Chapter 5. Methods for Solving Elliptic Equations

References: Tannehill et al Section 4.3.
Fulton et al (1986 MWR).

Recommended reading:
Chapter 7, Numerical Methods for Engineering Application. J. H. Perziger, 2nd Ed.

5.1. Introduction

In fluid dynamics, elliptic problems often arise. Two examples are given below.

Example 1:

When solving the incompressible equations, we have

$$\frac{\partial \vec{V}}{\partial t} + \vec{V} \cdot \nabla \vec{V} = -\nabla\left(\frac{p}{\rho_0}\right) + \vec{F} + K\nabla^2\vec{V} \qquad (1a)$$

where $\vec{F}$ is a body force and K is a constant viscosity.

$$\nabla \cdot \vec{V} = 0 \qquad (1b)$$

$$\frac{d\theta}{dt} = 0. \qquad (1c)$$

The pressure has to be obtained from a diagnostic elliptic equation, obtained by taking $\nabla\cdot(\ )$ of the momentum equation (1):

$$\nabla^2 p = -\nabla\cdot\left(\rho_0\vec{V}\cdot\nabla\vec{V}\right) + \nabla\cdot\left(\rho_0\vec{F}\right) + \nabla\cdot\left(\rho_0 K\nabla^2\vec{V}\right) \qquad (2)$$

Equation (2) is a Possion equation, which is a special case of elliptic equations. It is a boundary condition problem (no time integration therefore no I.C. needed).

Example 2:

When we solve 2-D equations in vorticity-streamfunction form, we define

$$u \equiv \frac{\partial\psi}{\partial y}, \quad v \equiv -\frac{\partial\psi}{\partial x}$$

where ψ is the streamfunction, and vorticity $\zeta = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} = \nabla^2\psi$. From the equations in example 1, we can obtain prognostic vorticity equation (assuming $\nabla\cdot\vec{V} = 0$):

$$\frac{d\zeta}{dt} = K\nabla^2\zeta \qquad (3)$$

and diagnostic elliptic equation

$$\nabla^2\psi = \zeta\,. \qquad (4)$$

Equation (2) is integrated forward for one time step first then (3) is solved to obtain ψ which is then used to calculated u and v. The new u and v are then used to advect the vorticity field to obtain ζ at the next time level.

Other examples of elliptic equations include the omega equation for diagnosing vertical motion in quasi-geostrophic synoptic dynamics, and the equation for diagnosing potential function from divergence. Due to the non-time matching nature of elliptic equations, we often refer to them as diagnostic equations.

5.2. Methods for Solving Elliptic Equations

5.2.1. Introduction

Many methods exist, some of them are available as 'canned' programs in standard libraries, such as IMSL. Commonly used methods can be divided into the following classes:

a. Iterative (relaxation) methods
b. ADI method
c. Multi-grid methods
d. Direct Methods
    - FFT
    - Gaussian elimination or block cyclic reduction
e. Preconditioned conjugate residual method (e.g, Skamarock et al 1997, MWR)
f. many more

We will only look at the first three.

Consider a simple form of an elliptic, the Possion equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \rho(x,y) \quad 0 \le x \le a,\ 0 \le y \le b. \tag{5}$$

<u>Boundary conditions</u> are crucial in elliptic problems (elliptic problems are boundary conditions problems in nature). As discussed previously, the B.C. can be written in a general form:

$$\alpha u_B + \beta \left(\frac{\partial u}{\partial n} \right)_B = \gamma \tag{6}$$

where α, β are constants, $\alpha^2 + \beta^2 \neq 0$. Depending on the value of α and β, we have Neumann, Dirichlet and Robin boundary conditions.

One of the most common forms of discretization for elliptic problems is the so-called 5-point formula which is based on second-order centered difference:

$$\delta_{xx} u + \delta_{yy} u = \rho. \tag{7}$$

Let the B.C be $u_B^n = f_{i,j}^n$.

Rearranging terms, our F.D. equation can be written as

$$u_{i,j} = c_x (u_{i+1,j} + u_{i-1,j}) + c_y (u_{i,j+1} + u_{i,j-1}) - C\rho_{i,j} \tag{8}$$

$$c_x = \frac{\Delta y^2}{2(\Delta x^2 + \Delta y^2)},\ c_y = \frac{\Delta x^2}{2(\Delta x^2 + \Delta y^2)},\ C = \frac{\Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)}.$$

We can also write this system in a matrix form as

$$[A]\,\vec{U} = \vec{B} \tag{9}$$

which can be solved using one of the standard methods for linear system of equations. However, because $[A]$ is a sparse matrix, it's generally more efficient to use methods designed for solving such sparse matrix problems.

5.3.2. Iterative Methods

Strategy: Make an initial guess for $u_{i,j}$ at the zeroth iteration, and then apply a correction to the guess based on a local error. Once the difference between successive iterations falls below a set tolerance, the iteration is considered converged and the solution found.

Method I: Jacobi Iteration

Let $u^v_{i,j} = v^{th}$ = solution after v iterations. If we write

residual = $R^v = \nabla^2 u^v - \rho$

or $$R^v_{i,j} = c_x(u^v{}_{i+1,j} + u^v{}_{i-1,j}) + c_y(u^v{}_{i,j+1} + u^v{}_{i,j-1}) - C\rho_{i,j} - u^v{}_{i,j} \tag{10}$$

then the solution is converged if $R^v_{i,j} = 0$. In practice, we can't make *R* exactly zero using iteration methods, but we want $\| R^v_{i,j} \| < \varepsilon$. However, we can always define a new estimate of the solution $u^{v+1}_{i,j}$ such that the residual is <u>temporarily</u> reduced to zero at (*i, j*):

$$c_x (u^v{}_{i+1,j} + u^v{}_{i-1,j}) + c_y (u^v{}_{i,j+1} + u^v{}_{i,j-1}) - C\rho_{i,j} - u^{v+1}{}_{i,j} = 0. \tag{11}$$

To achieve that, we need, from (11) – (10):

$$u^{v+1}_{i,j} = u^v_{i,j} + R^v_{i,j} \tag{12}$$

where $R^v_{i,j}$ is computed from (10).

(12) is the formula to obtain solution of *u* through iterations. The iterations are needed because even though the residual *R* has been reduced to zero by (12) at point (*i*, *j*), when *u* at the neighboring points are adjusted in a similar manner, the residual at (*i*, *j*) is messed up (because the equations at each point are not independent)! If one recompute *R* using new values of *u* at the neighboring points, it is no longer zero. But, we do this iteratively, i.e., perform the adjustment repeated, hopefully we can eventually reduce the residual at all points to below certain threshold!

This method can be shown to converge all the time, but unfortunately at a slow rate.

<u>Method II: Guass-Seidel or Sequential Relaxation</u>

This method is the same as the Jacobi method, except for the definition of the residual which is now given by

$$R^{v}_{i,j} = c_x \left(u^{v}_{i+1,j} + \enclose{circle}{u^{v+1}_{i-1,j}}\right) + c_y \left(u^{v}_{i,j+1} + \enclose{circle}{u^{v+1}_{i,j-1}}\right) - C\rho_{i,j} - u^{v}_{i,j}\,. \qquad (13)$$

The circled terms are different from the previous formula.

For a loop of the form:

```
DO j =2, ny-1
DO I =2, nx-1
```

this essentially means that the newest values of *u* are used in the calculation of *R*.

Implications:

- $R_{i,j}$ is calculated from the newest values of *u*, old values of *u* do not need to be stored – storage requirement is reduced.
- Because $R_{i,j}$ depends on the latest values of *u*, the algorithm is sequential and therefore cannot be vectorized.

Method III: Successive or Sequential Over-relaxation (SOR)

SOR is essentially the Gauss-Seidel method augmented by a relaxation parameter α:

$$u^{v+1}_{i,j} = u^{v}_{i,j} + \alpha R^{v}_{i,j}\,. \qquad (14)$$

$\alpha > 1$ is usually used to improve convergence.

How do we determine the optimal value of α?

It depends on the particular elliptic PDE being solved and its coefficients. Unfortunately, the convergence rate is very sensitive to the value of α.

Without going into the heavy theory behind, we simply note here that maximum rate of convergence is obtained from the smaller root of

$$\alpha_{opt}^2 t_m^2 - 16\alpha_{opt} + 16 = 0 \tag{15}$$

where α_{opt} = optimal over-relaxation parameter

$$t_m = \cos(\pi/M) + \cos(\pi/N) \tag{16}$$

where M and N are the number of grid intervals in the two directions. This formula says that the α is mostly dependent on the grid size.

Solving (15) →

$$\alpha_{opt} = \frac{8 - 4\sqrt{4 - t_m^2}}{t_m^2}. \tag{17}$$

If M and N are large, then

$$\alpha_{opt} \approx 2 - \frac{\pi}{2\sqrt{2}}\left(\frac{1}{M^2} + \frac{1}{N^2}\right)^{1/2}. \tag{18}$$

For a square domain (*M*=*N*)

M=*N*	α
10	1.84
100	1.98
1000	1.998

5.3.3. Alternate Direction Implicit (ADI) Method

We encountered this method when discussing methods for solving multi-dimensional heat transfer equations. With this method, we solve the elliptic equation one direction at a time, by doing the following:

$$c_x u_{i+1,j}^{v+1/2} - u_{i,j}^{v+1/2} + c_x u_{i-1,j}^{v+1/2} = C\rho_{i,j} - c_y(u^v{}_{i,j+1} + u^v{}_{i,j-1}) \tag{19a}$$

$$c_y u_{i,j+1}^{v+1} - u_{i,j}^{v+1} + c_y u_{i,j-1}^{v+1} = C\rho_{i,j} - c_y(u_{i-1,j}^{v+1/2} + u_{i+1,j}^{v+1/2}) \tag{19b}$$

Each of them is a tridiagonal system of equations. Repeat the steps until convergence.

Discussion – The memory and operation counts for a 2D problem on N x N grid (see Fulton et al MWR 1986).

Methods	Memory	Operation	Comments
Gaussian elimination	$O(N^4)$	$O(N^4)$	error accumulation problem
Jacobi and G-S	$O(N^2)$	$O(N^4)$	Flexible, easy to code
SOR	$O(N^2)$	$O(N^3)$	
ADI	$O(N^2)$	$O(N^2 \ln N)$	
Fast/Direct Solvers Cyclic reduction matrix decomposition or combination of two after using FFT		$O(N^2)$	Limited to simple problems with constant coefficients
Multi-grid method		$O(N^2)$	Suitable for more general problems

5.3.4. Multigrid Methods for Elliptic Equations

Multi-grid represents a particular strategy for elliptic problems, within which a variety of methods can be used. They were developed in the 1970's by Achi Brandt. Because of its flexibility and operational counts on the same order as the much more restrictive direct methods, it has become rather popular.

The methods work by approximating a problem on multiple overlapping grids with widely varying mesh sizes and cycling between these approximations, using relaxation to reduce the error on the scale of each grid. The efficiency is optimal – the operational count is proportionally to the number of unknowns.

Reading: Fulton et al. (1986, MWR).

Analysis of Error Reduction with Relaxation Methods

Let's analyze the error of G-S relaxation:

Assume $\Delta x=\Delta y$, $c_x = c_y= 1/4$, $C=\Delta x^2/4$, (13) becomes

$$\frac{1}{4}(u^{v}{}_{i+1,j}+u^{v+1}{}_{i-1,j}+u^{v}{}_{i,j+1}+u^{v+1}{}_{i,j-1})-u^{v}{}_{i,j}-\frac{\Delta x^2}{4}\rho_{i,j}=R^{v}_{i,j} \qquad (20)$$

$$u^{v+1}_{i,j}=u^{v}{}_{i,j}+R^{v}_{i,j} \qquad (21)$$

(20) - (21) gives

$$\frac{1}{4}(u^{v}{}_{i+1,j}+u^{v+1}{}_{i-1,j}+u^{v}{}_{i,j+1}+u^{v+1}{}_{i,j-1})-u^{v+1}{}_{i,j}-\frac{\Delta x^2}{4}\rho_{i,j}=0 \qquad (20a)$$

Assume v is the true (discrete) solution, the error $e = v - u$ and

$$\frac{1}{4}(v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1}) - v_{i,j} - \frac{\Delta x^2}{4}\rho_{i,j} = 0 \qquad (22)$$

(22) – (20a) →

$$e_{i,j}^{new} = \frac{1}{4}(e_{i+1,j} + e^{new}{}_{i-1,j} + e_{i,j+1} + e^{new}{}_{i,j-1}) \qquad (23)$$

We see that <u>the error at (i, j) is a simple average of the current errors at four surrounding points</u>, which suggests that small scale errors can be reduced by the averaging faster due to cancellation. Smooth error field is slow to reduce!

Let's quantify the result by using Fourier analysis. Let

$$e_{i,j} = A\exp[i(j\theta_1 + k\theta_2)] \qquad (24)$$

θ_1 and θ_2 are the discrete wave number in the two directions, and are integer multiple of $2\pi\Delta x$ (assuming $\Delta x=\Delta y$) between $-\pi$ and π.

Plug (24) into (23) →

$$A^{new} = \frac{1}{4}(e^{-i\theta_1}A^{new} + e^{i\theta_1}A + e^{-i\theta_2}A^{new} + e^{i\theta 2}A) \qquad (25)$$

Reduction in error after one iteration (remember the amplification factor?):

$$\mu(\theta) = \left|\frac{A^{new}}{A}\right| = \left|\frac{e^{i\theta_1} + e^{i\theta_2}}{4 - e^{-i\theta_1} - e^{-i\theta_2}}\right| \qquad (26)$$

Here, you want μ to be small – so that error is reduced faster!

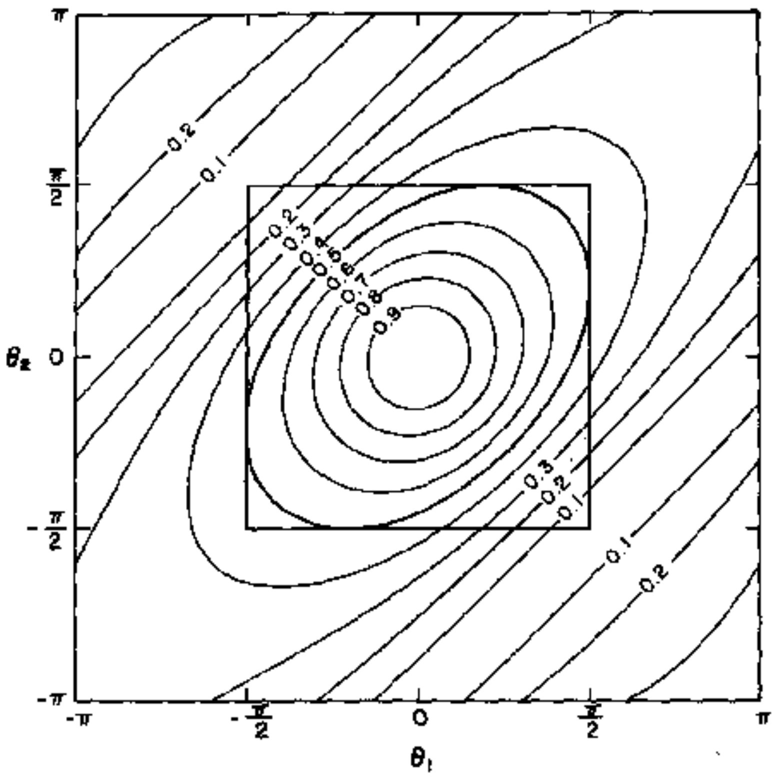

FIG. 2. Convergence factor $\mu(\theta)$ for the Poisson problem (2.3) using Gauss-Seidel relaxation with lexicographic ordering.

When $\theta_1 = \theta_2 = \pi$ (2Δ waves), $\mu = 1/3$

but $\mu = 1 - O(h^2)$ when θ_1 and $\theta_2 \sim O(h)$ where $h = \Delta x = \Delta y$.

We can see that convergence is slow for the lowest wave numbers, i.e., for the longest waves (in terms of the number of grid intervals).

The long waves can be represented rather accurately on a coarse resolution grid, on which the error can be reduced much faster – this is the idea behind the multi-grid method.

Before looking at the multi-grid method, let's first look at the two-grid method.

Two-grid method

Let's write the Possion equation as

$$Lv = f. \tag{27}$$

Let u be an approximate solution to $Lv=f \rightarrow$

$$Lu + r = f \tag{28}$$

where r is the residual.

Let v be the true solution to the finite difference form of equation, then error

$$e = v - u.$$

(27) – (28) →

$$Le = r. \tag{29}$$

On a grid with spacing h,

$$L^h e^h = r^h. \tag{30}$$

Goal: To find e so that the true solution $v = e + u$ can be obtained. We use relaxation plus iterations to reduce error e, so that eventually u approaches v.

After a number of iterations on grid h, e^h becomes smooth so that reduction in e becomes slow. The approximation of (30) on a coarser grid with spacing H (H>h) is

$$L^H e^H = I_h^H r^H \tag{31}$$

where I_h^H is an operator that transfers r from grid h to grid H (when $H = nh$, we transfer the values at overlapping points directly, and call the process injection).

Assume (31) is solved accurately on grid H, i.e., we found an e^H that satisfies (31), we interpolate it to h grid and adjust u so that

$$u_{new}^h = u^h + I_H^h e^H \tag{32}$$

where I_H^h is a transfer operator from H to h grid. This operator usually performs interpolation.

It can shown that with υ iterations on the fine grid and a *true* solution of e obatined on the coarse grid somehow, the error for ALL wave numbers are reduced by a factor of $(1/2)^{\upsilon}$ when G-S and $H/h=2$ are used!

When $\upsilon=3$, the error is reduced by a factor of 10 for ALL wave numbers!

Note that additional error is introduced during interpolation and injection. But usually, the error is relatively small.

Multi-grid method

Previous two-grid method did not say how to obtain the solution on the coarse grid. Since it has the same form as that on the fine grid, it can be solved the same way as the fine grid equations, by using a still coarser grid, e.g., $2H$. Continuing this process recursively until the coarsest grid is trivial to solve, we obtain a simple multigrid method.

Total number of grid points (storage) involved:

$$(1+1/4 +1/16+1/2^{N-1})\, N^2 < 4/3\ \ N^2 \sim O(N^2).$$

We have only shown a simplest procedure for solving an elliptic equation using multigrid methods. There are many variations and the method is very flexible and powerful. More details found in Fulton et al. (1986). Read the paper!