Elliptic Equations

Examples of Elliptic equations include

Laplace's Equation $\nabla^2\Phi = 0$

Poisson's Equation $\nabla^2\Phi = g$

Helmholz's Equation $\nabla^2\Phi + f\Phi = g$

For simplicity will look at 2-dimensional problems on a plane with coordinates x and y.

Boundary conditions:

Dirichlet: where Φ is specified at the boundary

Neumann where the gradient $\nabla_n\Phi$ is specified at the boundary

Mixed boundary conditions with a combination of the above.

Relaxation Methods

Consider the Diffusion Equation for *Φ(x,y)*

$$\frac{\partial \Phi}{\partial t} = \mu\left(\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2}\right)$$

In steady state, the solution will approach a solution

$$\lim_{t \to \infty} \Phi(x, y, t) = \Phi_s(x, y)$$

so we will have a solution to the equation

$$\frac{\partial^2 \Phi_s}{\partial x^2} + \frac{\partial^2 \Phi_s}{\partial y^2} = 0$$

algorithms that use a diffusion type of approach to get a steady state solution are called *relaxation methods*.

Jacobi Method

So starting with the equation

$$\frac{\partial \Phi}{\partial t} = \mu \left(\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} \right)$$

we discretize as before for the diffusion equation to get

$$\Phi_{i,j}^{n+1} = \Phi_{i,j}^{n} + \frac{\mu\tau}{h_x^2}\left(\Phi_{i+1,j}^{n} - 2\Phi_{i,j}^{n} + \Phi_{i-1,j}^{n}\right) + \frac{\mu\tau}{h_y^2}\left(\Phi_{i,j+1}^{n} - 2\Phi_{i,j}^{n} + \Phi_{i,j-1}^{n}\right)$$

where $\Phi_{i,j}^{n} \equiv \Phi(x_i, y_j, t_n)$ $\quad x_i = (i-1)h_x \quad y_i = (j-1)h_y$

$$t_n = (n-1)\tau$$

As before, the scheme is stable if $\frac{\mu\tau}{h_x^2} + \frac{\mu\tau}{h_y^2} \leq \frac{1}{2}$

Jacobi Method - 2

For simplicity, set $h = h_x = h_y$, so the stability condition becomes

$$\frac{\mu\tau}{h^2} \le \frac{1}{4}$$

If we use the maximum stability condition in the FTCS version of the diffusion equation we get

$$\Phi_{i,j}^{n+1} = \frac{1}{4}\left(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n} + \Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n}\right)$$

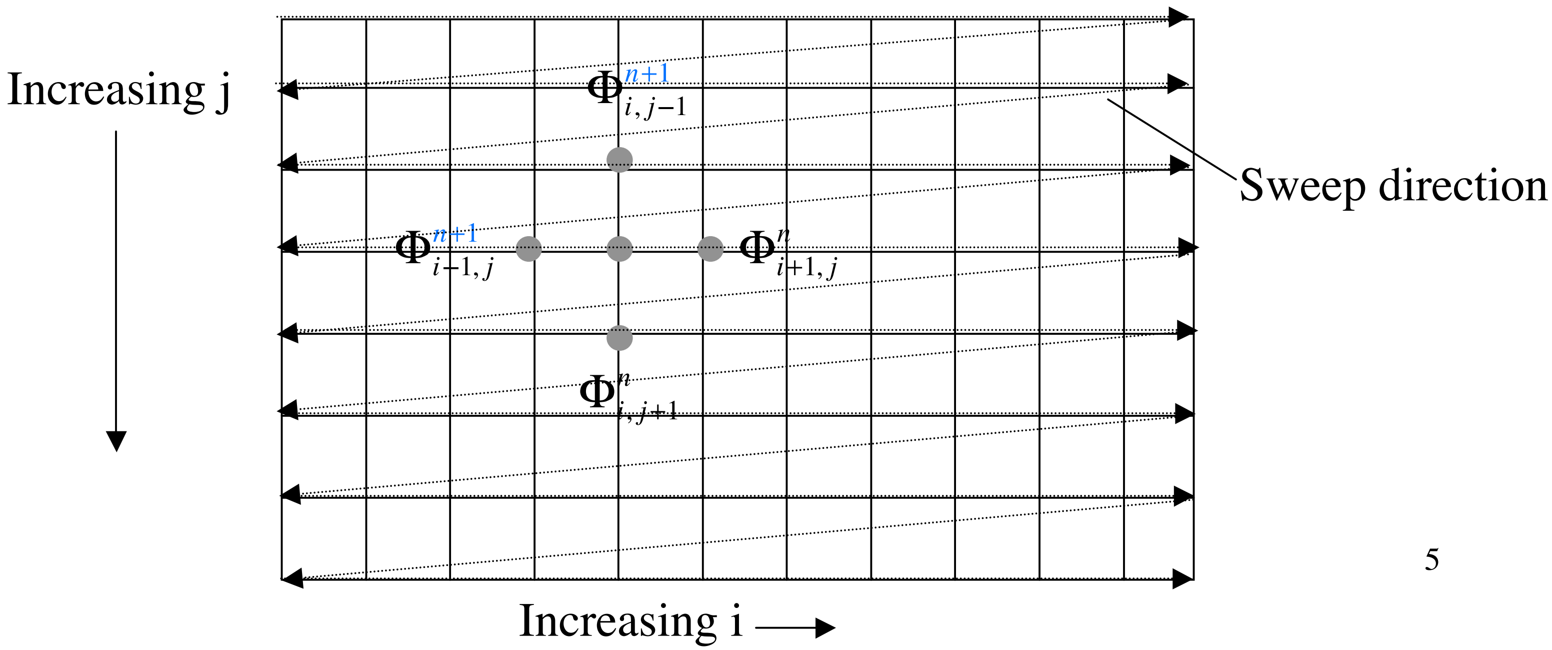
This is the *Jacobi method*.

Gauss-Seidel Method

A slight modification to the Jacobi Method uses updated values of Φ to iterate the solution

$$\Phi_{i,j}^{n+1} = \frac{1}{4}\left(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n+1} + \Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n+1}\right)$$

This method accelerates convergence and reduces memory usage since only one array of Φ is stored.



Simultaneous Overrelaxation (SOR)

To accelerate convergence even more one can use a parameter ω in the following way

$$\Phi_{i,j}^{n+1} = (1-\omega)\Phi_{i,j}^{n} + \frac{\omega}{4}\left(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n+1} + \Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n+1}\right)$$

The constant ω is called the overrelaxation parameter. Best convergence is found by optimal choices of ω.

- Using $\omega = 1$ is the same as doing Gauss-Seidel.
- Using $\omega < 1$ is called underrelaxation.
- Using $\omega > 2$ causes SOR to become unstable.
- Ideally ω_{opt} lies between 1 and 2.
- In general ω_{opt} is difficult to determine analytically, but a sophisticated program can find it.

SOR - optimal relaxation parameter

For a simple 2D problem size N_x x N_y it can be shown that

$$\omega_{opt} = \frac{2}{1+\sqrt{1-r^2}}$$

where

$$r = \frac{1}{2}\left(\cos\frac{\pi}{N_x} + \cos\frac{\pi}{N_y}\right)$$

for N_x & $N_y >> 1$

$$r \approx 1 - \frac{1}{4}\left(\frac{\pi}{N_x}\right)^2 - \frac{1}{4}\left(\frac{\pi}{N_y}\right)^2$$

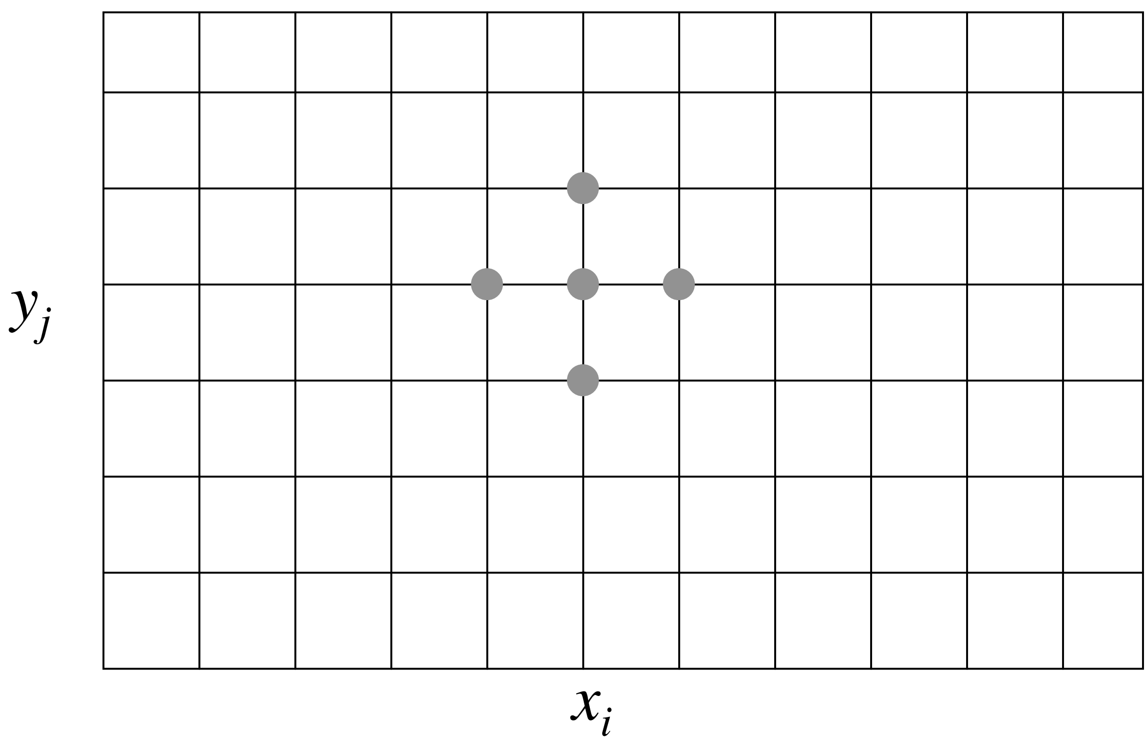
r is known as the *spectral radius*.

In most cases ω_{opt} has to be found empirically.

Matrix Methods

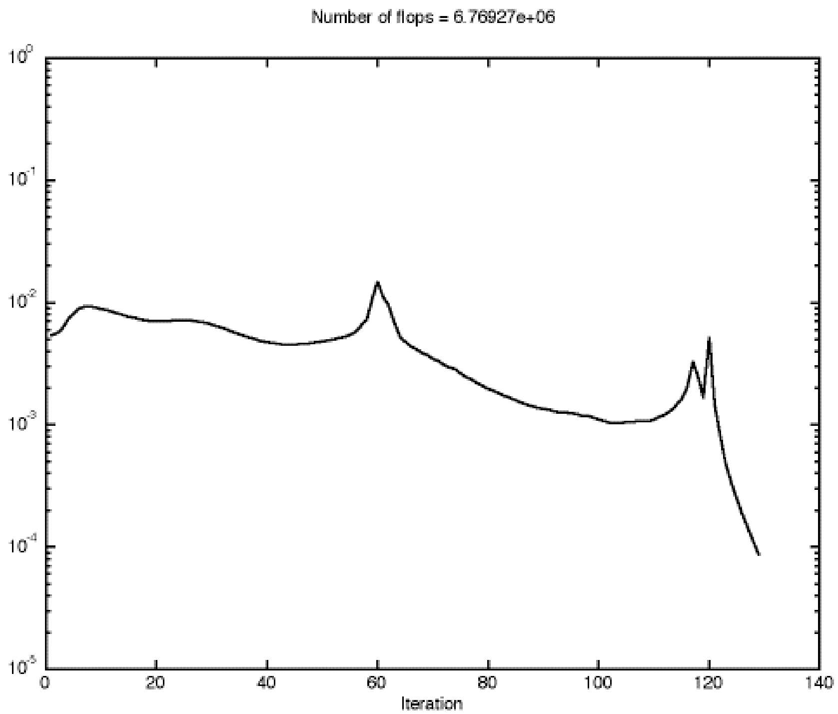
Recall that the discretization of Laplace's equation in 2D is given by

$$\nabla^2\Phi \approx \frac{1}{h_x^2}\left(\Phi_{i+1,j}^n - 2\Phi_{i,j}^n + \Phi_{i-1,j}^n\right) + \frac{1}{h_y^2}\left(\Phi_{i,j+1}^n - 2\Phi_{i,j}^n + \Phi_{i,j-1}^n\right) + O(h_x^2, h_y^2) = 0$$

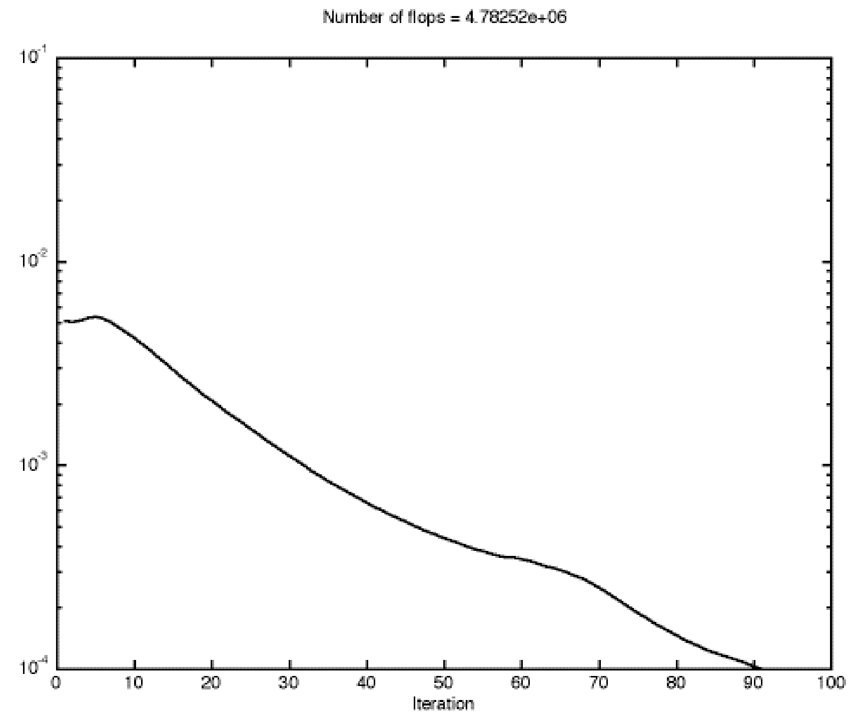


SOR Example

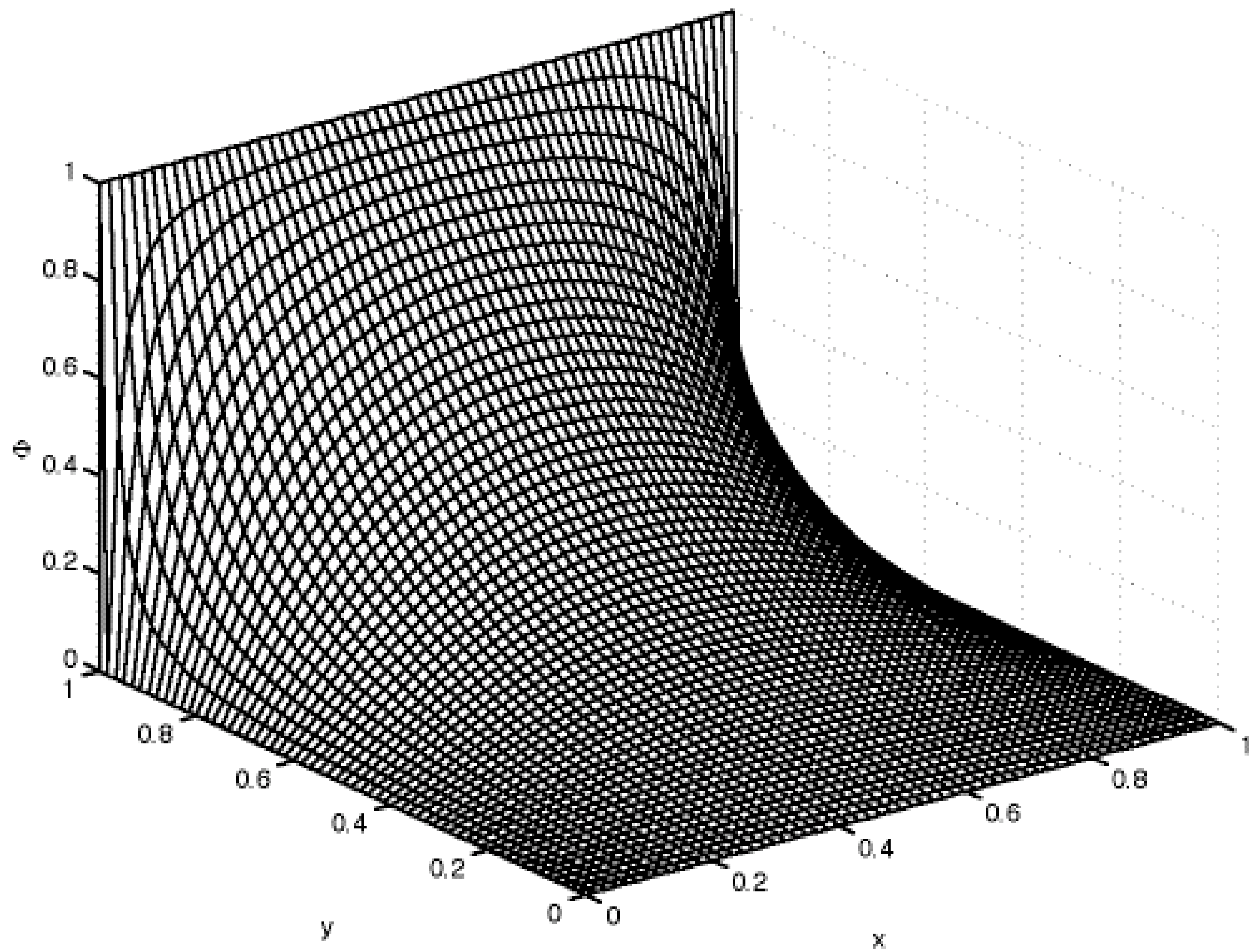
Program Relax



$\omega = 1.9$, $\omega_{opt} = 1.9$ N=61



$\omega = 1.8$, $\omega_{opt} = 1.9$ N=61



1
0.8
0.6
Φ
0.4
0.2
0
1
0.8
0.6
0.4
0.2
0
y
0
0.2
0.4
0.6
0.8
1
x

Matrix Methods - 2

Consider the following simple example, set $h_x = h_y = h$ and $N_x = N_y = N = 5$

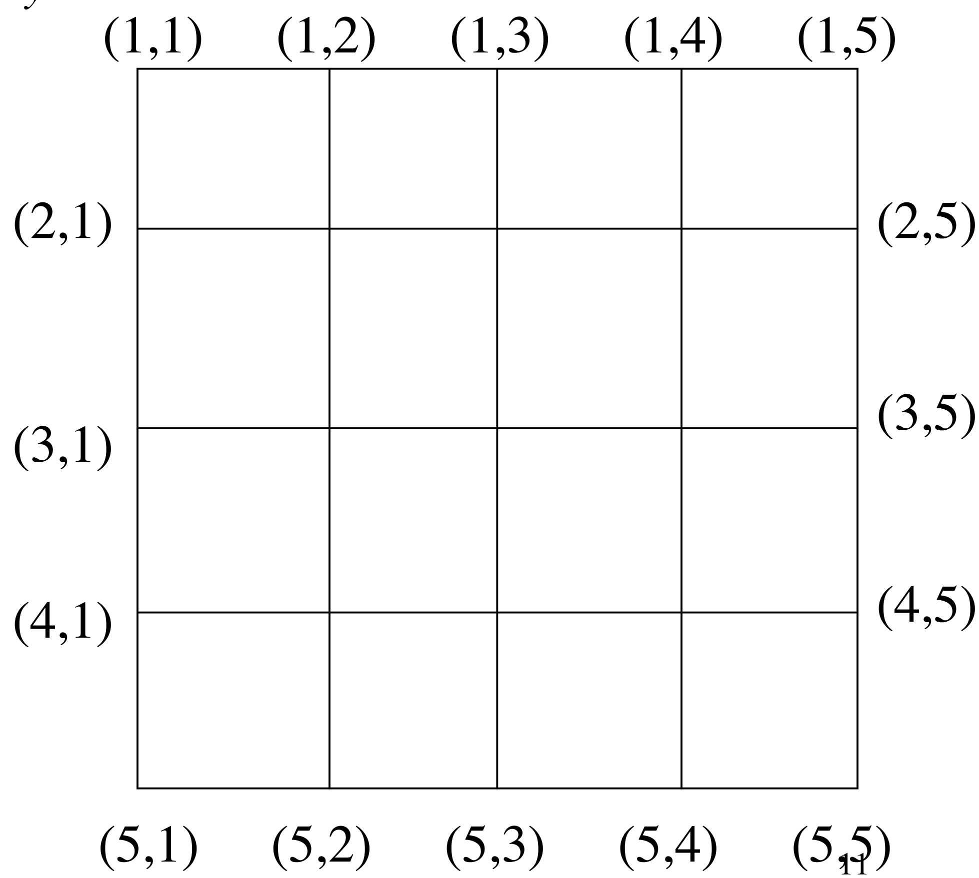
Define an index $n = j + (i-1) * N_y$

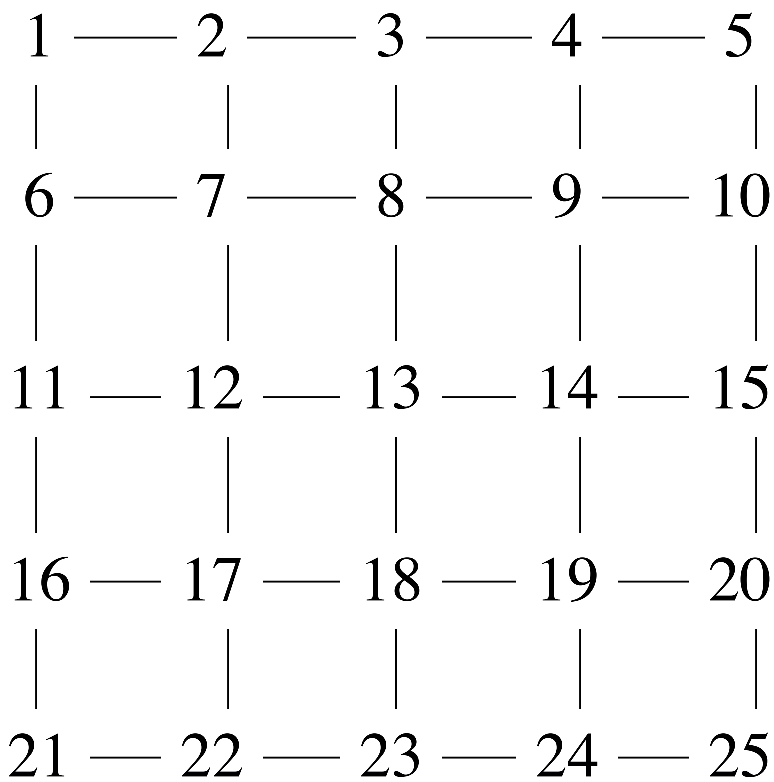
Let the solution Φ_n be defined at each of the vertices n

Note that given n you can infer (i,j)

$$i = \text{int}\left(\frac{n-1}{N_y}\right) + 1$$

$$j = n - (i-1)N_y$$





i	j	n
1	1	1
1	2	2
1	3	3
1	4	4
1	5	5
2	1	6
2	2	7
2	3	8
2	4	9
2	5	10
3	1	11
3	2	12
3	3	13
3	4	14
3	5	15
4	1	16
4	2	17
4	3	18
4	4	19
4	5	20
5	1	21
5	2	22
5	3	23
5	4	24
5	5	25

$$
\begin{bmatrix}
1 \\
&1 \\
&&1 \\
&&&1 \\
&&&&1 \\
&&&&&1 \\
&1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 \\
&&1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 \\
&&&1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 \\
&&&&&&&&&1 \\
&&&&&&&&&&1 \\
&&&&&&1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 \\
&&&&&&&1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 \\
&&&&&&&&1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 \\
&&&&&&&&&&&&&&1 \\
&&&&&&&&&&&&&&&1 \\
&&&&&&&&&&&1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 \\
&&&&&&&&&&&&1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 \\
&&&&&&&&&&&&&1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 \\
&&&&&&&&&&&&&&&&&&&1 \\
&&&&&&&&&&&&&&&&&&&&1 \\
&&&&&&&&&&&&&&&&&&&&&1 \\
&&&&&&&&&&&&&&&&&&&&&&1 \\
&&&&&&&&&&&&&&&&&&&&&&&1 \\
&&&&&&&&&&&&&&&&&&&&&&&&1
\end{bmatrix}
\begin{pmatrix}
\Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \\ \Phi_{10} \\ \Phi_{11} \\ \Phi_{12} \\ \Phi_{13} \\ \Phi_{14} \\ \Phi_{15} \\ \Phi_{16} \\ \Phi_{17} \\ \Phi_{18} \\ \Phi_{19} \\ \Phi_{20} \\ \Phi_{21} \\ \Phi_{22} \\ \Phi_{23} \\ \Phi_{24} \\ \Phi_{25}
\end{pmatrix}
=
\begin{pmatrix}
\Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ 0 \\ 0 \\ 0 \\ \Phi_{10} \\ \Phi_{11} \\ 0 \\ 0 \\ 0 \\ \Phi_{15} \\ \Phi_{16} \\ 0 \\ 0 \\ 0 \\ \Phi_{20} \\ \Phi_{21} \\ \Phi_{22} \\ \Phi_{23} \\ \Phi_{24} \\ \Phi_{25}
\end{pmatrix}
$$

Example of Neumann Boundary conditions

$$
\begin{bmatrix}
1 & & & & & & & & & & & & & & & & & & & & & & & & \\
& 1 & & & & & -1 & & & & & & & & & & & & & & & & & & \\
& & 1 & & & & & -1 & & & & & & & & & & & & & & & & & \\
& & & 1 & & & & & -1 & & & & & & & & & & & & & & & & \\
& & & & 1 & & & & & & & & & & & & & & & & & & & & \\
& & & & & 1 & -1 & & & & & & & & & & & & & & & & & & \\
& 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & & & & & & & & & & & & & \\
& & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & & & & & & & & & & & & \\
& & & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & & & & & & & & & & & \\
& & & & & & & & -1 & 1 & & & & & & & & & & & & & & & \\
& & & & & & & & & & 1 & -1 & & & & & & & & & & & & & \\
& & & & & & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & & & & & & & & \\
& & & & & & & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & & & & & & & \\
& & & & & & & & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & & & & & & \\
& & & & & & & & & & & & & -1 & 1 & & & & & & & & & & \\
& & & & & & & & & & & & & & & 1 & -1 & & & & & & & & \\
& & & & & & & & & & & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & & & \\
& & & & & & & & & & & & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & & \\
& & & & & & & & & & & & & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 0 & 1 & \\
& & & & & & & & & & & & & & & & & & -1 & 1 & & & & & \\
& & & & & & & & & & & & & & & & & & & & 1 & & & & \\
& & & & & & & & & & & & & & & & -1 & & & & & 1 & & & \\
& & & & & & & & & & & & & & & & & -1 & & & & & 1 & & \\
& & & & & & & & & & & & & & & & & & -1 & & & & & 1 & \\
& & & & & & & & & & & & & & & & & & & & & & & & 1
\end{bmatrix}
\begin{pmatrix}
\Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \\ \Phi_{10} \\ \Phi_{11} \\ \Phi_{12} \\ \Phi_{13} \\ \Phi_{14} \\ \Phi_{15} \\ \Phi_{16} \\ \Phi_{17} \\ \Phi_{18} \\ \Phi_{19} \\ \Phi_{20} \\ \Phi_{21} \\ \Phi_{22} \\ \Phi_{23} \\ \Phi_{24} \\ \Phi_{25}
\end{pmatrix}
=
\begin{pmatrix}
\Phi_1 \\ N_2 \\ N_3 \\ N_4 \\ \Phi_5 \\ N_6 \\ 0 \\ 0 \\ 0 \\ N_{10} \\ N_{11} \\ 0 \\ 0 \\ 0 \\ N_{15} \\ N_{16} \\ 0 \\ 0 \\ 0 \\ N_{20} \\ \Phi_{21} \\ N_{22} \\ N_{23} \\ N_{24} \\ \Phi_{25}
\end{pmatrix}
$$

Matrix Methods -2

This can be simplified on only include the unknown values:

$$
\begin{bmatrix}
-4 & 1 & & & & & & & \\
1 & -4 & 1 & 0 & 1 & & & & \\
 & 1 & -4 & 0 & 0 & 1 & & & \\
 & 0 & 1 & -4 & 1 & 0 & 1 & & \\
 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & \\
 & & 1 & 0 & 1 & -4 & 1 & 0 & 1 \\
 & & & 1 & 0 & 0 & -4 & 1 & 0 \\
 & & & & 1 & 0 & 1 & -4 & 1 \\
 & & & & & 1 & 0 & 1 & -4
\end{bmatrix}
\begin{pmatrix}
\Phi_7 \\ \Phi_8 \\ \Phi_9 \\ \Phi_{12} \\ \Phi_{13} \\ \Phi_{14} \\ \Phi_{17} \\ \Phi_{18} \\ \Phi_{19}
\end{pmatrix}
=
\begin{pmatrix}
-\Phi_1 - \Phi_2 \\ -\Phi_3 \\ -\Phi_4 - \Phi_{10} \\ -\Phi_{11} \\ 0 \\ -\Phi_{15} \\ -\Phi_{16} - \Phi_{22} \\ -\Phi_{23} \\ -\Phi_{20} - \Phi_{24}
\end{pmatrix}
$$

Multigrid Methods

The key to iterative solution to an elliptic equation is the propagation of the boundary condition to all of the solution domain.

- The number of iterations to do this is O(*N*)
- A coarser grid will propagate the solution more quickly, but the solution accuracy is poor

- The idea then behind multigrid methods is to use a number of different grids ranging from coarse to fine.
  - The numerical solution on the coarse grid can be computed quickly, but with low accuracy
  - Interpolate onto a finer grid
  - However, at one point of the the iteration one also goes from a fine grid to a coarser one, the coarser grid eliminates low frequency errors more efficiently.

Spectral Methods

Finite difference methods are not the only way to get numerical solutions to elliptic problems. If we wanted to solve the equation

$$\nabla^2 \Phi = g$$

And express the solution as a series

$$\Phi(x,y) = \sum_{k=1}^{K} a_k f_k(x,y) + T(x,y)$$

where first term on the rhs is the approximate solution and *T(x,y)* is the error term. For simplicity, let the trial functions be orthogonal

$$\int_0^L dx \int_0^L dy \;\; f_k(x,y) f_{k'}(x,y) = A_k \delta_{k,k'}$$

Spectral Methods - Fourier Galerkin Methods

Inserting into the differential equation

$$\nabla^2\left(\sum_{k=1}^{K} a_k f_k(x,y)\right) - g(x,y) = -\nabla^2 T(x,y) \equiv R(x,y) \qquad (*)$$

To get the solution, one needs to find the coefficients ak. We want to minimize the error. The *Galerkin* method imposes the condition $\int_0^L dx \int_0^L dy \;\; f_k(x,y) R(x,y) = 0$

For example, the solution to Poisson's equation (*f(x,y)*=-*ρ(x,y)/ε₀*) with Neumann boundary conditions $\nabla\Phi \cdot \hat{n} = 0$ at the boundary, the trial function would be

$$f_{m,n}(x,y) = \cos\left[\frac{(n-1)\pi x}{L}\right]\cos\left[\frac{(n-1)\pi y}{L}\right]$$

Spectral Methods - Fourier Galerkin Methods -2

Inserting into (*) we get

$$-\sum_{n=1}^{N}\sum_{m=1}^{M}\left[(m-1)^2+(n-1)^2\right]\frac{\pi^2}{L^2}f_{m,n}(x,y)+\frac{1}{\varepsilon_0}\rho(x,y)+R(x,y)$$

Applying orthogonality condition by applying

$$\int_0^L dx\int_0^L dy \quad f_{m',n'}(x,y)$$

to get the coefficients one gets

$$a_{n,m}=\frac{4}{\pi^2\varepsilon_0}\frac{1}{(m-1)^2+(n-1)^2}\frac{1}{(1+\delta_{m,1})(1+\delta_{n,1})}\times$$

$$\int_0^L dx\int_0^L dy \quad a_{n,m}\cos\left[\frac{(m-1)\pi x}{L}\right]\cos\left[\frac{(n-1)\pi y}{L}\right]$$

Finite Element Methods

The basic idea of Galerkin methods is 1 of several strategies that form the core of *Finite element methods*.

- The *Galerkin Method* requires that

$$\int_0^L dx \int_0^L dy \;\; w(x,y) R(x,y) = 0$$

where w(x,y) is some basis function

- *Collocation Methods* set $R(x_i,y_j)=0$. This gives a system of equations that have to be solved for the coefficients $a_{n,m}$
- Least squares methods, the square of the residual over the domain is set to a minimum, so one must solve an equation of the form

$$\frac{\partial}{\partial a_{n.m}} \int_0^L dx \int_0^L dy \;\; R(x,y)^2 = 0$$

Multiple Fourier Transform Methods

Discretize the Poisson Equation as

$$\frac{1}{h^2}\left(\Phi_{i+1,j} + \Phi_{i-1,j} - 2\Phi_{i,j}\right) + \frac{1}{h^2}\left(\Phi_{i,j+1} + \Phi_{i,j-1} - 2\Phi_{i,j}\right) = -\frac{\rho_{i,j}}{\varepsilon_0}$$

And apply the 2D Fourier transform

$$F_{n+1,m+1} = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} \Phi_{i+1,j+1} e^{-\alpha im - \alpha jn} \quad \rho_{n+1,m+1} = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} R_{i+1,j+1} e^{-\alpha im - \alpha jn}$$

and the inverse Fourier transform

$$\Phi_{n+1,m+1} = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} F_{i+1,j+1} e^{\alpha im + \alpha jn} \quad R_{n+1,m+1} = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} \rho_{i+1,j+1} e^{\alpha im + \alpha jn}$$

Placing into the discretized equation we get

$$\left\{e^{-\alpha(m-1)} + e^{\alpha(m-1)} + e^{-\alpha(n-1)} + e^{\alpha(n-1)} - 4\right\} F_{m,n} = -\frac{h^2}{\varepsilon_0} R_{m,n}$$

Multiple Fourier Transform Methods -2

This gives a matrix equation for $F_{m,n}$ where

$$F_{m,n} = -\frac{h^2 / 2\varepsilon_0}{\cos\left[\frac{2\pi(m-1)}{M}\right]\cos\left[\frac{2\pi(m-1)}{N}\right] - 2} R_{m,n}$$

and the inverse Fourier transform gives the potential