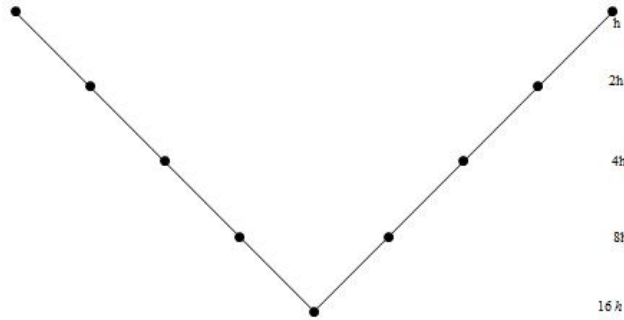


Figure 2.17

A  $V - cycle$  can also be designed recursively, which is given as follows.

**$V - cycle$  Scheme (Recursive Definition)**

1. Relax  $\nu_1$  times on  $A^h v^h = f^h$  with a given initial guess  $\tilde{v}^h$ .
2. If  $\Omega^h =$  Coarsest grid,

then go to step 4.

Else

$$f^{2h} \leftarrow I_h^{2h}(f^h - A^h \tilde{v}^h)$$

$$\tilde{v}^{2h} \leftarrow 0$$

$$\tilde{v}^{2h} \leftarrow V^{2h}(\tilde{v}^{2h}, f^{2h}).$$

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

4. Relax  $\nu_2$  times on  $A^h v^h = f^h$  with initial guess  $\tilde{v}^h$

Some other multigrid cycles are shown geometrically in the following figure.

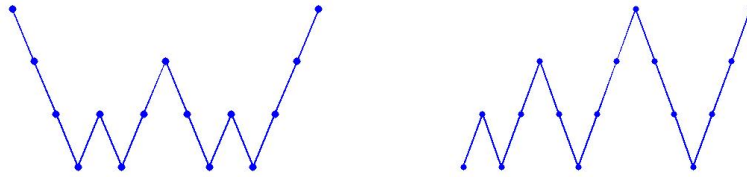


Figure 2.18: W cycle and Full Multigrid

There are many others multigrid schemes available. Detailed overview of multigrid methods and their applications can be found in [8],[12],[21].

# Chapter 3

## Integral Equations

### 3.1 Introduction

An integral equation is an equation in which an unknown function appears under an integral. There is a close connection between differential and integral equations, and some problems may be formulated either way. Practically most of the boundary value problems and initial value problems in differential calculus can be formulated as integral equations, and this conversion often gives easier and more efficient solution to the problem. One of the reasons is that the truncation errors of the solution tends to be averaged out by the process of quadrature while they tend to accumulate during the process of numerical integration employed in the solution of differential equations.

Integral equations are roughly divided into two categories:

- Fredholm Integral Equation;
- Volterra Integral Equation.

The difference between the two is defined by the domain of integration, namely, integral equations in which the integration domain vary with the independent variable are the Volterra

integral equations

$$u(x) = \int_a^x K(x, y)u(y)dy + f(x), \quad (3.1)$$

and the ones in which the integration domain is fixed are Fredholm integral equations

$$u(x) = \int_a^b K(x, y)u(y)dy + f(x). \quad (3.2)$$

In both cases,  $u(x)$  is the unknown function to be determined,  $f(x)$  is a known function and  $K(x, y)$  is a known function of two variables, called the **kernel**. Equations (3.1) and (3.2) are called the integral equation of the second kind. If  $f(x) \equiv 0$ , (3.1) and (3.2) are called the integral equation of the first kind.

There are various methods to solve integral equations. In the analytical setting these includes Adomian decomposition method [1], successive substitutions[13], Laplace transformation method[13], Picard's method are popular. We, however, are interested in the numerical solution of the integral equations.

## 3.2 Discretization

The discretization process starts with evaluating the integral numerically. The techniques of numerical integration involve a partition of the domain of integration, known as a mesh. If  $D = [a, b]$ , the partition is a set of points  $\{x_j, j = 0, \dots, n\}$  with  $x_0 = a$  and  $x_n = b$ . The points  $x_j$ , which are called the nodes, are selected in different ways in different numerical integration techniques. For example, Newton-Cotes formulas need equally spaced nodes, defined as,  $x_j = x_0 + ih$ , where  $h = \frac{b-a}{n}$  is the mesh size. Other choices are zeros of the Chebyshev polynomials.

A system of linear equations is then formed to evaluate the unknown function  $u(x)$  at the mesh points.

$$u = \mathcal{K}u + f.$$

A major difference between the discrete system resulted from an integral equation and that of a partial differential equation is that, unlike most of the partial differential equations, integral equations result in dense matrices. Consider the integral equation,

$$u(x) = \int_D K(x, y)u(y)dy + f(x), \quad y \in D. \quad (3.3)$$

The integral  $\int_D K(x, y)u(y)dy$  is replaced by a quadrature sum and that results in a system of linear equations where the unknowns are the values of the unknown function  $u(x)$  at the nodes. An approximation to the solution of the integral equation at some node is obtained solving the system. It is a common practice to use Newton-Cotes quadrature specially Trapezoidal rule or Gaussian quadrature in the discretization process.

### 3.2.1 Discretization Using Trapezoidal Rule

Consider a partition of the domain  $D$  of integration in (3.3) given by equidistant mesh points  $\{y_0, y_1, \dots, y_n\}$ . Using the Trapezoidal rule, the integral can be approximated as

$$\int_D K(x, y)u(y)dy \approx T_n(x) = \frac{1}{2}K(x, y_0)u(y_0) + \sum_{j=1}^{n-1} K(x, y_j)u(y_j) + \frac{1}{2}K(x, y_n)u(y_n).$$

$T_n(x)$  represents the approximation of the integral using Trapezoidal rule on a mesh of  $n$  subintervals. Corresponding to the approximation  $T_n(x)$  of the integral (3.3) let  $u_n$  be the approximation of the unknown function  $u(x)$ . So,  $u_n(x)$  is the exact solution of the following equation obtained by replacing the integral by  $T_n(x)$  and  $u$  by  $u_n$ :

$$u_n(x) = \frac{1}{2}K(x, y_0)u_n(y_0) + \sum_{j=1}^{n-1} K(x, y_j)u_n(y_j) + \frac{1}{2}K(x, y_n)u_n(y_n) + f(x), \quad x \in D. \quad (3.4)$$

As  $n \rightarrow \infty$ ,  $T_n(x) \rightarrow \int_D K(x, y)u(y)dy$ . The exact solution  $u_n(x)$  of equation (3.4) is an approximation to the unknown function  $u(x)$  and  $u_n(x) \rightarrow u(x)$  as  $n \rightarrow \infty$ . Evaluate the function  $u_n(x)$  at the nodes  $\{y_0, y_1, \dots, y_n\}$ ,

$$u_n(y_i) = \frac{1}{2}K(y_i, y_0)u_n(y_0) + \sum_{j=1}^{n-1} K(y_i, y_j)u_n(y_j) + \frac{1}{2}K(y_i, y_n)u_n(y_n) + f(y_i), \quad (3.5)$$

$$i = 0, 1, \dots, n.$$

Equation (3.5) represents a system of linear equations which can be written as,

$$u = \mathcal{K}u + f. \quad (3.6)$$

where,  $u = [u(y_0), u(y_1), \dots, u(y_n)]^T$  and  $f = [f(y_0), f(y_1), \dots, f(y_n)]^T$  and the Kernel matrix becomes,

$$\mathcal{K} = \begin{bmatrix} \frac{1}{2}K(y_0, y_0) & K(y_0, y_1) & \dots & \frac{1}{2}K(y_0, y_n) \\ \frac{1}{2}K(y_1, y_0) & K(y_1, y_1) & \dots & \frac{1}{2}K(y_1, y_n) \\ \dots & \dots & \dots & \dots \\ \frac{1}{2}K(y_n, y_0) & K(y_n, y_1) & \dots & \frac{1}{2}K(y_n, y_n) \end{bmatrix}$$

### 3.2.2 Discretization using Gaussian Quadrature

Gaussian quadrature is another technique to approximate a definite integral, formulated as,

$$\int_a^b g(x)dx \approx \sum_{j=1}^n w_j g(x_j). \quad (3.7)$$

An  $n$ -point Gaussian quadrature rule, named after Carl Friedrich Gauss, is a quadrature rule constructed to yield an exact result for polynomials of degree  $2n-1$  or less by a suitable choice of the points  $x_j$  and weights  $w_j$  for  $j = 1, \dots, n$ . Unlike the Newton-Cotes approach, the node points  $x_j$  are not chosen to be equally spaced rather they are chosen together with the weights to minimize the approximate error using the Legendre polynomials. The following theorem defines the choices  $x_j$  and weights  $w_j$ :

**Theorem 5.** *Suppose that  $x_1, x_2, \dots, x_n$  are the roots of the  $n$ th Legendre polynomial  $P_n(x)$  and that for each  $i = 1, 2, \dots, n$  the numbers  $w_i$  are defined by*

$$w_i = \int_{-1}^1 \prod_{j=1, j \neq i}^n \frac{x - x_j}{x_i - x_j} dx.$$

If  $P(x)$  is any polynomial of degree less than  $2n$ , then,

$$\int_{-1}^1 P(x)dx = \sum_{i=1}^n w_i P(x_i).$$

The nodes and the coefficients for  $n = 2, 3, 4, 5$  are given in the following table

Approximating the integral in equation (3.3) using Gaussian quadrature we get a system of linear equations in a similar manner as in Section 3.2.1.

### 3.3 Picard's Iteration

Picard's method is a successive approximation method to solve an integral equation.

$$u(x) = \int_a^b K(x, y)u(y)dy + f(x),$$

Table 3.1: Gaussian quadrature points and weights

n	$x_i$	$w_i$
2	0.5773502692	1.00000000
	-0.5773502692	1.00000000
3	0.7745966692	0.5555555556
	0.0000000000	0.8888888889
	-0.7745966692	0.5555555556
4	0.8611363116	0.3478548451
	0.3399810426	0.6521451549
	-0.3399810426	0.6521451549
	-0.8611363116	0.3478548451
5	0.9061798459	0.2369268850
	0.5384693101	0.4786286705
	0.0000000000	0.5688888889
	-0.5384693101	0.4786286705
	-0.9061798459	0.2369268850

The goal is to construct a sequence of successive approximations to  $u(x)$ . First choose an initial approximation  $u_0(x)$ . It is usual practice to choose  $u_0(x) = f(x)$ . Then define the sequence  $u_1(x), u_2(x), \dots, u_k(x)$  by,

$$\begin{aligned}
 u_1(x) &= \int_a^b K(x, y)u_0(y)dy + f(x), \\
 u_2(x) &= \int_a^b K(x, y)u_1(y)dy + f(x), \\
 &\dots \quad \dots \quad \dots \quad \dots \quad \dots \quad \dots \\
 u_k(x) &= \int_a^b K(x, y)u_{k-1}(y)dy + f(x)
 \end{aligned}$$

According to Picard's Existence theorem [18], under certain conditions on  $K(x, y)$ , there is a unique solution  $u(x)$  so that  $u(x) = \lim_{k \rightarrow \infty} u_k(x)$ . The following example illustrates Picard's method.

### 3.3.1 Example

Consider the integral equation

$$u(x) = \int_0^x 2t(1 + u(t))dt$$

As initial approximation we choose  $u_0(x) \equiv 0$ . We then get,

$$\begin{aligned}u_1(x) &= \int_0^x 2t(1 + u_0(t))dt = \int_0^x 2tdt = x^2, \\u_2(x) &= \int_0^x 2t(1 + u_1(t))dt = \int_0^x 2t(1 + t^2)dt = x^2 + \frac{1}{2}x^4, \\u_3(x) &= \int_0^x 2t(1 + u_2(t))dt = \int_0^x 2t(1 + t^2 + \frac{1}{2}t^4)dt = x^2 + \frac{1}{2}x^4 + \frac{1}{6}x^6, \\&\dots\dots\dots \\u_k(x) &= x^2 + \frac{x^4}{2} + \frac{x^6}{6} + \dots\dots\dots + \frac{x^{2k}}{k!}.\end{aligned}$$

We see that  $u(x) = \lim_{k \rightarrow \infty} u_k(x) = e^{x^2} - 1$ .

## 3.4 Picard's Method in a Discrete Setting

The widely used iterative method for solving (3.6) is the Picard's iteration, and it is also known as successive approximations, fixed point iteration. The Picard's iteration read,

$$u^{(k+1)} = \mathcal{K}u^{(k)} + f. \tag{3.8}$$

The Picard's iteration converges if  $\rho(\mathcal{K}) < 1$ , where  $\rho(\mathcal{K})$  is the spectral radius of the kernel matrix  $\mathcal{K}$  in the discretized equation (3.6). The spectral radius  $\rho(A)$  of a matrix  $A$  is defined by

$$\rho(A) = \max |\lambda|,$$

where  $\lambda$  is an eigenvalue of  $A$ .

### 3.4.1 Numerical Example

Consider the Fredholm integral equation

$$u(x) = \frac{1}{2} \int_0^{\frac{\pi}{2}} \sin xu(y)dy + \cos x. \quad (3.9)$$

We discretized the equation (3.9) using Trapezoidal rule with  $n = 16$ . The kernel has a spectrum radius  $\rho(\mathcal{K}) = 0.5634 < 1$ , the Picard's iteration converges and, because the spectrum radius is not small enough, the convergence is slow. For this example we use  $L_2$  norm to define the error,  $\|e_n\| = \|u_n - u\|$ , where  $u_n$  and  $e_n$  represents the approximation and the error respectively after  $n$  Picard's iterations, and  $u(x)$  is the exact solution given by  $u(x) = \sin x + \cos x$ .

Table 3.2: Convergence of Picard's iteration for equation (3.9).

Number of Pricar's iteration	5	10	15	20	25
Error	0.0953	0.0075	0.0048	0.0047	0.0047

The above table shows the slow convergence as expected. Let us consider another example. We take this example so that the spectral radius of the kernel is small enough for faster convergence.

$$u(x) = x + \int_0^1 \frac{x^2 - y^2}{10} u(y) dy.$$

We discretized the equation in a partition with 16 subintervals. The spectral radius  $\rho(\mathcal{K})$  of the kernel matrix is 0.0324. The error after 2 Picard's iteration is  $4.65 \times 10^{-4}$ .

## 3.5 Nyström Method

The Nyström method uses numerical integration of the integral operator in the equation

$$u(x) = \int_D K(x, y)u(y)dy + f(x), \quad y \in D. \quad (3.10)$$

The resulting solution is found first at the set of quadrature node points and then it is interpolated to all points in  $D$  using special interpolation scheme we discussed in this section. We assumed the kernel of the equation to be continuous.

For generalization let us consider the numerical integration scheme given by,

$$\int_D h(x)dx \approx \sum_{j=1}^{q_n} w_{n,j}h(x_{n,j}), \quad h \in C(D), x_{n,j} \in D.$$

In the numerical integration techniques it is assumed that for every  $h \in C(D)$ , the set of all continuous function in  $D$ , the numerical integrations converges to the true integral as  $n \rightarrow \infty$ . To simplify the notation, we will write  $w_{n,j}$  and  $x_{n,j}$  as  $w_j$  and  $x_j$  respectively. Using the above quadrature technique we approximate the integral in (3.10) to get a new equation:

$$u_n(x) = \sum_{j=1}^{q_n} w_j K(x, x_j)u_n(x_j) + f(x), \quad x \in D. \quad (3.11)$$

For  $x = x_j$ , (3.11) results in

$$u_n(x_i) = \sum_{j=1}^{q_n} w_j K(x_i, x_j)u_n(x_j) + f(x_i), \quad i = 1, \dots, q_n, \quad (3.12)$$

which is a system of linear equation of size  $q_n \times q_n$  with the unknown vector,

$$u_n = [u_n(x_1), u_n(x_2), \dots, u_n(x_{q_n})]^T.$$

Corresponding to each solution of (3.12), there is a unique solution of (3.11), that agrees at the node points. When given a discrete solution  $v = [v_1, v_2, \dots, v_{q_n}]^T$  to (3.12), a value

at  $x = x \neq x_i$  is given by

$$v(x) = \sum_{j=1}^{q_n} w_j K(x, x_j) v_j + f(x), \quad x \in D. \quad (3.13)$$

This is the Nyström interpolation formula. In the original paper [16], the author uses a highly accurate Gaussian quadrature formula with a very small number of quadrature nodes. He then uses (3.13) to extend the solution to all other points in the domain while retaining the accuracy found in the solution at the node points. The following example illustrates Nyström interpolation formula.

### 3.5.1 Example

Consider the integral equation

$$u(x) = \int_0^1 (1 - x \cos(xy)) u(y) dy + \sin(x), \quad 0 \leq x \leq 1, \quad (3.14)$$

with exact solution  $u(x) = 1$ .

First we approximate (3.14) using the three point Simpson rule with nodes  $\{0, 0.5, 1\}$ . The approximation we get

$$(\tilde{u}(0), \tilde{u}(0.5), \tilde{u}(1)) = (0.999869, 0.99992, 0.999667),$$

where  $\tilde{u}$  represents approximation using Simpson's rule. Then the errors approximation are given by

$$\begin{bmatrix} u(0) \\ u(0.5) \\ u(1) \end{bmatrix} - \begin{bmatrix} \tilde{u}(0) \\ \tilde{u}(0.5) \\ \tilde{u}(1) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0.999869 \\ 0.99992 \\ 0.999667 \end{bmatrix} = \begin{bmatrix} 0.000131 \\ 0.00008 \\ 0.00033 \end{bmatrix}.$$

If we interpolate using Nyström interpolation formula (3.13) to get the solution at the nodes  $\{0, 0.1, 0.2, \dots, 1\}$ ,

Table 3.3: Nyström interpolation with Simpson quadrature for equation(3.14).

$x$	approximation	error
0.	0.999869	0.000130669
0.1	0.999882	0.000117640
0.2	0.999895	0.000104921
0.3	0.999907	0.000093226
0.4	0.999916	0.000084059
0.5	0.999920	0.000080045
0.6	0.999915	0.000085201
0.7	0.999895	0.000105106
0.8	0.999853	0.000146977
0.9	0.999780	0.000219605
1.	0.999667	0.000333166

For comparison, we use Gauss-Legendre quadrature with three nodes,

$$\int_0^1 g(t)dt \approx \frac{1}{18} [5g(t_1) + 8g(t_2) + 5g(t_3)],$$

with,  $t_1 = \frac{1-\sqrt{0.6}}{2} \approx 0.11270167$ ,  $t_2 = 0.5$  and  $t_3 = \frac{1+\sqrt{0.6}}{2} \approx 0.88729833$  The error in solving 3.14 with Nyström method is now,

$$\begin{bmatrix} u(t_1) \\ u(t_2) \\ u(t_3) \end{bmatrix} - \begin{bmatrix} \tilde{u}(t_1) \\ \tilde{u}(t_2) \\ \tilde{u}(t_3) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0.9999998897 \\ 0.9999999305683689 \\ 0.9999997739907928 \end{bmatrix} = \begin{bmatrix} 1.10317 \times 10^{-7} \\ 6.94316 \times 10^{-8} \\ 2.26009 \times 10^{-8} \end{bmatrix}$$

which is much smaller than with Simpson's rule when using an equal number of node points. Interpolation of this more accurate approximation using the Nyström interpolation formula (3.13) gives more accurate approximation to the nodes  $\{0, 0.1, 0.2, \dots, 1\}$ .

Table 3.4: Nyström interpolation with Gaussian quadrature for equation(3.14).

x	approximation	error
0	0.99999988	$1.24 \times 10^{-7}$
0.1	0.99999989	$1.12 \times 10^{-7}$
0.2	0.99999990	$9.96 \times 10^{-8}$
0.3	0.99999991	$8.79 \times 10^{-8}$
0.4	0.99999992	$7.71 \times 10^{-8}$
0.5	0.99999993	$6.94 \times 10^{-8}$
0.6	0.99999993	$6.90 \times 10^{-8}$
0.7	0.99999991	$8.51 \times 10^{-8}$
0.8	0.99999987	$1.24 \times 10^{-7}$
0.9	0.99999976	$2.44 \times 10^{-7}$
1.0	0.99999954	$4.59 \times 10^{-7}$

Generally Gaussian Quadrature gives much better approximation than the Simpson's rule; but it results in the approximations being given at the Gauss-Legendre nodes, which is usually not a convenient choice for the subsequent use of the approximation. Quadratic interpolation can be used to extend the numerical solution to all other  $x \in [0, 1]$ , but it generally results in much larger errors.

# Chapter 4

## Multigrid Methods for Integral Equations

### 4.1 Introduction

Multigrid methods have been first developed for solving partial differential equations. The main idea of Multigrid scheme is to solve a system of linear equations resulted from discretization of the partial differential equations efficiently. Integral equations can also be discretized in a similar manner and so multigrid techniques are also applicable to solve integral equation numerically. There are some basic differences of the structure of differential and integral equations after discretization. Most importantly partial differential equations produce sparse matrix equation upon discretization while integral equation typically result in dense matrices yielding more expensive numerical solution. Secondly it is very important to choose the smoothing operation very carefully for a particular integral equation depending on the nature of the kernel function. In this chapter we discuss the development of Multigrid method for solving integral equations specially the Fredholm integral equations. We start with Atkinson-Brakhage two grid iteration technique.

## 4.2 Two Level Method

The study of two grid method begins with Atkinson-Brakhage algorithm [3],[6] for integral equations. First we will discuss the theoretical development of this method as introduced in the original paper. Consider the Fredholm integral equation of the second kind,

$$u(x) = \int_0^1 k(x,y)u(y)dy + f(x). \quad (4.1)$$

In our standard notation,  $k$  and  $f$  are given continuous functions and  $u \in \mathcal{C}[0,1]$  is the unknown function to be found. Equation (4.1) can also be written as,

$$u(x) = \mathcal{K}u(x) + g(x), \quad (4.2)$$

where  $\mathcal{K}$  represents the integral operator.

The operator  $\mathcal{K}$  in the discrete system corresponding the kernel  $K(x,y)$  is defined using the quadrature rules discussed in Section 2.2. The Atkinson-Brakhage algorithm begins with a sequence of quadrature rules, indexed by  $m$ , nodal points  $\{x_j^m\}_{j=1}^{N_m}$  and weights  $\{w_j^m\}_{j=1}^{N_m}$ . If the quadrature rule is convergent, that is,

$$\lim_{m \rightarrow \infty} \sum_{j=1}^{N_m} g(x_j^m)w_j^m = \int_0^1 g(x)dx,$$

then the sequence  $\{K_m\}$  defined by

$$K_m u(x) = \sum_{j=1}^{N_m} k(x, x_j^m)u(x_j^m)w_j^m,$$

is collectively compact [2] and convergent strongly to the operator  $K$ . With this approximation  $K_m$  of the operator  $K$ , the equation (4.2) can be written in the form

$$(I - K_m)u = f. \quad (4.3)$$

As discussed in Section 3.5, the equation (4.3) can be approximated by solving a discrete system,

$$u(x_i^m) - \sum_{j=1}^{N_m} k(x_i^m, x_j^m)u(x_j^m)w_j^m = f(x_i^m). \quad (4.4)$$

Solution of (4.4) will give the approximation of the unknown function  $u(x)$  at the nodal points and these can be used to obtain the approximation at an arbitrary  $x \in [0, 1]$  by Nyström interpolation (3.13). As discussed in [3], [15], the important consequences of collective compactness and strong convergence of  $K_m$  to  $K$  are,

1. There is  $M$  such that  $m \geq M$  implies  $I - K_m$  is nonsingular and  $(I - K_m)^{-1} \rightarrow (I - K)^{-1}$
2. For any  $\rho > 0$  there is  $M$  such that, if  $L \geq l \geq M$  the operator  $B_l^L = I + (I - K_l)^{-1}K_L$  satisfies

$$\|I - B_l^L(I - K_L)\| \leq \rho.$$

The first one implies that the solution of (4.3) converges uniformly to the solution of the original equation (4.1). The second one is the most important discovery in the Atkinson-Brakhage two-level method. In the notation  $B_l^L$ , the small letter  $l$  represents the coarse grid and the capital  $L$  represents the fine grid in the two grid scheme. The proposed two grid method basically solves the equation of the form  $u - K_L u = f$ . If we know the solution  $u_c$  in the coarse grid the transition to the solution in the fine grid is given by,

$$u_f = u_c - B_l^L(u_c - K_L u_c - f).$$

### 4.2.1 Application of Two Level Method

Generic multigrid technique is based on a smoothing process, a restriction and an interpolation operations. For partial differential equations, it is a common practice to use Jacobi like iterative process. Initially to handle integral equations, we use the Picard's iteration as smoothing step, as suggested by W. Hackbusch [12]. Two grid algorithm is as follows:

**Two grid iteration to solve  $u = Ku + f$ .**

Initial approximation  $u_0^h$

$$u_j^h := K^h u_j^h + f^h,$$

$$u_{j+1}^h = u_j^h - I_{2h}^h (I - K^{2h})^{-1} I_h^{2h} (u^h - K^h u_j^h - f^h),$$

$u_j^h$  is the  $j^{\text{th}}$  iteration defined on scale  $h$ .

The algorithm also can be defined recursively as follows:

Multigrid Cycle:  $MGM(l, u, f)$

The input: Number of levels,  $l$ ; current approximation  $u^h$

The output: Fine grid approximation,  $v^h$

If  $l = 0$  then  $u := (I - K^{2h})^{-1} f$  else

$$u := K^h u^h + f^h;$$

$$r := I_h^{2h} (u^h - K^h u^h - f);$$

$$v^{2h} := 0;$$

for  $i := 1, 2$  do  $MGM(l - 1, v, r)$ ;

$$u^h := u - I_{2h}^h \times v^{2h};$$

end;

### 4.3 Hemker-Schippers Modification

In this section, we consider the Fredholm integral equation

$$u(x) = \int_0^1 k(x, y)u(y)dy + f(x) \quad x \in [0, 1], \quad (4.5)$$

which can be written symbolically as,

$$Au = f, \quad f \in X, \quad (4.6)$$

where  $X$  is a Banach space and  $A = I - K$ , where  $I$  the identity operator on  $X$  and  $K$  the linear operator associated with the kernel  $k(x, y)$ . As Brakhage and Atkinson, Hemker and Schippers also use the defect correction principle [20] to formulate the iterative method for the solution of (4.6). Let us write the equation (4.6) as

$$A^h u^h = f^h, \quad (4.7)$$

with  $A^h = I^h - K^h$ , where  $K^h$  represents a discrete integral operator for the kernel on the fine grid with mesh size  $h$ . The iterative process defined by the defect correction principle can be written as

$$\begin{cases} u_0^h = 0, \\ u_{k+1}^h = B^h f + (I - B^h A^h)u_k^h, \end{cases} \quad (4.8)$$

where  $B^h$  denotes some approximate inverse of  $A^h$ ;  $u_{k+1}^h$  and  $u_k^h$  respectively are current and previous approximation. Notice that if  $B^h$  is exactly equal to  $(A^h)^{-1}$  then (4.8) implies that  $u_1^h$  is the exact solution. The characteristics that we impose to select such inverse  $B^h$  is that it has to be bijective and continuous in  $X$ . Theoretically, the solution  $u^h$  is the fixed point of (4.8) and the process converge to  $u^h$  if the rate of convergence  $\|I - B^h A^h\| < 1$ .

Choice  $B^h = I + (I - K^{2h})^{-1}K^h$ , (4.8) is the Brakhage formulation [6] of the iterative scheme (4.8). Atkinson's formulation [3] can be obtained taking  $B^h = I + (I - K^H)^{-1}K^h$  where  $H$  represents the coarsest grid. So, to determine the approximate inverse  $B^h$  of  $A^h$  both Brakhage and Atkinson use only two levels. The difference is Brakhage uses the fine grid  $\Omega^h$  and the next coarser  $\Omega^{2h}$  grid whereas Atkinson uses the fine grid  $\Omega^h$  and the coarsest grid  $\Omega^H$ , typically  $H \gg 2h$ .

For convenience of defining the Hemker-Schippers modification, we temporarily adopt their notations:  $K_l$  and  $B_l$  stand for operators  $K$  and  $B$  defined on level  $l$ , where  $l = 0$  stands for the finest level. Then operators  $B_l$  are given by,

$$B_0 = (I - K_0)^{-1},$$

$$B_l = Q_{l-1}(I - K_{l-1} + K_l), \quad l = 1, 2, \dots$$

with  $Q_l$  defined as

$$Q_l = \sum_{m=0}^{\gamma-1} (I - B_l A_l)^m B_l, \tag{4.9}$$

for some positive integer  $\gamma$ .

Equation (4.9) implies,

$$\begin{aligned} I - Q_l A_l &= I - (I - I + B_l A_l)^{-1} (I - B_l A_l)^\gamma B_l A_l \\ &= I - (I - B_l A_l)^\gamma, \end{aligned}$$

meaning that  $Q_l$  is also an approximate inverse of  $A_l$  and its application is equivalent with the application of  $\gamma$  steps of Brakhage or Atkinson's iteration with the use of approximate inverse  $B_l$ .

The algorithm using the Hemker-Schippers modification can be described as follows:

Multigrid Cycle:  $MGM(l, u, f)$

The input: Number of levels,  $l$ ; current approximation  $u$

The output: Fine grid approximation,  $v$

If  $l = 0$  then  $u := (I - K_0)^{-1}f$  else

$$r := u - K_l \times u - f;$$

$$u := u - (I - I_{2h}^h I_h^{2h})r;$$

$$r := ((I - K_{l-1}) \times I_h^{2h} - I_h^{2h} K_l) \times r;$$

$$v := 0; \text{ do } MGM(l - 1, v, r);$$

$$u^h := u - I_{2h}^h \times v^{2h};$$

end;

## 4.4 Numerical Experiments

In the two grid algorithm, it is a common practice to use Picard's iteration instead of relaxation in standard multigrid. In many cases, however, the Picard's iteration diverges specifically when  $\rho(\mathcal{K}) \not\leq 1$ , where  $\rho(\mathcal{K})$  is the spectral radius of the kernel matrix  $\mathcal{K}$ . Therefore, the pointwise iterations [17] in case of weakly singular kernels since the main diagonal and the neighboring diagonals contain entries of larger size than outside the diagonal band. For general case we are interested in a relaxation technique which is always convergent, and Kaczmarz iteration is one of them. Since Picard's method does not converge for a kernel with spectral radius greater than or equal to 1, and converges very slowly if the spectral radius is close to 1, it is advised to use only one iteration sweep of Picard's method. But since Kaczmarz is convergent for all matrices, we can use more than one Kaczmarz iteration sweep, which is helpful in order to get a smooth error and its corresponding residual on the fine grid so that it can be restricted properly to the coarser grid using full-weighting restriction operator.

Selecting proper relaxation technique is very important because efficiency of multigrid methods depends on the efficiency of the relaxation. In this section, we use Kaczmarz iteration as relaxation and analyze the result with some applications.

#### 4.4.1 Using Kaczmarz iteration

We consider the following Fredholm integral equation:

$$u(x) = \frac{\lambda}{2} \int_a^b e^{\lambda|x-y|} u(y) dy + f(x). \quad (4.10)$$

The exact solution of the equation 4.10 is given by

$$u(x) = C_1 + C_2 x + f(x) - \lambda^2 \int_a^x (x-t)f(t) dt.$$

where  $C_1$  and  $C_2$  are determined by the boundary conditions,

$$u'(a) + \lambda u(a) = f'(a) + \lambda f(a),$$

$$u'(b) - \lambda u(b) = f'(b) - \lambda f(b).$$

For  $f(x) = x$  coefficients  $C_1$  and  $C_2$  are given by

$$C_1 = \frac{\lambda^2}{3} + \lambda - 1, \quad C_2 = -\frac{\lambda^3}{3} - \lambda^2 + 2\lambda - 1.$$

We compare the use of Kaczmarz iteration with that of Picard's iteration. We have calculated number of multigrid cycles needed to get convergence rate  $10^{-6}$ . The convergence rate is defined as  $\frac{\|r_k\|}{\|r_0\|}$  where  $r_0$  and  $r_k$  are the residual corresponding to initial guess and approximation in  $k^{th}$  multigrid cycle, respectively.

Table 4.1: Observed multigrid convergence for equation (4.10) with  $\lambda = -1$ .

Fine Grid Size, $n = 2^7$		
Number of Grids	Number of Cycle Needed	
	Picard's	Kaczmarz
2	2	3 (1 Kaczmarz sweep)
		3 (2 Kaczmarz sweep)
3	2	3(1 Kaczmarz sweep)
		2 (2 Kaczmarz sweep)
4	2	3 (1 Kaczmarz sweep)
		2(2 Kaczmarz sweep)

Table 4.2: Observed multigrid convergence for equation (4.10) with  $\lambda = -10$ .

Fine Grid Size, $n = 2^7$		
Number of Grids	Number of Cycle Needed	
	Picard's	Kaczmarz
2	2	3 (1 Kaczmarz sweep)
		2 (3 Kaczmarz sweep)
3	2	4(1 Kaczmarz sweep)
		3 (2 Kaczmarz sweep)
		2 (3 Kaczmarz sweep)
4	3	4 (1 Kaczmarz sweep)
		2(2 Kaczmarz sweep)

Table 4.3: Observed multigrid convergence for equation (4.10) with  $\lambda = -100$ .

Fine Grid Size, $n = 2^7$		
Number of Grids	Number of Cycle Needed	
	Picard's	Kaczmarz
2	9	6 (1 Kaczmarz sweep) 4 (2 Kaczmarz sweep) 2 (5 Kaczmarz sweep)
3	9	6(1 Kaczmarz sweep) 4 (2 Kaczmarz sweep) 3 (3 Kaczmarz sweep)
4	18	6 (1 Kaczmarz sweep) 4 (2 Kaczmarz sweep) 3 (3 Kaczmarz sweep)

For  $\lambda = -1$ , and  $\lambda = -10$  use of Picard's iteration is preferable because two or three iteration sweeps of Kaczmarz gives similar result and Kaczmarz is more expensive than Picard's iteration. But for  $\lambda = -100$  we see that with the increase of number of grids we need more Picard's iteration sweeps to reach desired accuracy while two or three Kaczmarz iteration gives the accuracy in much less multigrid cycles than Picard's. In case of  $\lambda = -100$  the kernel has the property that  $\mathcal{K}_{ij} \ll 1$  when  $|i - j| \gg 1$  that is the components of kernel decrease with the distance from the diagonal band. We conclude that for a kernel with this property Kaczmarz iteration is more efficient.

### 4.4.2 Using Distributive relaxation

Consider the same example 4.10 as in the previous section. The following table shows the number of multigrid cycles needed to get the desired convergence rate  $10^{-6}$  using Picard's iteration and distributive relaxation.

Table 4.4: Comparison of Picard's iteration and distributive relaxation for equation (4.10) with  $\lambda = -1$ ,  $n = 2^7$ .

Number of Grids	Number of cycles needed		
	Picard's	Distributive Relaxation	
		Iteration sweeps	Multigrid Cycle
2	2	1	2
3	2	1	2
4	2	1	2

Table 4.5: Comparison of Picard's iteration and distributive relaxation for equation (4.10) with  $\lambda = -10$ ,  $n = 2^7$ .

Number of Grids	Number of cycles needed		
	Picard's	Distributive Relaxation	
		Iteration sweeps	Multigrid Cycle
2	3	1	4
		2	2
3	3	1	4
		2	2

Table 4.6: Comparison of Picard's iteration and distributive relaxation for equation (4.10) with  $\lambda = -100$ ,  $n = 2^7$ .

Number of Grids	Number of cycles needed		
	Picard's	Distributive Relaxation	
		Iteration sweeps	Multigrid Cycle
2	9	2	4
		3	3
		4	2
3	9	2	4
		3	3
		4	2
4	18	2	4
		3	3
		4	2

For  $\lambda = -1$  Picard's iteration and distributive relaxation gives similar result. If  $\lambda = -10$ , while Picard's needs three 2-grid or 3-grid cycles, the distributive relaxation needs two cycles with two relaxation sweeps in each cycle. Again in case of  $\lambda = -100$  the multigrid algorithm using distributive relaxation needs significantly less multigrid cycles. The following table containing the result of same example with fine grid of  $2^5 + 1$  points shows more interesting result. While the multigrid algorithm using Picard's iteration diverges for  $\lambda = -100$ , using distributive relaxation we get fast convergence.

Table 4.7: Comparison of Picard's iteration and distributive relaxation for equation (4.10) with  $\lambda = -1, -10, -100, n = 2^5$

$\lambda = -1, n = 2^5$			
Grid	Number of cycle needed		
	Picard's	Distributive Relaxation	
		Iteration	Cycle
2	2	1	3
		2	2
3	2	1	3
		2	2
4	2	1	3
		2	2
$\lambda = -10$			
2	5	1	6
		2	3
		4	2
3	5	1	6
		2	3
		4	2
4	7	1	6
		2	3
		4	2
$\lambda = -100$			
2	Diverge	1	4
		2	3
		3	2
3	Diverge	1	4
		3	2
4	Diverge	1	4
		3	2

A multigrid algorithm using Picard's iteration as a smoother often faces risk of slow convergence and in some cases of divergence while using of Kaczmarz the algorithm enjoys the property of always convergence. The distributive relaxation techniques are better in terms of computational complexity and convergence rate.

# Chapter 5

## Conclusion

Integral equations arise in many areas of science specially in physics, population dynamics, stochastic processes. It is equally important to be able to solve them numerically. Multigrid methods are efficient ways to solve integral equations. Most of the boundary value problem in partial differential equation can be transformed into an integral equation. An efficient technique of solving integral equation can provide the opportunity to solve partial differential equations as well.

In this thesis we discuss the basic ideas of multigrid techniques and preliminary topics of integral equations. We discuss the Atkinson-Brakhage two grid method and the Hemker-Schippers modification. The methods are based on the Nyström interpolation and Picard's iteration. We propose to use Kaczmarz and distributive relaxation instead of Picard's method. With an example we show that the distributive and Kaczmarz relaxation works better specially in the cases when Picard's method results divergence. To minimize the risk of divergence it is suggested to use one Picard's iteration sweep. But one Picard's sweep often does not provide with a smooth residual in the fine grid and as we discussed it is necessary for multigrid to work efficiently that the residual is smooth in the fine grid. For the always convergence property of Kaczmarz we can use more than one relaxation sweeps. At the

same time Kaczmarz iteration has higher computational complexity and so more than one sweeps would be expensive. So, it is necessary to find a balance depending on the problem. Our future research plan is to apply FMG-Full Multigrid with Kaczmarz and distributive relaxation.

# Bibliography

- [1] G. Adomian. *Solving Frontier Problems of Physics: The Decomposition Method*. Fundamental Theories of Physics. Springer, 1993.
- [2] Philip M Anselone and Joel Davis. *Collectively compact operator approximation theory and applications to integral equations*, volume 1971. Prentice-Hall Englewood Cliffs, NJ, 1971.
- [3] Kendall Atkinson. Iterative variants of the nyström method for the numerical solution of integral equations. *Numerische Mathematik*, 22(1):17–31, 1974.
- [4] Kendall E Atkinson. *The numerical solution of integral equations of the second kind*. Number 4. Cambridge university press, 1997.
- [5] Nikolai Sergeevich Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Mathematics and Mathematical Physics*, 6(5):101–135, 1966.
- [6] Helmut Brakhage. über die numerische behandlung von integralgleichungen nach der quadraturformelmethode. *Numerische Mathematik*, 2(1):183–196, 1960.
- [7] Achi Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390, 1977.
- [8] W.L. Briggs, V.E. Henson, and S.F. McCormick. *A Multigrid Tutorial: Second Edition*. Society for Industrial and Applied Mathematics, 2000.
- [9] E Cheney and David Kincaid. *Numerical mathematics and computing*. Cengage Learning, 2012.
- [10] Radii Petrovich Fedorenko. The speed of convergence of one iterative process. *USSR Computational Mathematics and Mathematical Physics*, 4(3):227–235, 1964.
- [11] W. Hackbusch. *Integral Equations: Theory and Numerical Treatment*. International Series of Numerical Mathematics. SPRINGER VERLAG NY, 1995.
- [12] Wolfgang Hackbusch. *Multi-grid methods and applications*, volume 4. Springer-Verlag Berlin, 1985.
- [13] A. Jerri. *Introduction to Integral Equations with Applications*. A Wiley-Interscience publication. Wiley, 1999.

- [14] S Kaczmarz. Approximate solution of systems of linear equations. *International Journal of Control*, 57(6):1269–1271, 1993.
- [15] CT Kelley. A fast multilevel algorithm for integral equations. *SIAM journal on numerical analysis*, 32(2):501–513, 1995.
- [16] E.J. Nyström. ber die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54(1):185–204, 1930.
- [17] B Oskam and JMJ Fray. General relaxation schemes in multigrid algorithms for higher-order singularity methods. *Journal of Computational Physics*, 48(3):423–440, 1982.
- [18] M. Rahman. *Integral equations and their applications*. WIT, 2007.
- [19] Y. Saad. *Iterative Methods for Sparse Linear Systems: Second Edition*. Society for Industrial and Applied Mathematics, 2003.
- [20] Hans J Stetter. The defect correction principle and discretization methods. *Numerische Mathematik*, 29(4):425–443, 1978.
- [21] U. Trottenberg, C.W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.