Multigrid Method

A Thesis

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Gunes Senel, B.S., M.S.

Graduate Program in Department of Mathematics and Mathematical Sciences

The Ohio State University

2017

Master's Examination Committee:

Dr. Edward Overman, Advisor

Dr. Chuan Xue

Abstract

The *multigrid method* is an algorithm aiming to provide a numerical solution to a system of equations. In this thesis we focus on its original purpose: solving differential equations. By discretizing a differential equation, one obtains a system of linear equations, which can be solved using multigrid methods. We will start our discussion in one dimension, considering ordinary differential equations. First, we will review standard pointwise relaxation methods, since they are a fundamental part of any multigrid method. We will compare them in different settings and see their shortcomings. This part also provides an insight of the need for a better method, and one of the better methods is the multigrid method. Then we will discuss elements of multigrid method in one dimension. We will apply the method on different problems, compare and discuss the results, providing proofs and calculations for easy but well-known phenomena, such as the reduction rate of the error and the exact outcome of the method on the homogeneous Poisson equation when damped Jacobi relaxation is chosen to be used in the multigrid method, and proof of the fact that the error for the same equation can be totally eliminated in one single iteration using the multigrid method with the choice of red-black Gauss-Seidel relaxation. We will consider a nontrivial ordinary differential equation in order to be convinced that the problems that the multigrid method can deal with and the conclusions we have arrived can be generalized. Afterwards we discuss the multigrid method in two dimensions, where

it is mostly used in real life applications. As we did in one dimension, we shortly review the relaxation methods in two dimensions. Then we introduce the elements of the multigrid method in two dimensions. We apply the multigrid method to some problems and tabulate the results, leaving the redundant cases out. We discuss the results and compare them with the one dimensional results.

Acknowledgments

First of all, I would like to express my deep gratitude to my advisor Dr. Edward Overman. He does not only guide my thesis and motivated me during this process, but also he kept believing in me and encouraged me at the most difficult phase of my education and life. He was there with all his tolerance, patience, understanding and kindness with his big heart.

I am grateful to my coadvisor Dr. Chuan Xue for sharing her valuable comments and questions throughout my Masters, and participating in my thesis committee. Without her motivation and support, I wouldn't be able to complete the degree.

I would like to thank Department of Mathematics of the Ohio State University for providing me financial support and giving me the chance to improve my teaching skills.

I would also like to thank Dr. R. Ryan Patel and Dr. Sadi Fox for their help, support and insistence. Without them I would have given up, however they never stop trying.

I am thankful to my family for their respect to my decisions, their support and understanding. Especially I am indebted to my grandparents for raising me up and their unconditional love and encouragement.

Vita

2010 ..... B.S. Mathematics & Physics, Bogazici University

2013 ..... M.S. Mathematics, Bogazici University

2013 to present ..... Graduate Teaching Associate, Department of Mathematics, The Ohio State University

Fields of Study

Major Field: Department of Mathematics
and Mathematical Sciences

Table of Contents

Page

List of Tables

Table **Page**

List of Figures

Chapter 1: Introduction

Mathematical equations, where a function is related with its derivatives, are called *differential equations*. In applications the function usually represents some physical quantity and the equation models some physical phenomena related to that physical quantity. For example, the diffusion equation, $u_t = \frac{\partial}{\partial x}(d(x,t)\frac{\partial u}{\partial x}(x,t))$, describes the free motion of "particles" with random movement, where $u(x,t)$ stands for the density of the particles and $d(x,t)$ is the diffusion coefficient. When $d(x,t)$ is constant, the equation is called the "heat equation", which describes the distribution of heat with u being the temperature and d representing thermal diffusivity. Similarly, the wave equation, $u_{tt} = c^2 u_{xx}$ with c constant, represents the "wave-like" behaviour, the transport equation, $u_t = -cu_x$ represents transportation of the particles, etc.

To understand, describe and predict physical phenomena, solving differential equations, finding a family of solutions, or even just finding the space that the solutions belong to, is very important. However, existence of the solutions of the differential equations is not guaranteed, and even if the solution exists, it might be difficult to get it explicitly. This is where numerical methods help: They offer us a way to formalize the equation and a chance to approximate the solution, whenever they can.

The *multigrid method* is one of these numerical algorithms, which was discovered in 1960s to solve boundary value problem of elliptic partial differential equations, i.e.

equations of the form

$$a_{11}u_{xx} + a_{12}u_{xy} + a_{22}u_{yy} + a_1u_x + a_2u_y + a_0u = f(x, y)$$

where $u(x, y)$ is the solution of the equation and $a_{12}^2 - 4a_{11}a_{22} < 0$. Multigrid method employs hierarchic grids and a smoothing algorithm, such as basic relaxation methods, and use these to solve the system of linear equations representing the discretization of differential equation "successfully", with smaller error and less iterations when compared with other iterative methods. In addition, multigrid methods are time-efficient, especially when compared to the algorithms of the direct methods, and are one of the fastest numerical methods for elliptic partial differential equations, with a great scope of generalizations.

In the second chapter of this thesis, we will start with a review of standard pointwise iterative relaxation methods, Jacobi and Gauss-Seidel, and their variations, which can also be used to solve differential equations. Although iterative methods are not as costly as direct methods, it takes many iterations to converge to the solution. We'll try to see whether and/or how much they can reduce the error, given "some information" about the error.

In the third chapter, we will start discussing the multigrid method in one dimension, where it is easier to analyze the method. We'll try to improve the damping rate of the two grid method with damped Jacobi relaxation applied on homogeneous Poisson equation with homogeneous Dirichlet boundary conditions, which was previously given in [1], and see that the rate is actually much better than what was suggested. We'll do some analysis to understand what is really happening and give a proof of the interesting fact that two grid method with a final red-black Gauss-Seidel relaxation zeroes out the error when used for the same problem. We'll examine the effects of

the two-grid method when applied to "basis functions". Then we'll introduce V and W cycles, the "building blocks" of the full multigrid method. Finally, we'll generalize V and W cycles to *full multigrid method*, where the main important change in the method is to be able to calculate a "good" initial guess.

Throughout the third chapter, we'll apply the methods discussed on simple ordinary differential equations, $-u_{xx} = f(x)$ and $-u_{xx} + u = f(x)$, to simplify the analysis. We end the third chapter with a general example of a ordinary differential equation, $a(x)u_{xx} + b(x)u_x + c(x) = f(x)$, to see how good multigrid method is in the general setting.

In the fourth chapter, we'll generalize the multigrid method to two dimensions. We will look at examples using the equation $-u_{xx} - u_{yy} = f(x, y)$, which is called Poisson's equation and frequently used in physics. We will review standard pointwise iterative relaxation methods in two dimensions and mention a variation of the Poisson equation, where the relaxations methods we had discussed so far are not very effective. This generates a problem, since relaxation methods are crucial in multigrid methods, as will be discussed in Chapter 3. We'll propose possible solutions to the problem occurred. After checking our methods in the two dimensional setting, we end the chapter with a discussion of the results and a comparison of the multigrid method in one and two dimensions.

In this thesis, a basic knowledge of finite difference discritization and linear algebra is assumed. A good reference for the basic linear algebra is [7]. A good reference of finite difference discritization is [4]. A basic introduction on differential equations is given in [2].

Chapter 2: Relaxation Methods

Relaxation methods are iterative methods to solve linear systems of equations. As in any iterative method, we start with an initial iterate, and construct a "sequence" of "improving" solutions, each of which depends on the previous iterate. The "solution" gets closer to the exact solution as we do more iterations if the sequence "converges".

We can write a system of equations in the matrix form as

$$A\mathbf{u} = \mathbf{f}.$$

For example, while trying to find a solution to a differential equation, when we apply finite difference discretization to the equation, we get a linear system of equations. A simple example of such an equation would be

$$-u_{xx} = f(x) \quad 0 < x < 1$$

with $n-1$ interior grid points $x_1, x_2, ..., x_{n-1}$, whose finite difference discretization will give the system of equations

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f_i \qquad 1 \leq i \leq n-1 \tag{2.1}$$

where $h = \frac{1}{n}$, $u_i = u(x_i)$ and $f_i = f(x_i)$. Given the homogeneous Dirichlet boundary conditions $u_0 = u_n = 0$, we can write this system as a matrix multiplication:

$$\frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}\begin{pmatrix} u_1 \\ \vdots \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_{n-1} \end{pmatrix} \tag{2.2}$$

which we can denote as $A\mathbf{u} = \mathbf{f}$.

To get the solution $\mathbf{u}$ from the equation $A\mathbf{u} = \mathbf{f}$, iterative methods split the matrix A into two parts: $A - B$ and B, where B is a simple matrix to deal with. Now our problem takes the form

$$B\mathbf{u} = (B - A)\mathbf{u} + \mathbf{f}$$

where iterations can be written as

$$B\mathbf{u}^{(k)} = (B - A)\mathbf{u}^{(k-1)} + \mathbf{f} \quad \text{or} \quad \mathbf{u}^{(k)} = (I - B^{-1}A)\mathbf{u}^{(k-1)} + B^{-1}\mathbf{f}$$

with the initial iterate $u^{(0)}$ is given.

In this chapter we are going to briefly discuss the standard relaxation methods Jacobi and Gauss-Seidel, and their variations, which play an important role in the multigrid method.

2.1 Jacobi method

The *Jacobi method* is one of the most common and easiest relaxation methods. Here we break A into diagonal (D), upper diagonal (U) and lower diagonal (L) parts, where $A = D + U + L$, and choose $B = D$. Leaving the diagonal part alone on the left hand side of the equation, we get

$$D\mathbf{u} = -(U + L)\mathbf{u} + \mathbf{f} \quad \text{or} \quad \mathbf{u} = -D^{-1}(U + L)\mathbf{u} + D^{-1}\mathbf{f}$$

which we can use to update our iterates. If we define *Jacobi iteration matrix* C_J as $C_J = -D^{-1}(U + L)$, we can write Jacobi method in the matrix form as

$$\mathbf{u}^{(k)} = C_J \mathbf{u}^{(k-1)} + D^{-1}\mathbf{f}$$

Looking at the matrix form (2.2) of the example problem, we see that in this case our iterations become

$$\begin{pmatrix} 2 & 0 & & & \\ 0 & 2 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & 2 & 0 \\ & & & 0 & 2 \end{pmatrix} \begin{pmatrix} u_1^{(k)} \\ \vdots \\ u_{n-1}^{(k)} \end{pmatrix} = \begin{pmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 1 & 0 \end{pmatrix} \begin{pmatrix} u_1^{(k-1)} \\ \vdots \\ u_{n-1}^{(k-1)} \end{pmatrix} + h^2 \begin{pmatrix} f_1 \\ \vdots \\ f_{n-1} \end{pmatrix}$$

When we look at a single row on both sides after the operations are performed, we see that we are actually calculating

$$u_i^{(k)} = \frac{u_{i-1}^{(k-1)} + u_{i+1}^{(k-1)} + h^2 f_i}{2} \qquad 1 \leq i \leq n-1$$

starting from $k = 1$. So the Jacobi iteration basically uses the j^{th} equation in (2.1) to update u_i. In other words, it uses the neighbour points of the j^{th} point and the right hand side of the equation to update u_i. Starting with an initial guess $\mathbf{u}^{(0)}$ for the solution, we try to get as close as possible to the actual solution, by using these updates.

An important aspect of the iterative methods is the fact that in order to be able to approximate the solution, the method has to converge. The sequence of iterates $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, ...$ defined by $\mathbf{u}^{(k)} = C\mathbf{u}^{(k-1)} + B^{-1}f$ where $C = I - B^{-1}A$ converges if and only if the spectral radius of C, denoted by $\rho(C)$, is less than 1, i.e. the maximum of the absolute values of the eigenvalues of C is less than 1. To see this, we can define the error at each iteration as $\mathbf{e}^{(k)} = \mathbf{u}^{(k)} - \bar{\mathbf{u}}$, where $\bar{\mathbf{u}}$ is the actual solution. The

error we defined satisfies the equation $\mathbf{e}^{(k)} = C\mathbf{e}^{(k-1)}$. So if $\mathbf{e}^{(k)}$ converges to 0, we get an iterate $\mathbf{u}^{(N)}$ very close to $\bar{\mathbf{u}}$. It is a well known result (Thm. 1.10 [10]) that a sequence $\mathbf{e}^{(1)}, \mathbf{e}^{(2)}, ...$ defined by $\mathbf{e}^{(k)} = C\mathbf{e}^{(k-1)}$ converges to 0 if and only if $\rho(C) < 1$. We can easily see why this is true for a nondefective matrix C:

Let $\lambda_1, \lambda_2, \cdots, \lambda_n$ denote the eigenvalues of C and $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n$ denote the corresponding eigenvectors. Since C is nondefective, eigenvectors of C construct a basis, so we can write $\mathbf{e}^{(1)}$ as a linear combination of eigenvalues,

$$\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \cdots + \alpha_n\mathbf{v}_n$$

By the recursion relation, we have

$$\begin{aligned} e^{(k)} =& C\mathbf{e}^{(k-1)} = C^2\mathbf{e}^{(k-2)} = \cdots = C^{k-1}\mathbf{e}^{(1)} \\ =& C^{k-1}(\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \cdots + \alpha_n\mathbf{v}_n) \\ =& \alpha_1\lambda_1^{k-1}\mathbf{v}_1 + \alpha_2\lambda_2^{k-1}\mathbf{v}_2 + \cdots + \alpha_n\lambda_n^{k-1}\mathbf{v}_n. \end{aligned}$$

$\rho(C) < 1$ if and only if $\max_{i\in 1,\cdots,n} |\lambda_i| < 1$ if and only if $\lim_{k\to\infty} \lambda_i^k = 0$. Hence the result follows.

When we want to check the eigenvalues λ_k and eigenvectors $\mathbf{v}_k$ of C_J to check convergence of our example problem, one possible way is following the path $C_J = -D^{-1}(U + L) = -D^{-1}(A - D) = I - D^{-1}A$ with I being the identity matrix, which gives us

$$\lambda(C_J) = \lambda(I) - \lambda(D^{-1}A) = 1 - \tfrac{h^2}{2}\lambda(A)$$

where $\lambda(M)$ stands for the eigenvalues of a matrix M. In our example problem, the eigenvalues of A are

$$\mu_k = \frac{4}{h^2}\sin^2(\tfrac{1}{2}\pi hk) \tag{2.3}$$

(can be calculated by the standard formula for the eigenvalues of a tridiagonal matrix, which can be found in many references such as in [8]). So we find that

$$\lambda_k = 1 - \frac{h^2}{2}\frac{4}{h^2}\sin^2\left(\frac{1}{2}\pi kh\right) = 1 - 2\sin^2\left(\frac{1}{2}\pi kh\right) = \cos(\pi kh)$$

where $1 \leq k \leq n-1$, since A and C_J are $(n-1)\times(n-1)$ matrices. Since $|\cos(\pi kh)| < 1$ for $1 \leq k \leq (n-1)$, Jacobi iterations converge in our problem.

At this point, we can remember the fact that the family of nontrivial solutions to the equation $-u_{xx} = \alpha^2 u$ with the homogeneous Dirichlet boundary condition is $\sin(k\pi x)$, where k is an integer. We relate this differential equation as a continuous eigenvalue problem of the operator $-\frac{d^2}{dx^2}$, where eigenvalues are $\alpha^2 = (k\pi)^2$ and eigenfunctions are $\sin(k\pi x)$. Evaluating $\sin(k\pi x)$ on the interior points of a mesh of $n+1$ points on $[0,1]$, $\{x_j | x_j = \frac{j}{n} = jh, j = 0, 1, \cdots, n\}$, with h being the distance between two grid points, as usual, we get the vector $\mathbf{v}_k$,

$$\mathbf{v}_k = \begin{pmatrix} \sin(\pi kh) \\ \sin(2\pi kh) \\ \vdots \\ \sin((n-1)\pi kh) \end{pmatrix} \tag{2.4}$$

When we calculate the eigenvectors of the matrix A, which represents the discretized version of $-\frac{d^2}{dx^2}$, we get the same set of vectors $\mathbf{v}_k$, which were the eigenfunctions of the continuous case evaluated on the mesh. For Jacobi iteration matrix C_J we get the same eigenvectors, since

$$C_J\mathbf{v}_k = (I - D^{-1}A)\mathbf{v}_k = \mathbf{v}_k - D^{-1}\mu_k\mathbf{v}_k = \mathbf{v}_k - \mu_k\tfrac{h^2}{2}\mathbf{v}_k = (1 - \tfrac{h^2}{2}\mu_k)\mathbf{v}_k.$$

Since $\{\mathbf{v}_k | 1 \leq k \leq n-1\}$ is a basis for $\mathfrak{R}^{n-1}$, we can write any vector as a linear combination of $\mathbf{v}_k$'s. When we apply Jacobi method on any vector, λ_k determines

how much the k^{th} component of the error can be reduced:

$$\begin{aligned} \mathbf{y} &= \xi_1\mathbf{v}_1 + \xi_2\mathbf{v}_2 + \cdots + \xi_{n-1}\mathbf{v}_{n-1} \\ C_J\mathbf{y} &= \xi_1\lambda_1\mathbf{v}_1 + \xi_2\lambda_2\mathbf{v}_2 + \cdots + \xi_{n-1}\lambda_{n-1}\mathbf{v}_{n-1} \end{aligned} \tag{2.5}$$

For k close to 0 or n, since $|\cos(0)| = |\cos(\pi)| = 1$ and cosine is a continuous function, $|\lambda_k| = |\cos(\pi kh)| \approx 1$, so the effect of the Jacobi method on $\mathbf{v}_k$, where k is close to n or 0, is not very helpful. For example, we don't get much/enough reduction if we have $\mathbf{v}_k$ with a k close to n as our error. Therefore we need some improvement/variation and this leads us to our next method: *damped Jacobi method*.

2.1.1 Weighted (Damped) Jacobi Method

A very simple but important modification of the Jacobi method adds an extra step to the iteration. For this method, we still calculate

$$\mathbf{u}^{(k-1/2)} = C_J\mathbf{u}^{(k-1)} + D^{-1}f$$

but use it as an intermediate step to calculate $\mathbf{u}^{(k)}$. $\mathbf{u}^{(k)}$ will be the weighted average of $\mathbf{u}^{(k-1)}$ and $\mathbf{u}^{(k-1/2)}$:

$$\begin{aligned} \mathbf{u}^{(k)} &= w\mathbf{u}^{(k-1/2)} + (1-w)\mathbf{u}^{(k-1)} \\ &= w(C_J\mathbf{u}^{(k-1)} + D^{-1}\mathbf{f}) + (1-w)\mathbf{u}^{(k-1)} \\ &= ((1-w)I + wC_J)\mathbf{u}^{(k-1)} + wD^{-1}\mathbf{f} \end{aligned}$$

with weight w. We denote the *weighted Jacobi iteration matrix* as

$$C_J(w) = (1-w)I + wC_J.$$

Note that when $w = 1$, we obtain the standard Jacobi method.

Looking to our example problem, we calculate our iterates as a combination:

$$\mathbf{u}^{(k)} = w\mathbf{u}^{(k-1/2)} + (1-w)\mathbf{u}^{(k-1)}$$

where the intermediate step corresponds to

$$u_i^{(k-1/2)} = \frac{u_{i-1}^{(k-1)} + u_{i+1}^{(k-1)} + h^2 f_i}{2} \qquad 1 \le i \le n-1$$

Calculation of the eigenvalues $\lambda_k(w)$ and eigenvectors $\mathbf{v}_k(w)$ of the weighted Jacobi matrix is straightforward after realizing that the eigenvectors for the Jacobi iteration matrix are also eigenvalues of the weighted Jacobi iteration matrix with eigenvalues:

$$C_J(w)\mathbf{v}_k = (1-w)\mathbf{v}_k + wC_J\mathbf{v}_k = (1-w+w\lambda_k)\mathbf{v}_k$$

which gives us in our example problem

$$\begin{aligned} C_J(w)\mathbf{v}_k &= ((1-w) + w\cos(\pi kh))\mathbf{v}_k \\ &= (1-w(1-\cos(\pi kh)))\mathbf{v}_k = (1-2w\sin^2(\tfrac{1}{2}\pi kh))\mathbf{v}_k \end{aligned}$$

In weighted Jacobi iteration, choosing the "right" w plays an important role. When we check $\lambda_k(w)$'s with different w's, we see Figure 2.1.

Looking at the Figure 2.1, we see that the eigenvalues are different for different w's. For example, $\lambda_{\frac{n}{2}}(1) = 0$, so when $w = 1$, we can get rid of the $\frac{n}{2}^{th}$ component of the error immediately, since by Equation (2.5), $C_J(1)(\xi_1\mathbf{v}_1 + \cdots + \xi_{n-1}\mathbf{v}_{n-1}) = \xi_1\lambda_1\mathbf{v}_1 + \cdots + \xi_{\frac{n}{2}-1}\lambda_{\frac{n}{2}-1}\mathbf{v}_{\frac{n}{2}-1} + \xi_{\frac{n}{2}}\lambda_{\frac{n}{2}}\mathbf{v}_{\frac{n}{2}} + \xi_{\frac{n}{2}+1}\lambda_{\frac{n}{2}+1}\mathbf{v}_{\frac{n}{2}+1} + \cdots + \xi_{n-1}\lambda_{n-1}\mathbf{v}_{n-1}$ (Recall that $w = 1$ case was the standard Jacobi method). Since when k is small, there is not much difference between those curves (eigenvalues are close to 1), we may aim to reduce $k \ge \frac{n}{2}$ components of the error. Setting $\lambda_{\frac{n}{2}}(w)$ and $-\lambda_n(w)$ equal to each

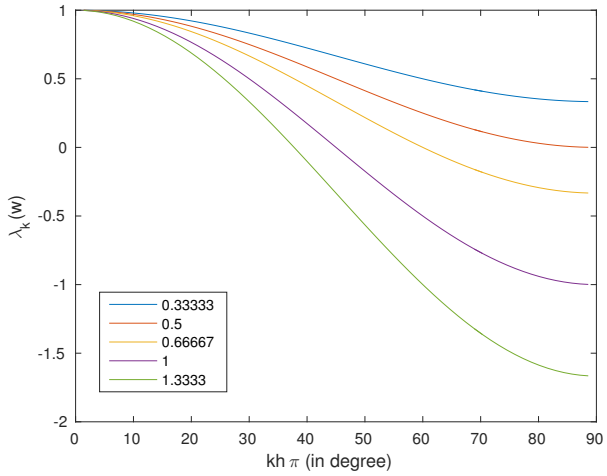


Figure 2.1: Eigenvalues of weighted Jacobi iteration matrix with different w's. Each curve corresponds to the w given in the legend

other (so that eigenvalues cannot go far away from 0), we get

$$1 - 2w\sin^2(\tfrac{nh\pi}{4}) = -1 + 2w\sin^2(\tfrac{nh\pi}{2})$$

$$1 - w = -1 + 2w$$

using the fact that $nh = 1$. So the optimal w is $\frac{2}{3}$ for the components with $k \geq \frac{n}{2}$.

From now on, we will call the components of the error with $k \geq \frac{n}{2}$ as "*high frequency error*", and the ones with $k < \frac{n}{2}$ as "*low frequency error*".

As we will discuss in the next chapter, being a high or low frequency error depends on n, the number of intervals in the discretization. For example, the $\mathbf{v}_{10}$ component of the error is a low-frequency error when $n = 32$, but a high frequency error when $n = 16$. So one way to reduce the low frequency errors with relaxation methods is to use a coarser grid, where some components of the low frequency error in the finer grid becomes high frequency errors. We will make use of this change of grid technique in the multigrid method.

2.2 Gauss-Seidel method

Gauss-Seidel is another very common relaxation method, where the choice of B is now $B = L + D$, where $A = D + L + U$ as above. Now our iterations become

$$(L + D)\mathbf{u}^{(k)} = -U\mathbf{u}^{(k-1)} + \mathbf{f} \quad \text{or} \quad \mathbf{u}^{(k)} = -(L + D)^{-1}U\mathbf{u}^{(k-1)} + (L + D)^{-1}\mathbf{f}.$$

We define the *Gauss-Seidel iteration matrix* as

$$C_{GS} = -(L + D)^{-1}U.$$

What is different for Gauss-Seidel method from Jacobi iteration is, here we are going to use updated values immediately. In other words, to update j^{th} element for k^{th} time, we can use $1^{st}, ..., (j-1)^{st}$ elements which are already updated for the k^{th} time.

For our example problem, given the linear system of equations in Equation (2.2), we can formalize the Gauss-Seidel method as

$$\frac{1}{h^2}\begin{pmatrix} 2 & & & & \\ -1 & 2 & & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & \\ & & & -1 & 2 \end{pmatrix}\begin{pmatrix} u_1^{(k)} \\ \vdots \\ u_{n-1}^{(k)} \end{pmatrix} = \frac{1}{h^2}\begin{pmatrix} 0 & 1 & & & \\ & & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & & & 1 \\ & & & & \end{pmatrix}\begin{pmatrix} u_1^{(k-1)} \\ \vdots \\ u_{n-1}^{(k-1)} \end{pmatrix} + \begin{pmatrix} f_1 \\ \vdots \\ f_{n-1} \end{pmatrix} \tag{2.6}$$

where each row is

$$-u_{i-1}^{(k)} + 2u_i^{(k)} = u_{i+1}^{(k-1)} + h^2 f_i$$

or

$$u_i^{(k)} = \frac{u_{i-1}^{(k)} + u_{i+1}^{(k-1)} + h^2 f_i}{2} \quad 1 \leq i \leq n-1$$

starting from $k = 1$. This looks very similar to the equations we get with Jacobi method, however here the value we use for the left node, i.e. $u_{i-1}^{(k)}$, is different than the one in Jacobi, i.e. $u_{i-1}^{(k-1)}$.

C_{GS} can be written explicitly as

$$\begin{pmatrix} 0 & \frac{1}{2} & & & & \\ 0 & \frac{1}{4} & \frac{1}{2} & & & \\ 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ & & & & \frac{1}{4} & \frac{1}{2} \\ & & & & \frac{1}{8} & \frac{1}{4} \end{pmatrix}$$

since $u_0^{(k)} = u_n^{(k)} = 0$ for any k and

$$\begin{aligned} u_1^{(i)} &= \tfrac{1}{2} u_2^{(i-1)} \\ u_2^{(i)} &= \tfrac{1}{2} u_1^{(i)} + \tfrac{1}{2} u_3^{(i-1)} = \tfrac{1}{4} u_2^{(i-1)} + \tfrac{1}{2} u_3^{(i-1)} \\ u_3^{(i)} &= \tfrac{1}{2} u_2^{(i)} + \tfrac{1}{2} u_4^{(i-1)} = \tfrac{1}{2}(\tfrac{1}{4} u_2^{(i-1)} + \tfrac{1}{2} u_3^{(i-1)}) + \tfrac{1}{2} u_2^{(i-1)} = \tfrac{1}{8} u_2^{(i-1)} + \tfrac{1}{4} u_3^{(i-1)}) + \tfrac{1}{2} u_4^{(i-1)} \\ &\vdots \end{aligned}$$

When we check the eigenvalues and eigenvectors of C_{GS}, we get

$$\begin{aligned} C_{GS}\mathbf{z}_k &= -(L+D)^{-1} U \mathbf{z}_k = \gamma_k \mathbf{z}_k \\ -U\mathbf{z}_k &= (L+D)\gamma_k \mathbf{z}_k \\ 0 &= (\gamma_k L + \gamma_k D + U)\mathbf{z}_k \end{aligned}$$

where $\gamma_k L + \gamma_k D + U$ is a tridiagonal matrix slightly different than A. Checking its eigenvalues gives us $2\gamma_k - 2\sqrt{\gamma_k}\cos(k\pi h) = 0$, i.e.

$$\gamma_k = \begin{cases} \cos^2(kh\pi) & 1 \le k < \frac{n}{2} \\ 0 & \frac{n}{2} \le k \le n-1 \end{cases}$$

as eigenvalues. Again we can see that $|\gamma_k| < 1$ for $1 \le k \le n-1$, so Gauss-Seidel iterations converge, too.

Knowing the eigenvalues, eigenvectors of the Gauss-Seidel iteration matrix can be found by solving the equation directly to be

$$\mathbf{z}_k = \begin{pmatrix} \cos(kh\pi)\sin(kh\pi) \\ \cos^2(kh\pi)\sin(2kh\pi) \\ \vdots \\ \cos^{n-1}(kh\pi)\sin((n-1)kh\pi) \end{pmatrix}$$

for $1 \leq k < \frac{n}{2}$, corresponding to the nonzero eigenvalues. The matrix is defective since the only eigenvector for 0 is $\mathbf{e}_1 = [1, 0, \cdots, 0]^T$. Note that the eigenvectors of Gauss-Seidel iteration matrix are different from the eigenvectors of A or C_J.

2.2.1 Red-black Gauss-Seidel

Again, a very simple but important modification of the Gauss-Seidel method is the *red-black Gauss-Seidel method*. Here we simply select every other node to update it using the neighbouring nodes, then start over and update the nodes that were the "neighbours", hence we skipped without updating, in the same way. With red-black Gauss-Seidel relaxation nodes are divided into two nonintersecting sets and updating one set of nodes only require information from the other set. Hence the order of update in each set does not matter. First updating the odd numbered nodes, then the even numbered nodes, gives us the system of equations, starting from $k = 1$:

$$u_i^{(k)} = \begin{cases} \frac{1}{2}\left(u_{i-1}^{(k-1)} + u_{i+1}^{(k-1)} + h^2 f_i\right) & i = 1, 3, 5, \cdots, n-1 \\ \frac{1}{2}\left(u_{i-1}^{(k)} + u_{i+1}^{(k)} + h^2 f_i\right) & i = 2, 4, 6, \cdots, n-2 \end{cases}$$

where $\mathbf{u}^{(0)}$. is the initial iterate.

Seeing these system of equations in matrix form is not as obvious as the above cases. However, considering one step at a time, we see that the matrix we have is

$$C_{RB} = \begin{pmatrix} 1 & & & & & & & & \\ 0.5 & 0 & 0.5 & & & & & & \\ 0 & 0 & 1 & & & & & & \\ & & 0.5 & 0 & 0.5 & & & & \\ & & & & \ddots & & & & \\ & & & & & 1 & & & \\ & & & & & 0.5 & 0 & 0.5 & \\ & & & & & & & 1 & \end{pmatrix} \begin{pmatrix} 0 & 0.5 & & & & & & & \\ 0 & 1 & & & & & & & \\ 0 & 0.5 & 0 & 0.5 & & & & & \\ 0 & 0 & 0 & 1 & & & & & \\ & & & 0.5 & 0 & 0.5 & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & 0 & \\ & & & & & & 0.5 & 0 & \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0.5 & 0 & \cdots & & & & & & \\ \vdots & 0.5 & 0 & 0.25 & 0 & \cdots & & & & \\ & 0.5 & 0 & 0.5 & 0 & \ddots & 0 & 0 & \cdots & \\ & 0.25 & 0 & 0.5 & 0 & \ddots & 0.25 & 0 & \cdots & \\ & 0 & 0 & 0.5 & \vdots & \ddots & 0.5 & \vdots & \ddots & \\ & \vdots & \vdots & 0.25 & & & 0.5 & 0 & 0.25 & 0 \\ & & & 0 & & & 0.5 & 0 & 0.5 & 0 \\ & & & \vdots & & & 0.25 & 0 & 0.5 & 0 \\ & & & & & & 0 & 0 & 0.5 & 0 \end{pmatrix}.$$

Eigenvalues of C_{RB} are the same as the eigenvalues of C_{GS}. Eigenvectors of C_{RB} corresponding to the nonzero eigenvalues can be written explicitly as

$$\mathbf{y}_k = \begin{pmatrix} \cos(kh\pi)\sin(kh\pi) \\ \cos^2(kh\pi)\sin(2kh\pi) \\ \cos(kh\pi)\sin(3kh\pi) \\ \cos^2(kh\pi)\sin(4kh\pi) \\ \vdots \\ \cos(kh\pi)\sin((n-1)kh\pi) \end{pmatrix}$$

for $1 \leq k < \frac{n}{2}$, and $\mathbf{e}_1$, $\mathbf{e}_3$, $\cdots$, $\mathbf{e}_{n-3}$, $\mathbf{e}_{n-1}$ are the remaining eigenvectors, where $\mathbf{e}_i$ stands for the i^{th} element of the standard basis.

2.3 Comparison of the methods

Recall that the eigenvectors of the matrix A, given by the Equation 2.4, construct a basis, and we can express our error as a linear combination of them. Now let's

check how much our relaxation methods can reduce the error, by trying them on the individual basis elements, $\mathbf{v}_k$, which are of high-frequency if $k \geq \frac{n}{2}$ and of low-frequency if $k < \frac{n}{2}$.

We can relate this process with "finite" Fourier series as follows: Recall that a Fourier series is an expansion/a representation of a periodic function $f(x)$ in terms of an infinite sum of $\sin(m\pi x)$, $\cos(m\pi x)$, where m is a positive integer. For example, if $f(x)$ is periodic over the interval $[-1, 1]$, we can write the Fourier series of $f(x)$ as

$$f(x) = \tfrac{1}{2}a_0 + \sum_{m=1}^{\infty} a_m \cos(m\pi x) + \sum_{m=1}^{\infty} b_m \sin(m\pi x)$$

with

$$\begin{aligned} a_0 &= \int_{-1}^{1} f(x)dx \\ a_m &= \int_{-1}^{1} f(x)\cos(m\pi x)dx \\ b_m &= \int_{-1}^{1} f(x)\sin(m\pi x)dx \end{aligned}$$

where the orthogonality of $\sin(m\pi x)$, $\cos(m\pi x)$ plays the main role in determining the coefficients a_m, b_m. With 0's on the boundary, we have to work only with the sine terms in the Fourier representation of the error function. Remember that $\sin(k\pi x)$ evaluated on the mesh was our eigenvectors $\mathbf{v}_k$ for A and C_J, when $k = 1, \cdots, n-1$. Now the relation is clear: Checking the coeficients of the Fourier representation of the error, we can see the "contribution" of each $\mathbf{v}_k$ we are interested in to the error.

We extend our interval from $[0, 1]$ to $[-1, 1]$, keeping our "function" antisymmetric about the origin like sine, in order to be able to apply discrete Fourier transform. For calculation, MATLAB's built-in fast Fourier transform function "fft" is used, which uses a very efficient and famous algorithm to compute the discrete Fourier transform

of a sequence. Only half of the results we get from the transform is plotted, since the result is symmetric, due to the fact that our input is real.

Let's start our discussion with the Jacobi method:

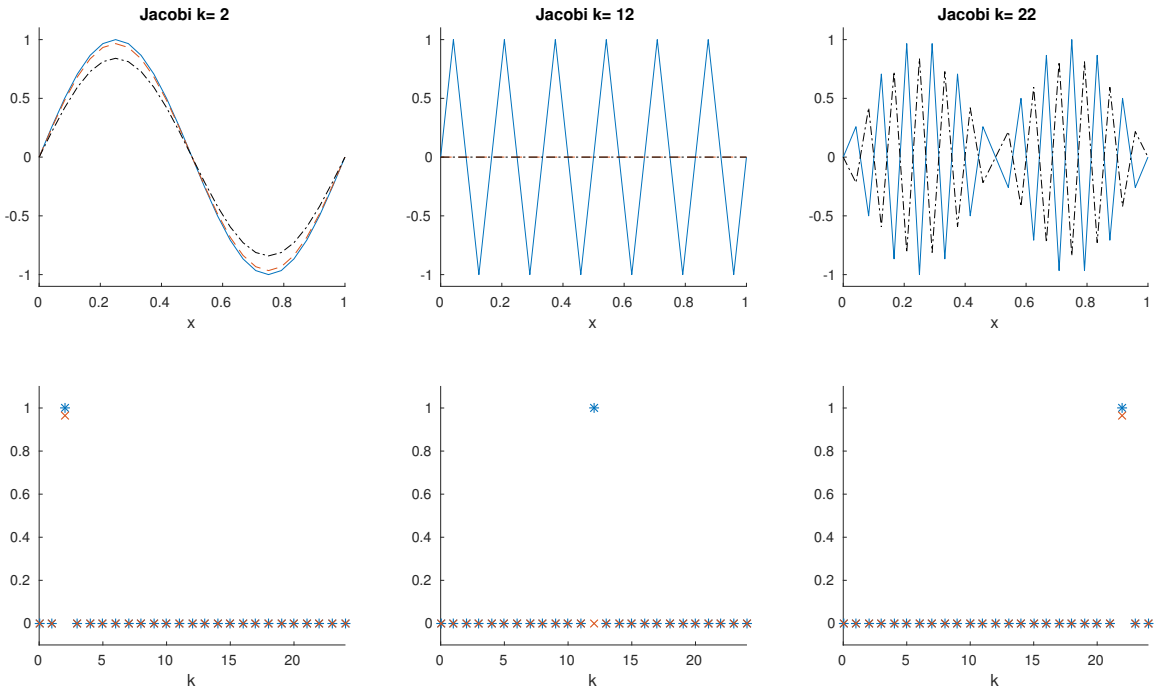


Figure 2.2: Top row: Initial error (solid line), error after one relaxation (dashed line) and error after 5 relaxations (dash-dot line) with Jacobi method for modes k=2 (low), k=12 (medium) and k=22 (high) where n=24; Second row: Fourier modes of the initial error (*) and error after **one** relaxation (x) with Jacobi method

We see in Figure 2.2, Jacobi method doesn't help with low or high frequency errors much, since its eigenvalues with k near n and near 1 are close to 1 in magnitude (Recall Figure 2.1 with $w = 1$.) From the Fourier mode graphs, we can also see that no other mode than the mode we started with is excited, since $\mathbf{v}_k$'s, sine-basis of the Fourier modes, are also eigenvectors of the Jacobi matrix.

Next, damped Jacobi method with $w = 2/3$:

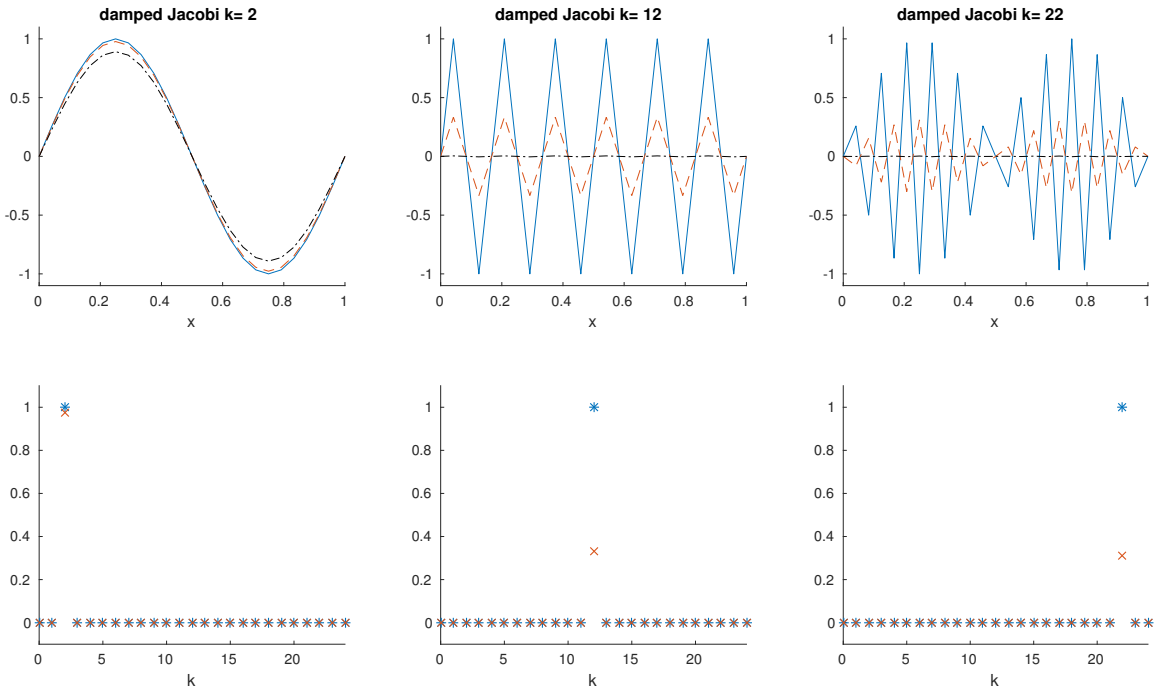


Figure 2.3: Top row: Initial error (solid line), error after one relaxation (dashed line) and error after 5 relaxations (dash-dot line) with damped Jacobi method with weight 2/3 for modes k=2 (low), k=12 (medium) and k=22 (high) where n=24; Second row: Fourier modes of the initial error (*) and error after **one** relaxation (x) with damped Jacobi method

This time, we see that damped Jacobi method is quite successful in damping high frequency errors. Again, from Figure 2.1, we can see that the magnitude of the eigenvalues of the damped Jacobi matrix with weight 2/3 is less than or equal to 1/3 for $k \geq n/2$. However, we cannot say the same thing for the low frequency errors. The magnitude of the eigenvalues of the damped Jacobi matrix with weight

2/3 are close to 1 for small k's, hence we cannot expect a good damping rate for low frequencies.

We get similar results from Gauss-Seidel iteration:

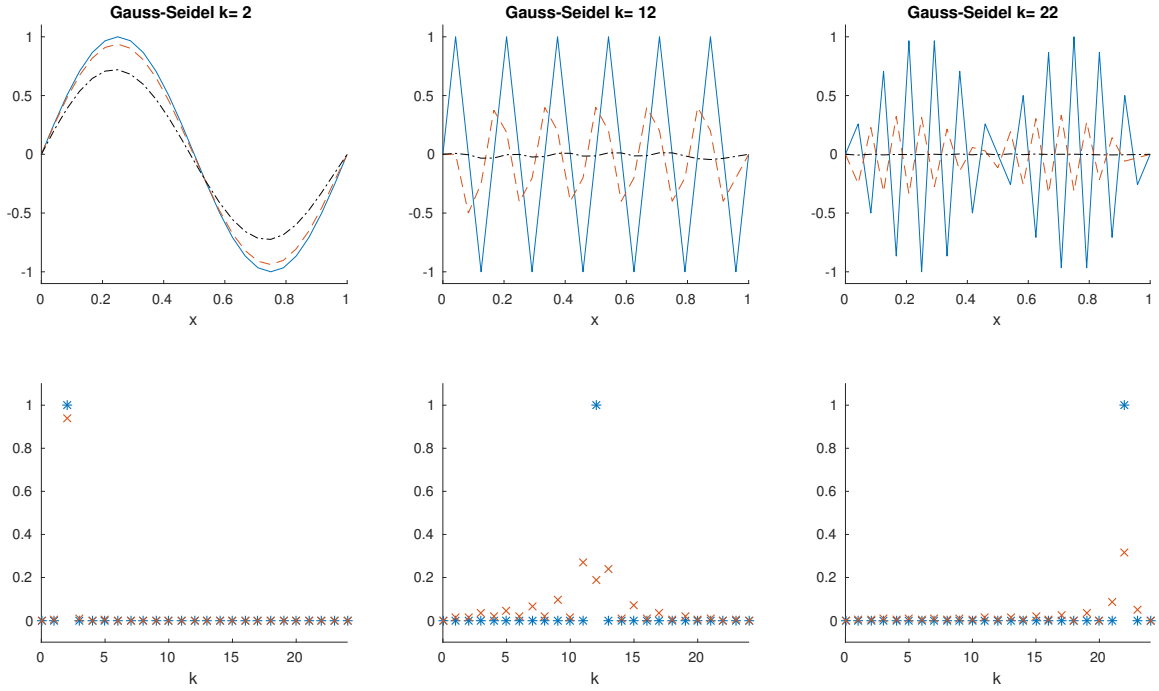


Figure 2.4: Top row: Initial error (solid line), error after one relaxation (dashed line) and error after 5 relaxations (dash-dot line) with Gauss-Seidel method for modes k=2 (low), k=12 (medium) and k=22 (high) where n=24; Second row: Fourier modes of the initial error (*) and error after **one** relaxation (x) with Gauss-Seidel method

We see that Gauss-Seidel method is good for high frequencies, but not that good for low frequencies (See Figure 2.4). Unlike Jacobi or damped Jacobi methods, Gauss-Seidel excites a couple of different modes. We can explain this fact by looking back to our eigenvalue-eigenvector discussion: Our initial error has only one mode, $\mathbf{v}_k$ for some k. Since $\mathbf{v}_k$ is not an eigenvector of C_{GS}, $C_{GS}\mathbf{v}_k$ is a "new" and different vector,

which can be represented with a new combination of $\mathbf{v}_l$'s. This is what we see here: a combination of different modes.

The last relaxation method we discussed was the red-black Gauss-Seidel method:

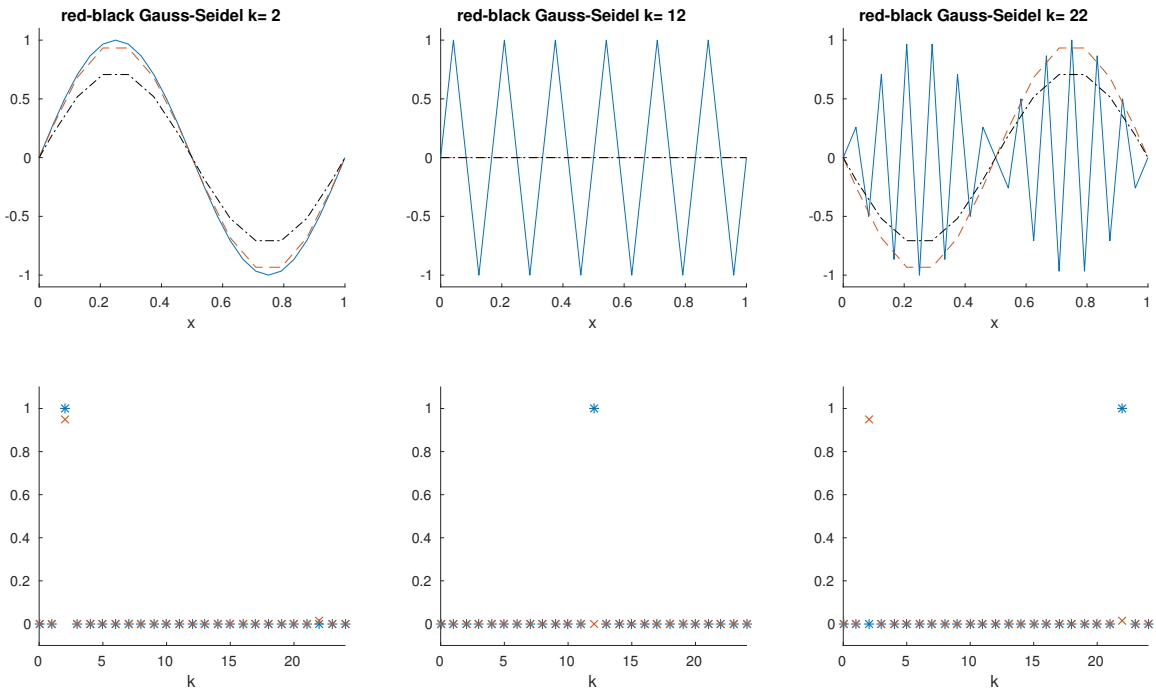


Figure 2.5: Top row: Initial error (solid line), error after one relaxation (dashed line) and error after 5 relaxations (dash-dot line) with red-black Gauss-Seidel method for modes k=2 (low), k=12 (medium) and k=22 (high) where n=24; Second row: Fourier modes of the initial error (*) and error after **one** relaxation (x) with red-black Gauss-Seidel method

As the other methods, we see that it also doesn't help much with the low frequency errors. Something surprising happens with the high modes: We get rid of the high frequency error right after the first iteration, but we see a low frequency error at

$(n-k)$th mode, which has a magnitude not very different from the initial error. Doing consecutive red-black Gauss-Seidel iterations on the low frequency errors decreases the error with a very slow rate. So for high frequency errors, although red-black Gauss-Seidel damps them immediately, we cannot say the method is successful, since we will have the low frequency "correspondent" of the error, which is not small in magnitude and cannot be damped effectively.

Red-black Gauss-Seidel method sounds a little disappointing, in the sense that we were hoping to get a significant improvement when it was used. Soon we'll see that its "shifting the high frequency to low frequency" behaviour helps a lot, since we have another way to handle low frequency errors. We'll discuss this "way" starting in the next chapter.

To summarize, although we can get some success over the high frequency errors, unfortunately, none of our basic relaxation methods are effectively damping the low frequency errors. This leads us to the question: "What else can be done? Can we use the relaxation methods in another way or with another method so that we can always damp the error, regardless of it is being of low or high frequency?" The answer is, as you can guess, yes. We'll focus on one of these ways/methods in the coming chapters.

Chapter 3: Multigrid in One Dimension

As shown in the previous chapter, the relaxation methods we discused are not very helpful if we want to reduce the low frequency error. But they can be used in a way that, with the help of a direct solver, low frequency errors are also damped. In this chapter, we'll discuss this way and its efficiency on the simple one dimensional setting.

This way, namely multigrid, is based on using relaxation on the finer grids, where it is costly to use a direct method, then calculating the error after the relaxation and projecting it down to a coarser grid. These relaxation-projection steps are continuously done until we reach a "coarse enough" grid, where we can apply a direct solver without a high cost. We use the result from the direct solver to correct our solution at each step while we are going back to the initial fine grid. During the journey back to the initial grid, it is possible to make use of the relaxation methods as well.

3.1 Two Grid Method

We start discussing multigrid method with the two grid method, where we only make use of two different grids, by going down to a coarser grid to use a direct method, and going up to the initial grid immediately after that. The steps of the two grid method to solve the discritized problem $A\mathbf{u} = \mathbf{f}$ are as follows:

1. Start with an initial guess $\mathbf{u}^{h(0)}$ for the solution

2. On the initial grid, apply the chosen relaxation method on $A^h\mathbf{u}^h = \mathbf{f}^h$ with the initial iterate $\mathbf{u}^{h(0)}$ ν_1 times, where h is the distance between two grid points.

3. Calculate the resudual $\mathbf{r}^h = \mathbf{f}^h - A^h\mathbf{u}^{h(\nu_1)}$, where $\mathbf{u}^{h(\nu_1)}$ is the solution after ν_1 relaxation steps.

4. Project the residual $\mathbf{r}^h$ down to the coarse grid by the projection operator I_h^{2h}, which will be defined below, to get $\mathbf{r}^{2h}$. Similar as above, $2h$ is the distance between two grid points in the coarse grid.

5. Use a direct method to solve $A^{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$.

6. Interpolate $\mathbf{e}^{2h}$ to get $\mathbf{e}^h$ on the fine grid, and update the solution using $\mathbf{e}^h$ by $\mathbf{u}^{h(\nu_1)} = \mathbf{u}^{h(\nu_1)} + \mathbf{e}^h$.

7. Apply the chosen relaxation method on $A^h\mathbf{u}^h = \mathbf{f}^h$ with the initial iterate $\mathbf{u}^{h(\nu_1)}$ ν_2 times.

One can apply two grid method as many times as wanted, using the solution of the last one as the initial guess of the next one.

The projection operator I_h^{2h} mentioned above is used to transfer the data to a coarser grid as a weighted average of the data on three consecutive grid points on the fine grid as

$$x_j^{2h} = 0.25x_{2j-1}^h + 0.5x_{2j}^h + 0.25x_{2j+1}^h \quad \text{where} \quad j = 1, 2, ..., \frac{n}{2} - 1$$

We can write this relation in the matrix form $\mathbf{x}^{2h} = I_h^{2h}\mathbf{x}^h$, where I_h^{2h} is a $(\frac{n}{2}-1)\times(n-1)$ matrix.

$$I_h^{2h} = \begin{pmatrix} 0.25 & 0.5 & 0.25 & 0 & & & & & \cdots & 0 \\ 0 & 0 & 0.25 & 0.5 & 0.25 & 0 & & & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0.25 & 0.5 & 0.25 & 0 & \cdots & 0 \\ & & & & & \ddots & \ddots & \ddots & & \\ 0 & & & & & \cdots & 0 & 0.25 & 0.5 & 0.25 \end{pmatrix}.$$

There is another way to transfer data from the fine grid to coarse grid, which is called *injection*. In injection, we simply ignore every other grid point on the fine grid. The transfer is made with the correspondence

$$x_j^{2h} = x_{2j}^h \quad \text{where} \quad j = 1, 2, ..., \frac{n}{2} - 1$$

The corresponding $(\frac{n}{2}-1)\times(n-1)$ matrix for this transfer is

$$I_h^{2h} = \begin{pmatrix} 0 & 1 & 0 & 0 & & & & & \cdots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & & & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ & & & & & \ddots & \ddots & \ddots & & \\ 0 & & & & & \cdots & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Although simpler than projection, injection might be a bad choice to transfer data between grids. The problem occurs when our data that we want to transfer is of high frequency: We observe aliasing.

For example, let us have the data $\mathbf{v}_{k,j}^h = \sin(\pi jkh)$, where $j \in \{1, \cdots, n-1\}$ and $k > \frac{n}{2}$. If we use injection, on the coarser grid we get

$$v_{k,j}^{2h} = v_{k,2j}^h = \sin(2\pi jkh)$$

Since $n > k > \frac{n}{2}$, $2n > 2k > n$, so $\sin(2\pi jkh) = -\sin(\pi(2n-2k)jh) = -\sin(2\pi(n-k)jh)$. Denoting $n-k$ as k', which is less than $\frac{n}{2}$, we have

$$v_{k,j}^{2h} = -\sin(2\pi k'jh) = -v_{k',j}^{2h}$$

So as it turned out, the high frequency error we started with on the fine grid, ended up being a low frequency error on the coarse grid! When we want to transfer the error back up to the finer grid, we carry the "new" low frequency error and try to deal with that, rather than working on reducing the original high frequency error.

If we use projection to change grids, we also observe an unavoidable shift in frequencies while reducing the number of grid points. However, this is not a big problem as it is with injection: Focusing on the same example, this time we have

$$\begin{aligned} v_{k,j}^{2h} =& \tfrac{1}{4}\left(v_{k,2j-1}^{h} + 2v_{k,2j}^{h} + v_{k,2j+1}^{h}\right) \\ =& \tfrac{1}{4}\left(\sin(\pi(2j-1)kh) + 2\sin(\pi 2jkh) + \sin(\pi(2j+1)kh)\right) \\ =& \tfrac{1}{4}\left(\sin(\pi 2jkh)\cos(\pi kh) + 2\sin(\pi 2jkh) + \sin(\pi 2jkh)\cos(\pi kh)\right) \\ =& \tfrac{1}{4}2\sin(\pi 2jkh)(\cos(\pi kh)+1) = \tfrac{1}{2}(-\sin(\pi 2jk'h))(-\cos(\pi k'h)+1) \\ =& -\sin^2(\tfrac{1}{2}\pi k'h)v_{k',j}^{2h} \end{aligned}$$

Although we see the same shift, this time it is damped with a factor of $\sin^2(\frac{1}{2}\pi k'h)$, where $k' < \frac{n}{2}$, so $\sin^2(\frac{1}{2}\pi k'h) < \sin^2(\frac{\pi}{4}) = \frac{1}{2}$, hence its "contribution" to the error is less.

In the other direction, from the coarser grid to the finer grid, we transfer data by

$$x_{2j}^{h} = x_{j}^{2h} \quad \text{where} \quad j = 1, 2, ..., \frac{n}{2} - 1$$

$$x_{2j+1}^{h} = 0.5(x_{j}^{2h} + x_{j+1}^{2h}) \quad \text{where} \quad j = 0, 1, 2, ..., \frac{n}{2} - 1$$

which can be represented by the $(n-1) \times \left(\frac{n}{2}-1\right)$ *interpolation* matrix I_{2h}^{h} as

$$I_{2h}^{h} = \begin{pmatrix} 0.5 & 0 & & & \cdots & 0 \\ 1 & 0 & & & \cdots & 0 \\ 0.5 & 0.5 & 0 & & \cdots & 0 \\ 0 & 1 & 0 & & \cdots & 0 \\ 0 & 0.5 & 0.5 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & & 0 \\ 0 & 0 & 0.5 & \ddots & & 0 \\ 0 & 0 & 0 & \ddots & & 0.5 \\ & & & & & 1 \\ & & & & & 0.5 \end{pmatrix}$$

which is 2 times the transpose of the projection matrix I_{h}^{2h}.

Figure 3.1 shows an example of data transfer between grids. On both graphs initial error, arbitrarily chosen as $\sin(57\pi x)$, is plotted with solid lines on a grid with 65 points, i.e. $n = 64$. After transfering the data to the coarser grid, we get the circles, which construct the curve of $\sin(7\pi x)$ when connected where $7 = 64-57 = n-k = k'$, agreeing with our calculations. On the left graph, injection was chosen as the method of transfer and we see that the maximum error doesn't decrease. On the other side, projection was chosen as the method of transfer and we see a very strong damping of the error. In both graphs, we observe the frequency shift. Lastly, the dashed lines represent interpolation, transfering the data back to the finer grid.

Now, let's take a look at how "useful/powerful" two grid method is (or is not). We'll start with a theorem from [1]:

Theorem 3.1. *If we want to solve $-u_{xx} = f(x)$ with $u(x) = 0$ on the boundaries, we get the discretization $A^h \boldsymbol{u}^h = \boldsymbol{f}^h$ where A^h is a tridiagonal matrix with $2/h^2$ on the diagonal and $-1/h^2$ on the 1st diagonals above and below the main diagonal. If we apply the two grid method with the choice of weighted Jacobi method with $w = 2/3$*

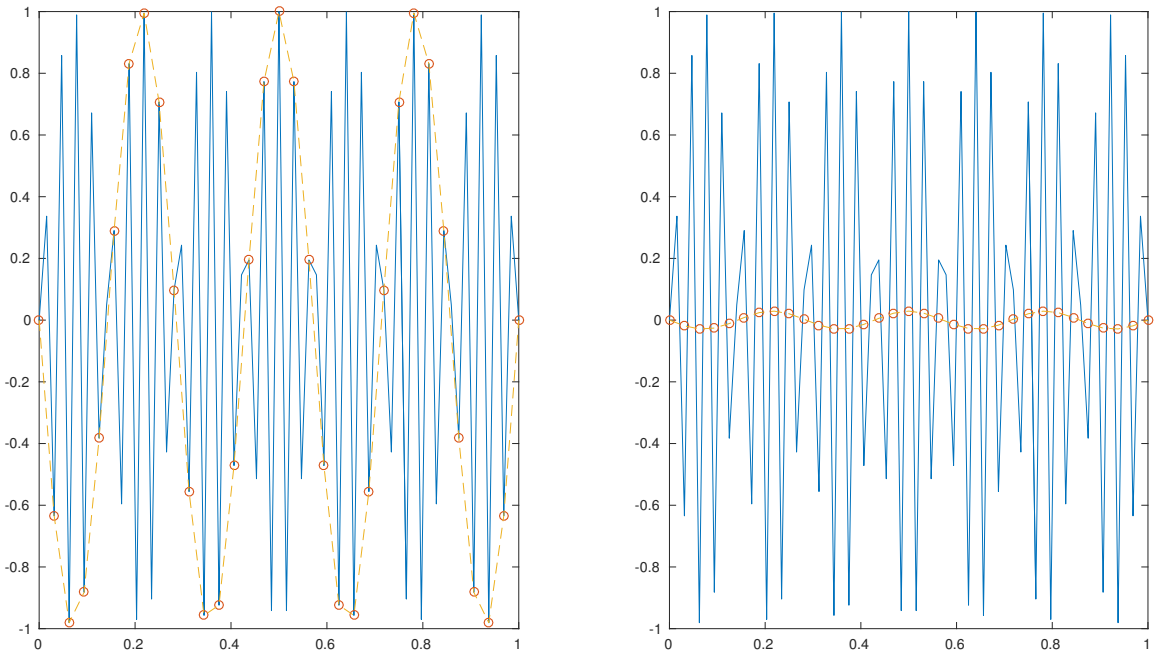


Figure 3.1: Solid lines: initial, circles: left-after injection, right-after projection, dashed lines: after interpolation. $n = 64$. initial error: $\sin(57\pi x)$

for relaxations, we get

$$\|\boldsymbol{e}^{(k)}\|_2 < \sqrt{\left(\frac{1}{4} + \frac{1}{2}\frac{1}{3^{\nu_1}}\right)\left(1 + \frac{1}{9^{\nu_2}}\right)}\|\boldsymbol{e}^{(k-1)}\|_2$$

where $\boldsymbol{e}^{(k)}$ *is the error after kth two grid method has been applied.*

For example, when $\nu_1 = 2$ *and* $\nu_2 = 0$*, we get*

$$\|\boldsymbol{e}^{(k)}\|_2 < 0.8\|\boldsymbol{e}^{(k-1)}\|_2$$

Proof. If we recall the relaxation methods from the previous chapter, for $A\mathbf{u} = \mathbf{f}$, we were writing A as $B+(A-B)$ with a suitable matrix B, so that our problem becomes

$$B\mathbf{u} + (A - B)\mathbf{u} = \mathbf{f}$$

and to update the solution, we use the iterations

$$\mathbf{u}^{(k)} = B^{-1}(B - A)\mathbf{u}^{(k-1)} + B^{-1}\mathbf{f}$$

where $\mathsf{C} = B^{-1}(B - A)$ is the relaxation matrix. It is easy to show that errors satisfy the iterative relation $\mathbf{e}^{(k)} = \mathsf{C}\mathbf{e}^{(k-1)}$, where $\mathbf{e}^{(k)}$ is defined by the difference between the calculated solution after the kth iteration and true solution, i.e. $\mathbf{e}^{(k)} = \mathbf{u}^{(k)} - \bar{\mathbf{u}}$ with $\bar{\mathbf{u}}$ being the true solution:

$$\begin{aligned}
\mathbf{e}^{(k)} &= \mathbf{u}^{(k)} - \bar{\mathbf{u}} \quad = \quad B^{-1}(B - A)\mathbf{u}^{(k-1)} + B^{-1}\mathbf{f} - \bar{\mathbf{u}} \\
&= \mathsf{C}\mathbf{u}^{(k-1)} + B^{-1}(B + (A - B))\bar{\mathbf{u}} - \bar{\mathbf{u}} \\
&= \mathsf{C}\mathbf{u}^{(k-1)} + \bar{\mathbf{u}} - \mathsf{C}\bar{\mathbf{u}} - \bar{\mathbf{u}} \quad = \quad \mathsf{C}(\mathbf{u}^{(k-1)} - \bar{\mathbf{u}}) \\
&= \mathsf{C}\mathbf{e}^{(k-1)} \qquad (3.1)
\end{aligned}$$

So if we start with some initial guess $\mathbf{u}^{h(0)}$ with an initial error $\mathbf{e}^{h(0)}$ on the fine grid, following the steps of two grid method we get

Relaxation: $\mathbf{e}^{h(\nu_1)} = \mathsf{C}^{\nu_1}\mathbf{e}^{h(0)}$

Residual: $\mathbf{r}^h = \mathbf{f}^h - A^h\mathbf{u}^{h(\nu_1)} = \mathbf{f}^h - A^h(\bar{\mathbf{u}} + \mathbf{e}^{h(\nu_1)})$

$= \mathbf{f}^h - A^h\bar{\mathbf{u}} - A^h\mathbf{e}^{h(\nu_1)} = -A^h\mathbf{e}^{h(\nu_1)} = -A^h\mathsf{C}^{\nu_1}\mathbf{e}^{h(0)}$

Projection: $\mathbf{r}^{2h} = I_h^{2h}\mathbf{r}^h = -I_h^{2h}A^h\mathsf{C}^{\nu_1}\mathbf{e}^{h(0)}$

Direct Solution: $\mathbf{e}^{2h} = (A^{2h})^{-1}\mathbf{r}^{2h} = -(A^{2h})^{-1}I_h^{2h}A^h\mathsf{C}^{\nu_1}\mathbf{e}^{h(0)}$

Interpolation and update: $\mathbf{u}^h = \mathbf{u}^{h(\nu_1)} + I_{2h}^h\mathbf{e}^{2h}$

$= \bar{\mathbf{u}} + \mathsf{C}^{\nu_1}\mathbf{e}^{h(0)} - I_{2h}^h(A^{2h})^{-1}I_h^{2h}A^h\mathsf{C}^{\nu_1}\mathbf{e}^{h(0)}$

i.e. $\mathbf{e}^h = \mathbf{u}^h - \bar{\mathbf{u}} = (I - I_{2h}^h(A^{2h})^{-1}I_h^{2h}A^h)\mathsf{C}^{\nu_1}\mathbf{e}^{h(0)}$

Relaxation: $\mathbf{e}^{h(1)} = \mathsf{C}^{\nu_2}(I - I_{2h}^h(A^{2h})^{-1}I_h^{2h}A^h)\mathsf{C}^{\nu_1}\mathbf{e}^{h(0)}$

where I is the identity matrix and

$$G = I - I_{2h}^{h}(A^{2h})^{-1}I_{h}^{2h}A^{h} \tag{3.2}$$

is the matrix representing the grid change and direct solution of the error on the coarser grid.

Here we will make use of the fact that A^h and C for the damped Jacobi method have the same eigenvectors $\mathbf{v}_k^h$ where

$$\mathbf{v}_k^h = \begin{pmatrix} \sin(\pi kh) \\ \sin(2\pi kh) \\ \vdots \\ \sin((n-1)\pi kh) \end{pmatrix}$$

with different eigenvalues,

$$\lambda_k = \frac{4}{h^2}\sin^2(\tfrac{1}{2}\pi kh) \quad \text{and} \quad \mu_k = 1 - 2w\sin^2(\tfrac{1}{2}\pi kh)$$

with $w = \frac{2}{3}$, correspondingly. (From now on, we'll use the notation $\mu_k = 1 - \frac{4}{3}\sin^2(\theta_k)$ where $\theta_k = \frac{1}{2}\pi kh$.) We will also use the following properties of the projection and interpolation matrices:

$$\begin{aligned} I_{2h}^{h}\mathbf{v}_k^{2h} &= \cos^2(\theta_k)\mathbf{v}_k^h - \sin^2(\theta_k)\mathbf{v}_{k'}^h \\ I_{h}^{2h}\mathbf{v}_k^{h} &= \begin{cases} \cos^2(\theta_k)\mathbf{v}_k^{2h} & 1 \le k < \frac{n}{2} \\ -\sin^2(\theta_{k'})\mathbf{v}_{k'}^{2h} & \frac{n}{2} \le k \le n-1 \end{cases} \end{aligned}$$

where $k' = n - k$. Using all these to get a simple expression for $G\mathbf{v}_k^h$, we get

$$G\mathbf{v}_k^h = \begin{cases} \sin^2(\theta_k)(\mathbf{v}_k^h + \mathbf{v}_{k'}^h) & 1 \le k < \frac{n}{2} \\ \cos^2(\theta_{k'})(\mathbf{v}_k^h + \mathbf{v}_{k'}^h) & \frac{n}{2} \le k \le n-1 \end{cases} \tag{3.3}$$

Here we can see that special "symmetric" nature of two grid method: Starting with the k^{th} eigenvector, we end up getting k^{th} and $(n-k)^{th}$ eigenvectors together. This step damps the low frequency errors relatively well even though it creates its high

frequency symmetry, but it is not that effective for high frequency errors and might increase the error by not damping the high frequency error much and creating a symmetric one.

Since the eigenvectors of A^h (or of C) creates an orthonormal basis (due to the orthogonality of the $\sin(m\pi kh)$ functions), we can represent our initial error as a linear combination of $\{\mathbf{v}_k\}$s on the fine grid:

$$e^{(0)} = \gamma_1 \mathbf{v}_1 + \gamma_2 \mathbf{v}_2 + \cdots + \gamma_{n-1}\mathbf{v}_{n-1}.$$

Finally, taking the relaxation steps into consideration, which can be represented with the matrix C with eigenvalues λ_k and eigenvectors $\mathbf{v}_k$, we get

$$\mathsf{C}^{\nu_2} G \mathsf{C}^{\nu_1} \mathbf{e}^{(0)} = \mathsf{C}^{\nu_2} \left[\sum_{k=1}^{(n/2)-1} \gamma_k \lambda_k^{\nu_1} \sin^2(\theta_k)(\mathbf{v}_k + \mathbf{v}_{k'}) + \sum_{k=n/2}^{n-1} \gamma_k \lambda_k^{\nu_1} \cos^2(\theta_{k'})(\mathbf{v}_k + \mathbf{v}_{k'}) \right]$$

$$= \sum_{k=1}^{(n/2)-1} \lambda_k^{\nu_2} \left[\gamma_k \lambda_k^{\nu_1} \sin^2(\theta_k) + \gamma_{k'} \lambda_{k'}^{\nu_1} \cos^2(\theta_k) \right] \mathbf{v}_k$$

$$+ \sum_{k=n/2}^{n-1} \lambda_k^{\nu_2} \left[\gamma_k \lambda_k^{\nu_1} \cos^2(\theta_{k'}) + \gamma_{k'} \lambda_{k'}^{\nu_1} \sin^2(\theta_{k'}) \right] \mathbf{v}_k.$$

When looking at the two-norm of the above expression, i.e., $\|\mathsf{C}^{\nu_2} G \mathsf{C}^{\nu_1} \mathbf{e}^{(0)}\|_2 = \sqrt{< \mathsf{C}^{\nu_2} G \mathsf{C}^{\nu_1} \mathbf{e}^{(0)}, \mathsf{C}^{\nu_2} G \mathsf{C}^{\nu_1} \mathbf{e}^{(0)} >}$, while keeping in mind that $\mathbf{v}_k$ forms an orthonormal basis, $\sin^2(\theta_k)$,$\cos^2(\theta_{k'}) \leq 0.5$ when $k \leq \frac{n}{2}$, $|\lambda_k| \leq \frac{1}{3}$ when $k \geq \frac{n}{2}$ for weighted Jacobi with $w = 2/3$, and $2|\gamma_k \gamma_{k'}| \leq \gamma_k^2 + \gamma_{k'}^2$, we see that we can bound the ratio of the errors by

$$\sqrt{\left(\frac{1}{4} + \frac{1}{2}\frac{1}{3^{\nu_1}}\right)\left(1 + \frac{1}{9^{\nu_2}}\right)}$$

Considering the given example in the statement of the theorem, when we evalute the above expression with $\nu_1 = 2$ and $\nu_2 = 0$, we get $\sqrt{2\left(\frac{1}{4} + \frac{1}{2}\frac{1}{9}\right)}$, which is equal to 0.7817. So we have $\|e^{(k)}\|_2 < 0.8\|e^{(k-1)}\|_2$, as stated in the theorem. □

According to this bound, our error can be reduced with a rate of $\sim 20\%$. But when we actually apply two grid method, we see that the error is reduced at a higher rate. The upper bounds we used above for $\sin^2(\theta_k)$,$\cos^2(\theta_{k'})$ and λ_k when $k < \frac{n}{2}$ are not very good approximations for many k's. Especially for the latter, we used $\lambda_k = 1$ for $k < \frac{n}{2}$, which ignores the contribution of the relaxation. This leads us to have a deeper look into the two grid method.

Corollary 3.2. *The results of Theorem 3.1 when $\nu_1 = 2$ and $\nu_2 = 0$ can be improved to*

$$\|\boldsymbol{e}^{(k)}\|_2 < \frac{\sqrt{2}}{9}\|\boldsymbol{e}^{(k-1)}\|_2 = 0.157\|\boldsymbol{e}^{(k-1)}\|_2$$

by an analysis of the actual expression of the errors.

Proof. We can start by considering the basis elements $\mathbf{v}_k$ one by one. When we apply the two grid method on each one of the basis elements, we can see how much each basis element is damped. Since in this case only one γ_k will be nonzero, most of the terms vanish and we are left with:

$$\mathsf{C}^{\nu_2} G \mathsf{C}^{\nu_1} \mathbf{v}_k = \begin{cases} \sin^2(\theta_k)\lambda_k^{\nu_1}(\lambda_k^{\nu_2}\mathbf{v}_k + \lambda_{k'}^{\nu_2}\mathbf{v}_{k'}) & k < \frac{n}{2} \\ \cos^2(\theta_{k'})\lambda_k^{\nu_1}(\lambda_k^{\nu_2}\mathbf{v}_k + \lambda_{k'}^{\nu_2}\mathbf{v}_{k'}) & k \geq \frac{n}{2} \end{cases}$$

whose 2-norm is

$$\|\mathsf{C}^{\nu_2} G \mathsf{C}^{\nu_1} \mathbf{v}_k\|_2 = \begin{cases} \sin^2(\theta_k)|\lambda_k^{\nu_1}|\sqrt{\lambda_k^{2\nu_2} + \lambda_{k'}^{2\nu_2}} & k < \frac{n}{2} \\ \cos^2(\theta_{k'})|\lambda_k^{\nu_1}|\sqrt{\lambda_k^{2\nu_2} + \lambda_{k'}^{2\nu_2}} & k \geq \frac{n}{2} \end{cases}$$

Let us consider some particular cases to see the actual power of two grid method. For example, looking at the case where ν_1 is arbitrary and $\nu_2 = 0$, we see that the error becomes

$$\|\mathsf{C}^0 G \mathsf{C}^{\nu_1} \mathbf{v}_k\|_2 = \begin{cases} \sin^2(\theta_k)|\lambda_k^{\nu_1}|\sqrt{2} & k < \frac{n}{2} \\ \cos^2(\theta_{k'})|\lambda_k^{\nu_1}|\sqrt{2} & k \geq \frac{n}{2} \end{cases}$$

Recalling that for weighted Jacobi method, we had $\lambda_k = 1 - \frac{4}{3}\sin^2(\theta_k)$ where $w = \frac{2}{3}$ and $\theta_k = \frac{1}{2}\pi kh$, we can rewrite the error (using the identity $\sin(\theta_{k'}) = \cos(\theta_k)$) as

$$\|\mathsf{C}^0 G \mathsf{C}^{\nu_1} \mathbf{v}_k\|_2 = \begin{cases} \sin^2(\theta_k)|(1 - \frac{4}{3}\sin^2(\theta_k))^{\nu_1}|\sqrt{2} & k < \frac{n}{2} \\ \cos^2(\theta_{k'})|(1 - \frac{4}{3}\sin^2(\theta_k))^{\nu_1}|\sqrt{2} & k \geq \frac{n}{2} \end{cases}$$

To compare the result above with the previous estimate, we again focus on our test case, when $\nu_1 = 2$. We want to know, what is the maximum error we can get from any of these basis functions. To find that value, we are going to take the derivative of the error:

$$\begin{aligned} \frac{d}{d\theta}\|\mathsf{C}^0 G \mathsf{C}^2 \mathbf{v}_k\|_2 &= \begin{cases} \sqrt{2}\left(1 - \frac{16}{3}\sin^2(\theta) + \frac{16}{3}\sin^4(\theta)\right) 2\sin(\theta)\cos(\theta) & k < \frac{n}{2} \\ \sqrt{2}\left(1 - \frac{16}{3}\cos^2(\theta) + \frac{16}{3}\cos^4(\theta)\right) 2\cos(\theta)(-\sin(\theta) & k \geq \frac{n}{2} \end{cases} \\ &= \begin{cases} \sqrt{2}\left(1 - \frac{16}{3}\sin^2(\theta)(1 - \sin^2(\theta))\right)\sin(2\theta) & k < \frac{n}{2} \\ \sqrt{2}\left(1 - \frac{16}{3}\cos^2(\theta)(1 - \cos^2(\theta))\right)(-\sin(2\theta) & k \geq \frac{n}{2} \end{cases} \\ &= \begin{cases} \sqrt{2}\left(1 - \frac{4}{3}\sin^2(2\theta)\right)\sin(2\theta) & k < \frac{n}{2} \\ \sqrt{2}\left(1 - \frac{4}{3}\sin^2(2\theta)\right)(-\sin(2\theta)) & k \geq \frac{n}{2} \end{cases} \end{aligned}$$

Setting the derivative equal to zero gives us two options in our interval of interest, $(0, \frac{\pi}{2})$: One is $2\theta_k = kh\pi$ can be a multiple of π, which is true when k is a multiple of n. But our k is between 1 and $n-1$, so this is not an option. The second option is when $\sin(2\theta) = \frac{\sqrt{3}}{2}$, which is true when $2\theta = \frac{\pi}{3}, \frac{2\pi}{3}$, i.e. $\theta = \frac{\pi}{6}, \frac{\pi}{3}$. Checking their second derivatives, we see that $\theta = \frac{\pi}{6}$ corresponds to a local maximum, whereas at $\theta = \frac{\pi}{3}$ we have a local minimum. When $\theta = \frac{\pi}{6} = \frac{1}{2}\pi kh$, k is approximately $\frac{n}{3}$. At $k = \frac{n}{3}$, the ratio is $\frac{\sqrt{2}}{9}$. Comparing this value with the endpoints, we see that we can

get very close to $\frac{\sqrt{2}}{9}$ on the right end, where the function is nondecreasing. Hence, the maximum ratio we can get is $\frac{\sqrt{2}}{9} = 0.157$.

This means, with just one iteration of two grid method with $\nu_1 = 2$ and $\nu_2 = 0$, the error is reduced at least by 84%. Since any $\mathbf{e}^{(0)}$ can be written as a linear combination of $\mathbf{v}_k$'s, and since this bound is true for any $\mathbf{v}_k$, this ratio of errors is also valid for any $\mathbf{e}^{(0)}$.

Even if we started with one single basis element, we end up with two components after the first iteration, since going down and up gives us a symmetric outcome. The final output might not be symmetric, if ν_2 is nonzero, though. Applying the two grid method for a second time on the initial input, i.e. once on the first outcome with two components, we get

$$\begin{aligned}(\mathsf{C}^{\nu_2} G \mathsf{C}^{\nu_1})^2 \mathbf{v}_k =& \mathsf{C}^{\nu_2} G \mathsf{C}^{\nu_1} (\lambda_k^{\nu_1} \sin^2(\theta_k)(\lambda_k^{\nu_2}\mathbf{v}_k + \lambda_{k'}^{\nu_2}\mathbf{v}_{k'})) \\ &= \lambda_k^{\nu_1} \sin^2(\theta_k)(\lambda_k^{\nu_2}\mathbf{v}_k + \lambda_{k'}^{\nu_2}\mathbf{v}_{k'})(\sin^2(\theta_k)\lambda_k^{\nu_1+\nu_2} + \sin^2(\theta_{k'})\lambda_{k'}^{\nu_1+\nu_2}) \\ &= (\sin^2(\theta_k)\lambda_k^{\nu_1+\nu_2} + \sin^2(\theta_{k'})\lambda_{k'}^{\nu_1+\nu_2})\mathsf{C}^{\nu_2} G \mathsf{C}^{\nu_1}\mathbf{v}_k\end{aligned}$$

In fact we get the same reduction rate between two consecutive iterations after the first iteration, and this rate is independent of the chosen norm, and only dependent on k and $\nu_1 + \nu_2$:

$$(\mathsf{C}^{\nu_2} G \mathsf{C}^{\nu_1})^m \mathbf{v}_k = \left(\lambda_k^{\nu_1+\nu_2} \sin^2(\theta_k) + \lambda_{k'}^{\nu_1+\nu_2} \sin^2(\theta_{k'})\right) (\mathsf{C}^{\nu_2} G \mathsf{C}^{\nu_1})^{m-1} \mathbf{v}_k$$

for any $m \geq 2$.

Again, considering our test case $\nu_1 + \nu_2 = 2$, we get

$$\lambda_k^2 \sin^2(\theta_k) + \lambda_{k'}^2 \sin^2(\theta_{k'}) = (1 - \tfrac{4}{3}\sin^2(\theta_k))^2 \sin^2(\theta_k) + (1 - \tfrac{4}{3}\sin^2(\theta_{k'}))^2 \sin^2(\theta_{k'})$$

which can be written in a simple form as

$$(1-\tfrac{4}{3}w)^2w+(1-\tfrac{4}{3}z)^2z$$

where $w=\sin^2(\theta_k)$ and $z=\sin^2(\theta_{k'})=\cos^2(\theta_k)$. Note that $w+z=1$ and $w'=-z'$. When we take the derivative of the above expression to find where the maximum ratio occurs, we get

$$\begin{aligned}&\left(-\tfrac{8}{3}(1-\tfrac{4}{3}w)w+(1-\tfrac{4}{3}w)^2\right)w'+\left((1-\tfrac{4}{3}z)^2-\tfrac{8}{3}(1-\tfrac{4}{3}z)z\right)z'\\&=\left((1-\tfrac{4}{3}w)(-\tfrac{8}{3}w+1-\tfrac{4}{3}w)\right)w'-\left((1-\tfrac{4}{3}z)(-\tfrac{8}{3}z+1-\tfrac{4}{3}z)\right)w'\end{aligned}$$

When we use the identity $w+z=1$, we see that

$$\begin{aligned}&(1-\tfrac{4}{3}w)(-\tfrac{8}{3}w+1-\tfrac{4}{3}w)=(1-\tfrac{4}{3}(1-z))(-4(1-z)+1)\\&=(-\tfrac{1}{3}+\tfrac{4}{3}z)(-3+4z)=-\tfrac{1}{3}(1-4z)(-3+4z)=(1-4z)(1-\tfrac{4}{3}z)\end{aligned}$$

So the above equation, i.e. the derivative, is 0 for any k. This means the ratio of consecutive errors after the first cycle is always the same with $\nu_1+\nu_2=2$, regardless of what k is. When we do a quick calculation, we actually see that this ratio is $\frac{1}{9}=0.111$.

Therefore if $\nu_1=2$ and $\nu_2=0$, the ratio of the consecutive errors has its maximum possible value in the first iteration, and that value is $\frac{\sqrt{2}}{9}=0.157$. □

When we check the damping rate with different basis functions and different ν_1 and ν_2, we see the relationship with $\theta_k=\frac{1}{2}\pi kh$ and the first damping rate in Figure 3.2.

We can observe that with only one relaxation in the beginning, we get the highest error for big k's, but if we do two, we have a better control on the high frequency

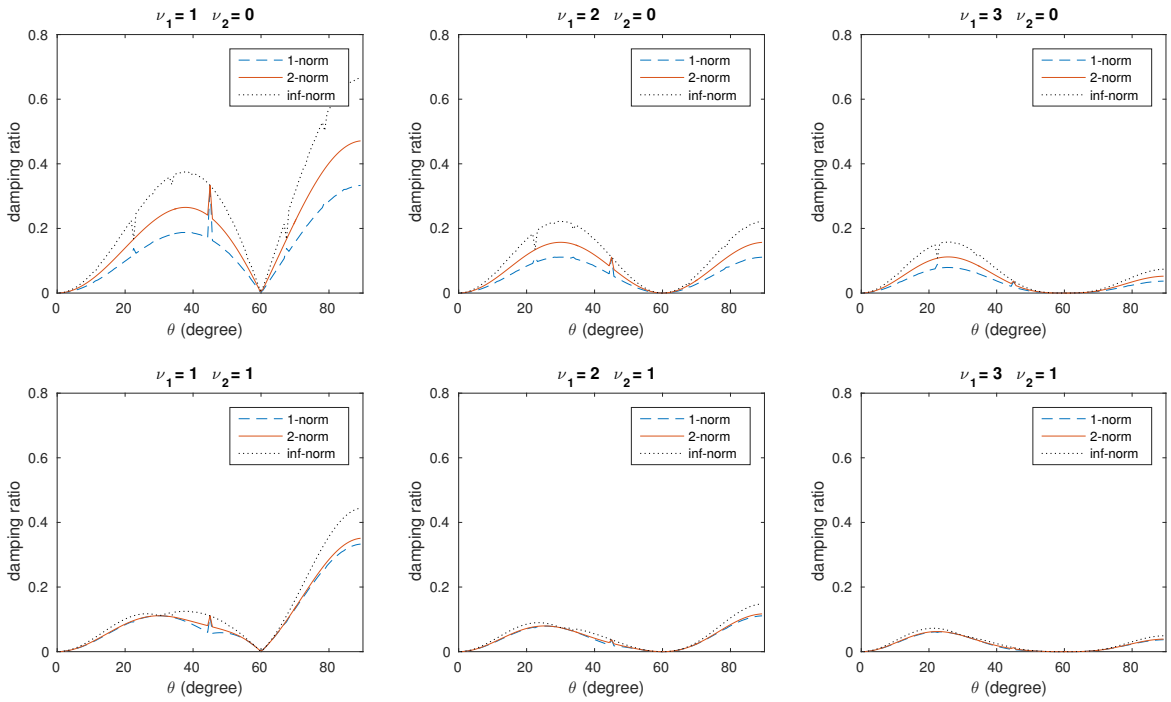


Figure 3.2: Ratio of errors in different norms before and after the first cycle of two grid method with damped Jacobi relaxation, $\omega = \frac{2}{3}$, ν_1: number of initial relaxations, ν_2: number of final relaxations, initial error is $\mathbf{v}_k$ where $k = \frac{2\theta n}{\pi}$, $n = 128$.

errors. We can compare the cases where the total number of iterations, i.e. $\nu_1 + \nu_2$, is the same and see that, doing the iteration in the beginning helps more than doing it at the end. This is related to the fact, that our relaxation methods are effective on the high frequency errors. When we start with a high frequency error, after the first relaxation(s), we get its symmetrical copy by going down and up. This symmetrical copy is in the low frequency region, so the relaxations at the end don't help it much, and its contribution to the error.

In Figure 3.3, we can see that after the first iteration, the ratio does not depend on ν_1 or ν_2 independently, but it depends on their sum, as shown in the proof of the

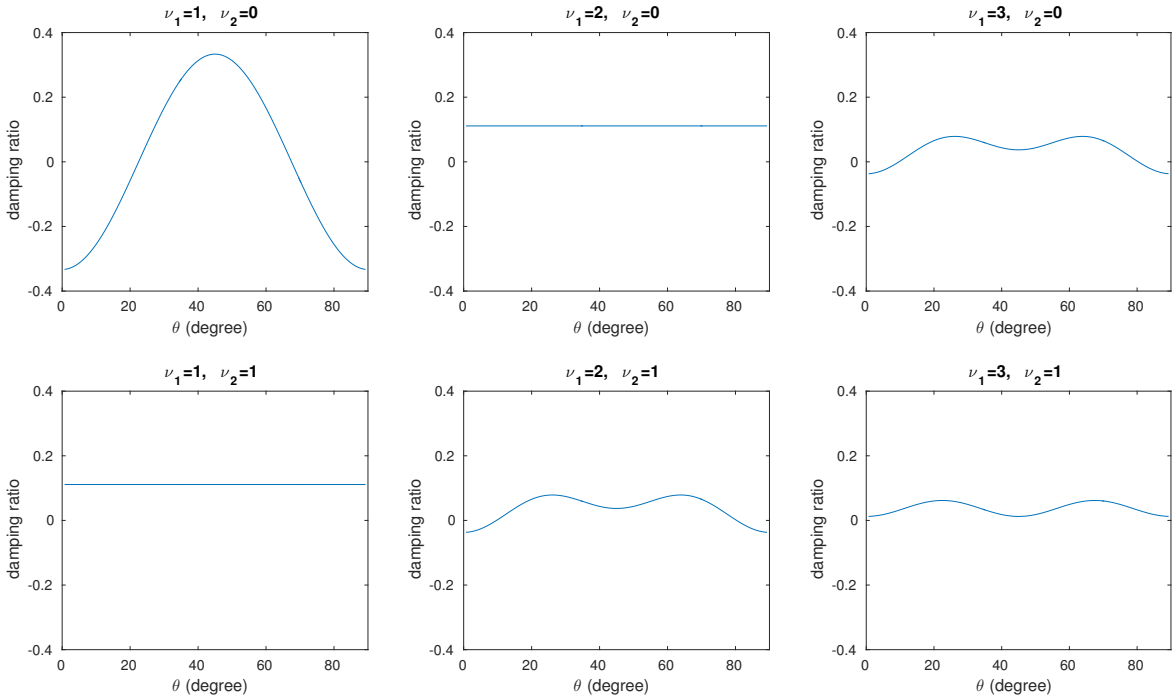


Figure 3.3: Ratio of errors before and after one cycle of two grid method after the first cycle, with damped Jacobi relaxation, $\omega = \frac{2}{3}$, ν_1: number of initial relaxations, ν_2: number of final relaxations.

theorem. Another fact is, as mentioned above, the two grid method works very well when we have low frequency errors, but we should be aware of the fact that it might be not the best for high frequencies. For example, when $\nu_1 = 1$ and $\nu_2 = 0$, for k close to n and initial guess is $\mathbf{v}_k$, we can damp the maximum error by 0.1179 with one cycle of two grid method with damped Jacobi relaxation, where relaxation part already brought down the error to 0.1381. However, instead of applying two grid method, if we just do two damped Jacobi relaxations, the maximum error goes down to 0.02.

Although increasing the number of relaxations will give a better result, it is not preferable due to its cost. But setting $\nu_1 = \nu_2 = 0$ is not a good idea either: Since some of the eigenvalues of G are equal to 1 (for example, $G\mathbf{v}^h_{\frac{n}{2}} = textbf{v}^h_{\frac{n}{2}}$), where G, defined in the equation 3.2, is the matrix representing two grid method without the relaxations, we need at least one relaxation to guarantee convergence. When we look at Figure 3.2 and Figure 3.3, we see that when $\nu_1 + \nu_2 = 2$, the damping ratio is significantly better than the case $\nu_1 + \nu_2 = 1$. However, if we increase the sum by 1, making $\nu_1 + \nu_2 = 3$, we don't get that much benefit. Therefore 2 is a good choice. As discussed above, doing the relaxation in the beginning helps more than doing it at the end, so $\nu_1 = 2$ and $\nu_2 = 0$ would be a wise choice.

So far we have observed the decrease of the error in our model problem ($-u_{xx} = f(x)$ with $u(x) = 0$ on the boundaries), using two-grid method with damped Jacobi relaxation. Let's look at the same problem using Gauss-Seidel method.

As seen in Figure 3.4, the choice of $\nu_1 = 2$, $\nu_2 = 0$ is significantly better than $\nu_1 = 1$, $\nu_2 = 1$ for high frequency errors. Although for low frequencies $\nu_1 = \nu_2 = 1$ is better, overall $\nu_1 = 2$, $\nu_2 = 0$ is a better choice for random errors compared to $\nu_1 = \nu_2 = 1$ due to its "homogenity". It is also "not that significantly" worse than $\nu_1 = 2$, $\nu_2 = 1$, so it is still a reasonable choice for the number of relaxations.

When we apply the two grid method on the same problem with red-black Gauss-Seidel relaxation, we see that the error "disappears" as soon as we relax it after the first time we went up to the finer grid.

In Figure 3.5, first graph on the top left shows us the initial error, which can be chosen at random. Second graph on the top right is the error we get after one initial relaxation. Third graph, on the bottom left shows the error after we go down and

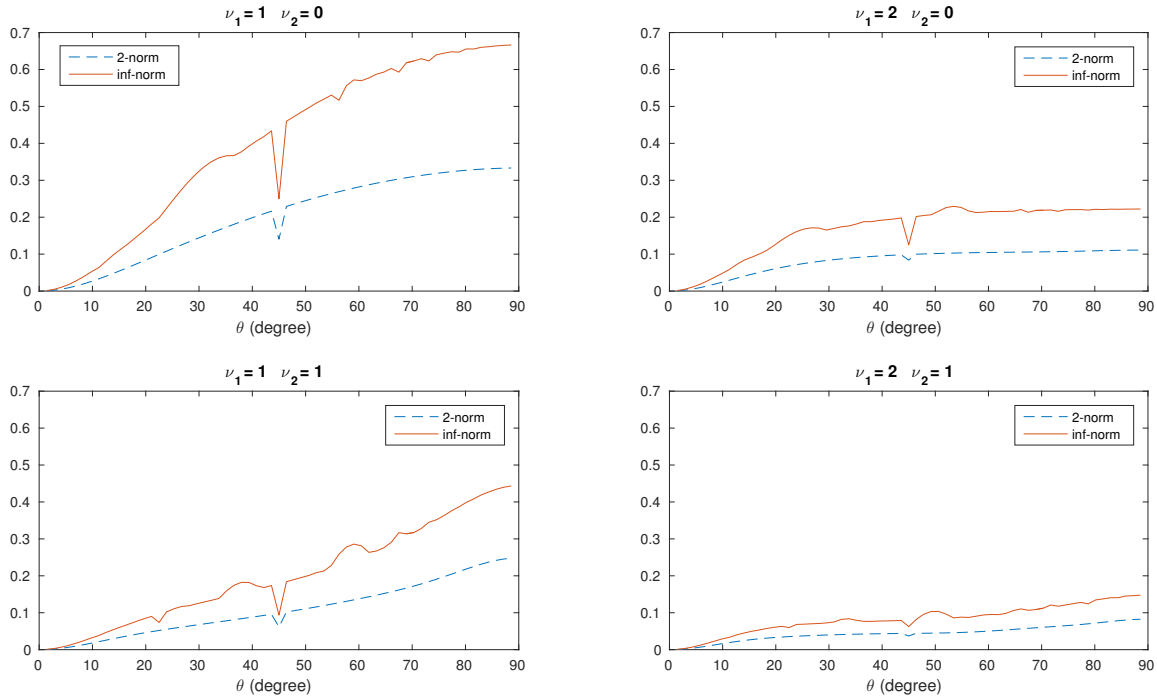


Figure 3.4: Two and infinity norms of errors after one cycle of two grid method with Gauss-Seidel relaxation. ν_1: number of initial relaxations, ν_2: number of final relaxations. Initial guess: $\sin(k\pi x)$, $k = \frac{2\theta}{\pi h}$

up. The last graph on the bottom right shows the error after one relaxation. When we check the infinity norm of the error in the 4^{th} plot in Figure 3.5, the number we get is of order 10^{-15}, even though it was around 0.3 for the previous step. To get a better understanding of why this happens, we check the Fourier modes of the error in Figure 3.6. Graphs in Figure 3.6 follow the same order as Figure 3.5. We see that if we have started with k^{th} node, after one initial relaxation we get both k^{th} and $(n-k)^{th}$ nodes, with different magnitudes. Going down and back up, we get the symmetry of the nodes! The first relaxation applied after we get the symmetry, makes the error disappear. Our first claim is:

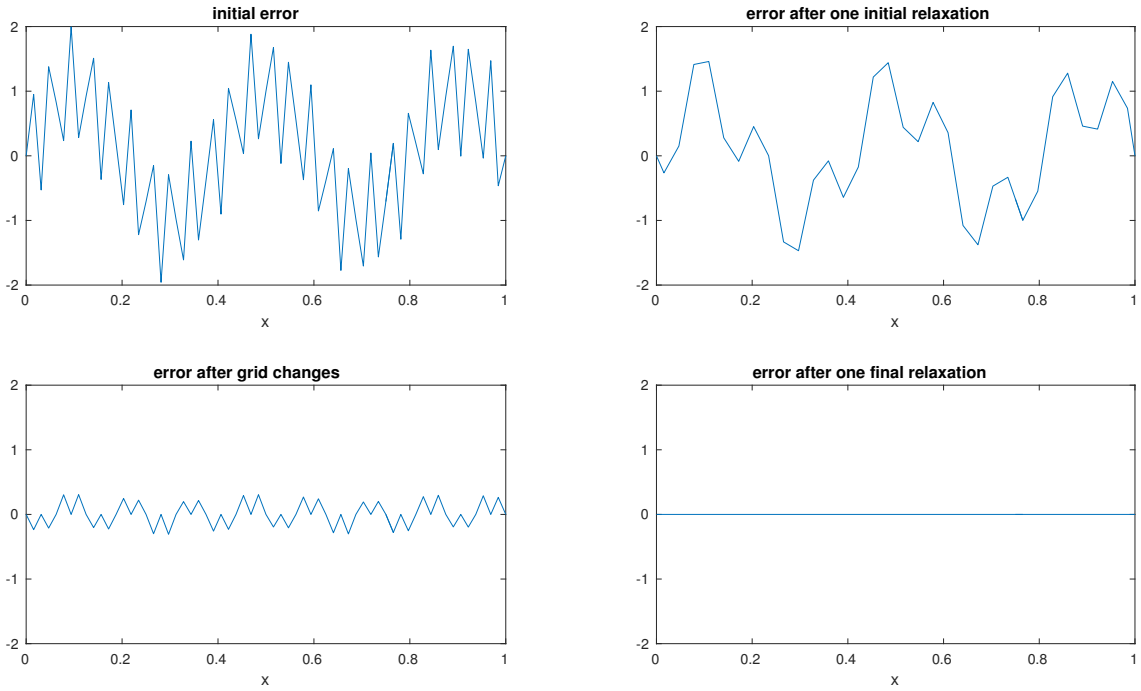


Figure 3.5: Errors during one two grid cycle with $\nu_1 = \nu_2 = 1$, relaxation method: red-black Gauss-Seidel. Top left: Initial error, chosen at random. Top right: After initial relaxation. Bottom left: After going down to coarser grid and coming back to the finer grid. Bottom right: After final relaxation

Corollary 3.3. *The red-black Gauss-Seidel relaxation zeros out any Fourier-mode-symmetric input, i.e. inputs of the form* $c(\boldsymbol{v}_k + \boldsymbol{v}_{n-k})$ *where* c *is a constant.*

Proof. To prove this, we'll look at the Fourier analysis. As we did in the previous chapter, we extend our error vector to its double length while keeping it antisymmetric about the origin and use the periodic boundary conditions. The $2n + 1$ vector

$$\mathbf{e} = (e_{-n}, e_{-n+1}, \cdots, e_0, e_1, \cdots, e_{n-1}, e_n)^T$$

represents the error, where $e_j = e(x_j)$ with $j = -n, -n+1, \cdots, n-1, n$. Because of the periodicity, $e_{-n} = e_n$, so we can consider the $2n$ vector, dropping one of the same

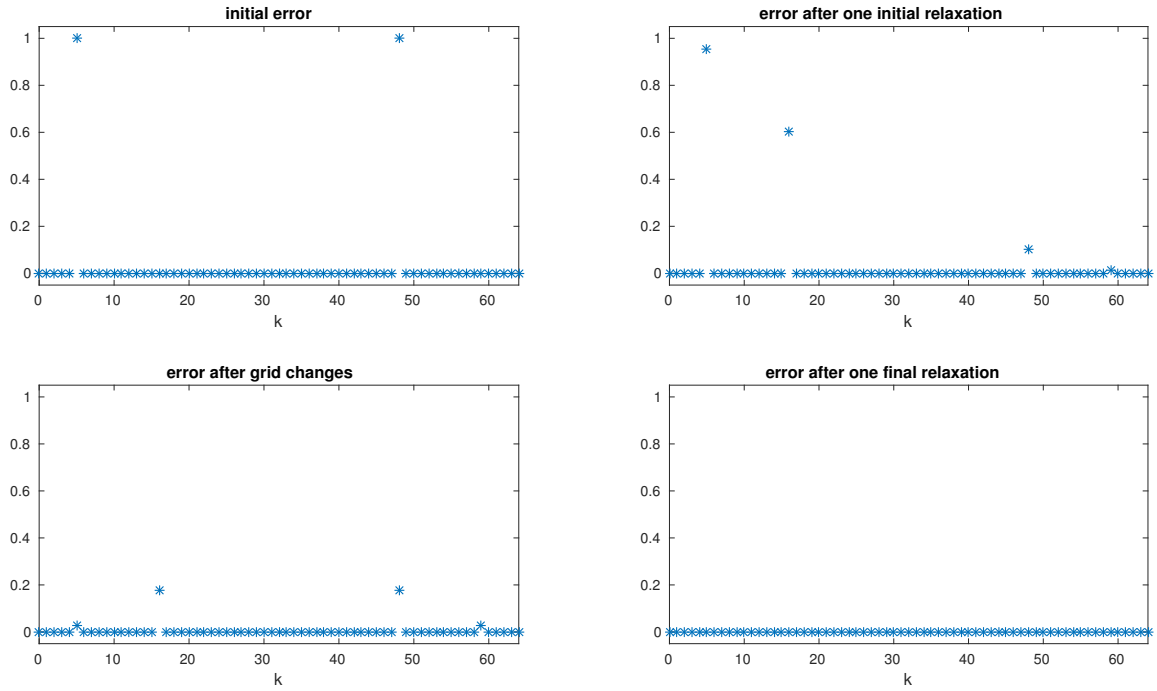


Figure 3.6: Fourier modes of the errors in Figure 3.5. Top left: Initial error. Top right: After initial relaxation. Bottom left: After going down to coarser grid and coming back to the finer grid. Bottom right: After final relaxation

terms. The Fourier representation of $\mathbf{e}^{(\nu)}$ is

$$e_j^{(\nu)} = \sum_{k=-n}^{n-1} c_k^{(\nu)} e^{\pi i k x_j}$$

Applying red-black Gauss-Seidel relaxation, the equations to update the components of the error will be

$$\begin{aligned} e_j^{(\nu+1)} &= \begin{cases} \frac{1}{2}(e_{j-1}^{(\nu)} + e_{j+1}^{(\nu)}) & \text{if } j \text{ is odd} \\ \frac{1}{2}(e_{j-1}^{(\nu+1)} + e_{j+1}^{(\nu+1)}) & \text{if } j \text{ is even} \end{cases} \\ &= \begin{cases} \frac{1}{2}(e_{j-1}^{(\nu)} + e_{j+1}^{(\nu)}) & \text{if } j \text{ is odd} \\ \frac{1}{4}(e_{j-2}^{(\nu)} + 2e_j^{(\nu)} + e_{j+2}^{(\nu)}) & \text{if } j \text{ is even} \end{cases} \end{aligned}$$

Replacing e_j in the above equations with its Fourier series representation, we get

$$\begin{aligned}
\sum_{k=-n}^{n-1} c_k^{(\nu+1)} e^{\pi i k x_j} &= \begin{cases} \sum_{k=-n}^{n-1} c_k^{(\nu)} \frac{1}{2} \left(e^{\pi i k x_{j-1}} + e^{\pi i k x_{j+1}} \right) & \text{if } j \text{ is odd} \\ \sum_{k=-n}^{n-1} c_k^{(\nu)} \frac{1}{4} \left(e^{\pi i k x_{j-2}} + 2e^{\pi i k x_j} + e^{\pi i k x_{j+2}} \right) & \text{if } j \text{ is even} \end{cases} \\
&= \begin{cases} \sum_{k=-n}^{n-1} c_k^{(\nu)} \frac{1}{2} \left(e^{\pi i k x_j} e^{-\pi i k h} + e^{\pi i k x_j} e^{\pi i k h} \right) & \text{if } j \text{ is odd} \\ \sum_{k=-n}^{n-1} c_k^{(\nu)} \frac{1}{4} \left(e^{\pi i k x_j} e^{-2\pi i k h} + 2e^{\pi i k x_j} + e^{\pi i k x_j} e^{2\pi i k h} \right) & \text{if } j \text{ is even} \end{cases} \\
&= \begin{cases} \sum_{k=-n}^{n-1} c_k^{(\nu)} e^{\pi i k x_j} \frac{1}{2} \left(e^{-\pi i k h} + e^{\pi i k h} \right) & \text{if } j \text{ is odd} \\ \sum_{k=-n}^{n-1} c_k^{(\nu)} e^{\pi i k x_j} \frac{1}{4} \left(e^{-\pi i k h} + e^{\pi i k h} \right)^2 & \text{if } j \text{ is even} \end{cases} \\
&= \begin{cases} \sum_{k=-n}^{n-1} c_k^{(\nu)} e^{\pi i k x_j} \cos(\pi k h) & \text{if } j \text{ is odd} \\ \sum_{k=-n}^{n-1} c_k^{(\nu)} e^{\pi i k x_j} cos^2(\pi k h) & \text{if } j \text{ is even} \end{cases}
\end{aligned}$$

where we used Euler's formula, $e^{i\theta} = \cos(\theta) + i\sin(\theta)$ and the fact that $x_{j+1} = x_j + h$.

Summing both sides over j and using the identities

$$\begin{aligned}
\sum_{j=-n}^{n-1} e^{\pi i K x_j} = \sum_{j=0}^{2n-1} e^{\pi i K x_j} = \sum_{j=0}^{2n-1} e^{\pi i K j h} = \sum_{j=0}^{2n-1} \left(e^{\pi i K h} \right)^j \\
= \begin{cases} 2n & \text{if } K = 0 \\ \frac{\left(e^{\pi i K h} \right)^{2n} - 1}{e^{\pi i K h} - 1} & \text{otherwise} \end{cases} = \begin{cases} 2n & \text{if } K = 0 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
\sum_{\substack{j=-n \\ j \text{ even}}}^{n-1} e^{\pi i K x_j} = \sum_{\substack{j=-n \\ j \text{ even}}}^{2n-1} e^{\pi i K x_j} = \sum_{j=0}^{n-1} e^{\pi i K 2 j h} = \sum_{j=0}^{n-1} \left(e^{2\pi i K h} \right)^j \\
= \begin{cases} n & \text{if } K = 0 \text{ or } K = -n \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

$$\sum_{\substack{j=-n \\ j \text{ odd}}}^{n-1} e^{\pi i K x_j} = \sum_{\substack{j=-n \\ j \text{ odd}}}^{2n-1} e^{\pi i K x_j} = \sum_{j=0}^{n-1} e^{\pi i K(2j+1)h} = e^{\pi i K h} \sum_{j=0}^{n-1} \left(e^{2\pi i K h}\right)^j$$

$$= \begin{cases} n & \text{if } K = 0 \\ -n & \text{if } K = -n \\ 0 & \text{otherwise} \end{cases}$$

we get

$$\sum_{k=-n}^{n-1} c_k^{(\nu+1)} \sum_{j=-n}^{n-1} e^{\pi i(k-l)x_j}$$

$$= \sum_{k=-n}^{n-1} c_k^{(\nu)} \cos(\pi k h) \sum_{\substack{j=-n \\ j \text{ odd}}}^{n-1} e^{\pi i(k-l)x_j} + \sum_{k=-n}^{n-1} c_k^{(\nu)} \cos^2(\pi k h) \sum_{\substack{j=-n \\ j \text{ even}}}^{n-1} e^{\pi i(k-l)x_j}$$

$$= c_l^{(\nu)} \cos(\pi l h) n - c_{l-n}^{(\nu)} \cos(\pi(l-n)h) n + c_l^{(\nu)} \cos^2(\pi l h) n + c_{l-n}^{(\nu)} \cos^2(\pi(l-n)h) n$$

where

$$\sum_{k=-n}^{n-1} c_k^{(\nu+1)} \sum_{j=-n}^{n-1} e^{\pi i(k-l)x_j} = 2n c_l^{(\nu+1)}$$

Therefore,

$$2c_l^{(\nu+1)} = c_l^{(\nu)} \cos(\pi l h) - c_{l-n}^{(\nu)} \cos(\pi(l-n)h) + c_l^{(\nu)} cos^2(\pi l h) + c_{l-n}^{(\nu)} \cos^2(\pi(l-n)h)$$

$$c_l^{(\nu+1)} = \frac{1}{2}\left[(\cos(\pi l h) + \cos^2(\pi l h)) c_l^{(\nu)} + (-\cos(\pi(l-n)h) + \cos^2(\pi(l-n)h)) c_{l-n}^{(\nu)}\right]$$

$$= \frac{1}{2}\left[(\cos(\pi l h) + \cos^2(\pi l h)) c_l^{(\nu)} - (-\cos(\pi(n-l)h) + \cos^2(\pi(n-l)h)) c_{n-l}^{(\nu)}\right]$$

since cosine is an even function and $c_k = -c_{-k}$ due to our antisymmetry and boundary conditions.

If $e_j^{(\nu)} = \sin(\pi k x_j) = \frac{-i}{2}\left(e^{\pi i k x_j} - e^{-\pi i k x_j}\right)$, i.e. $c_l^{(\nu)} = -\frac{i}{2}\delta_{l,k} + \frac{i}{2}\delta_{l,-k}$, then

$$c_l^{\nu+1} = \tfrac{1}{2}(\cos(\pi l h) + \cos^2(\pi l h))(-\tfrac{i}{2}\delta_{l,k} + \tfrac{i}{2}\delta_{l,-k})$$

$$- \tfrac{1}{2}(-\cos(\pi(n-l)h) + \cos^2(\pi(n-l)h))(-\tfrac{i}{2}\delta_{n-l,k} + \tfrac{i}{2}\delta_{n-l,-k})$$

$$
\begin{aligned}
e_j^{(\nu+1)} &= \tfrac{-i}{4}(\cos(\pi kh)+\cos^2(\pi kh))e^{i\pi kx_j} + \tfrac{i}{4}(\cos(\pi(-k)h)+\cos^2(\pi(-k)h))e^{-i\pi kx_j} \\
&+\tfrac{i}{4}(-\cos(\pi kh)+\cos^2(\pi kh))e^{i\pi(n-k)x_j} - \tfrac{i}{4}(-\cos(\pi(-k)h)+\cos^2(\pi(-k)h))e^{i\pi(n+k)x_j} \\
&= \tfrac{-i}{4}(1+\cos(\pi kh))\cos(\pi kh)(e^{i\pi kx_j}-e^{-i\pi kx_j}) \\
&\qquad + \tfrac{i}{4}(-1+\cos(\pi kh))\cos(\pi kh)e^{i\pi nx_j}(e^{-i\pi kx_j}-e^{i\pi kx_j}) \\
&= [\tfrac{-i}{4}(1+\cos(\pi kh))\cos(\pi kh) - \tfrac{i}{4}(-1+\cos(\pi kh))\cos(\pi kh)e^{i\pi njh}](2i\sin(\pi kx_j)) \\
&= \tfrac{1}{2}[(1+\cos(\pi kh))+(-1+\cos(\pi kh))(-1)^j]\cos(\pi kh)\sin(\pi kx_j)
\end{aligned}
$$

Similarly, for $k' = n-k$, using $\cos((n-k)\pi x) = -\cos(k\pi x)$ and $\sin(a-b) = \sin(a)\cos(b) - \sin(b)\cos(a)$, we get

$$
\begin{aligned}
\bar{e}_j^{(\nu+1)} &= \tfrac{1}{2}[1+\cos(\pi(n-k)h) - (-1)^j \\
&\qquad + (-1)^j\cos(\pi(n-k)h)]\cos(\pi(n-k)h)\sin(\pi(n-k)x_j) \\
&= \tfrac{-1}{2}\left[1-\cos(\pi kh)-(-1)^j-(-1)^j\cos(\pi kh)\right]\cos(\pi kh)(-\sin(\pi kx_j)\cos(\pi nhj)) \\
&= \tfrac{-1}{2}\left[1-\cos(\pi kh)-(-1)^j-(-1)^j\cos(\pi kh)\right]\cos(\pi kh)(-\sin(\pi kx_j)(-1)^j) \\
&= \tfrac{1}{2}\left[(-1)^j-(-1)^j\cos(\pi kh)-1-\cos(\pi kh)\right]\cos(\pi kh)\sin(\pi kx_j)
\end{aligned}
$$

Summing these two "symmetric" errors, we get

$$
\begin{aligned}
e_j^{(\nu+1)} + \bar{e}_j^{(\nu+1)} &= \tfrac{1}{2}(1+\cos(\pi kh)-(-1)^j+(-1)^j\cos(\pi kh))\cos(\pi kh)\sin(\pi kx_j) \\
&\quad + \tfrac{1}{2}\left[(-1)^j-(-1)^j\cos(\pi kh)-1-\cos(\pi kh)\right]\cos(\pi kh)\sin(\pi kx_j) \\
&= 0
\end{aligned}
$$

□

Now we are ready to prove what we've just observed:

Corollary 3.4. *If we apply two grid method to the equation $-u_{xx} = f$ with 0s on the boundary, with the choice of red-black Gauss-Seidel as the relaxation method, the first relaxation after going back to the finer grid eliminates the error.*

Proof. As argued above, with 3.3, we know that the matrix G, which is responsible from going down and up, gives a Fourier mode symmetric output, whatever the input is; since any input can be written as a linear combination as $\mathbf{v}_k$'s.

$$G\mathbf{v}_k = \begin{cases} \sin^2(\theta_k)(\mathbf{v}_k + \mathbf{v}_{n-k}) & 1 \leq k < \frac{n}{2} \\ \cos^2(\theta_{n-k})(\mathbf{v}_k + \mathbf{v}_{n-k}) & \frac{n}{2} \leq k \leq n-1 \end{cases}$$

By Corollary 3.3, the result follows. □

We need to keep in mind that the above results are for a specific problem. Unfortunately, for a different problem, we might not get a Fourier mode symmetric result when we go back up to the finer grid. For example, if we had $-u_{xx} + u = f$, the corresponding equation would be $(A + I)\mathbf{u} = \mathbf{f}$. Then G would be equal to $I + I_{2h}^h(A^{2h} + I)^{-1}I_h^{2h}(A^h + I)$, and we wouldn't get the same expression for $G\mathbf{v}$ as in Equation 3.3. Instead, after following the same steps as in Equation 3.3, we would get

$$G\mathbf{v}_k^h = \begin{cases} \mathbf{v}_k^h - \frac{(4\sin^2(\theta_k)+h^2)\cos^2(\theta_k)}{\sin^2(2\theta_k)+h^2}[\cos^2(\theta_k)\mathbf{v}_k^h - \sin^2(\theta_k)\mathbf{v}_{k'}^h] & 1 \leq k < \frac{n}{2} \\ \mathbf{v}_k^h - \frac{(4\sin^2(\theta_k)+h^2)\sin^2(\theta_{k'})}{\sin^2(2\theta_{k'})+h^2}[\cos^2(\theta_{k'})\mathbf{v}_{k'}^h - \sin^2(\theta_{k'})\mathbf{v}_k^h] & \frac{n}{2} \leq k \leq n-1 \end{cases} \tag{3.4}$$

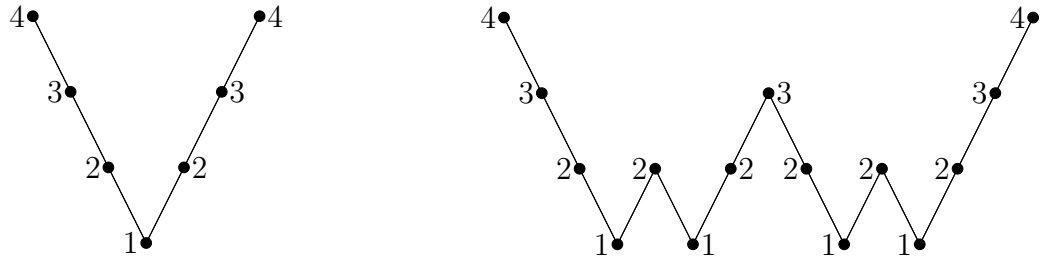
which is clearly not Fourier mode symmetric.

3.2 V and W cycles

In the previous section, we went down only one step. If our initial grid is "really" fine, the coarser grid we get might not be coarse enough: direct calculation of the error might still be costly, or the low frequency errors stay as low frequency errors, so

relaxations cannot help. The solution is to just keep going down until we reach some "suitable" grid- it might even consist of 3 grid points! (Of course, with 3 grid points, calculating the error with a direct method becomes trivial.)

Multigrid V-cycle is based on this idea, we start from the finest grid, go down to the coarsest grid we want, and go straightly back up to the finest grid. A slight variation of multigrid V-cycle is the *multigrid W-cycle*, which spends more time on coarser grids to reduce the low-frequency errors better.



(a) V-cycle (b) W-cycle

Figure 3.7: V and W-cycles

In many references, for instance in [11], V- and W-cycles are given with a recursive algorithm, which starts from the finest grid, as given in Algorithm 1. Increasing the number of recursions, i.e. calling the function multiple times, leads to spending more time on coarser grids. Algorithm 1 calls itself once for a V-cycle and twice for a W-cycle. The main idea is calculating large-scale errors on coarser grids, and correcting/finetuning them on the finer grids.

Algorithm 1 V/W cycle

function V/W($\mathbf{u}^h, \mathbf{f}^h$)
  if not on the coarsest grid **then**
    $\mathbf{u}^h \leftarrow \nu_1 \times$ relaxation($A^h\mathbf{u}^h = \mathbf{f}^h$) with initial iterate $\mathbf{u}^h$
    $\mathbf{f}^{2h} \leftarrow$ Projection/Injection($\mathbf{f}^h - A^h\mathbf{u}^h$)
    $\mathbf{u}^{2h} \leftarrow \mathbf{0}$
    $\mathbf{u}^{2h} \leftarrow V/W(\mathbf{u}^{2h}, \mathbf{f}^{2h})$
    if W cycle **then**
      $\mathbf{u}^{2h} \leftarrow V/W(\mathbf{u}^{2h}, \mathbf{f}^{2h})$
    end if
    $\mathbf{u}^h \leftarrow \mathbf{u}^h +$ Interpolation($\mathbf{u}^{2h}$)
    $\mathbf{u}^h \leftarrow \nu_2 \times$ relaxation($A^h\mathbf{u}^h = \mathbf{f}^h$) with initial iterate $\mathbf{u}^h$
  else
    Solve $A^h\mathbf{u}^h = \mathbf{f}^h$ for $\mathbf{u}^h$
  end if
  return $\mathbf{u}^h$
end function

These paths can easily be generated by "connecting" different V-cycles, which does not have to have the same entry and exit level. An algorithm for a V-cycle starting from level "sl" and ending at level "el" can be the following:

At this point, let us have a quick look at the costs of V and W-cycles. If the coarsest grid of the V-cycle is not "coarse enough", solving $A\mathbf{e} = \mathbf{f}$ directly might be costly. (For example, for a $n \times n$ matrix A, the computational cost for solving $A\mathbf{e} = \mathbf{f}$ with Gaussian elimination is $\frac{2}{3}n^3$.) So let us assume that the coarsest grid where we solve $A\mathbf{e} = \mathbf{f}$ directly has so many grid points that the cost of solving for $\mathbf{e}$ is a constant number and can be ignored. We also want to avoid matrix multiplications, instead we do elementwise calculations for relaxations, projections and interpolations if possible. In that case, a relaxation costs $C_1(n-1)$, projection costs $C_2(\frac{n}{2}-1)$ and interpolation costs $C_3(2n-1)$, where $n+1$ is the number of the grid points on the current level.

Algorithm 2 V cycle

procedure GENERAL V($sl, el, \mathbf{u}^h$)
  $\mathbf{u}^h \leftarrow \nu_1 \times$ Relaxation($A^h\mathbf{u}^h = \mathbf{f}^h$) with initial iterate $\mathbf{u}^h$
  $\mathbf{f}^{2h} \leftarrow$ Projection/Injection($\mathbf{f}^h - A^h\mathbf{u}^h$)
  $\alpha \leftarrow 1$;
  for $level := sl - 1$ **down to** coarsest level $+ 1$ **do**
    $\alpha \leftarrow 2 \times \alpha$
    $\mathbf{u}^{\alpha h} \leftarrow \nu_1 \times$ Relaxation($A^{\alpha h}\mathbf{u}^{\alpha h} = \mathbf{f}^{\alpha h}$) with initial iterate $\mathbf{0}$
    $\mathbf{f}^{2\alpha h} \leftarrow$ Projection/Injection($\mathbf{f}^{\alpha h} - A^{\alpha h}\mathbf{u}^{\alpha h}$)
  end for
  Solve $A^{2\alpha h}\mathbf{u}^{2\alpha h} = \mathbf{f}^{2\alpha h}$ for $\mathbf{u}^{2\alpha h}$
  for $level :=$ coarsest level $+ 1$ **to** el **do**
    $\mathbf{u}^{\alpha h} \leftarrow \mathbf{u}^{\alpha h} +$ Interpolation($\mathbf{u}^{2\alpha h}$)
    $\mathbf{u}^{\alpha h} \leftarrow \nu_2 \times$ Relaxation($A^{\alpha h}\mathbf{u}^{\alpha h} = \mathbf{f}^{\alpha h}$) with initial iterate $\mathbf{u}^{\alpha h}$
    $\alpha \leftarrow \frac{\alpha}{2}$
  end for
  return $\mathbf{u}^{2\alpha h}$
end procedure

In a V-cycle, at each level we do $\nu_1 + \nu_2$ relaxations. If we have m grids, we will go down and up $m - 1$ times. Adding all these together , for a V-cycle the cost is approximately ($n - 1 \approx n$ for big n, so -1's dropped)

$$(\nu_1 + \nu_2)C_1(n + \tfrac{n}{2} + \tfrac{n}{4} + \cdots + \tfrac{n}{2^{m-2}}) + C_2(\tfrac{n}{2} + \tfrac{n}{4} + \cdots + \tfrac{n}{2^{m-1}})$$
$$+C_3(\tfrac{n}{2^{m-2}} + \tfrac{n}{2^{m-2}} + \cdots + \tfrac{n}{2} + n) \quad < \quad 2(\nu_1 + \nu_2)C_1 n + C_2 n + 2C_3 n$$

so the time complexity is *O(n)*. Notice that it doesn't depend on m, the number of grids; but it depends on the number of grid points on the finest grid and the total number of relaxations done at each level.

For a W-cycle we expect a higher cost, since we do more than what we do in a V-cycle, if we start from the same grid. This time, at each level, we do 2 times more relaxations than the previous grid. Similarly, the number of projections and interpolations also depend on the level. For a W-cycle, we approximately get the cost

as

$$(\nu_1+\nu_2)C_1(n+2\tfrac{n}{2}+4\tfrac{n}{4}+\cdots+2^{m-2}\tfrac{n}{2^{m-2}})+C_2(\tfrac{n}{2}+2\tfrac{n}{4}+\cdots+2^{m-2}\tfrac{n}{2^{m-1}})$$

$$+C_3(2^{m-2}\tfrac{n}{2^{m-2}}+2^{m-3}\tfrac{n}{2^{m-3}}+\cdots+2\tfrac{n}{2}+n)=(m-1)\left[(\nu_1+\nu_2)C_1n+\tfrac{1}{2}C_2n+C_3n\right]$$

which has the time complexity *O(mn)*. For a W-cycle, the cost depends on the number of grids we have, since the coarser we go, the longer time we spend on that level.

We talked a lot about V and W-cycles. Now let's check the results when we apply V- and W-cycles to our example problem, $-u_{xx}=0$ on $0\le x\le 1$ with homogeneous Dirichlet boundary conditions, using various number of relaxations, ν_1 and ν_2.

		damped Jacobi				Gauss-Seidel				red-black GS			
		V-cycle		W-cycle		V-cycle		W-cycle		V-cycle		W-cycle	
ν_1	ν_2	pro	inj	pro	inj	pro	inj	pro	inj	pro	inj	pro	inj
1	0	23	-	23	186	23	-	23	113	19	-	2	-
0	1	25	-	23	186	24	211	23	120	1	-	1	-
2	0	17	57	12	29	14	27	13	29	14	-	2	-
1	1	15	56	12	30	14	33	13	30	1	-	1	-
0	2	18	67	13	31	15	30	13	31	1	-	1	-
3	0	13	20	10	17	10	18	10	19	11	-	2	-
2	1	12	20	10	19	10	18	10	19	1	-	1	-
1	2	12	21	10	19	10	20	10	20	1	-	1	-
0	3	15	25	10	21	11	20	11	21	1	-	1	-
4	0	12	15	9	14	9	14	7	14	10	-	2	-
3	1	11	14	9	14	8	14	7	14	1	-	1	-
2	2	11	15	9	15	8	14	7	15	1	-	1	-
1	3	11	16	9	15	9	17	7	15	1	-	1	-
0	4	14	19	10	16	9	16	8	16	1	-	1	-

Table 3.1: Number of cycles needed to reduce the maximum error by 10^{-11}, i.e. the ratio of the initial error and the final error is less than 10^{-11}. Same initial random guess is used for all configurations. Started with $n=128$, went down the coarsest possible grid, i.e. the grid with 3 points.

There are some observations we can make from Table 3.1:

When $\nu_1 + \nu_2$ is the same, there is no significant change between different cases, so we can still say the sum of the number of relaxations determines the behaviour. While with projection, a balanced combination is a better choice, with injection, having more initial relaxations gives a better result. Here we can remember the discussion on injection: Since injection shifts high frequency errors to low frequency errors without any damping in magnitude, the high frequency errors should be damped before they are transferred, if possible.

Although increasing $\nu_1 + \nu_2$ improves the results, the improvement is not as big as the one we have from $\nu_1 + \nu_2 = 1$ to 2, for the rest of the table. This also supports the general convention of choosing $\nu_1 + \nu_2 = 2$.

As expected, projection is better than injection to transfer the data to the coarser grids. Injection might even cause the multigrid method solution to diverge, which is the case when red-black Gauss-Seidel relaxation method is used or when we have a V-cycle with not "enough" number of relaxations.

Red-black Gauss-Seidel method is either does a great job with the same idea we proved in two grid method (with projection), or does not converge (with injection). We need to remember that the "1"s on the table are problem specific. In general, red-black Gauss-Seidel with projection is the best choice, but it might take some iterations to reduce the error for an arbitrary problem. Corollary 3.4 holds for $-u_{xx} = f$, but not for any problem, as discussed at the end of Section 1. In the next section we'll see what happens if we have a different problem.

An interesting question might be, if we need to go all the way down to the coarsest grid. Time complexity of V-cycles does not depend on the number of grids, however time complexity of W-cycles does. To reduce the time complexity, one can consider

stopping at a certain "level". For example, when we use only three grids instead of seven where $n = 128$, the results we get are in Table 3.2.

		damped Jacobi				Gauss-Seidel				red-black GS			
		V-cycle		W-cycle		V-cycle		W-cycle		V-cycle		W-cycle	
ν_1	ν_2	pro	inj	pro	inj	pro	inj	pro	inj	pro	inj	pro	inj
1	0	23	98	23	95	23	88	23	97	12	-	2	-
0	1	23	98	23	104	24	90	23	99	1	-	1	-
2	0	14	33	12	29	14	27	13	30	10	-	2	-
1	1	14	34	12	30	14	28	13	31	1	-	1	-
0	2	15	36	13	32	14	29	13	32	1	-	1	-
3	0	11	18	10	17	10	18	10	19	9	-	2	-
2	1	11	18	10	19	10	19	10	19	1	-	1	-
1	2	11	20	10	19	10	19	10	20	1	-	1	-
0	3	12	20	10	19	11	20	11	21	1	-	1	-
4	0	10	12	9	14	8	14	7	14	8	-	2	-
3	1	10	14	9	14	8	14	7	14	1	-	1	-
2	2	10	14	9	15	8	14	7	15	1	-	1	-
1	3	10	14	9	15	8	15	7	15	1	-	1	-
0	4	11	16	10	16	9	15	8	16	1	-	1	-

Table 3.2: Number of cycles needed to reduce the maximum error by 10^{-11}, i.e. the ratio of the initial error and the final error is less than 10^{-11}. Same initial random guess is used for all configurations. Started with $n = 128$, used only 3 grids, i.e. the coarsest grid had 33 points.

Comparing Table 3.1 and Table 3.2, we see that the results are very similar, for some cases even better on Table 3.2. This is another expected fact, since solving directly on a "closer" grid to the finest grid improves the solution. The drawback of using fewer grids is the cost of direct solution on the coarsest grid. This can be avoided by choosing a "coarsest grid" to stop, where it is not very costly to calculate the solution directly and we have a reasonable number of grid points.

3.3 Full Multigrid cycle

An important improvement can be made by starting with an educated guess, instead of a random one on the finest grid. On the coarsest grid, we solve $A\mathbf{e} = \mathbf{f}$ easily. Why don't we try to solve $A\mathbf{u} = \mathbf{f}$ instead, and use $\mathbf{u}$ as our initial guess? We can keep interpolating $\mathbf{u}$ until we reach to the finest grid. To get a better estimate, as we reach some level, we can do a V- or W-cycle starting from that level down to the coarsest grid, using the "interpolation" as a initial guess. That will reduce the error at every level. Once we reach the finest grid, we can do one more V- or W- cycle and *full multigrid cycle* is complete!

A full multigrid cycle is called *Full Multigrid V-cycle (FMV)* or *Full Multigrid W-cycle (FMW)*, depending on the type of cycle used at each level as we go up. Figure 3.8 sketches the path for each in case of 3 grids.

(a) FMV cycle (b) FMW cycle

Figure 3.8: FMV and FMW cycles

The algorithm for FMV or FMW is straightforward, once we know the algorithm of a V- or W-cycle.

The computational cost of a FMV and FMW can be quickly calculated using the costs of V and W-cycles. For a V cycle, the cost was $O(n)$, where $n + 1$ was the

Algorithm 3 FMV/FMW

procedure FMV/FMW
  Knowing A and $\mathbf{f}$ on the coarsest grid, solve $A\mathbf{u}_1 = \mathbf{f}$ for $\mathbf{u}_1$
  $\mathbf{u}_2 \leftarrow$ Interpolate $\mathbf{u}_1$
  $l \leftarrow 2$
  while number of grid points at level $l \leq$ number of grid points on the finest level **do**
    do a V/W-cycle where $endgrid = startgrid = l$ with initial guess $\mathbf{u}_l$
    if not on the finest grid **then**
      $\mathbf{u}_{l+1} \leftarrow$ Interpolate $\mathbf{u}_l$
    end if
    $l \leftarrow l + 1$
  end while
  return u on the finest grid
end procedure

number of grid points on the finest grid. In a FMV cycle, for each grid a V-cycle is done, and we need one interpolation between each consecutive levels. So the total cost will be

$$\bar{C}(n + \tfrac{n}{2} + \cdots + \tfrac{n}{2^{m-2}}) + \bar{D}n < 2\bar{C}n + \bar{D}n$$

where $\bar{C}$ and $\bar{D}$ are constants. Hence the cost of a FMV cycle is still *O(n)*!

For a W-cycle, the cost was *O(mn)*, where m was the number of grids. Doing a W-cycle at each level and one interpolation between each consecutive levels, the total cost of a FMW cycle will be

$$\begin{aligned}
&\tilde{C}(mn + (m-1)\tfrac{n}{2} + (m-2)\tfrac{n}{4} + \cdots + 2\tfrac{n}{2^{m-2}}) + \bar{D}n \\
=&\tilde{C}n(m + \tfrac{m-1}{2} + \tfrac{m-2}{4} + \tfrac{m-3}{8} + \cdots + \tfrac{2}{2^{m-2}}) + \bar{D}n \\
<&\tilde{C}n\left[m(1 + \tfrac{1}{2} + \tfrac{2}{4} + \tfrac{3}{8} + \cdots) - (\tfrac{1}{2} + \tfrac{2}{4} + \tfrac{3}{8} + \cdots)\right] + \bar{D}n \\
<&\tilde{C}n\left[m(\tfrac{3}{2} + 1 + \tfrac{1}{2} + \tfrac{1}{4} + \cdots) - \tfrac{1}{2}(\tfrac{3}{2} + 1 + \tfrac{1}{2} + \tfrac{1}{4} + \cdots)\right] + \bar{D}n \\
=&\tilde{C}n\left[m(\tfrac{3}{2} + 2) - (\tfrac{3}{4} + 1)\right] + \bar{D}n
\end{aligned}$$

where $\tilde{C}$ is a constant. Similar to FMV, the cost for FMW is still *O(mn)*.

Now it is time to check the results when we apply FMV- and FMW-cycles, using various number of relaxations, ν_1 and ν_2. This time we will look at the equation $-u_{xx} + u = f$. In the tables below, the particular choice is $f(x) = 3(x - e^{2x})$, $u(0) = 1$ and $u(1) = e^2 + 3$, so the exact solution of the equation is $u(x) = e^{2x} + 3x$.

		damped Jacobi				Gauss-Seidel				red-black GS			
		FMV		FMW		FMV		FMW		FMV		FMW	
ν_1	ν_2	pro	inj	pro	inj	pro	inj	pro	inj	pro	inj	pro	inj
1	0	18	-	17	113	17	-	17	98	13	-	2	-
0	1	19	-	17	115	17	172	17	98	5	-	3	-
2	0	13	40	9	23	9	21	9	23	8	-	2	-
1	1	11	41	9	24	9	26	9	25	2	-	2	-
0	2	14	44	10	24	10	23	9	25	4	-	3	-
3	0	10	16	7	15	7	14	7	15	7	-	2	-
2	1	9	16	7	15	7	15	7	15	2	-	2	-
1	2	8	16	7	15	7	16	7	15	2	-	2	-
0	3	11	17	8	15	9	16	8	15	4	-	3	-
4	0	9	12	7	11	6	11	5	11	6	-	2	-
3	1	7	11	6	11	6	12	5	11	2	-	2	-
2	2	8	12	6	12	6	12	5	12	2	-	2	-
1	3	8	12	6	12	6	13	5	12	2	-	2	-
0	4	10	14	7	12	8	12	6	12	4	-	3	-

Table 3.3: Number of cycles needed to reduce the maximum error by 10^{-11}, i.e. the ratio of the initial error and the final error is less than 10^{-11}. Started with $n = 128$, went down the coarsest possible grid, i.e. the grid with 3 points.

As we see on Table 3.3, the "behaviour" is not very different than Table 3.1. Note that we are working on a different problem than before, where G, defined in Equation (3.2), does not return a Fourier symmetric result as discussed at the end of the first section, so Corollary 3.4 is not applicable. This is exactly what we see on Table 3.3 under red-black Gauss-Seidel.

Applying the full multigrid method with 3 grids, we get the same conclusions from the Table 3.4 as in V- and W-cycles case in Section 2.

		damped Jacobi				Gauss-Seidel				red-black GS			
		FMV		FMW		FMV		FMW		FMV		FMW	
ν_1	ν_2	pro	inj	pro	inj	pro	inj	pro	inj	pro	inj	pro	inj
1	0	17	72	17	74	17	65	17	71	5	-	2	-
0	1	17	73	17	75	18	68	17	74	3	-	3	-
2	0	10	24	9	23	9	20	9	23	4	-	2	-
1	1	10	26	9	24	9	22	9	25	2	-	2	-
0	2	11	26	9	24	10	22	9	24	3	-	3	-
3	0	8	14	7	15	7	14	7	15	4	-	2	-
2	1	8	14	7	15	7	15	7	15	2	-	2	-
1	2	7	16	7	15	7	15	7	15	2	-	2	-
0	3	9	15	8	16	8	15	8	16	3	-	3	-
4	0	7	10	7	11	6	10	5	11	4	-	2	-
3	1	7	10	6	11	6	10	5	11	2	-	2	-
2	2	7	12	6	12	6	10	5	12	2	-	2	-
1	3	7	12	6	12	6	12	5	12	2	-	2	-
0	4	8	12	7	12	6	12	6	12	3	-	3	-

Table 3.4: Number of cycles needed to reduce the maximum error by 10^{-11}, i.e. the ratio of the initial error and the final error is less than 10^{-11}. Started with $n = 128$, used only 3 grids, i.e. the coarsest grid had 33 points.

So far we must have been convinced that projection is a superior method compared to injection. In the remaining discussion, we will only consider cycles with projection, unless it is necessary or interesting to check the results with injection.

Another simple but interesting question is whether full multigrid provides a big improvement over regular V and W cycles. Surely it does if we start with a "bad" initial guess in a V or W cycle. What if we had a relatively good guess, such as given the boundary conditions, we start the cycle with the linear interpolation of the boundary values? Table 3.5 summarizes the results, when we have the same problem

as above, when we use only 3 grids as above, and when we start V and W cycles with the linear interpolation of boundary conditions.

		damped Jacobi		Gauss-Seidel		red-black GS	
		V-cycle	W-cycle	V-cycle	W-cycle	V-cycle	W-cycle
ν_1	ν_2	pro	pro	pro	pro	pro	pro
1	0	18	18	18	18	10	3
0	1	17	17	18	17	3	3
2	0	11	10	11	10	8	3
1	1	11	10	10	10	3	2
0	2	11	10	10	9	3	3
3	0	9	8	8	8	7	3
2	1	9	7	8	8	3	2
1	2	9	7	8	8	3	2
0	3	9	8	8	8	3	3
4	0	8	7	7	6	7	3
3	1	8	7	7	6	3	2
2	2	8	7	6	5	3	2
1	3	8	7	6	5	3	2
0	4	8	7	7	6	3	3

Table 3.5: Number of cycles needed to reduce the maximum error by 10^{-11}, i.e. the ratio of the initial error and the final error is less than 10^{-11}. Linear initial guess. Started with $n = 128$, used only 3 grids, i.e. the coarsest grid had 33 points.

Comparing Table 3.5 with Table 3.4, we see that regular V and W cycles do a slightly worse job compared to full multigrid cycles, even if we start with an educated guess. But the difference is very small, which makes V and W cycles with good initial iterates an alternative to full multigrid cycles.

Last thing we'll have a look at is the following question: How good is the calculated initial iterate in full multigrid method? After reaching to the finest grid, there is nothing different in the full multigrid method from regular V or W cycles. But we've

just seen that full multigrid is a "better" option. So this difference should be due to the initial guess.

		damped Jacobi				Gauss-Seidel				red-black GS			
		FMV		FMW		FMV		FMW		FMV		FMW	
ν_1	ν_2	b	a	b	a	b	a	b	a	b	a	b	a
1	0	3.3	1.1	1.3	0.35	2.1	0.66	2.1	0.65	0.90	0.54	0.87	$7e^{-3}$
0	1	6.8	2.5	2.7	0.64	4.1	1.4	2.5	0.64	0.85	0.02	0.85	0.02
2	0	2.0	0.44	1.1	0.17	1.1	0.15	0.99	0.12	0.86	$9e^{-3}$	0.86	$4e^{-5}$
1	1	0.9	0.16	0.89	0.19	0.81	0.23	0.81	0.20	0.86	$1e^{-5}$	0.86	$3e^{-8}$
0	2	5.0	1.5	2.4	0.48	3.7	0.99	2.4	0.46	0.85	0.02	0.85	0.02
3	0	1.2	0.14	0.96	0.05	0.89	0.04	0.89	0.04	0.86	$4e^{-4}$	0.86	$3e^{-5}$
2	1	1.2	0.15	1	0.07	1	0.09	0.97	0.06	0.86	$9e^{-6}$	0.86	$3e^{-8}$
1	2	0.84	0.13	0.81	0.15	0.8	0.16	0.81	0.15	0.86	$1e^{-4}$	0.86	$3e^{-8}$
0	3	3.1	0.76	2.3	0.42	2.5	0.53	2.3	0.42	0.85	0.02	0.85	0.02
4	0	1.1	0.1	0.94	0.04	0.86	0.03	0.86	0.02	0.86	$3e^{-3}$	0.86	$3e^{-5}$
3	1	0.95	0.04	0.89	0.02	0.86	0.02	0.86	0.02	0.86	$6e^{-6}$	0.86	$3e^{-8}$
2	2	1.1	0.11	1	0.05	0.99	0.07	0.94	0.05	0.86	$6e^{-6}$	0.86	$3e^{-8}$
1	3	0.82	0.13	0.8	0.14	0.8	0.15	0.8	0.14	0.86	$6e^{-6}$	0.86	$3e^{-8}$
0	4	2.9	0.66	2.2	0.4	2.4	0.48	2.2	0.40	0.85	0.02	0.85	0.02

Table 3.6: Relative error $\times 10^{-4}$, b: when reached to the finest level for the first time, a: at the end of first full multigrid cycle. $e = 10$. Started with $n = 128$, went down the coarsest possible grid with projection, i.e. the grid with 3 points.

Keeping in mind that the relative errors are 10^{-4} times the values on Table 3.6, we see that after we reach the finest grid with damped Jacobi or Gauss-Seidel relaxation, doing one complete V or W cycle as the last step of full multigrid method does not contribute as much ($\approx 10^{-1}$ to 10^{-2} reduction) as getting to the good "initial guess" to the reduction of the error. If we had started with the linear interpolation of the boundary conditions as our initial guess, the maximum relative errors we get after one regular V cycle are around 0.01-0.03 for damped Jacobi relaxation and 0.023-0.004 for Gauss-Seidel relaxation, which are worse than the results on Table 3.6. Doing the

same thing with one regular W cycle, we get $5-10\times10^{-5}$ for damped Jacobi relaxation and $4.3-16.3\times10^{-5}$ for Gauss-Seidel relaxation for the relative errors, which are still worse compared to the table. When we calculate the relative errors with red-black Gauss-Seidel relaxation, there is a significant decrease in relative errors at the last step of full multigrid method given that "enough" initial relaxations were applied. After a similar check with regular V and W cycles, we get around $0.42-390\times10^{-5}$ for the relative error after a V-cycle and $17-210\times10^{-5}$ after a W cycle.

Another observation from Table 3.6 points out for the importance of initial relaxations. Usually a balanced number of initial and final relaxations provides a better initial guess, however generally the more initial relaxations we have, the better results we get at the end of one full cycle. On the other hand, applying no initial relaxations results in the highest relative errors.

3.4 Another Example

So far, our discussion focused on standard differential equations. We'll conclude Chapter 3 with an arbitrary example of a ordinary differential equation with nonconstant coefficients and Dirichlet boundary conditions, where the equation is

$$-(1+\tfrac{1}{2}\cos(\pi x))u_{xx}+xu_x+(2+\sin(\pi x))u=(x-1-\cos(\pi x))2e^{2x}+9x+(3x+e^{2x})\sin(\pi x)$$

on $0\leq x\leq 1$ with $u(0)=1$ and $u(1)=e^2+3$ as the boundary conditions. The actual solution is $u(x)=e^{2x}+3x$ as before.

So far we observed that injection is not a preferable method of transfer and there is a small difference between using all possible grids and using only a few. Table 3.7 excludes injection and uses all grids possible to find the number of cycles we need to have to decrease the error by 10^{-11}.

		damped Jacobi				Gauss-Seidel				red-black GS			
ν_1	ν_2	V	W	FMV	FMW	V	W	FMV	FMW	V	W	FMV	FMW
1	0	25	15	18	14	19	15	15	14	22	4	13	3
0	1	25	15	20	15	16	15	15	15	9	3	6	3
2	0	18	9	12	8	12	9	8	8	15	3	8	2
1	1	16	8	10	8	10	8	8	8	8	3	4	2
0	2	18	8	14	8	12	8	10	8	7	3	5	3
3	0	15	7	9	6	10	7	6	6	12	3	6	2
2	1	14	7	9	6	9	7	6	6	7	3	4	2
1	2	14	6	8	6	9	6	6	6	7	3	4	2
0	3	15	6	11	6	10	6	8	6	6	3	5	3
4	0	13	6	8	6	9	5	5	4	11	3	6	2
3	1	12	6	7	5	8	5	5	4	7	3	4	2
2	2	12	6	8	5	8	5	6	4	7	3	4	2
1	3	12	6	7	5	8	5	6	5	7	3	4	2
0	4	13	6	10	6	9	5	8	5	6	3	5	3

Table 3.7: Number of cycles needed to reduce the maximum error by 10^{-11}, i.e. the ratio of the initial error and the final error is less than 10^{-11}. Started with $n = 512$, used all possible grids, i.e. the coarsest grid had 3 points.

Table 3.7 summarizes what we've observed so far and gives us confidence that this is the case even if our problem is some nontrivial differential equation:

- Initial relaxations are important. Preferably we should choose a balanced combination of initial and final relaxations.
- Starting with a "good" initial guess, like linear interpolation of the boundary values, a regular W cycle "competes with" a FMV cycle.
- $\nu_1 + \nu_2 = 2$ is still a "reasonable" choice, in the sense that the results are clearly better with $\nu_1 + \nu_2 = 2$ compared to $\nu_1 + \nu_2 = 1$, but not too worse than $\nu_1 + \nu_2 > 2$.

This ends our discussion of one dimensional multigrid method, where it was easier to do analysis. In the next chapter, we'll generaize these concepts to two dimensions and check whether our conclusions from the one dimensional case are still valid.

Chapter 4: Multigrid in Two Dimensions

Now we will try to generalize the results in one dimension to two dimensions. Similar to one dimensional case, if at least in one of the dimensions high frequency modes occur, we'll call it as *high frequency mode*. In other words, considering the Fourier basis elements $\sin(\pi kx)$, $\cos(\pi kx)$, $\sin(\pi ly)$, $\cos(\pi ly)$, if k or l or both are greater than or equal to $\frac{n}{2}$, we have a high frequency mode.

4.1 Relaxation Methods Revisited

Our relaxation method discussion was a general discussion, so it still applies in two dimensions. To look at an example, we'll again look at the homogeneous Poisson equation with homogeneous Dirichlet boundary conditions, which can be written as

$$-u_{xx} - u_{yy} = 0 \tag{4.1}$$

with 0's on the boundaries. To bring this equation to $A\mathbf{u} = f$ form, we need to "vectorize" $\mathbf{u}$ and $\mathbf{f}$, by just converting the $(n-1) \times (n-1)$ matrices to $(n-1)^2 \times 1$ vectors. If we do this "vectorization" by going along columns or rows, the matrix A will be

$$\frac{1}{h^2}\left[\begin{array}{c|c|c|c|c} T & -I & & & \\ \hline -I & T & -I & & \\ \hline & \ddots & \ddots & \ddots & \\ \hline & & -I & T & -I \\ \hline & & & -I & T \end{array}\right] \tag{4.2}$$

which is a tridiagonal block matrix with another tridiagonal matrix, T, on the main diagonal and and $-I$ on the first diagonal above and below the main diagonal, where I stands for the identity matrix and T can be written as

$$\frac{1}{h^2}\begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix} \tag{4.3}$$

As in one dimension, this is a result of the finite difference discritization applied on our equation, which will be

$$\frac{-u_{i-1,j} + 2u_{i,j} - u_{i+1,j}}{h^2} + \frac{-u_{i,j-1} + 2u_{i,j} - u_{i,j+1}}{h^2} = f_{i,j} \qquad 1 \le i, j \le n-1 \tag{4.4}$$

Matrix descriptions of relaxation methods still apply. To have a better idea, let's look at the individual equations to update $u_{i,j}$.

- Jacobi method:

$$u_{i,j}^{(k)} = \frac{1}{4}\left[u_{i-1,j}^{(k-1)} + u_{i+1,j}^{(k-1)} + u_{i,j-1}^{(k-1)} + u_{i,j+1}^{(k-1)} + h^2 f_{i,j}\right] \qquad 1 \le i, j \le n-1$$

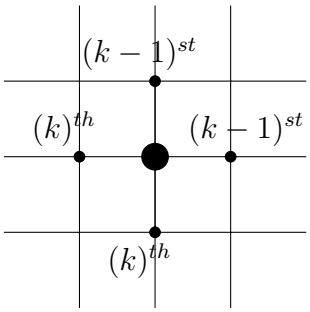
  All values used to update $u_{i,j}$ on k^{th} iteration are from the results of the previous, $(k-1)^{st}$ iteration.

- Damped Jacobi method:

$$u_{i,j}^{(k-1/2)} = \frac{1}{4}\left[u_{i-1,j}^{(k-1)} + u_{i+1,j}^{(k-1)} + u_{i,j-1}^{(k-1)} + u_{i,j+1}^{(k-1)} + h^2 f_{i,j}\right] \qquad 1 \le i, j \le n-1$$
$$u_{i,j}^{(k)} = w u_{i,j}^{(k-1/2)} + (1-w) u_{i,j}^{(k-1)}$$

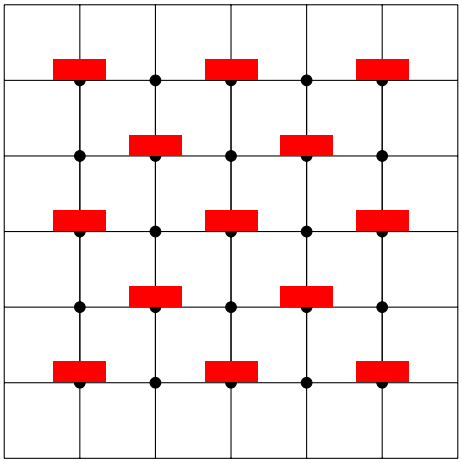
- Gauss-Seidel method:

$$u_{i,j}^{(k)} = \frac{1}{4}\left[u_{i-1,j}^{(k)} + u_{i+1,j}^{(k-1)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k-1)} + h^2 f_{i,j}\right] \qquad 1 \le i, j \le n-1$$



- Red-black Gauss-Seidel method:

$$u_{i,j}^{(k)} = \begin{cases} \frac{1}{4}\left[u_{i-1,j}^{(k-1)} + u_{i+1,j}^{(k-1)} + u_{i,j-1}^{(k-1)} + u_{i,j+1}^{(k-1)} + h^2 f_{i,j}\right] & \text{if } i+j \text{ is even} \\ \frac{1}{4}\left[u_{i-1,j}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} + h^2 f_{i,j}\right] & \text{if } i+j \text{ is odd} \end{cases}$$



The modes with even $i+j$ (red blocks in the picture above) are updated first, then the non-updated nodes (black points) will be updated using the most recent values of the other nodes.

One thing we didn't discuss above is the "ideal" choice of w. If we remember from the one-dimensional case, the aim was to reduce the high frequency errors. Now our frequency depends on two parameters, each "representing" their corresponding spaces. If one of them has a value greater than or equal to $\frac{n}{2}$, we have a high frequency mode.

Doing a Fourier analysis on the damped Jacobi method as we did in the previous chapter on red-black Gauss-Seidel method, we plug in $e_{j_1,j_2} = \sum_{k,l=-n}^{n-1} c_{k,l} e^{i\pi k x_{j_1}} e^{i\pi l x_{j_2}}$ into the damped Jacobi equation, i.e.

$$e_{j_1,j_2}^{(\nu+1)} = \frac{1}{4} w \left[e_{j_1-1,j_2}^{(\nu)} + e_{j_1+1,j_2}^{(\nu)} + e_{j_1,j_2-1}^{(\nu)} + e_{j_1,j_2+1}^{(\nu)} \right] + (1-w) e_{j_1,j_2}^{(\nu)}$$

to get

$$\begin{aligned} c_{k,l}^{(\nu+1)} &= \left[\tfrac{1}{4} w (e^{-i\pi kh} + e^{i\pi kh} + e^{-i\pi lh} + e^{i\pi lh}) + (1-w) \right] c_{k,l}^{(\nu)} \\ &= \left[\tfrac{1}{4} w (2\cos(\pi kh) + 2\cos(\pi lh)) + (1-w) \right] c_{k,l}^{(\nu)} \\ &= \left[\tfrac{1}{2} w (\cos(\pi kh) + \cos(\pi lh)) + (1-w) \right] c_{k,l}^{(\nu)} \end{aligned}$$

To minimize this expression for the high frequencies, we check the extreme cases, where both k and l are $n-1$ and where one is 0, the other is $\frac{n}{2}$. As in one dimension, we want the magnetude of the $\frac{1}{2} w (\cos(\pi kh) + \cos(\pi lh)) + (1-w)$ to be equal at the extreme cases, but they should be of different sign, so that we can get a overall good damping rate. These cases suggest that

$$\begin{aligned} \tfrac{1}{2} w + 1 - w &= -(-2\tfrac{1}{2} w + 1 - w) = 2w - 1 \\ 1 - \tfrac{1}{2} w &= 2w - 1 \\ w &= \tfrac{4}{5} \end{aligned}$$

Therefore in two dimensions a suitable choice for w is 0.8.

For our example problem, these methods work well. Now let us look at a "slightly" different problem in formulation:

$$u_{xx} + \epsilon u_{yy} = f \tag{4.5}$$

with 0's on the boundary: the same equation as 4.1, with an $0 < \epsilon \ll 1$ added as a coefficient. Let us check the eigenvalues of the damped Jacobi iteration matrix to get

an idea of how well damped Jacobi relaxation works for this problem. To calculate the eigenvalues, we'll use the eigenfunctions, which can obtained by using separation of variables technique to solve the differential equation (see [2]). The eigenfunctions turn out to be of the form

$$v_{k,l}(x, y) = \sin(k\pi x)\sin(l\pi y)$$

which are just a generalization of the one dimensional case. Discritizing, we get

$$v_{k,l}(x_i, y_j) = \sin(ikh\pi)\sin(jlh\pi)$$

Now we want to check the eigenvalues of the iteration matrix for the damped Jacobi iteration matrix, i.e. we want to find $\lambda_{k,l}$ satisfying $(I - \omega D^{-1}A)\mathbf{v} = \lambda_{k,l}\mathbf{v}$.

Multiplying both sides by D and seeing A as a sum of two matrices, one representing the $-u_{xx}$ term and the other representing $-\epsilon u_{yy}$ term, we get

$$(1 - \lambda_{k,l})(2 + 2\epsilon)\tfrac{1}{h^2}\mathbf{v} = \omega\tfrac{4}{h^2}(\sin^2(\tfrac{k\pi h}{2}) + \epsilon\sin^2(\tfrac{l\pi h}{2})$$

which can be simplified as

$$\lambda_{k,l} = 1 - \frac{2\omega}{1+\epsilon}\left(\sin^2(\frac{kh\pi}{2}) + \epsilon\sin^2(\frac{lh\pi}{2})\right) \tag{4.6}$$

For k close to 0 and l close to n, we get $\lambda_{k,l} \approx 1 - \frac{2\epsilon\omega}{1+\epsilon} \approx 1$. This means, we cannot reduce the error much in this case. Even with k close to 0 and arbitrary l, this is true: We see that the second term in the paranthesis does not contribute much because of its coefficient, ϵ.

One alternative to solve this problem is to use *line relaxation methods*, which are a variation of the regular relaxation methods in two dimensions.

4.1.1 Line Relaxation Methods

The methods we discuss so far were called *point relaxation methods*, as they were updating the solution pointwise. As the name suggests, in *line relaxation methods*, we treat the "lines" as one "element" to update, instead of single nodes.

Since we have two dimensions now, the way we split up our matrix A depends on along which direction we will choose the lines to update. If we want to choose horizontal lines, i.e. x direction, we need to keep the components on the same line of the discritization together. If we want to "vectorize" our solution moving along the horizontal direction, i.e. convert the $(n-1) \times (n-1)$ matrix

$$\begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n-1} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,n-1} \\ \vdots & \vdots & \ddots & \\ u_{n-1,1} & u_{n-1,2} & \cdots & u_{n-1,n-1} \end{pmatrix}$$

into the $(n-1)^2 \times 1$ vector

$$\left[\begin{array}{cccc|cccc|c|cccc} u_{1,1} & u_{1,2} & \cdots & u_{1,n-1} & u_{2,1} & u_{2,2} & \cdots & u_{2,n-1} & \cdots & u_{n-1,1} & \cdots & u_{n-1,n-1} \end{array} \right]^T$$

we see that the neighbors along the x direction will be multiplied by those elements of A, which are closer to the main diagonal. Hence, for example, in two dimensional Poisson equation with homogeneous Dirichlet boundary conditions, if we "vectorize" our solution along horizontal direction, our choices for D_x, L_x and U_x, with $A = L_x + D_x + U_x$, will be:

D_x: Block diagonal matrix where each diagonal block is T, which was defined in equation 4.3

L_x: Block lower triangular matrix where each block on the first diagonal below is $-I$

U_x: Block lower triangular matrix where each block on the first diagonal above is $-I$

When we check a single row for line Jacobi relaxation applied along the horizontal direction, i.e. a single row of the equation $D_x\mathbf{u} = -(L_x + U_x)\mathbf{u} + \mathbf{f}$, in our example problem, we get

$$-u_{i,j-1}^{(k)} + 4u_{i,j}^{(k)} - u_{i,j+1}^{(k)} = u_{i-1,j}^{(k-1)} + u_{i+1,j}^{(k-1)} + h^2 f_{i,j} \qquad 1 \leq i, j \leq n-1$$

We can clearly see in this elementwise notation that to update the i^{th} row, we use the values on the lines above and below.

Similarly, with line Gauss-Seidel relaxation method, i.e. $(D_x + L_x)\mathbf{u} = -U_x\mathbf{u} + \mathbf{f}$, we get

$$-u_{i,j-1}^{(k)} + 4u_{i,j}^{(k)} - u_{i,j+1}^{(k)} = u_{i-1,j}^{(k)} + u_{i+1,j}^{(k-1)} + h^2 f_{i,j} \qquad 1 \leq i, j \leq n-1$$

where we use the updated values of the previous line immediately after calculating.

If we want to choose vertical lines, i.e. y direction, either we can simply reorder our "vectorized solution" and the right-hand side of the equation by going along the vertical lines, and constructing the matrix A accordingly, or we can keep the same ordering, and break A down as $A = L_y + D_y + U_y$ where

D_y: (main diagonal of A)+L_x+U_x, L_x and U_x defined as block lower/upper block triangular matrices with $-I$ on the first diagonal block below/above

L_y: lower triangular part of $A - D_y$, i.e. block diagonal matrix with lower triangular part of T on the diagonal

U_y: upper triangular part of $A - D_y$, i.e. block diagonal matrix with upper triangular part of T on the diagonal

Coming back to our "problematic" equation, $u_{xx} + \epsilon u_{yy} = f$, we can check the eigenvalues of the damped Jacobi iteration matrix when line relaxation is applied along x- and y-directions. For x-direction, following the steps we used to get 4.6, we

get

$$\begin{aligned}(D-\omega A)\mathbf{v}_{k,l} =& D\lambda_{k,l}\mathbf{v}_{k,l}\\ D(\lambda_{k,l}-1)\mathbf{v}_{k,l} = -\omega A\mathbf{v}_{k,l} =& -\omega\left(\tfrac{4}{h^2}\sin^2\left(\tfrac{kh\pi}{2}\right)+\epsilon\tfrac{4}{h^2}\sin^2\left(\tfrac{lh\pi}{2}\right)\right)\mathbf{v}_{k,l}\\ =&(\lambda_{k,l}-1)\left(\tfrac{4}{h^2}\sin^2\left(\tfrac{\pi kh}{2}\right)+\tfrac{2\epsilon}{h^2}\right)\mathbf{v}_{k,l}\\ \lambda_{k,l} =& 1-\frac{\omega\left(\tfrac{4}{h^2}\sin^2\left(\tfrac{kh\pi}{2}\right)+\epsilon\tfrac{4}{h^2}\sin^2\left(\tfrac{lh\pi}{2}\right)\right)}{\tfrac{4}{h^2}\sin^2\left(\tfrac{\pi kh}{2}\right)+\tfrac{2\epsilon}{h^2}}\\ =&1-\frac{2\omega\left(\sin^2\left(\tfrac{kh\pi}{2}\right)+\epsilon\sin^2\left(\tfrac{lh\pi}{2}\right)\right)}{2\sin^2\left(\tfrac{\pi kh}{2}\right)+\epsilon}\end{aligned}$$

Therefore,

$$\lambda_{k,l} = 1-\frac{2\omega}{2\sin^2(\frac{k\pi h}{2})+\epsilon}\left(\sin^2\left(\frac{kh\pi}{2}\right)+\epsilon\sin^2\left(\frac{lh\pi}{2}\right)\right) \tag{4.7}$$

When k is small, ignoring the $\sin^2(\frac{k\pi h}{2})$ terms, we get $1-2\omega\sin^2(\frac{l\pi h}{2})$ for the eigenvalue, which is the same with the one dimensional case! Similarly, if k is "not that small", ϵ terms can be ignored, leaving us $1-w$! So choosing $\omega\approx\frac{2}{3}$ as in one dimensional case would give an approximate damping ratio of $\frac{1}{3}$, as we had in the one dimensional problem. So applying damped line Jacobi relaxation helps when we choose the lines along x-direction, i.e. direction of strong coupling.

When we choose the lines in the other direction, direction of weak coupling, this time we get

$$\lambda_{k,l} = 1-\frac{2\omega}{1+2\epsilon\sin^2(\frac{l\pi h}{2})}\left(\sin^2(\frac{kh\pi}{2})+\epsilon\sin^2(\frac{lh\pi}{2})\right) \tag{4.8}$$

following the same steps as the above calculation. As in the pointwise damped Jacobi relaxation, when k is small, we get $\lambda_{k,l}\approx 1-2\omega\epsilon\sin^2(\frac{l\pi h}{2})\approx 1$, therefore we can't get a good damping ratio.

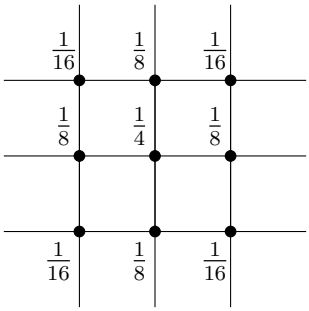
4.2 Two Grid Method In Two Dimensions

Two dimensional two grid method is not very different from the one dimensional two grid method, except we need to define the way we project and interpolate the data between fine and coarse grids, since here we "add and delete" grid points in both dimensions.

As in one dimensional case, we have two ways to transfer data to a coarser grid. The "standard" and "better" way is *projection*, where we use a weighted average of the data on that point and on the neighboring points on the fine grid:

$$x_{i,j}^{2h} = \frac{1}{16}\Big[4x_{2i,2j}^{h} + 2\left(x_{2i-1,2j}^{h} + x_{2i,2j-1}^{h} + x_{2i+1,2j}^{h} + x_{2i,2j+1}^{h}\right) + x_{2i-1,2j-1}^{h} + x_{2i+1,2j-1}^{h} + x_{2i-1,2j+1}^{h} + x_{2i+1,2j+1}^{h}\Big]$$

where $i, j = 1, \cdots, \frac{n}{2} - 1$. Projection can be visualized as



In the above picture, we see the nodes on the finer grid with their corresponding weights involved in the projection. Calculating their weighted average, we get the data at the node, which is in the middle of the sketched nodes, on the coarser grid. (On the coarser grid, the nodes around the middle one "dissappear".)

Another method, which is very straightforward, is *injection*. As in one dimension, injection transfers the data to the coarser grid by just carrying the data at the

corresponding nodes on the finer grid, i.e.

$$x_{i,j}^{2h} = x_{2i,2j}^{h} \quad \text{where} \quad i,j = 1,\cdots,\tfrac{n}{2}-1$$

Going in the other direction, from the coarser grid to the finer grid, again we take the average of the data at the neighboring nodes, which are available on the coarser grid:

$$\begin{aligned} x_{2i,2j}^{h} &= x_{i,j}^{2h} \\ x_{2i+1,2j}^{h} &= \tfrac{1}{2}(x_{i,j}^{2h} + x_{i+1,j}^{2h}) \\ x_{2i,2j+1}^{h} &= \tfrac{1}{2}(x_{i,j}^{2h} + x_{i,j+1}^{2h}) \\ x_{2i+1,2j+1}^{h} &= \tfrac{1}{4}(x_{i,j}^{2h} + x_{i+1,j}^{2h} + x_{i,j+1}^{2h} + x_{i+1,j+1}^{2h}) \end{aligned}$$

where $i,j = 0,1,\cdots,\frac{n}{2}-1$. Notice that this *interpolation* is the "reverse" of projection, and its operator is 4 times the transpose of the projection operator.

Using these operators, we follow the same steps given in Section 2.1 to apply the two-grid method.

To check the efficiency of the two grid method in two dimensions, it is a good idea to look at the maximum errors we get with different initial guesses. The example in Figure 4.1 is with 2 initial relaxations and no final relaxation.

Let us check what happens when we do one initial and one final relaxation, which is shown in Figure 4.2.

It seems like overall one initial relaxation and one final relaxation does a better job than two initial relaxations! However, for some special cases, two initial relaxations eliminates the error better than one initial and one final relaxations. For example, if $k+l \approx n$, choosing red-black Gauss-Seidel relaxation would be beneficial compared

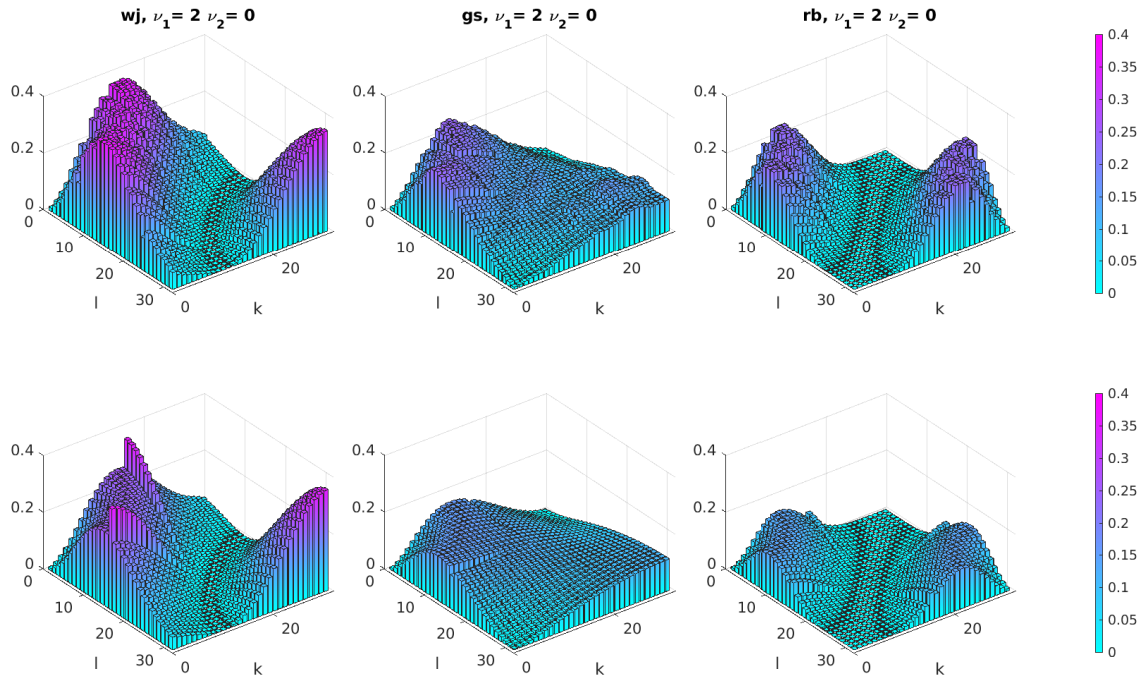


Figure 4.1: Top row: Absolute maximum error, second row: 2-norm of the error, with two initial relaxations for homogeneous Poisson equation with homogeneous Dirichlet boundary conditions. Initial guess: $\sin(k\pi x)\sin(l\pi y)$ where k and l are represented on x and y axis.

to the same relaxation method with one initial and one final relaxations. Again, the choice depends on the error in hand, keeping in mind that for arbitrary errors one initial and one final relaxation is a better choice. Although the number of basis elements that two initial relaxations can damp more efficiently is higher, if we don't have any information about the error, it is better to try to reduce the error for any arbitrary case.

So as we see in Figure 4.1 and Figure 4.2, although red-black Gauss-Seidel does a better job for more k and l's compared to other relaxation methods, there is no such

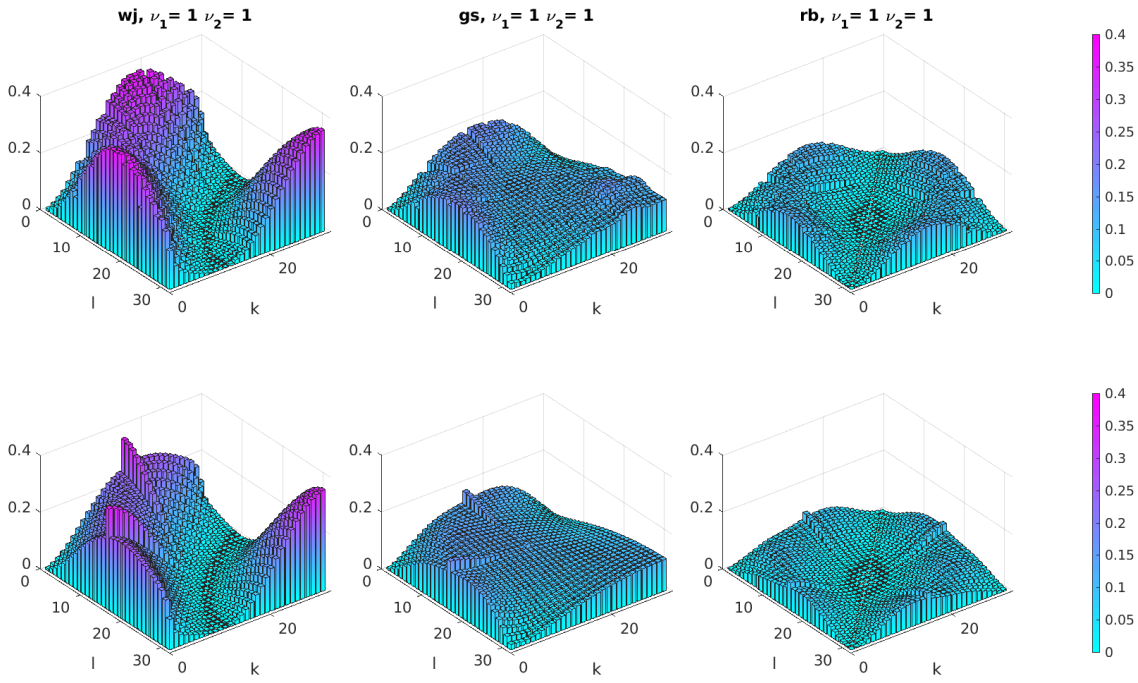


Figure 4.2: Top row: Absolute maximum error, second row: 2-norm of the error, with one initial relaxation and one final relaxation for homogeneous Poisson equation with homogeneous Dirichlet boundary conditions. Initial guess: $\sin(k\pi x)\sin(l\pi y)$ where k and l are represented on x and y axis.

"magic" as the red-black Gauss-Seidel method in the one dimensional case, where the error "disappears". Moreover, in some cases choosing damped Jacobi relaxation might be advantageous over choosing red-black Gauss-Seidel relaxation, as seen in the figures. But again, for arbitrary errors, since we cannot guarantee that we have one of those cases, it is safer to choose red-black Gauss-Seidel relaxation, probably with one initial and one final relaxations.

4.3 V and W and Full Multigrid cycles

In two dimension, V and W and full multigrid cycles have the same algorithm, so we will jump to conclusions. As above we check Poisson equation $-u_{xx} - u_{yy} = f(x, y)$, but this time right hand side of the equation is nonzero. Instead, we have $f(x, y) = -\frac{17}{4}e^{2x+\frac{y}{2}}$, $0 \le x, y \le 1$ and the actual solution is $u(x, y) = e^{2x+\frac{y}{2}}$.

Tables 4.1 and 4.2 compare the number of cycles needed to reduce the error by 10^{-11} with various relaxation methods and relaxation numbers, the first one using all possible grids and the second one using only 4 grids. For V and W cycles, the initial guess is two dimensional linear interpolation of the boundary conditions. To transfer data from finer grid to the coarser grid, projection is used.

		damped Jacobi				Gauss-Seidel				red-black GS			
ν_1	ν_2	V	W	FMV	FMW	V	W	FMV	FMW	V	W	FMV	FMW
1	0	51	47	41	34	26	26	21	19	26	18	20	13
0	1	52	47	41	33	28	27	22	20	27	18	21	13
2	0	27	24	21	17	17	13	12	10	16	10	12	7
1	1	27	24	20	17	15	12	11	10	13	10	9	7
0	2	28	24	22	17	18	12	14	10	17	10	13	8
3	0	20	17	15	12	14	10	10	8	13	8	10	7
2	1	19	16	14	12	13	10	9	8	11	8	8	6
1	2	19	16	15	12	12	10	9	8	11	8	8	6
0	3	21	17	17	12	14	10	12	8	14	8	11	7
4	0	17	13	12	10	12	9	9	7	12	7	9	6
3	1	16	13	12	9	11	9	8	7	10	7	7	6
2	2	16	13	12	9	11	9	8	7	10	7	7	6
1	3	17	13	13	9	11	9	8	7	10	7	7	6
0	4	18	13	14	10	13	9	10	7	12	7	9	6

Table 4.1: Number of cycles needed to reduce the maximum error by 10^{-11}, i.e. the ratio of the initial error and the final error is less than 10^{-11}. Started with $n = 128$, used all possible grids, i.e. the coarsest grid had 3^2 points.

The results in table 4.1 are very similar to one dimensional results. For example, still W cycle "competes with" FMV cycle, though this time FMV cycle is favoured. We see that $\nu_1 + \nu_2 = 2$ is still a reasonable choice of the total relaxations, considering both the reduction of the error and the computational complexity of the method.

In one dimension, as long as we had enough initial relaxations, we were getting good results. In Table 4.1 it is clear that we need a balanced number of initial and final relaxations to get slightly better results.

Next, we will check the results of the methods in Table 4.1 with reduced the number of grids. This time we'll only use 4 grids.

		damped Jacobi				Gauss-Seidel				red-black GS			
ν_1	ν_2	V	W	FMV	FMW	V	W	FMV	FMW	V	W	FMV	FMW
1	0	50	47	38	34	26	26	19	19	24	18	18	13
0	1	50	47	38	33	27	27	21	20	24	18	18	13
2	0	26	24	20	17	14	13	11	10	13	10	10	7
1	1	26	24	21	17	14	12	13	10	11	10	9	7
0	2	27	24	20	17	15	12	12	10	14	10	11	8
3	0	19	17	16	14	11	10	10	8	11	8	10	7
2	1	18	16	14	12	11	10	9	8	9	8	7	6
1	2	19	16	15	12	11	10	10	8	9	8	8	6
0	3	19	17	15	12	12	10	10	8	11	8	9	7
4	0	15	13	13	11	10	9	9	8	10	7	9	6
3	1	15	13	11	9	10	9	8	7	8	7	7	6
2	2	15	13	13	9	10	9	9	7	8	7	7	6
1	3	15	13	12	9	10	9	8	7	9	7	7	6
0	4	16	13	13	10	11	9	9	7	10	7	8	6

Table 4.2: Number of cycles needed to reduce the maximum error by 10^{-11}, i.e. the ratio of the initial error and the final error is less than 10^{-11}. Started with $n = 128$, used only 4 grids, i.e. the coarsest grid had 17^2 points.

The results on Table 4.2 are better than the ones on Table 4.1. The same reasoning from the one dimensional case is still valid: Evaluating the error by a direct method at a level closer to the finest grid gives us a better approximation for the error. But we need to keep in mind that the coarsest grid in Table 4.2 has $17^2 = 289$ points, therefore solving the system directly requires solving a matrix equation with a matrix of size 289×289, which may increase the time complexity of the method.

A balanced number of relaxations looks like a better choice, as previous results, with an exception of the case with a total of 2 damped Jacobi or Gauss-Seidel relaxations.

Lastly, comparing the relative errors at the finest grid as in Table 3.6, i.e. when reached to the finest level for the first time and at the end of the full multigrid cycle, we get the results listed in Table 4.3.

Other than the facts that we've already observed, like the preference for the balance of initial and final relaxations, Table 4.3 also shows us how useful are the initial steps of the full multigrid method. As in one dimensional case, keeping in mind that the values on Table 4.3 are 10^{-5} times the actual relative errors, we see that the "initial guess" we get is already quite accurate, and the rate of further reduction of the error by regular V or W cycles are not high. If we take the two dimensional linear interpolation of the boundary values as our initial iterate, we see that the relative error after a regular V cycle with $1 < \nu_1 + \nu_2 < 5$ is between 0.06-0.14 with damped Jacobi relaxation, 0.03-0.1 with Gauss-Seidel relaxation and 0.02-0.07 with red-black Gauss-Seidel relaxation. All these errors are clearly higher than the error we get just after having the first guess on the finest grid, which is given under the columns labeled

		damped Jacobi				Gauss-Seidel				red-black GS			
		FMV		FMW		FMV		FMW		FMV		FMW	
ν_1	ν_2	b	a	b	a	b	a	b	a	b	a	b	a
1	0	336	182	55	11.6	70.8	21.6	24.7	4.8	82.5	29.1	24.5	3.4
0	1	425	224	49.7	9.7	190	76.6	29.8	4.4	127	47.4	23.1	3.4
2	0	92.9	32.8	32.4	5.6	23.5	5.0	16	2.0	27.7	5.2	17.2	1.5
1	1	88.8	30.2	26.3	3.8	25.8	4.5	16.3	1.1	17.6	1.8	13.2	0.38
0	2	147	55.6	27.7	3.8	69	20.4	23	3.4	50.1	13.2	20.5	3.3
3	0	47.1	12.3	24.3	3.4	17.9	2.1	14.5	0.99	19.4	2.4	14.5	0.81
2	1	40.9	9.7	20.1	2.2	17.3	1.8	14	0.5	14.4	0.91	12.9	0.25
1	2	46.4	11.8	19.6	2.0	18.8	2.2	14.7	0.76	14.3	0.94	12.8	0.17
0	3	84.6	26.5	22.9	3.4	43.4	10.8	20.8	3.32	34.3	7.7	20.3	3.3
4	0	32.2	6.6	20.1	2.3	15.8	1.4	13.9	0.84	16.5	1.6	13.4	0.65
3	1	27.6	5.0	17.2	1.5	15.3	1.2	13.2	0.43	13.5	0.62	12.8	0.18
2	2	27.9	5.1	16.6	1.3	15.5	1.3	13.4	0.44	13.5	0.62	12.7	0.18
1	3	32.5	6.8	17.5	1.4	16.5	1.5	13.9	0.57	13.3	0.64	12.6	0.12
0	4	59.2	16.3	20.8	3.3	34.4	7.7	20.3	3.3	29	6.0	20.1	3.3

Table 4.3: Relative error $\times 10^{-5}$, b: when reached to the finest level for the first time, a: at the end of first full multigrid cycle. $e = 10$. Started with $n = 128$, went down the coarsest possible grid with projection, i.e. the grid with 3^2 points.

with "b". So the first steps of the full multigrid method provide a better "solution" compared to the regular V or W cycles in two dimensions.

Checking regular W cycles with $5 > \nu_1 + \nu_2 > 1$ to see the error after one regular W cycle, we get the following relative errors: 0.03-0.07 for damped Jacobi relaxation, 0.01-0.05 for Gauss-Seidel relaxation and 0.004-0.03 for red-black Gauss-Seidel relaxation. Therefore the above conclusions for V-cycles are also valid for W cycles. If we remember, although W cycles can compete with FMV cycles in one dimension, in two dimension we saw and clearly state that FMV cycles are favoured.

Chapter 5: Conclusion

So far we had a "survey" of the multigrid methods in one and two dimensions, discussing the results of some example differential equations with Dirichlet boundary conditions. Recall that those boundary conditions do not have to be homogeneous, and the coefficients in the equation do not have to be constant.

This thesis serves the purpose of an introductory documentation of the multigrid method. Since multigrid method needs a relaxation method, we also summarized basic pointwise iterative relaxation methods, which can be used in the multigrid method. Relaxation methods which can be used in the multigrid method is not restricted to these basic methods. The better our choice of relaxation method is, the better the results of the multigrid method. For example, in our discussion one can see the difference between when damped Jacobi or red-black Gauss-Seidel relaxations are chosen.

We also proved some facts in one dimensional multigrid method. Our proofs were problem specific, but the procedure can be applied to any particular differential equation to see the outcomes. We considered simple cases, where it was easy to prove facts. As the equations get complicated (for example, with nonconstant coefficients), obviously the analysis will become harder.

In two dimensions, Fourier analysis can be applied for theoretical discussions.

A well-known fact is that the multigrid method works well on elliptic equations with Dirichlet boundary conditions. A straightforward generalization would be applying multigrid methods on other types of differential equations. Examples can be found in literature, where multigrid methods are applied on nonlinear problems [11], on the convection diffusion equation [9] or on Navier-Strokes equations [5].

Similarly, we can increase the number of dimensions of the domain of the problem. However, two and three dimensions are of special interest of physical applications. Examples of three dimensional multigrid method can be found in [9] and [5].

Another aspect of the multigrid method is the "discretization". Throughout this thesis, we used "structured", uniform discretization, where we divide the domain into equal "pieces". However, this doesn't have to be the case. Different discretizations can be used to partition the domain. Unstructured partitions are helpful if the domain has curved boundaries.

In *Adaptive multigrid* [9] the grid refinements are determined dynamically during the process, depending on the intermediate results. The idea behind adaptive multigrid is to have finer grids only where needed, thus to reduce the computations and increase the efficiency. It is very useful especially if we don't know the behaviour of the solution in advance.

In this text, we considered finite difference discretizations of the differential equations to get a system of linear equations and used multigrid method on this system. Similarly, we could have used finite element method to convert our differential equation to a system of linear equations and use multigrid method on it [11]. [3] is one of the introductory books for finite element methods one can refer to.

Although multigrid method is sequential in the sense that it follows a sequence while moving along grids, it can be parallelized (for more information on parallel computing, see [6]). Parallelization for multigrids is described in [9], with variations found in many sources (see [5]).

Bibliography

[1] G. R. Baker and E. A. Overman. *The Art of Scientific Computing*. preprint, First edition, 2014.

[2] W. E. Boyce and R. C. DiPrima. *Elementary Differential Equations and Boundary Value Problems*. New York: Wiley, Eighth edition, 2005.

[3] Graham E. Carey Eric B. Becker and J. Tinsley Oden. *Finite Elements: An Introduction*. Prentice Hall, First edition, 1981.

[4] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations : steady-state and time-dependent problems*. Society for Industrial and Applied Mathematics, First edition, 2007.

[5] Steve F. McCormick. *Multigrid Methods: Theory, Applications, and Supercomputing*. Marcel Dekker, Inc., First edition, 1988.

[6] Peter S. Pacheco. *A Introduction to Parallel Programming*. Morgan Kaufmann, First edition, 2011.

[7] Gilbert Strang. *Linear Algebra and its Applications*. Thomson, Brooks/Cole, Fourth edition, 2006.

[8] J. W. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*. Springer, First edition, 1995.

[9] C. W. Oosterlee U. Trottenberg and A. Schüller. *Multigrid*. Academic Press, First edition, 2001.

[10] Richard S. Varga. *Matrix Iterative Analysis*. Springer, Second edition, 2000.

[11] Van Emden Henson William L. Briggs and Steve F. McCormick. *A Multigrid Tutorial*. Siam, Second edition, 2000.