

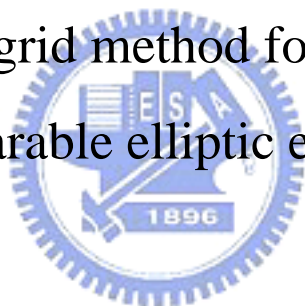
國立交通大學

應用數學系
碩士論文

多重網格法解一些

不可分離的橢圓方程式

Multigrid method for some
nonseparable elliptic equations



研究生：莊勝凱

指導老師：賴明治 教授

中華民國九十四年十月

多重網格法解一些
不可分離的橢圓方程式

Multigrid method for some
nonseparable elliptic equations

研究生：莊勝凱
指導教授：賴明治

Student : Sheng-kai Chuang
Advisor : Ming-Chih Lai

國立交通大學

應用數學系

碩士論文



Submitted to Department of Applied Mathematics

College of Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

in

Applied Mathematics

October 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年十月


多重網格法解一些不可分離的橢圓方程式

學生：莊勝凱

指導教授：賴明治

國立交通大學應用數學系(研究所)碩士班

摘 要



這篇論文主要之目的是使用多重網格法來解一些不可分離的橢圓方程式有著 Dirichlet 條件在矩形的區域上(當然這種方法也可應用在其他的邊界條件下)。首先，我們會學習基本的多重網格法。再來，我們簡要地介紹預加條件共軛梯度法和 Concus and Golub 法。最後，我們會給一些例子並且列出數值結果其中包含了達到判停條件所需的計算時間和迭代次數，然後做出結論。

Multigrid method for some nonseparable elliptic equations

student : Sheng-kai Chuang Advisor : Dr. Ming-Chih Lai

Department (Institute) of Applied Mathematics
National Chiao Tung University



The primary objective of this thesis is to use multigrid method (MG) for solving nonseparable elliptic equations with Dirichlet boundary condition on a rectangle. (Of course, this method can be applied with any boundary conditions.) First, we study elements of multigrid method. Next, we introduce roughly the preconditioned conjugate gradient (PCG) method and Concus and Golub's method to compare with MG. Finally, we give some examples and show numerical results including CPU time and the number of necessary iterations to achieve stopping criterion, and the conclusion follows.

誌 謝

這篇論文的完成首先要感謝我的指導老師 賴明治教授，從一開始的題目選定，還有後來參考資料的協尋，以及完成後的數據分析，都給了我很大的幫助。此外，在這兩年來，老師除了在學問上的指導令我獲益良多外，其對於研究事物的態度更是令我留下了深刻的印象，謹此致上我最誠摯的敬意與謝意。口試期間，也承蒙黃聰明老師、吳金典老師及張書銘老師費心審閱並提供了寶貴的意見，使得本論文得以更加的完備，永誌於心。

在這兩年求學的過程中，也要感謝昱豪學長在我遇到問題時，總是細心及耐心地給予我意見及靈感的啟發；另外，也要感謝同學們及學弟妹給我的支持與鼓勵，讓我在這些日子過的並不孤單，跟你們一起出遊聚餐還有打球更是令我開心且難忘的寶貴回憶，隨著畢業的分離，心中卻也增添了一分的不捨，也希望大家都能實現自己的理想，有著美好的未來。

最後，要感謝的不只陪伴了我兩年而是陪伴了我二十多年的家人們，你們不但讓我有良好的生活環境，讓我可以求學的路上走得更專心，也總是不辭辛苦的照料我幫忙我，讓我能夠克服種種的困難繼續向前邁進，因為有你們，也讓我多了一份必須更上進的責任。再次地感謝所有幫助過我及關心過我的人，謝謝你們！

目 錄

中文提要	i
英文提要	ii
誌謝	iii
目錄	iv
1.	Introduction	1
2.	Multigrid Method	2
3.	Preconditioned Conjugate Gradient Method	7
4.	Concus and Golub's Method	11
5.	Numerical Examples	12
6.	Conclusion	18
References	19



1 Introduction

Solving Helmholtz equations is a fundamental problem of scientific computing. Generalized Helmholtz equations

$$\begin{aligned} \Delta v - g(x, y)v &= f(x, y) & \text{in } & \Omega \\ v &= p(x, y) & \text{on } & \partial\Omega, \end{aligned} \quad (1)$$

arise frequently in fields such as optic, geophysical, and plasma physics. In addition, non-separable elliptic equations of the form

$$\nabla \cdot (a(x, y)\nabla u) - b(x, y)u = c(x, y) \quad (2)$$

also can be transformed to the form of a generalized Helmholtz equation (1) through a change of variable $v = a^{1/2}u$, when $a(x, y)$ is positive in the domain of definition.

$$\begin{aligned} v &= a^{1/2}u \Rightarrow u = a^{-1/2}v \\ \nabla u &= a^{-1/2}\nabla v - \frac{1}{2}a^{-3/2}v\nabla a \\ a\nabla u &= a^{1/2}\nabla v - \frac{1}{2}a^{-1/2}v\nabla a \\ \nabla \cdot (a\nabla u) &= a^{1/2}\nabla^2 v + \nabla a^{1/2} \cdot \nabla v - \frac{1}{2}a^{-1/2}\nabla a \cdot \nabla v - v\nabla \cdot \left(\frac{1}{2}a^{-1/2}\nabla a\right) \\ &= a^{1/2}\nabla^2 v - \nabla^2 a^{1/2}v \end{aligned}$$

So, we only need to be absorbed in eq. (1).

Multigrid methods have become a common approach for solving system arising from discretizing elliptic equation. The basic multigrid principle is that the smoother damps the oscillatory high frequency errors whereas the coarse grid correction reduces the smooth low frequency errors. Multigrid method begins with a two-grid process. First, iterative relaxation is applied, whose effect is to smooth the error. Then a coarse-grid correction is applied, in which the smooth error is determined on a coarser grid. This error is interpolated to the fine grid and used to correct the fine-grid approximation. Applying this method recursively to solve the coarse-grid problem leads to multigrid.

2 Multigrid Method

We usually use *iterative methods* for solving linear system when the matrix arising from problem is sparse. A clear advantage of using iterative methods is that they require far less computational effort. The principle of iterative method is beginning with an initial guess, and improve approximation successively until it is as accurate as desired. Most of iterative methods can reduce efficiently oscillatory components of the error, but it is much less effective as smooth components remained (sometimes relaxation is also called smoother). The notion of smooth and oscillatory components are relative to the grid size which the solution is defined. The smooth error on the fine grid is more oscillatory when projected on the coarse grid. Therefore, we might relax on the fine grid to reduce oscillatory components of the error and then relax on the coarse grid when smooth components remained.

What is the relationship between the error and the approximation? An important scheme: *residual correction* gives a appropriate answer. Suppose that the system $A\mathbf{u} = \mathbf{f}$ has a unique solution and that \mathbf{v} is a computed approximation to \mathbf{u} . It is easy to compute the residual $\mathbf{r} = \mathbf{f} - A\mathbf{v}$. The error $\mathbf{e} = \mathbf{u} - \mathbf{v}$ satisfies the *residual equation*: $A\mathbf{e} = \mathbf{r}$. To improve the approximation \mathbf{v} , we might solve the residual equation for \mathbf{e} , and then compute a new approximation using the definition of the error $\mathbf{u} = \mathbf{v} + \mathbf{e}$.

Now, we can combine these two ideas to produce *two-grid correction scheme*, as outlined below [1].

Two-Grid Correction Scheme


$$\mathbf{v}^h \leftarrow TG(\mathbf{v}^h, \mathbf{f}^h)$$

- Relax ν_1 times on $A^h\mathbf{u}^h = \mathbf{f}^h$ on Ω^h with initial guess \mathbf{v}^h .
- Compute the fine-grid residual $\mathbf{r}^h = \mathbf{f}^h - A^h\mathbf{v}^h$ and restrict it to the coarse grid by $\mathbf{r}^{2h} = I_h^{2h}\mathbf{r}^h$.
- Solve $A^{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$ on Ω^{2h} .
- Interpolate the coarse-grid error to the fine grid by $\mathbf{e}^h = I_{2h}^h\mathbf{e}^{2h}$ and correct the fine-grid approximation by $\mathbf{v}^h \leftarrow \mathbf{v}^h + \mathbf{e}^h$.
- Relax ν_2 times on $A^h\mathbf{u}^h = \mathbf{f}^h$ on Ω^h with initial guess \mathbf{v}^h .

Ω^{2h} denotes coarse-grid has twice the grid spacing of the fine grid Ω^h . The procedure transferring the vectors from a fine-grid to a coarse-grid is called *restriction*, and this operator is denoted by I_h^{2h} ; the procedure transferring the vectors from a coarser-grid to a fine-grid is called *interpolation* or *prolongation*, and this operator is denoted by I_{2h}^h . The arrow notation stands for replacement or overwriting. The integers ν_1 and ν_2 are parameters in the scheme that control the number of relaxation sweeps before and after

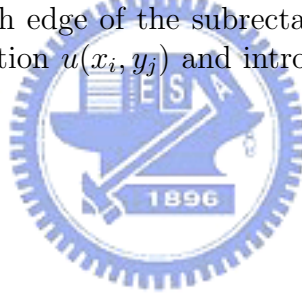
visiting the coarse grid. They are usually fixed at the start, based on either theoretical considerations or on past experimental results, and they are often small.

In general, injection and full weighting operators are more common restriction we used; linear and cubic interpolations are popular methods. The issue of intergrid transfers is discussed at some books about multigrid method [1][2]. And some basic iterative method like Jacobi, Gauss-Seidel (GS) or red-black Gauss-Seidel (RBGS) can be found in elementary numerical books. Therefore, we do not mention all of them here. We only give definition of full weighting restriction, linear interpolation and red-black Gauss-Seidel method for two-dimensional problem since we will use these tools in the later numerical experiment.

In order to illustrate them more conveniently, we consider a simple second-order boundary value problem

$$\begin{cases} -\Delta u = f(x, y), & 0 < x < 1, 0 < y < 1 \\ u = 0, & \text{on the boundary} \end{cases}$$

on rectangle. The domain of the problem $\{(x, y) : 0 \leq x \leq 1, 0 \leq y \leq 1\}$ is partitioned into $n \times n$ subrectangle by introducing the grid points $x_i = ih, y_j = jh$, where $h = 1/n$ is the constant width of the each edge of the subrectangle. We also introduce v_{ij} as an approximation to the exact solution $u(x_i, y_j)$ and introduce f_{ij} as the value of $f(x, y)$ at (x_i, y_j) .



- Linear interpolation

If we let $I_{2h}^h \mathbf{v}^{2h} = \mathbf{v}^h$, then the components of \mathbf{v}^h are given by

$$\begin{aligned} v_{2i,2j}^h &= v_{ij}^{2h}, \\ v_{2i+1,2j}^h &= \frac{1}{2}(v_{ij}^{2h} + v_{i+1,j}^{2h}), \\ v_{2i,2j+1}^h &= \frac{1}{2}(v_{ij}^{2h} + v_{i,j+1}^{2h}), \\ v_{2i+1,2j+1}^h &= \frac{1}{4}(v_{ij}^{2h} + v_{i+1,j}^{2h} + v_{i,j+1}^{2h} + v_{i+1,j+1}^{2h}), \quad 0 \leq i, j \leq \frac{n}{2} - 1. \end{aligned}$$

Linear interpolation is effective when the vector is smooth.

- Full weighting restriction

If we let $I_h^{2h} \mathbf{v}^h = \mathbf{v}^{2h}$, then the components of \mathbf{v}^{2h} are given by

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

The values of the coarse-grid vector are weighted averages of values at neighboring fine-grid points.

- Red-black Gauss-Seidel relaxation

This method may be expressed in component form as below procedure. First, update red points v_{ij} by

$$v_{ij} \leftarrow \frac{1}{4}(v_{i-1,j} + v_{i+1,j} + v_{i,j-1} + v_{i,j+1} + h^2 f_{ij}),$$

where sum of index i and j is even. Then update black points v_{ij} by

$$v_{ij} \leftarrow \frac{1}{4}(v_{i-1,j} + v_{i+1,j} + v_{i,j-1} + v_{i,j+1} + h^2 f_{ij}),$$

where sum of index i and j is odd. Fig. 1 shows red and black points. Red-black Gauss-Seidel has a clear advantage in terms of parallel computation.

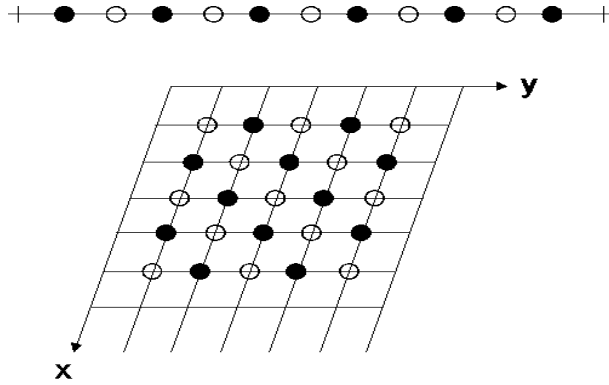


Figure 1: A one-dimensional grid (top) and a two-dimensional grid (bottom) showing the red points \circ and the black points \bullet for red-black relaxation.

Two-grid correction is the basis of multigrid. Of course, we can repeat this process on successively coarser grids until arriving the coarsest grid. This is so-called *V-cycle*. To obtain more benefit from the coarser grids, where computations are cheaper, the *W-cycle* zigzags among the lower-level grids before moving back up to the finest grid. We can also join *nested iteration* idea which uses V-cycle on coarse grids to obtain improved initial guesses for fine grids. This is so-called *full multigrid V-cycle* (FMG).

The schedule of grids for V-cycle, W-cycle, and FMG show in Fig. 2. The complete algorithms of them are given in Algorithm 1,2,3 respectively. To describe Algorithm conveniently, we change some notations. We call the right-side vector of the residual equation \mathbf{f} , rather than \mathbf{r} , because it is just another right-side vector. Instead of calling the solution of the residual equation \mathbf{e} , we use \mathbf{u} because it is just a solution vector. We can then use \mathbf{v} to denote approximations to \mathbf{u} .

Algorithm 1 $V^h(\mathbf{v}^h, \mathbf{f}^h)$	V-cycle Method
if (h =coarsest) then	
$\mathbf{v}^h = (A^h)^{-1}\mathbf{f}^h$	{solve $A^h\mathbf{u}^h = \mathbf{f}^h$ directly}
else	
$\mathbf{v}^h = \text{Relax}(\mathbf{v}^h, \mathbf{f}^h, \nu_1)$	{pre-relax ν_1 times with initial \mathbf{v}^h }
$\mathbf{f}^{2h} = I_h^{2h}(\mathbf{f}^h - A^h\mathbf{v}^h)$	{compute \mathbf{f}^{2h} by restricting residual}
$\mathbf{v}^{2h} = \mathbf{0}$	
$\mathbf{v}^{2h} = V^{2h}(\mathbf{v}^{2h}, \mathbf{f}^{2h})$	{V-cycle on the coarser grid }
$\mathbf{v}^h = \mathbf{v}^h + I_{2h}^h\mathbf{v}^{2h}$	{correct \mathbf{v}^h by interpolating \mathbf{v}^{2h} }
$\mathbf{v}^h = \text{Relax}(\mathbf{v}^h, \mathbf{f}^h, \nu_2)$	{post-relax ν_2 times with modified \mathbf{v}^h }
end if	

Algorithm 2 $W^h(\mathbf{v}^h, \mathbf{f}^h)$	W-cycle Method
if (h =coarsest) then	
$\mathbf{v}^h = (A^h)^{-1}\mathbf{f}^h$	{solve $A^h\mathbf{u}^h = \mathbf{f}^h$ directly}
else	
$\mathbf{v}^h = \text{Relax}(\mathbf{v}^h, \mathbf{f}^h, \nu_1)$	{pre-relax ν_1 times with initial \mathbf{v}^h }
$\mathbf{f}^{2h} = I_h^{2h}(\mathbf{f}^h - A^h\mathbf{v}^h)$	{compute \mathbf{f}^{2h} by restricting residual}
$\mathbf{v}^{2h} = \mathbf{0}$	
$\mathbf{v}^{2h} = V^{2h}(\mathbf{v}^{2h}, \mathbf{f}^{2h})$ 2 times	{V-cycle on the coarser grid 2 times}
$\mathbf{v}^h = \mathbf{v}^h + I_{2h}^h\mathbf{v}^{2h}$	{correct \mathbf{v}^h by interpolating \mathbf{v}^{2h} }
$\mathbf{v}^h = \text{Relax}(\mathbf{v}^h, \mathbf{f}^h, \nu_2)$	{post-relax ν_2 times with modified \mathbf{v}^h }
end if	

Algorithm 3 $FMG^h(\mathbf{f}^h)$	Full Multigrid V-cycle Method
-----------------------------------	-------------------------------

if (h =coarsest) then	
$\mathbf{v}^h = (A^h)^{-1}\mathbf{f}^h$	{solve $A^h\mathbf{u}^h = \mathbf{f}^h$ directly}
else	
$\mathbf{f}^{2h} = I_h^{2h}(\mathbf{f}^h)$	{compute \mathbf{f}^{2h} by restricting \mathbf{f}^h }
$\mathbf{v}^{2h} = FMG^{2h}(\mathbf{f}^{2h})$	{FMG on the coarser grid }
$\mathbf{v}^h = I_{2h}^h\mathbf{v}^{2h}$	{compute \mathbf{v}^h by interpolating \mathbf{v}^{2h} }
$\mathbf{v}^h = V^h(\mathbf{v}^h, \mathbf{f}^h)$	{V-cycle with modified \mathbf{v}^h }
end if	

In this section, we have studied the elements of multigrid method. Although we don't introduce all the details of multigrid method, we have enough ability to solve Helmholtz equations. [1] has complete introduction of multigrid method.

In brief, multigrid method integrate some easy ideas and schemes. In fact, these ideas may have some individual defects. Multigrid method arrange them skillfully such that they can work together, and result in a very useful numerical method.

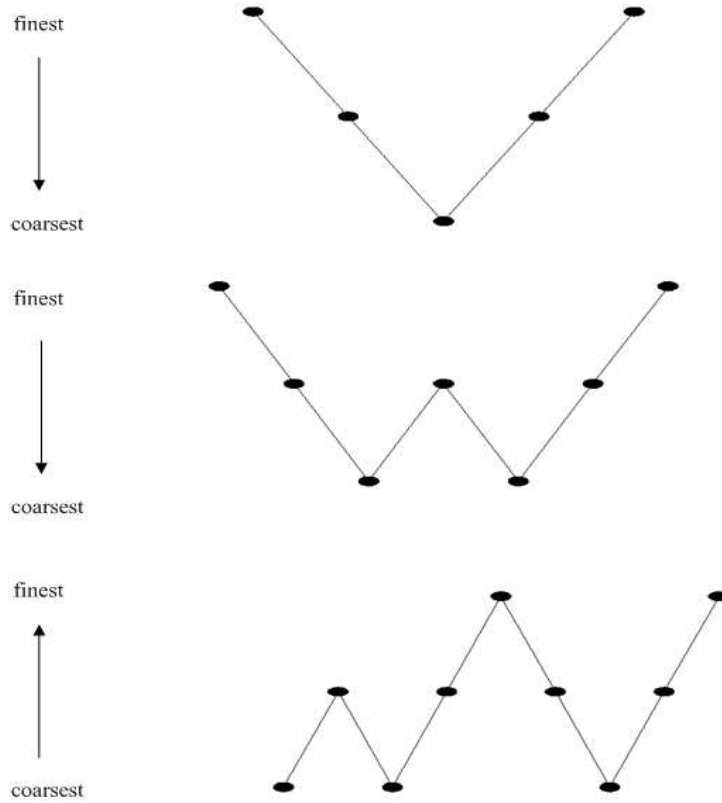


Figure 2: *Schedule of grids for V-cycle, W-cycle and FMG method from top to bottom.*

3 Preconditioned Conjugate Gradient Method

The conjugate gradient method is one of the popular methods for solving linear system $A\mathbf{x} = \mathbf{b}$. It is very suitable for large-scale sparse matrices arising from FD or FE approximation of boundary-value problems. When the $n \times n$ symmetric positive definite matrix has been preconditioned to make the calculations more effective, good results are obtained in only about \sqrt{n} steps.

In this section, we introduce roughly the conjugate gradient method, and its complete derivation can be found in [5].

First, we define the quadratic function

$$\phi(\mathbf{x}) = \frac{1}{2}\langle A\mathbf{x}, \mathbf{x} \rangle - \langle \mathbf{b}, \mathbf{x} \rangle = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T \mathbf{b},$$

and let

$$h(\alpha) = \phi(\mathbf{x} + \alpha\mathbf{s}).$$

We can prove that \mathbf{x}^* is a solution to $A\mathbf{x} = \mathbf{b}$ if and only if \mathbf{x}^* minimizes $\phi(\mathbf{x})$ [5]. We can

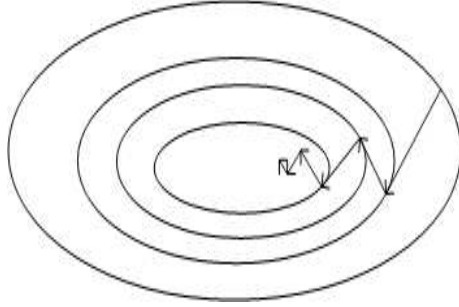


Figure 3: *Convergence of steepest descent.*

also find that $h(\alpha)$ has a minimal value when

$$\alpha = \frac{\langle \mathbf{s}, \mathbf{b} - A\mathbf{x} \rangle}{\langle \mathbf{s}, A\mathbf{s} \rangle}$$

Now, we denote \mathbf{x} an approximate solution to $A\mathbf{x}^* = \mathbf{b}$, and vector $\mathbf{s} \neq \mathbf{0}$ gives a search direction to improve the approximation. Let $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ be the residual vector associated with \mathbf{x} and

$$\alpha = \frac{\langle \mathbf{s}, \mathbf{b} - A\mathbf{x} \rangle}{\langle \mathbf{s}, A\mathbf{s} \rangle} = \frac{\langle \mathbf{s}, \mathbf{r} \rangle}{\langle \mathbf{s}, A\mathbf{s} \rangle}.$$

If $\mathbf{r} \neq \mathbf{0}$ and if \mathbf{s} and \mathbf{r} are not orthogonal, then $\phi(\mathbf{x} + \alpha\mathbf{s})$ is smaller than $\phi(\mathbf{x})$ and $\mathbf{x} + \alpha\mathbf{s}$ is presumably closer to \mathbf{x}^* than \mathbf{x} . This suggests the following method.

Let \mathbf{x}_0 be an initial approximation to \mathbf{x}^* , and let $\mathbf{s}_0 \neq \mathbf{0}$ be an initial search direction. For $k = 0, 1, 2, \dots$, we compute

$$\alpha_k = \frac{\langle \mathbf{s}_k, \mathbf{b} - A\mathbf{x}_k \rangle}{\langle \mathbf{s}_k, A\mathbf{s}_k \rangle},$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$$

and choose a new search direction \mathbf{s}_{k+1} . The object is to make this selection so that the sequence of approximations $\{\mathbf{x}_k\}$ converges rapidly to \mathbf{x}^* .

Our direct idea is using $-\nabla\phi(\mathbf{x})$ as a search direction because it is the direction of greatest decrease in the value of $\phi(\mathbf{x})$. And it is just the direction of the residual \mathbf{r} . The method that choose

$$\mathbf{s}_{k+1} = \mathbf{r}_{k+1} = \mathbf{b} - A\mathbf{x}_{k+1}$$

is called *the method of steepest descent*. Unfortunately, the convergence rate of steepest descent is often very poor owing to repeated searches in the same directions (see Fig. 3). Therefore, the search direction requires some modifications not negative gradient direction any more. *The conjugate gradient method* chooses search directions $\{\mathbf{s}_0, \dots, \mathbf{s}_{n-1}\}$ so that they are *A-orthogonal* set; that is,

$$\langle \mathbf{s}_i, A\mathbf{s}_j \rangle = 0, \quad \text{if } i \neq j.$$

Now, we use \mathbf{r}_{k+1} to generate \mathbf{s}_{k+1} by setting

$$\mathbf{s}_{k+1} = \mathbf{r}_{k+1} + \beta_{k+1}\mathbf{s}_k.$$

We want to choose β_{k+1} so that

$$\langle \mathbf{s}_k, A\mathbf{s}_{k+1} \rangle = 0.$$

We can obtain

$$\beta_{k+1} = -\frac{\langle \mathbf{s}_k, A\mathbf{r}_{k+1} \rangle}{\langle \mathbf{s}_k, A\mathbf{s}_k \rangle}.$$

It can be shown that with this choice of β_{k+1} , $\{\mathbf{s}_0, \dots, \mathbf{s}_{k+1}\}$ is an A-orthogonal set. Then we can simplify

$$\alpha_k = \frac{\langle \mathbf{r}_k, \mathbf{r}_k \rangle}{\langle \mathbf{s}_k, A\mathbf{s}_k \rangle}.$$

Thus,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{s}_k.$$

To compute \mathbf{r}_{k+1} , we multiply by A and subtract \mathbf{b} to obtain

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{s}_k.$$

Then we can change

$$\beta_{k+1} = \frac{\langle \mathbf{r}_{k+1}, \mathbf{r}_{k+1} \rangle}{\langle \mathbf{r}_k, \mathbf{r}_k \rangle}.$$

Above derivation is the process of conjugate gradient method. The algorithm of conjugate gradient method is given in algorithm 4.

Algorithm 4 Conjugate Gradient Method

```

 $\mathbf{x}_0$  =initial guess
 $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ 
 $\mathbf{s}_0 = \mathbf{r}_0$ 
for  $k = 0, 1, 2, \dots$ 
     $\alpha_k = \mathbf{r}_k^T \mathbf{r}_k / \mathbf{s}_k^T A\mathbf{s}_k$            {compute search parameter}
     $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$            {update solution}
     $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{s}_k$          {compute new residual}
     $\beta_{k+1} = \mathbf{r}_{k+1}^T \mathbf{r}_{k+1} / \mathbf{r}_k^T \mathbf{r}_k$ 
     $\mathbf{s}_{k+1} = \mathbf{r}_{k+1} + \beta_{k+1} \mathbf{s}_k$      {compute new search direction}
end

```

Although conjugate gradient method has a significant improvement over steepest descent, it can still converge very slowly if the matrix A is ill-condition. The convergence of

conjugate gradient method can be accelerated by *preconditioning*. Preconditioning means that choose a matrix M for which systems of the form $M\mathbf{z} = \mathbf{y}$ are easily solved. And $M^{-1} \approx A^{-1}$ so that $M^{-1}A$ is relatively well-conditioned. In fact, we should apply conjugate gradient to $L^{-1}AL^{-T}$ instead of $M^{-1}A$ to preserve symmetry of matrix, where $M = LL^T$. However, the algorithm can be suitably rearrange so that only M is used and the corresponding matrix L is not required explicitly. The algorithm of preconditioned conjugate gradient method is given in algorithm 5.

Algorithm 5 Preconditioned Conjugate Gradient Method

```

 $\mathbf{x}_0$  =initial guess
 $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ 
 $\mathbf{s}_0 = M^{-1}\mathbf{r}_0$ 
for  $k = 0, 1, 2, \dots$ 
     $\alpha_k = \mathbf{r}_k^T M^{-1}\mathbf{r}_k / \mathbf{s}_k^T A\mathbf{s}_k$            {compute search parameter}
     $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$            {update solution}
     $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{s}_k$            {compute new residual}
     $\beta_{k+1} = \mathbf{r}_{k+1}^T M^{-1}\mathbf{r}_{k+1} / \mathbf{r}_k^T M^{-1}\mathbf{r}_k$ 
     $\mathbf{s}_{k+1} = M^{-1}\mathbf{r}_{k+1} + \beta_{k+1}\mathbf{s}_k$    {compute new search direction}
end

```

Choosing an appropriate preconditioner is very important. The choice of preconditioner depends on the usual trade-off between the gain in the convergence rate and the increased cost per iteration that results from applying the preconditioner. Many types of preconditioner can be find in [6]. We only introduce some of them.

- **Jacobi:** M is taken to be a diagonal matrix with diagonal entries equal to those of A . This is the easiest preconditioner since M^{-1} is also a diagonal matrix (can be regarded as vector when we compute) whose entries are reciprocal of the diagonal entries of A . Although it only increases less storage and cost per iteration, it need more number of necessary iterations than the following incomplete Cholesky factorization.
- **Incomplete Cholesky factorization:** Ideally, we can factor A into LL^T by Cholesky factorization, but this may incur unacceptable fill. We may instead compute an approximate factorization $A \approx \hat{L}\hat{L}^T$ that allows little or no fill (e. g. restricting the nonzero entries of \hat{L} to be in the same positions as those of the lower triangle of A) to save storage and CPU time, then use $M = \hat{L}\hat{L}^T$ as a preconditioner.
- **Multigrid preconditioner:** We choose directly $M = A$ and find $M^{-1}\mathbf{r}$ by multigrid method. This is so-called *multigrid conjugate gradient method* (MGCG). This

method is also researched widely in many papers. We will use MGCG in the later numerical experiments.

Up to now, We have introduced MG and PCG. These methods belong to iterative method. In fact, fast direct methods have been developed for solving general poisson equation [9]. The following Concus and Golub method make a little modification to Helmholtz equation so that fast direct method can be used.

4 Concus and Golub Method

Concus and Golub propose an iterative scheme which uses fast direct solvers for the repeated solution of a Helmholtz problem [7]. Suppose original equation is

$$\Psi u = -\Delta u + g(x, y)u = f(x, y).$$

Concus and Golub provide a approach to utilize a modified form of the iterative procedure

$$-\Delta u_{n+1} = -\Delta u_n - \tau(\Psi u_n - f),$$

where τ is a parameter. We use the shifted iteration

$$(-\Delta + K)u_{n+1} = (-\Delta + K)u_n - \tau(\Psi u_n - f).$$

The discrete form is given by

$$(-\Delta_h + KI)V_{n+1} = (-\Delta_h + KI)V_n - \tau[(-\Delta_h + G)V_n - F],$$

where K is a parameter, V is approximation to exact u , $-\Delta_h$ is a matrix from operator $-\Delta$ discretization with mesh space h , G is a diagonal matrix with elements $G_{ij} = g(ih, jh)$, F is a vector with elements $F_{ij} = f(ih, jh)$, and I is the identity matrix.

Now, we want to find spectral radius ρ for above iteration. We denote $M \equiv -\Delta_h + G$ operator, Φ is a vector, and ν_m, ν_M are the minimum and maximum eigenvalues of eigenvalue problem $M\Phi = \nu(-\Delta_h + KI)\Phi$. To obtain it,

$$\rho(I - \tau[-\Delta_h + KI]^{-1}M) = \max\{|1 - \tau\nu_m|, |1 - \tau\nu_M|\}.$$

To estimate ν_m and ν_M , we use the Rayleigh quotient for ν ,

$$\frac{\Phi^T M \Phi}{\Phi^T (-\Delta_h + KI) \Phi} = 1 + \frac{\Phi^T (G - KI) \Phi}{\Phi^T (-\Delta_h + KI) \Phi}$$

then

$$1 + \min\left\{\frac{\beta - K}{\lambda_m + K}, \frac{\beta - K}{\lambda_M + K}\right\} \leq \nu_m \leq \nu_M \leq 1 + \max\left\{\frac{B - K}{\lambda_m + K}, \frac{B - K}{\lambda_M + K}\right\},$$

where β and B are the minimal and maximal of function $g(x, y)$, λ_m and λ_M are the smallest and largest eigenvalue of $-\Delta_h$, and K is between β and B .

In this, the Lemma in [7] help us finding the optimal choice of τ

$$\tau = \frac{2}{\nu_m + \nu_M} = \frac{2(\lambda_m + K)}{(2\lambda_m + B + \beta)}$$

therefore

$$\rho \leq \frac{B - \beta}{2\lambda_m + B + \beta}.$$

We will use this formula to observe the convergence rate of Concus and Golub method in the later numerical examples.

In an attempt to make the operator $-\Delta + K$ on the left-hand side agree closely with Ψ , we usually set K the so-called min-max value,

$$\frac{1}{2}(\min(g(x, y)) + \max(g(x, y)))$$

so the optimal $\tau = 1$. And then, the shifted iteration formula can be simplified

$$(-\Delta_h + KI)V_{n+1} = (KI - G)V_n + F.$$

According to above discrete form, we give an initial guess first, and compute right-hand side. Now, we can solve ‘‘Poisson equation’’ by FPS to gain the approximation and then update right-hand side again. Repeat this action until it reaches stopping criterion.

In fact, K can also be optimized for higher rates of convergence. The efficiency of Concus and Golub’s method can be increased by extending its formulation to accommodate the use of a parameter K which is a one-dimensional function instead of a constant [7]. Unfortunately, for a special solution, that would have required a prohibitively large computational time since there is no fast Helmholtz solver available for variable coefficient.

In the past it has been demonstrated that the number of necessary iterations can vary dramatically, depending on the function $g(x, y)$ which has a critical role on the rate of convergence. The smoother $g(x, y)$ is, the faster rate of convergence. We will give a example in the next section.

5 Numerical Examples

We will solve Helmholtz equations with Dirichlet boundary condition

$$\begin{aligned} -\Delta u(x, y) + g(x, y)u &= f(x, y), & (x, y) \in \Omega \\ u(x, y) &= p(x, y), & (x, y) \in \partial\Omega \end{aligned} \quad (3)$$

on domain $\Omega = [-1, 1] \times [-1, 1]$.

First, we use *five-point finite difference formula* to discretize e.q. (3). Assume domain is partitioned into $n \times n$ subdomain by introducing the grid points

$$x_i = ih, \quad y_j = jh, \quad 0 \leq i, j \leq n$$

, where $h = \frac{2}{n}$ is the uniform mesh size.

We denote

- v_{ij} : as an approximation to the exact solution $u(x_i, y_j)$
- F_{ij} : as the value of $f(x, y)$ at (x_i, y_j)
- G_{ij} : as the value of $g(x, y)$ at (x_i, y_j)
- P_{ij} : as the value of $p(x, y)$ at (x_i, y_j) .

Then, we can obtain discrete component form

$$\frac{-1}{h^2}(v_{i-1,j} + v_{i+1,j} + v_{i,j-1} + v_{i,j+1} - 4v_{ij}) + G_{ij}v_{ij} = F_{ij},$$

and we can use iterative method to improve approximation v_{ij} according to

$$v_{ij} = \frac{1}{4 + h^2G_{ij}}(v_{i-1,j} + v_{i+1,j} + v_{i,j-1} + v_{i,j+1} + h^2F_{ij}),$$

for $i = 1, \dots, n-1, \quad j = 1, \dots, n-1$
with

$$v_{0j} = P_{0j}, \quad v_{nj} = P_{nj}, \quad \text{for } j = 0, \dots, n$$

and

$$v_{i0} = P_{i0}, \quad v_{in} = P_{in}, \quad \text{for } i = 0, \dots, n.$$

We may also represent system in matrix form as $A\mathbf{v} = \mathbf{b}$.
 A is an $(n-1) \times (n-1)$ block matrix, T is an $(n-1) \times (n-1)$ matrix and I is the identity matrix of order $n-1$.

$$A = \frac{1}{h^2} \begin{bmatrix} T_1 & I & & & & \\ I & T_2 & I & & & \\ & \ddots & \ddots & \ddots & & \\ & & I & T_{n-2} & I & \\ & & & I & T_{n-1} & \end{bmatrix},$$

$$T_j = \begin{bmatrix} 4 + h^2G_{1j} & -1 & & & & \\ -1 & 4 + h^2G_{2j} & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 4 + h^2G_{(n-2),j} & -1 & \\ & & & -1 & 4 + h^2G_{(n-1),j} & \end{bmatrix}.$$

And the unknown vector \mathbf{v} is defined by

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_{n-2} \\ \mathbf{v}_{n-1} \end{bmatrix}, \quad \mathbf{v}_j = \begin{bmatrix} v_{1j} \\ v_{2j} \\ \vdots \\ v_{(n-2),j} \\ v_{(n-1),j} \end{bmatrix}.$$

And the right-hand side vector \mathbf{b} is defined by

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{n-2} \\ \mathbf{b}_{n-1} \end{bmatrix}, \quad \mathbf{b}_j = \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{(n-2),j} \\ b_{(n-1),j} \end{bmatrix}.$$

Then set

$$\begin{aligned} b_{ij} &= F_{ij}, \quad \text{for } i, j = 2, \dots, n-2 \\ b_{1j} &= F_{1j} + P_{0j}/h^2, \quad b_{n-1,j} = F_{n-1,j} + P_{nj}/h^2, \quad \text{for } j = 2, \dots, n-2 \\ b_{i1} &= F_{i1} + P_{i0}/h^2, \quad b_{i,n-1} = F_{i,n-1} + P_{i,n}/h^2, \quad \text{for } i = 2, \dots, n-2 \end{aligned}$$

and

$$\begin{aligned} b_{1,1} &= F_{1,1} + P_{0,1}/h^2 + P_{1,0}/h^2, \quad b_{n-1,1} = F_{n-1,1} + P_{n-1,0}/h^2 + P_{n,1}/h^2 \\ b_{1,n-1} &= F_{1,n-1} + P_{0,n-1}/h^2 + P_{1,n}/h^2, \quad b_{n-1,n-1} = F_{n-1,n-1} + P_{n-1,n}/h^2 + P_{n,n-1}/h^2. \end{aligned}$$

Our stopping criterion is $\frac{\|\mathbf{r}\|_h}{\|\mathbf{b}\|_h} < 10^{-8}$, where $\mathbf{r} = \mathbf{b} - A\mathbf{v}$ is residual, and $\|\bullet\|_h$ means discrete L^2 norm.

We use the following methods for solving later examples.

- **Concus and Golub method:** Choose K equal to min-max value, and use “GEN-BUN” routine which is in “FISHPACK” as our fast poisson solver.
- **V-cycle method:** Use RBGS relaxation with 2 sweeps before the correction step and 1 sweep after the correction step. And use full weighting restriction and linear interpolation operators.
- **ICCG method:** Use Incomplete Cholesky factorization as a preconditioner of conjugate gradient method.
- **MGCG method:** Use V-cycle method as a preconditioner of conjugate gradient method.

We choose two coefficient functions

$$g_s(x, y) = -\frac{\sin(\pi(x + y))}{\frac{3}{2} + \sin(\pi(x + y))}$$

and

$$g_o(x, y) = -\frac{80 \sin(2\pi(x + y))}{\frac{3}{2} + \sin(2\pi(x + y))}.$$

Clearly, $g_s(x, y)$ is a smoother function. The maximum of $g_s(x, y)$ is about equal to 2.0, and the minimum of $g_s(x, y)$ is about equal to -4.0 on the domain $[-1, 1] \times [-1, 1]$. On the contrary, $g_o(x, y)$ is a large amplitude function. The maximum and minimum of $g_o(x, y)$ are 160 and -32 respectively.

We also choose two functions

$$u_s(x, y) = \sin(\pi x) + \cos(\pi y)$$

and

$$u_o(x, y) = (y^2 - 1)(y - 1)(e^{5y} \sin(4\pi x) + e^{-5y} \cos(4\pi x))$$

as our exact solutions. The function $u_s(x, y)$ is smoother than $u_o(x, y)$. This also means that the approximation of $u_s(x, y)$ is more accurate than the approximation of $u_o(x, y)$ under the same conditions.

Now, we use these functions $g_s(x, y)$, $g_o(x, y)$, $u_s(x, y)$ and $u_o(x, y)$ to result four examples, and show numerical results including CPU time (sec.) and necessary iterations to achieve stopping criterion on different grid numbers $N = 128^2, 256^2, 512^2, 1024^2$.

Example 1.1: $g_s(x, y)$ $u_o(x, y)$

Iterations	Concus and Golub	ICCG	MGCG	V-cycle
$N = 128^2$	5	117	7	7
$N = 256^2$	5	218	7	7
$N = 512^2$	5	428	7	6
$N = 1024^2$	5	796	7	6

CPU time	Concus and Golub	ICCG	MGCG	V-cycle
$N = 128^2$	0.3	0.7	0.6	0.6
$N = 256^2$	1.1	4.6	2.3	2.2
$N = 512^2$	4.8	34.0	9.2	7.8
$N = 1024^2$	23.0	264.0	36.6	31.0

In this example, we can find that ICCG is not a good method since it need longer CPU time to achieve stopping criterion. So incomplete Cholesky factorization is not a good preconditioner for conjugate gradient method here. We can try to use *modified incomplete*

Cholesky factorization (MIC) for construction of the preconditioning matrix, see [6]. We also find that the number of iterations of ICCG increases when N becomes larger.

The other methods like Concus and Golub, MGCG, V-cycle converge with very few iterations. And their number of iterations is almost independent of N . Therefore, the CPU time needed for these methods reach stopping criterion is much less than ICCG when N is very large.

Up to now, Concus and Golub method seem to converge rapidly. It is even better than multigrid method. Let us continue seeing the next example.

Example 1.2: $g_o(x, y) u_o(x, y)$

Iterations	Concus and Golub	ICCG	MGCG	V-cycle
$N = 128^2$	271	144	9	11
$N = 256^2$	267	282	9	10
$N = 512^2$	266	533	9	9
$N = 1024^2$	266	1006	9	8

CPU time	Concus and Golub	ICCG	MGCG	V-cycle
$N = 128^2$	9.0	0.73	0.7	0.8
$N = 256^2$	36.0	5.2	2.8	3.0
$N = 512^2$	158.0	39.0	11.6	10.8
$N = 1024^2$	784.0	305.0	46.0	40.0

This example uses a high amplitude function $g_o(x, y)$ as coefficient. Clearly, the iterations and CPU time of all methods increase. But we note that Concus and Golub method results in very very slow convergence. Its CPU time needed is even longer than ICCG. This phenomenon can be explained by below inequality introduced in the last section.

$$\rho \leq \frac{M - m}{2\lambda + M + m} < 1,$$

where M and m denote the maximum and minimum of $g(x, y)$ respectively. λ is the smallest eigenvalue of the discrete matrix of Laplace operator, and ρ is the spectral radius of iterative matrix of Concus and Golub method. In example 1.1

$$\frac{M - m}{2\lambda + M + m} \approx 0.08,$$

but in example 1.2

$$\frac{M - m}{2\lambda + M + m} \approx 0.93.$$

That is why Concus and Golub method need so many iterations to reach stopping criterion here. In next two examples 2.1,2.2 we replace $u_o(x, y)$ in examples 1.1,1.2 with $u_s(x, y)$.

Example 2.1: $g_s(x, y) u_s(x, y)$

Iterations	Concus and Golub	ICCG	MGCG	V-cycle
$N = 128^2$	9	124	7	7
$N = 256^2$	9	232	7	6
$N = 512^2$	9	442	7	6
$N = 1024^2$	9	869	6	6

CPU time	Concus and Golub	ICCG	MGCG	V-cycle
$N = 128^2$	0.3	0.6	0.5	0.5
$N = 256^2$	1.3	4.1	1.9	1.6
$N = 512^2$	5.6	30.6	7.8	6.4
$N = 1024^2$	28.3	257.0	27.3	25.6

Example 2.2: $g_o(x, y)$ $u_s(x, y)$

Iterations	Concus and Golub	ICCG	MGCG	V-cycle
$N = 128^2$	394	144	9	12
$N = 256^2$	393	282	9	10
$N = 512^2$	392	554	9	10
$N = 1024^2$	392	1078	9	8

CPU time	Concus and Golub	ICCG	MGCG	V-cycle
$N = 128^2$	12.0	0.63	0.6	0.8
$N = 256^2$	53.0	4.9	2.6	2.6
$N = 512^2$	228.0	38.0	10.2	10.6
$N = 1024^2$	1153.0	317.0	41.0	35.0

After observing results of these two examples, we get the same conclusion discussed before. But we can find another phenomenon by comparing with example 1.1 and example 1.2. The number of iterations and CPU time needed of Concus and Golub method and ICCG increase, but MGCG and V-cycle almost remain the same number of iterations even if $u_s(x, y)$ may cause smoother error in these examples. That is because multigrid method can reduce smooth error efficiently.

In this thesis, we do not compare MGCG and V-cycle (ordinary multigrid) methods. Which is the better method? What is the difference of behaviors between the MCCG and multigrid methods? We can find answer in [10]. In [10], it gives a special example. The matrix produced by the discretization of this example has scattered eigenvalues. This scattered eigenvalues prevent the ordinary multigrid method from converging rapidly. We can observe the eigenvalue's distribution after multigrid preconditioning from [10]. Almost all eigenvalues are clustered, and a few eigenvalues are scattered. The conjugate gradient method hides the defect of the multigrid method. Therefore, the MGCG method is superior to the multigrid in this situation.

6 Conclusion

After testing above examples, we know that multigrid method suits to be used for solving general nonseparable elliptic equations. Since no matter $g(x, y)$ is smooth or oscillatory and smooth or oscillatory error resulted, multigrid method always converges in very few iterations. This means that multigrid method convergence stably. We also know an important advantage of multigrid method from our numerical examples. That is the number of necessary iterations is independent of mesh size.

There is another advantage of multigrid method. It is easy to be parallelized. We do not introduce the parallelism of multigrid method in this thesis. The content of parallel multigrid can be studied from [11]. These above reasons explain why we use multigrid method for solving Helmholtz equation here.



References

- [1] William L. Briggs Van Emden Henson Steve F. McCormick, *A multigrid tutorial 2nd ed.* 2000.
- [2] A. BRANDT, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, GMD-Studien Nr. 85, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Bonn, 1984.
- [3] Michael T. Heath, *Scientific computing :an introductory survey 2nd ed.* 2002.
- [4] W. M. Pickering and P. J. Harley , *Numerical solution of non-separable elliptic equations by iterative application of FFT methods.* Int. Jnl. Computer Math. 55, 211-222, 1995.
- [5] Richard L. Burden, J. Douglas Faires, *Numerical analysis 7th ed.* 2001.
- [6] Yousef Saad, *Iterative methods for sparse linear systems*, 1996.
- [7] Paul Concus and Gene H. Golub, *Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations*, SIAM J. Numer. Anal. 10 pp. 1103-1120, 1973.
- [8] Dimitropoulos, C.D. and Beris, A.N. *An efficient and robust spectral solver for non-separable elliptic equations* J. Comp. Phys. 133 pp. 186-191, 1997.
- [9] B. Buzbee, G. Golub, and C. Nielson, *On direct methods for solving Poissons equation*, SIAM J. Numer. Anal. 7 pp. 627-656, 1970.
- [10] Osamu Tatebe, *The multigrid preconditioned conjugate gradient method*
- [11] Jim E. Jones, *A Parallel Multigrid tutorial.*