Scientific Computing: Lecture 23

- Introduction to relaxation methods
  - Jacobi
  - Gauss-Seidel
  - Successive Overrelaxation
- Example: Laplace Equation (electric potential)
- Source Terms: Poisson's Equation

CLASS NOTES

- HW09 due **Monday** (last HW!).
- You should have proposals back through Box.
- Work on projects!
- Parallel programming and GUIs coming up next

Relaxation Methods

- Generally **relaxation** methods are useful for systems in **equilibrium**.
- Consider the 2D diffusion (or thermal) equation:

$$\frac{\partial T(x,y,t)}{\partial t} = \kappa \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

- LHS represents a temperature change (energy flux). When the system is in equilibrium (as $t \rightarrow \infty$), the net flux is 0.
- Then temperature would be given by:

$$\kappa \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) = 0$$

Laplace Equation

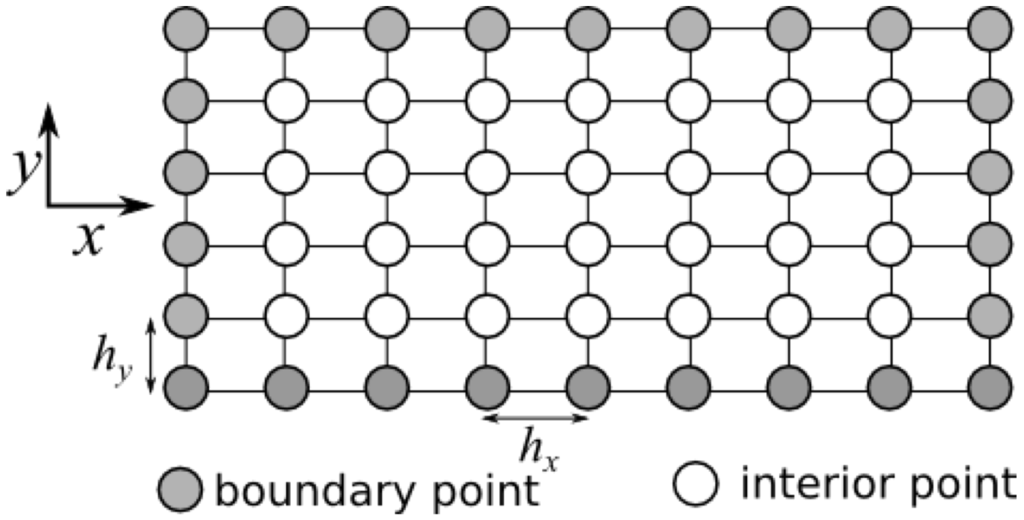
- Laplace Equation gives the electrostatic potential. The 'static' here implies equilibrium.

$$\mu\left(\frac{\partial^2\Phi}{\partial x^2}+\frac{\partial^2\Phi}{\partial y^2}\right)=0$$

- Note it has the same form as the diffusion equation for $t \rightarrow \infty$.
- Algorithms based on this principle are called **relaxation methods**.
- These are iterative procedures – like time steps but really just gradually approaching the equilibrium state.

Boundary Conditions

- Boundary conditions really define the system here.
- BCs are needed on ALL edges of your computational domain.
- Examples are the temperature of the walls of a container or the constant electric potential at each edge.
- Theses BCs must be kept constant over the entire iterative process until the solution is found.



J.R. Gladden, Phys212, Univ. of Mississippi

Jacobi Method

- In the normal way, discretize each of the 2nd order derivative. We include the time derivative term because we are "letting the solution evolve" until equilibrium is reached.

$$\begin{aligned}\Phi^{n+1}_{i,j} = \Phi^n_{i,j} \quad &+ \quad \frac{\mu\tau}{h_x^2}(\Phi^n_{i+1,j} + \Phi^n_{i-1,j} - 2\Phi^n_{i,j}) \\ &+ \quad \frac{\mu\tau}{h_y^2}(\Phi^n_{i,j+1} + \Phi^n_{i,j-1} - 2\Phi^n_{i,j})\end{aligned}$$

- Here n is a "time" index, "i" is the x index, and "j" is the y index.
- This is based on the FTCS method we discussed last time.

Jacobi Method

- From our FTCS stability requirement:

$$\frac{\mu\tau}{h_x^2} + \frac{\mu\tau}{h_y^2} \leq \frac{1}{2}$$

- We want the largest stable time step to get to equilibrium as fast as possible.
- This is the Jacobi method. Note the solution at each step and point is nothing more than computing the **average** of the solution at each point bounding it (assume $h_x = h_y$).

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

Final state and Initial Guess

- Final State
  - This step will be repeated until the solution “stops” changing – indicating equilibrium has been reached.
  - Often the solution never actually stops. Need to provide a tolerance metric and target .
- Initial Guess
  - Obvious initial values for all interior points is 0. The farther this is away from final solution, the longer it will take.
  - Often possible to make a more intelligent guess based in BCs. Can dramatically reduce number of iterations!
    - Horizontal plane with average of all BCs
    - Flat plane sloping from high to low side of BCs.

Gauss-Seidel Method

- A simple change can speed convergence.
- We already have updated solutions for half of the bounding points – lets use them!
- This is called Gauss-Seidel method. It speeds convergence and reduces the amount of memory required. We only need to store half the number of points.

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

Simultaneous Overrelaxation

- Another idea is to take bigger steps toward the equilibrium solution.
- Before, all steps were in the same direction. How about overshooting the solution and coming back.
- This is called Simultaneous Overrelaxation (SOR).

$$\begin{aligned}\Phi_{i,j}^{n+1} &= (1-\omega^2)\Phi_{i,j}^{n} + \\ &\frac{\omega}{4}(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n+1} + \Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n+1})\end{aligned}$$

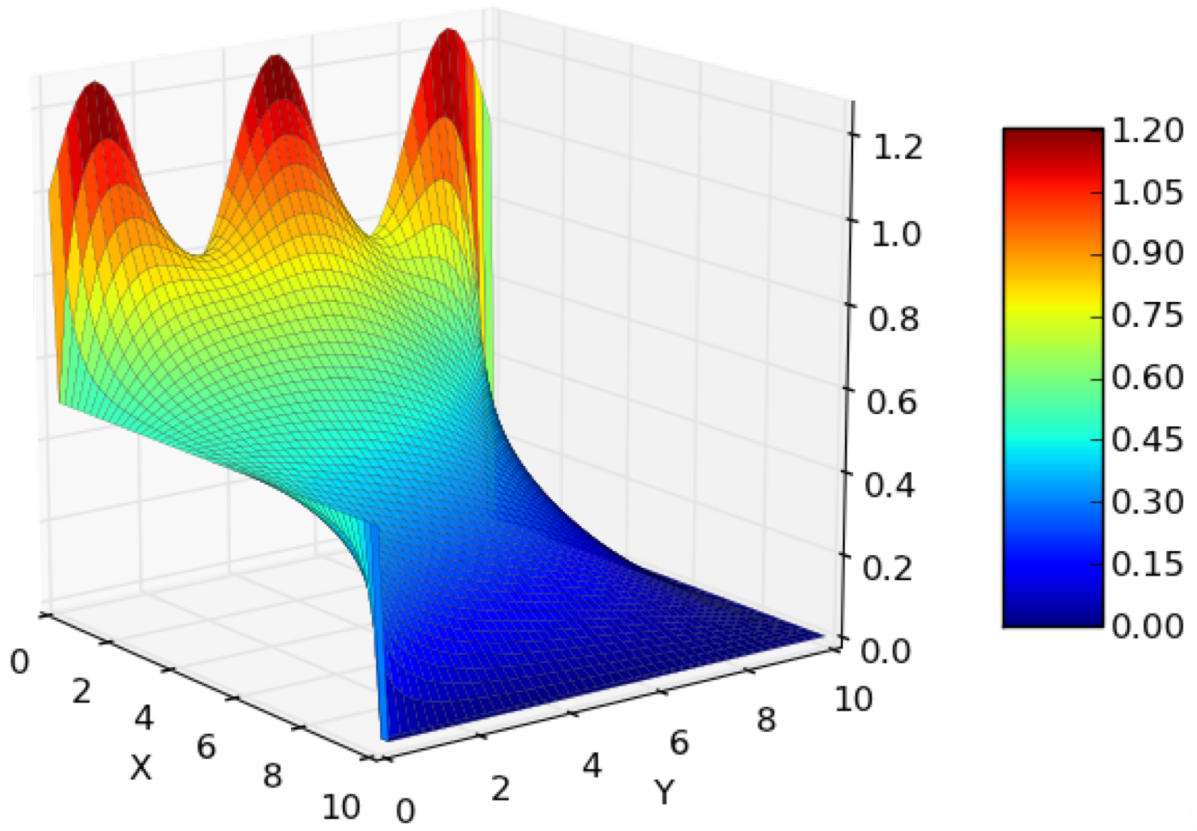
Overrelaxation Parameter

- Trick is to find a good value for the overrelaxation parameter ω. For stability, it MUST be between 1 and 2.
- Optimal solution for a N_x X N_y grid is

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

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

Sample Output



Poisson Equation

- The Poisson equation allows for interior source terms.
- Here the sources of potential come in the form of a charge density over the interior points.

$$\frac{\partial^2 \Phi(x,y)}{\partial x^2} + \frac{\partial^2 \Phi(x,y)}{\partial y^2} = -\frac{1}{\epsilon_0}\rho(x,y)$$

- We discretize using the same procedure as Laplace.
- Here ρ gives the charge density over the interior points.

$$\frac{1}{h_x^2}\left(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n+1} - 2\Phi_{i,j}\right) + \frac{1}{h_y^2}\left(\Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n+1} - 2\Phi_{i,j}\right) = -\frac{1}{\epsilon_0}\rho_{i,j}$$

Gauss-Seidel for Poisson

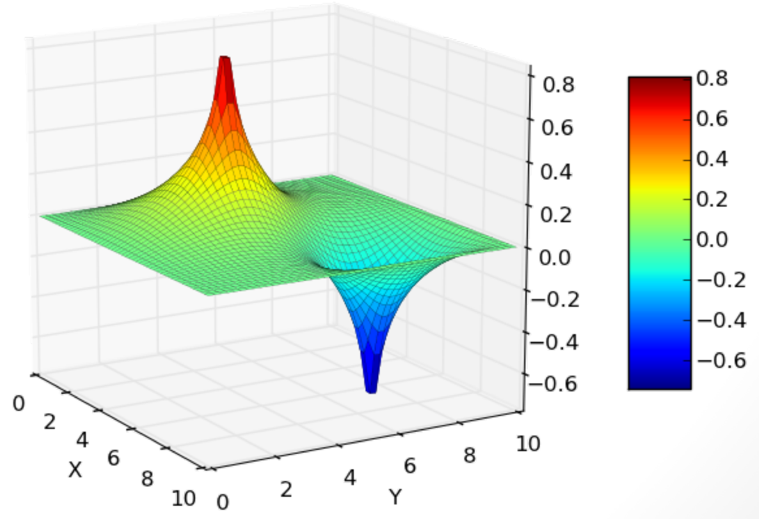
- Using the GS method, the discrete equation we need is

$$\Phi_{i,j}^{n+1} = \frac{1}{4}(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n+1} + \Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n+1} + \frac{1}{\epsilon_0}h^2\rho_{i,j})$$

- Here we have assumed $h_x = h_y = h$
- We can define a python function to return the charge density which will be called inside the interior points loop.

Exercise: Poisson Equation

- Take the Laplace code and adjust it to solve the Poisson equation.
- Define a charge density of your choosing and run it.
  - It may be easiest to initially try a simple dipole with opposite sign charges at two different points.
  - Then try a ring of charge near the center of the domain.



0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
X
Y
0
2
4
6
8
10