Elliptic Equations

Examples of Elliptic equations include

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

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

Helmholtz's Equation $\nabla^2\Phi + \mathrm{f}\Phi = g$

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 – Derivation from the Diffusion equation

Consider the Diffusion Equation in 2D for $\Phi(x, y)$

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

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 - Derivation

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

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

Where

$$\Phi_{i,j}^{n} \equiv \Phi(x_i, y_j, t_n)$$

And $x_i = (i-1)h_x$, $y_j = (j-1)h_y$ and $t_n = (n-1)\tau$

Where h_x and h_y are the x and y grid spacing, respectively. The scheme is stable if

$$\mu\tau\left(\frac{1}{h_x^2} + \frac{1}{h_y^2}\right) \leq \frac{1}{2}$$

Jacobi Method - 2

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

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

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

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

This is the *Jacobi method*. It basically states that in order to satisfy Laplace's equation on an even spaced 2D grid, each grid point should be the average of the 4 points that surround it. It is an iterative scheme, where you have to sweep the entire grid multiple times until some convergence criteria has been met. For example, the solution stops changing to some specified tolerance.

Jacobi Method – An alternative view

The discretized version of Laplace's equation looks like

$$\frac{1}{h_x^2}(\Phi_{i+1,j} - 2\Phi_{i,j} + \Phi_{i-1,j}) + \frac{1}{h_y^2}(\Phi_{i,j+1} - 2\Phi_{i,j} + \Phi_{i,j-1}) = 0$$

Moving all the terms in that involves $\Phi_{i,j}$ out on one side and rewriting we get

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

And solving for $\Phi_{i,j}$

$$\Phi_{i,j} = \frac{\frac{1}{h_x^2}(\Phi_{i+1,j} + \Phi_{i-1,j}) + \frac{1}{h_y^2}(\Phi_{i,j+1} + \Phi_{i,j-1})}{\frac{2}{h_x^2} + \frac{2}{h_y^2}}$$

Jacobi Method – An alternative view

In other words

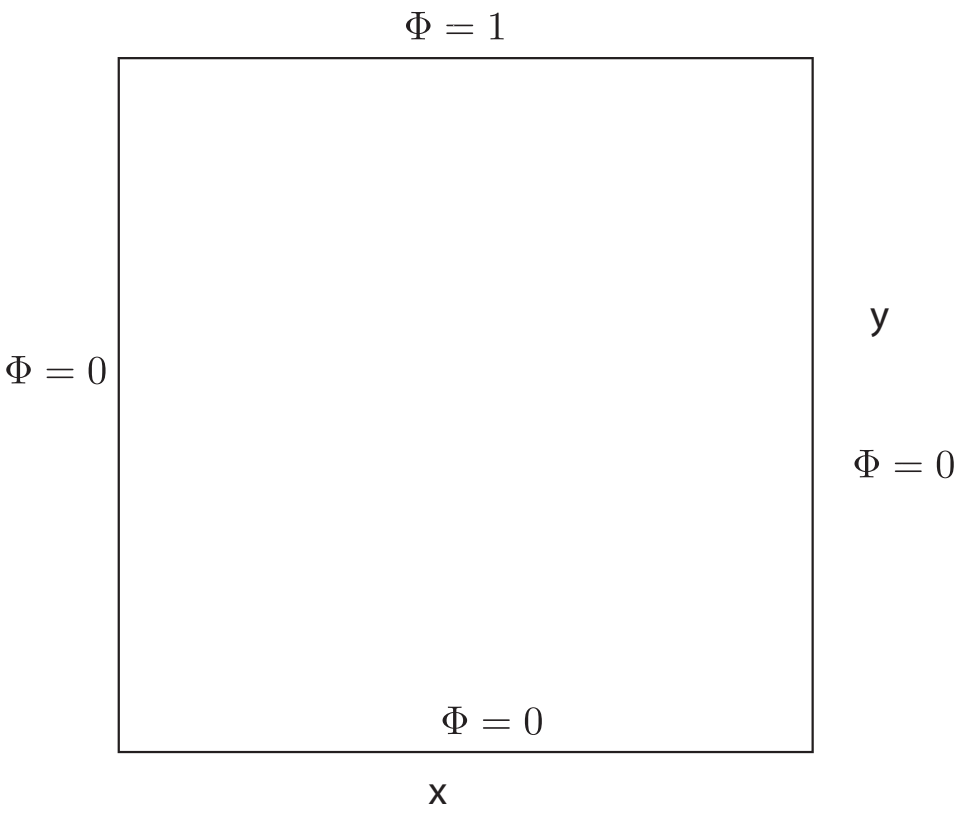
$$\Phi_{i,j}^{\mathrm{n}+1} = \frac{\frac{1}{h_x^2}(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{\mathrm{n}}) + \frac{1}{h_y^2}(\Phi_{i,j+1}^{\mathrm{n}} + \Phi_{i,j-1}^{\mathrm{n}})}{\frac{2}{h_x^2} + \frac{2}{h_y^2}}$$

This is a basic scheme for iterating. When the grid spacing is the same in both directions, i.e., $h_x = h_y = h$ we get

$$\Phi_{i,j}^{\mathrm{n}+1} = \frac{\Phi_{i+1,j}^{\mathrm{n}} + \Phi_{i-1,j}^{\mathrm{n}} + \Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n}}{4}$$

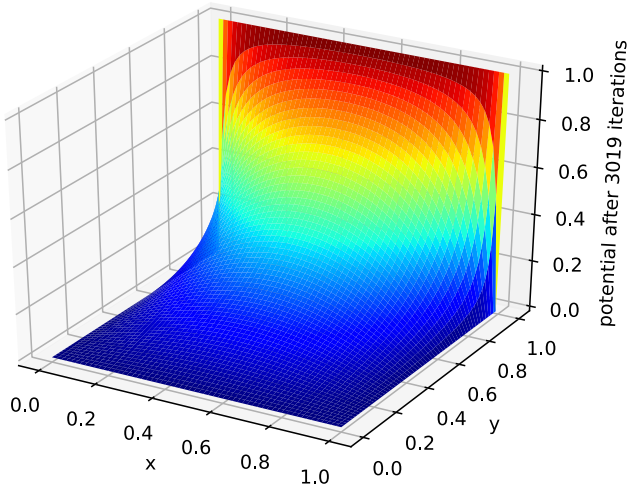
As before

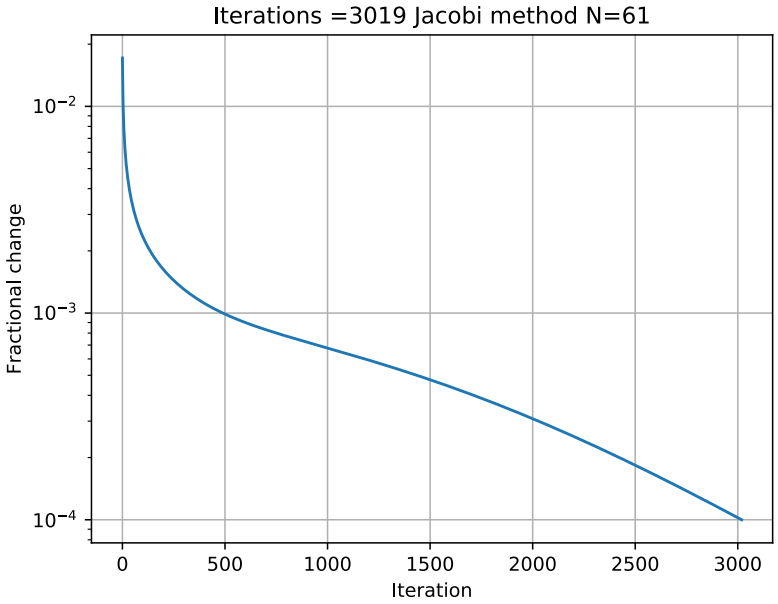
Potential Problem from Chapter 8



- Initial guess sets the potential to zero
- You can also use the first term in the analytic solution to set the guess – it is in the code but commented out
  - I will leave you to try that out

Jacobi Method



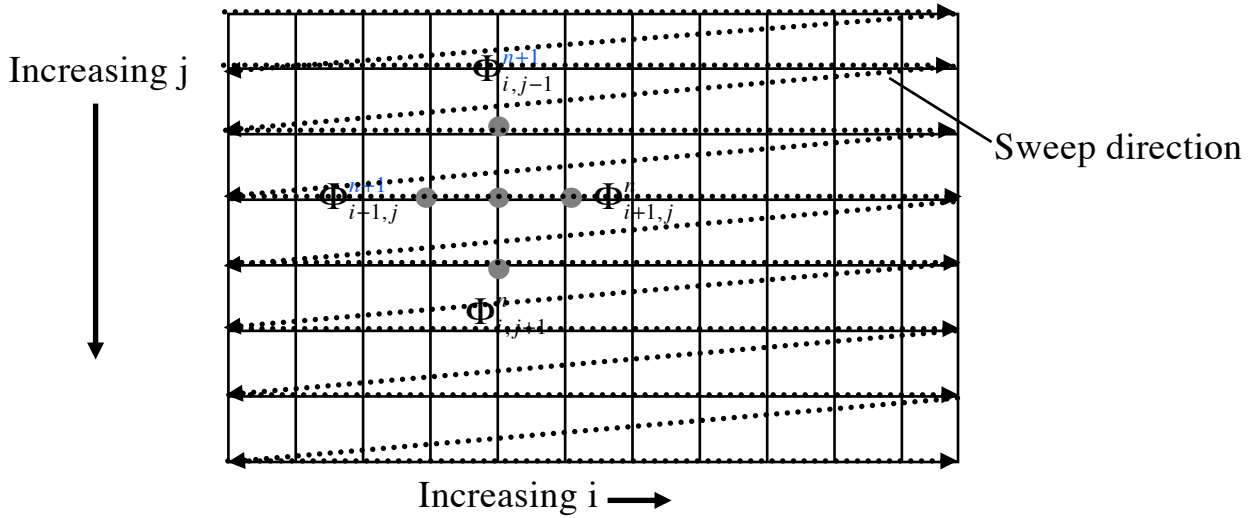


Gauss-Seidel Method

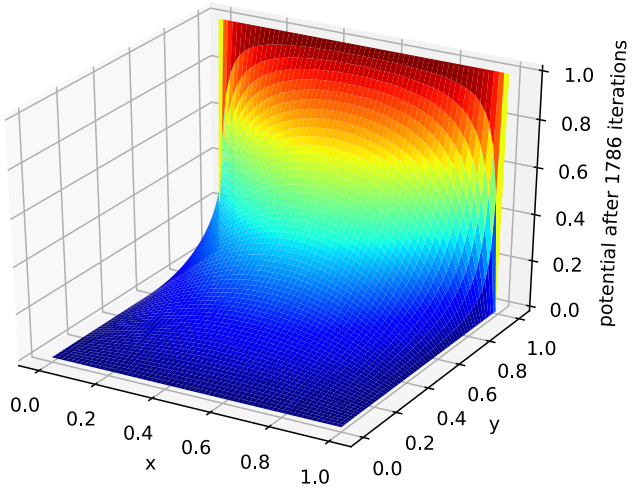
A slight modification to the Jacobi Method uses updated values of Φ to iterate the solution

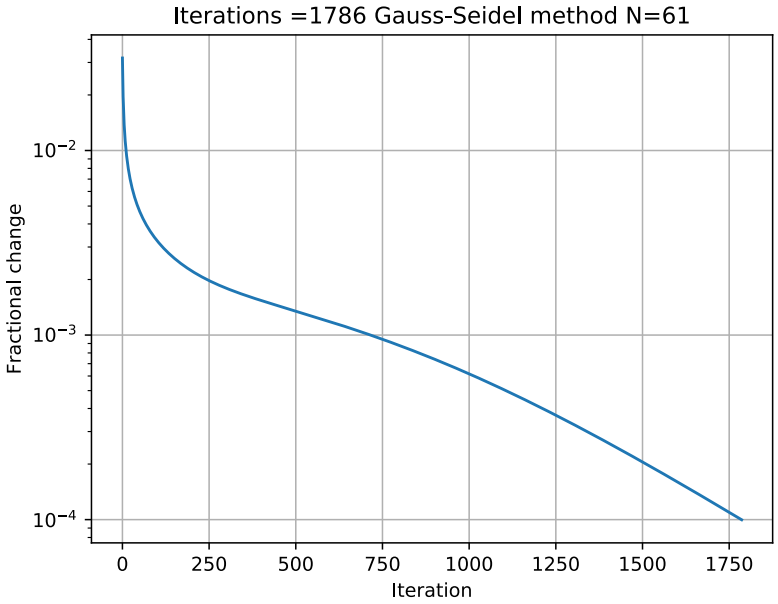
$$\Phi_{i,j}^{n+1} = \frac{1}{4}(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n+1} + \Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n+1})$$

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



Gaus Seidel Method





Simultaneous Overrelaxation (SOR)

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

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

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, which is sometimes useful for nonlinear Elliptic equations
- Using $\omega > 2$ causes SOR to become unstable
- Ideally ω_{opt} lies between 1 and 2, but not always
- In general ω_{opt} is difficult to determine analytically, but a program can find it empirically

SOR - optimal relaxation parameter

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

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

Where

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

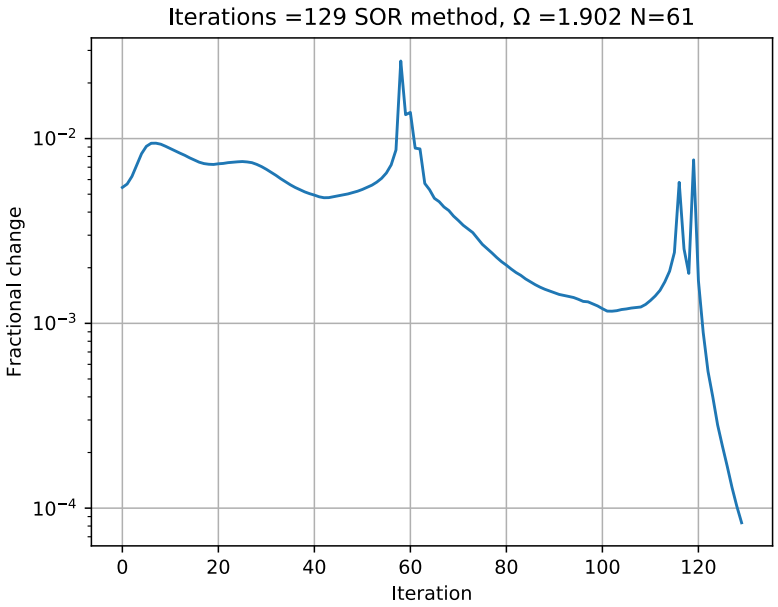
for N_x & N_y>>1

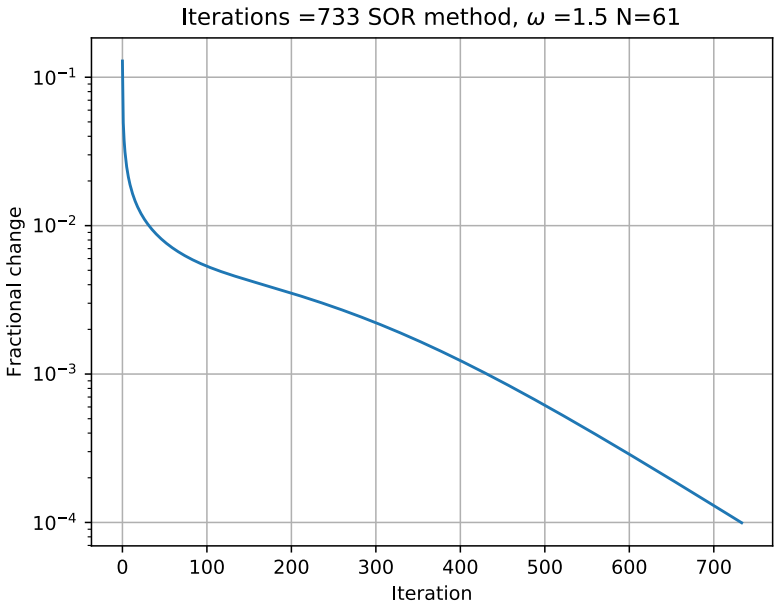
$$r \approx 1 - \frac{1}{4}\left(\frac{\pi}{N_x}\right)^2 - \frac{1}{4}\left(\frac{\pi}{N_y}\right)^2$$

r is known as the *spectral radius*.

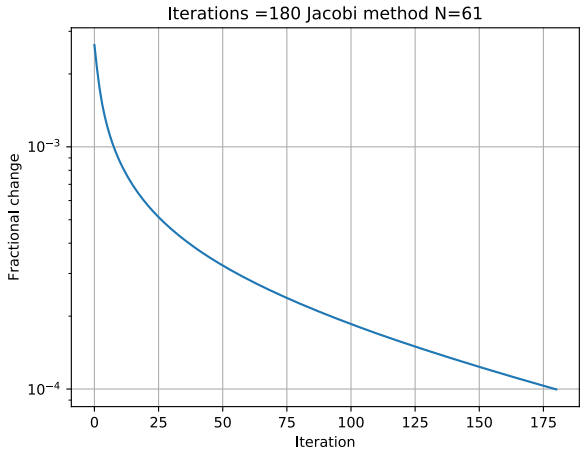
In many cases ω_{opt} has to be found empirically.

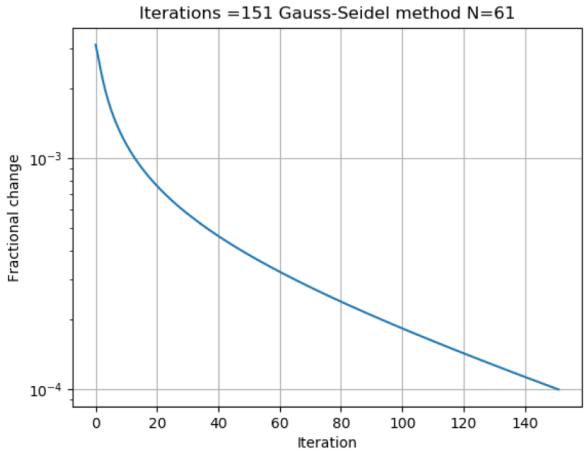
SOR method

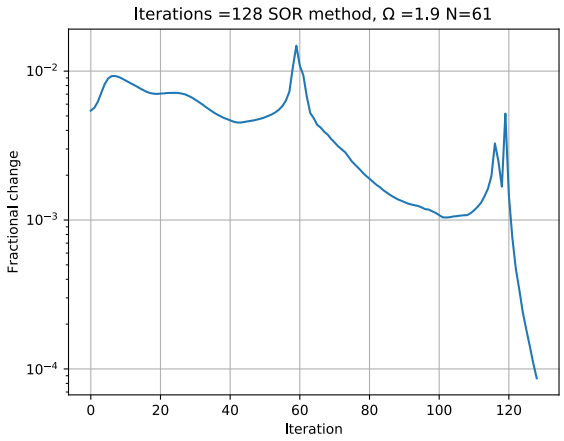

Iterations =129 SOR method, Ω =1.902 N=61
Fractional change
10^{-2}
10^{-3}
10^{-4}
0
20
40
60
80
100
120
Iteration


Iterations =733 SOR method, ω =1.5 N=61
Fractional change
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
100
200
300
400
500
600
700
Iteration

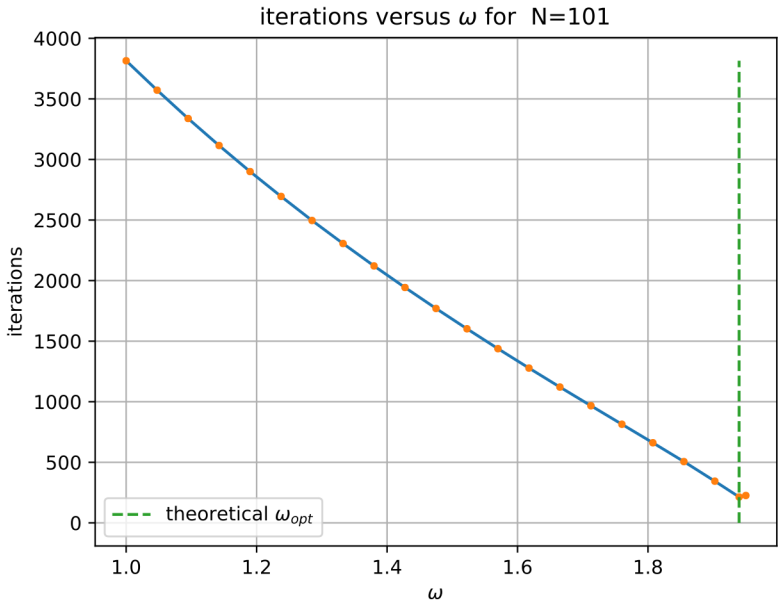
Using the first term of the analytic solution can dramatically speed up convergence



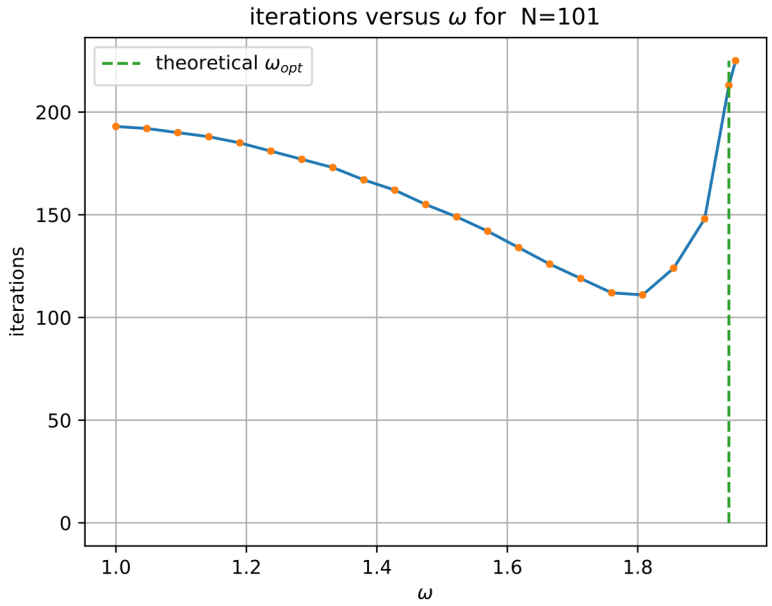




SOR - Optimal convergence estimate depends on initial guess

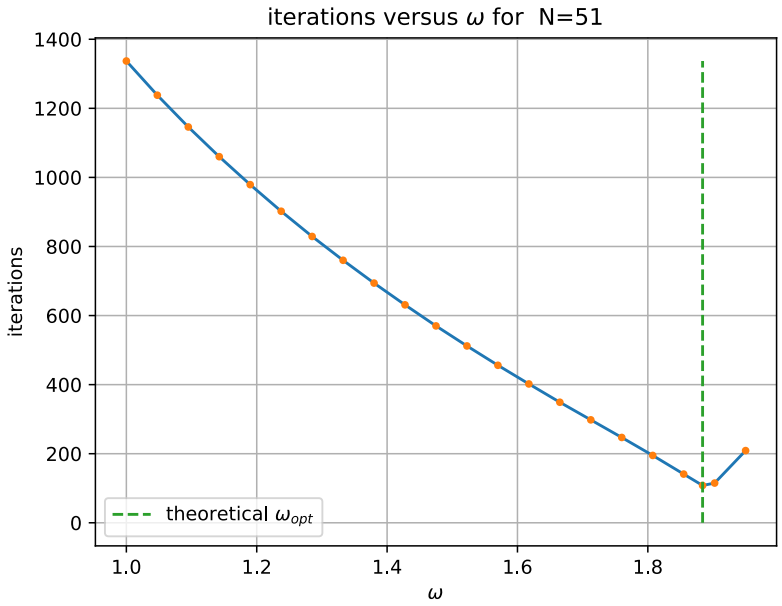


Initial guess, potential =1

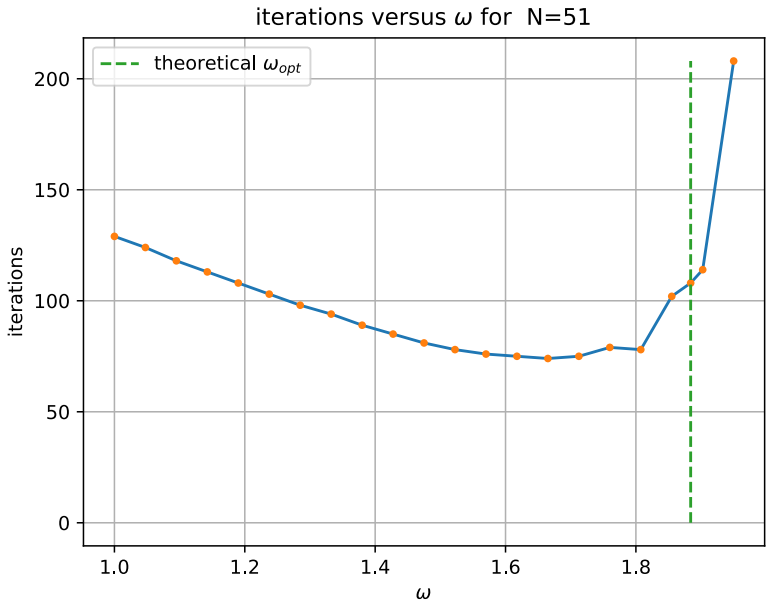


Initial guess uses part of analytic solution

It also depends on the size of the problem



Initial guess, potential =1



Initial guess uses part of analytic solution

Extra Notes

Analytic solution (from text)

The analytic solution can be found be separation of variables, so that the solution is split as $\Phi(x,y) = X(x)Y(y)$

$$\frac{1}{X(x)}\frac{d^2X(x)}{dx^2} = -k^2 = -\frac{1}{Y(y)}\frac{d^2Y(y)}{dy^2}$$

the general solution of which is

$$X(x) = C_s \sin kx + C_c \cos kx$$
$$Y(y) = C_s' \sinh ky + C_c' \cosh ky$$

Analytic solution (from text)-2

For the problem in `relax.m/.py` the boundary conditions are

$$\Phi(x = 0, y) = \Phi(x = L_x, y) = \Phi(x, y = 0) = 0$$
$$\Phi\left(x, y = L_y\right) = \Phi_0$$

the solution takes for series form

$$\Phi(x, y) = \sum_{n=1}^{\infty} c_n \sin\left(\frac{n\pi x}{L_x}\right) \sinh\left(\frac{n\pi y}{L_x}\right)$$

which satisfies the boundary condition where the potential is zero (i.e., the top set of boundary conditions above). For the bottom boundary condition, we have

$$\Phi_0 = \sum_{n=1}^{\infty} c_n \sin\left(\frac{n\pi x}{L_x}\right) \sinh\left(\frac{n\pi L_y}{L_x}\right)$$

which we can use to solve for c_n

Analytic solution (from text)-3

Next we integrate out the x-component by multiplying both sides by $\sin\left(\frac{m\pi}{L_x}\right)$ and integrating between 0 and

$$L_x \int_0^{L_x} \Phi_0 sin\left(\frac{m\pi x}{L_x}\right) dx = \sum_{n=1}^{\infty} c_n \sinh\left(\frac{n\pi L_y}{L_x}\right) \int_0^{L_x} sin\left(\frac{m\pi x}{L_x}\right) sin\left(\frac{n\pi x}{L_x}\right) dx$$

using the orthogonality properties of the sin function

$$\int_0^{L_x} \sin\left(\frac{m\pi x}{L_x}\right) dx = \begin{cases} \frac{2L_x}{m}, & m \text{ odd} \\ 0, & m \text{ even} \end{cases}$$

and

$$\int_0^{L_x} \sin\left(\frac{m\pi x}{L_x}\right) \sin\left(\frac{n\pi x}{L_x}\right) dx = \frac{L_x}{2} \delta_{m.n}$$

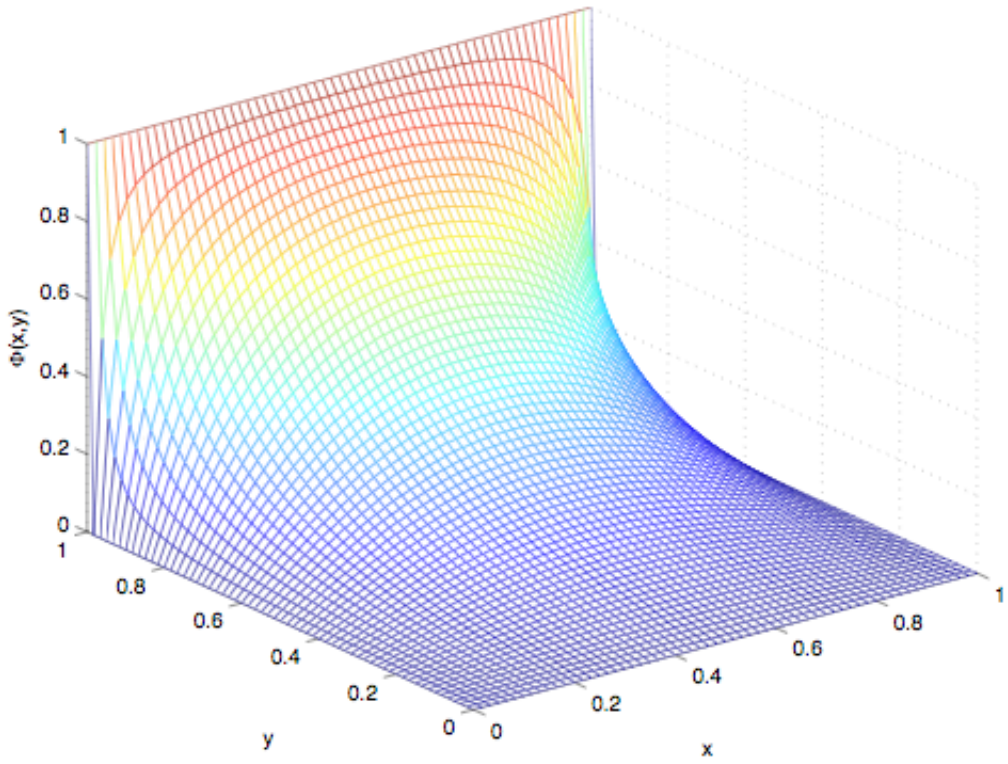
Analytic solution (from text)-4

Solving for c_n then yields

$$c_n = \frac{4\Phi_0}{\pi n \sinh\left(\frac{n\pi L_y}{L_x}\right)} \text{ n odd}$$

So we have the solution as

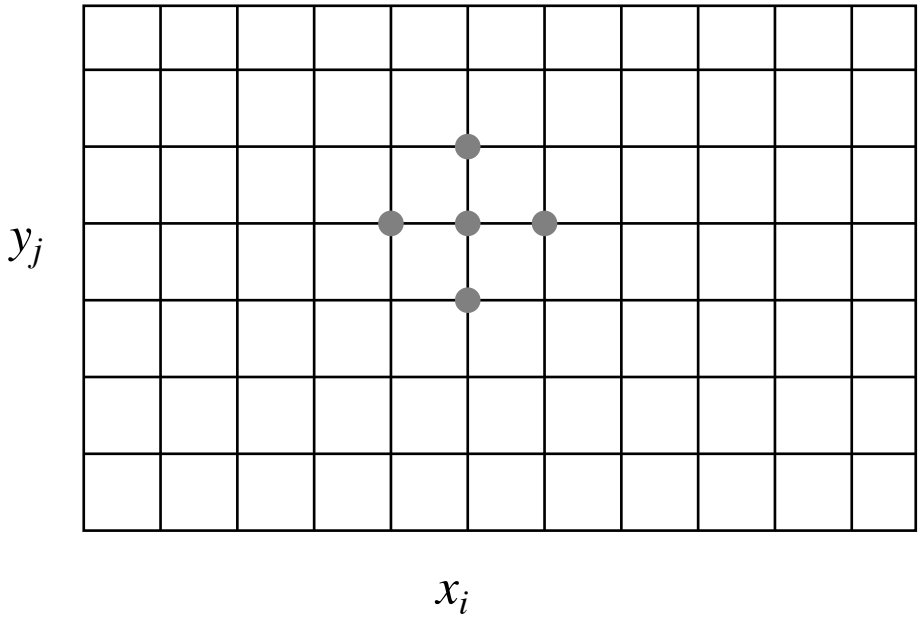
$$\Phi(x,y) = \Phi_0 \sum_{n=1,n\ odd}^{\infty} \frac{4}{n\pi} \sin\left(\frac{n\pi x}{L_x}\right) \frac{\sinh\left(\frac{n\pi y}{L_x}\right)}{\sinh\left(\frac{n\pi L_y}{L_x}\right)}$$


Φ(x,y)
1
0.8
0.6
0.4
0.2
0
y
x

Matrix Methods

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

$$\nabla^2\Phi \approx \frac{1}{h_x^2}\left(\Phi_{i+1,j}^n - 2\Phi_{i,j}^n + \Phi_{i-1,j}^n\right) + \frac{1}{h_y^2}\left(\Phi_{i,j+1}^n - 2\Phi_{i,j}^n + \Phi_{i,j-1}^n\right) + O(h_x^3, h_y^3)$$



Matrix Methods - 2

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

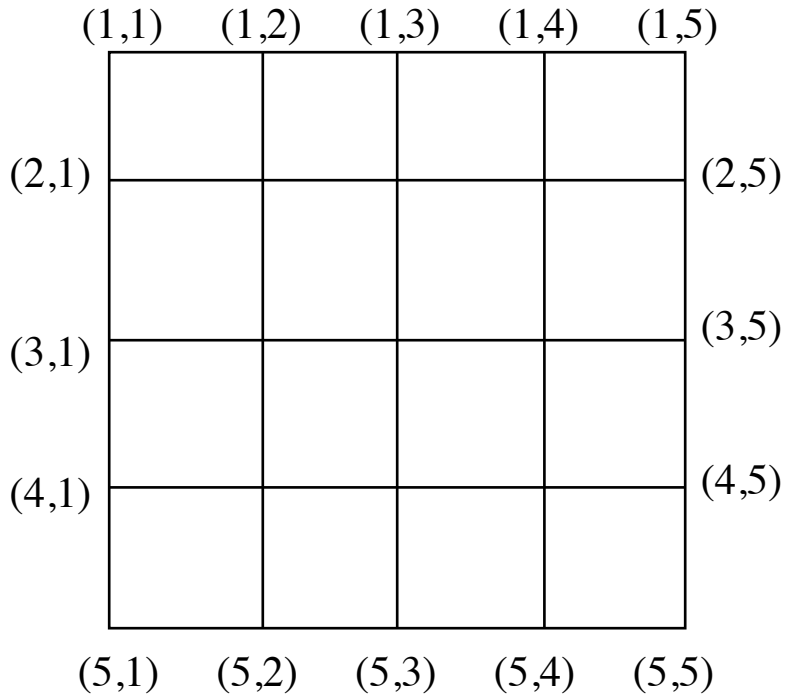
Define an index $n = j + (i-1) * N_y$

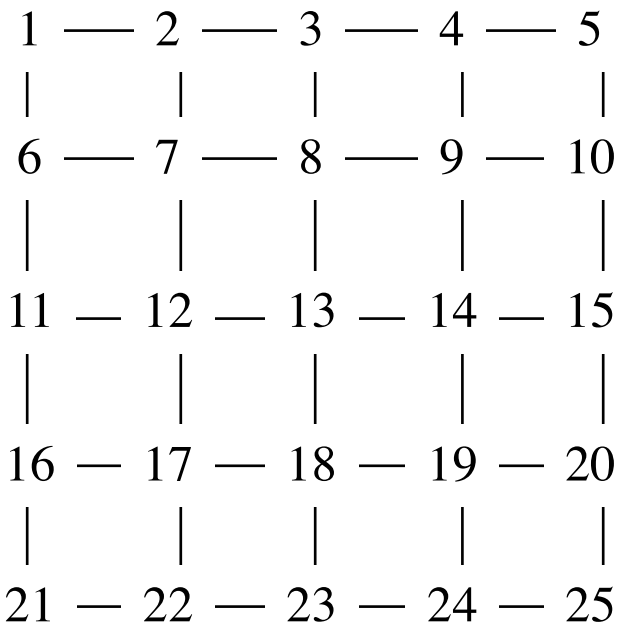
Let the solution Φ_n be defined at each of the vertices n

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

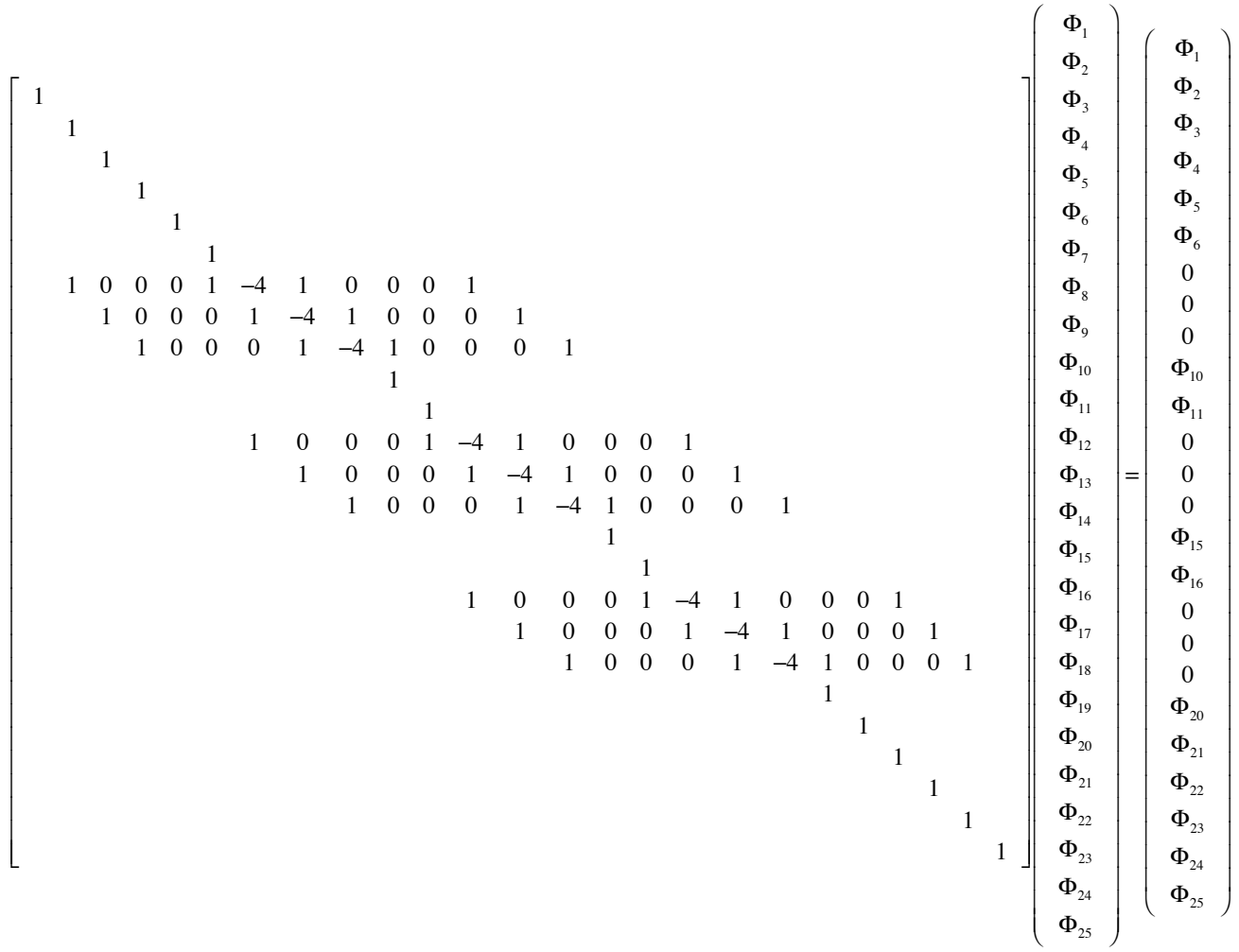
$$i = \text{int}\left(\frac{n-1}{N_y}\right) + 1$$

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





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


1
1
1
1
1
1 0 0 0 1 –4 1 0 0 0 1
1 0 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 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
Φ1 Φ2 Φ3 Φ4 Φ5 Φ6 Φ7 Φ8 Φ9 Φ10 Φ11 Φ12 Φ13 Φ14 Φ15 Φ16 Φ17 Φ18 Φ19 Φ20 Φ21 Φ22 Φ23 Φ24 Φ25
=
Φ1 Φ2 Φ3 Φ4 Φ5 Φ6 0 0 0 Φ10 Φ11 0 0 0 Φ15 Φ16 0 0 0 Φ20 Φ21 Φ22 Φ23 Φ24 Φ25

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

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

To get the solution, one needs to find the coefficients a_k. 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 ($\nabla^2\Phi - \frac{\rho}{\epsilon_0}$) with Neumann boundary conditions $\nabla\Phi \cdot \hat{n} = 0$ at the boundary, the trial function would be

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

Spectral Methods - Fourier Galerkin Methods -2

Inserting into (*) we get

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

Applying orthogonality condition by applying

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

to get the coefficients one gets

$$a_{n,m}=\frac{4}{\pi^2\varepsilon_0}\frac{1}{[(m-1)^2+(n-1)^2][1+\delta_{m,1}][1+\delta_{n,1}]}\times$$
$$\int_0^L dx\int_0^L dy\,\rho(x,y)\cos\left[\frac{(m-1)\pi x}{L}\right]\cos\left[\frac{(n-1)\pi y}{L}\right]$$

Spectral Methods - Fourier Galerkin Methods -3

Finally, we end up with the approximate solution as

$$\Phi_a(x,y) = \sum_{m=1}^{M}\sum_{n=1}^{M} a_{m,n} \cos\left[\frac{(m-1)\pi x}{L}\right] \cos\left[\frac{(n-1)\pi y}{L}\right]$$

To get the coefficients $a_{m,n}$ one has to compute the integrals

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

Which has to be done numerically

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

For this approach, we start with the discretized the Poisson Equation as

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

And apply the 2D Fourier transform

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

and the inverse Fourier transform

$$\Phi_{m+1,n+1} = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \mathrm{F}_{i+1,j+1} e^{\alpha i m + \alpha j n} \quad \mathrm{R}_{m+1,n+1} = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \rho_{i+1,j+1} e^{\alpha i m + \alpha j n}$$

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

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

and the inverse Fourier transform gives the potential

The program `fftpoi` that is available on the book author's website solves the Poisson problem for a dipole in a box problem. (algarcia.org)