MINNESOTA STATE UNIVERSITY MOORHEAD

Physics 350: Computational Methods for Physical Science

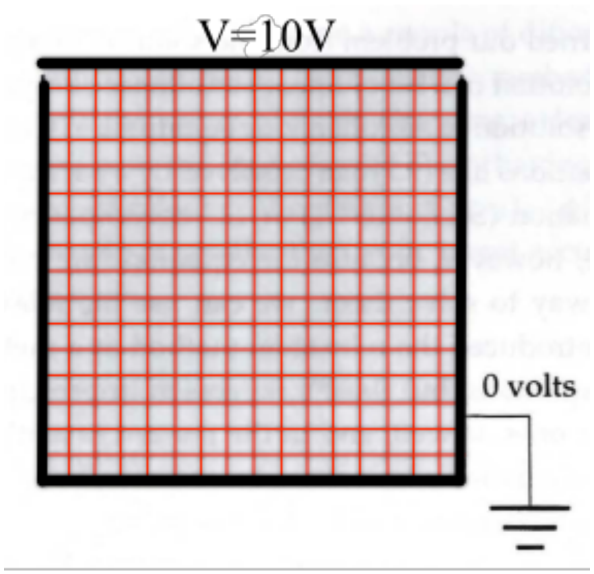
Lecture 20: Solving Partial Differential Equations

Spring Semester 2019

Matt Craig and Steve Lindaas

Destination

- Goal is to calculate electric field in region of space
- Similar to finding gravitational field or heat conduction



Quick Review of Electric Fields

- Electric potential ϕ is the electric potential energy per charge (often called V in intro physics)
- In 1D,

$$U = -\int \vec{F} \cdot d\vec{x}$$

$$q\phi = -q\int \vec{E} \cdot d\vec{x}$$

$$\phi = -\int \vec{E} \cdot d\vec{x} = -\int E_x \, dx$$

$$d\phi = -E_x dx \longrightarrow \boxed{E_x = -\frac{\partial \phi}{\partial x}}$$

Electric Fields and Potentials

MINNESOTA STATE UNIVERSITY MOORHEAD

- In 3D: $\vec{E} = E_x\hat{x} + E_y\hat{y} + E_z\hat{z}$

$$= -\frac{\partial\phi}{\partial x}\hat{x} - \frac{\partial\phi}{\partial y}\hat{y} - \frac{\partial\phi}{\partial z}\hat{z}$$

$$\boxed{\vec{E} = -\vec{\nabla}\phi} \qquad \text{where } \vec{\nabla} = \frac{\partial}{\partial x}\hat{x} + \frac{\partial}{\partial y}\hat{y} + \frac{\partial}{\partial z}\hat{z}$$

Field, Potential and Charge

- Gauss' Law for electric fields

$$\int \vec{E} \cdot d\vec{A} = Q_{encl}/\varepsilon_0$$

- Charge in terms of charge density, ρ

$$Q = \int \rho dV$$

- So

$$\int \vec{E} \cdot d\vec{A} = \int \frac{\rho}{\varepsilon_0} dV$$

Field, Potential and Charge 2

- Divergence theorem says that

$$\int \vec{E} \cdot d\vec{A} = \int \frac{\rho}{\varepsilon_0} dV$$

is equivalent to

$$\boxed{\vec{\nabla} \cdot \vec{E} = \frac{\rho}{\epsilon_0}}$$

- In terms of potential

$$\vec{\nabla} \cdot \vec{E} = \frac{\rho}{\epsilon_0} \longrightarrow -\nabla^2 \phi = \frac{\rho}{\epsilon_0} \longrightarrow \boxed{\nabla^2 \phi = -\frac{\rho}{\epsilon_0}}$$

Names

- Poisson Equation

$$\nabla^2 \phi = -\frac{\rho}{\varepsilon_0}$$

- Special case (no charge), ρ=0, called Laplace's Equation

$$\nabla^2 \phi = 0$$

Laplacian

- What does $\nabla^2\phi$ mean?

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}$$

- Solving Laplace's Equation means finding φ so that

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

Solving PDEs via Relaxation

MINNESOTA STATE UNIVERSITY
MOORHEAD

Setup: An Electrostatics PDE Problem

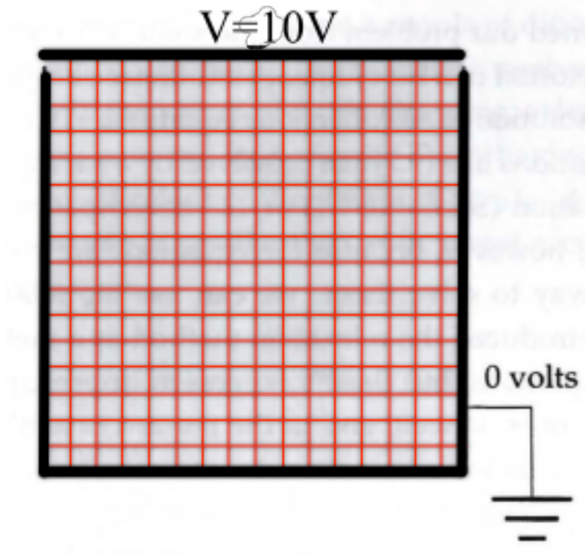


Figure 9.1: A simple electrostatics problem. A box has conducting walls, all of which are grounded at 0 volts except for the wall at the top, which is at some other voltage V. The small gaps between the top wall and the others are intended to show that they are insulated from one another. We'll assume in our calculations that these gaps have negligible width.

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial x^2} = \rho(x, y)/\epsilon_0$$

- Given the PDE for $\phi(x,y)$ and its boundary conditions (fixed).
- Divide space into a square grid with spacing a.
- We'll then use the **method of finite differences** to solve for $\phi(x,y)$ inside the boundary.

Approximation for 2nd derivative

- Consider one dimensional function f
- What is approximate formula for $f''(x)$?
- Assume values of f known at a set of grid points, $f(0), f(h), \ldots, f(x-h), f(x), f(x+h), \ldots$
- Start by thinking about "separate" function $g(x)$, which is really $f'(x)$…

2nd Derivative via Central Difference Method

- Given one-dimensional function g
- Assume you know values at $g(x+h/2)$ and $g(x-h/2)$
- What is central difference formula for $g'(x)$?

$$g'(x) =$$

Back to *f*...

- What is central difference approximation for $f'(x+h/2)$?
  - We know $f(x-h)$, $f(x)$, $f(x+h)$, etc.

$$f'(x+h/2) =$$

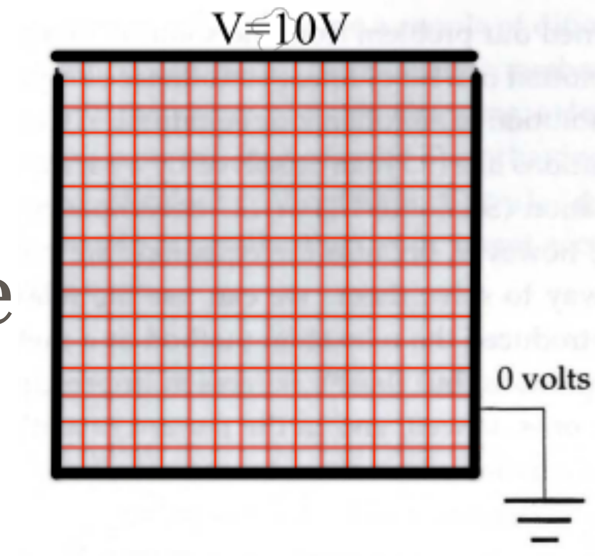
Combine these results

- Recall $g'(x) = \dfrac{g(x+h/2) - g(x-h/2)}{h}$
- and $g=f'$
- so $f''(x) = \dfrac{f'(x+h/2) - f'(x-h/2)}{h}$

- Just found $f'(x+h/2)=$
- Put it together and…

$f''(x) \approx$

Back to Poisson Equation



- Assuming grid has spacing a, we estimate can estimate second derivative of potential

$$\frac{\partial^2 \phi}{\partial x^2} \approx \frac{\phi(x+a,y)+\phi(x-a,y)-2\phi(x,y)}{a^2}$$

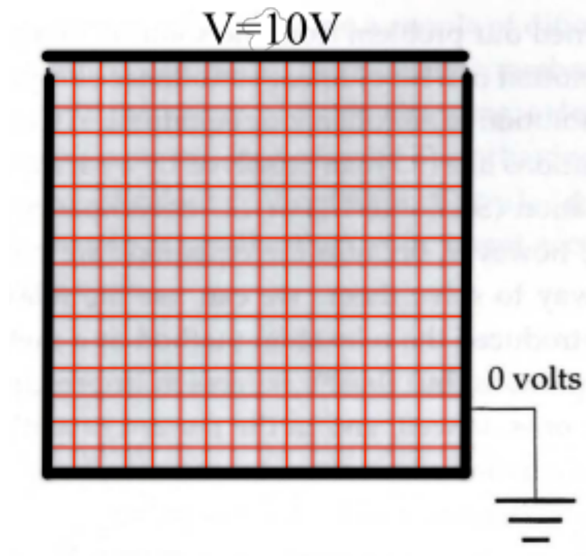
y derivative is similar

$$\frac{\partial^2 \phi}{\partial y^2} \approx \frac{\phi(x,y+a)+\phi(x,y-a)-2\phi(x,y)}{a^2}$$

2D Poisson Equation

- Put these into Poisson's equation

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial x^2} = \rho(x,y)/\epsilon_0$$



$$\frac{\partial^2\phi}{\partial x^2} \approx \frac{\phi(x+a,y)+\phi(x-a,y)-2\phi(x,y)}{a^2}$$

$$\frac{\partial^2\phi}{\partial y^2} \approx \frac{\phi(x,y+a)+\phi(x,y-a)-2\phi(x,y)}{a^2}$$

$$\phi(x+a,y)+\phi(x-a,y)+\phi(x,y+a)+\phi(x,y-a)-4\phi(x,y) = a^2\rho(x,y)/\epsilon_0$$

Electrostatic Potential Solution

- Given the previous result, we then know the solution to Laplace's equation here is:

$$\phi(x,y) = \frac{1}{4}\left(\phi(x+a,y) + \phi(x-a,y) + \phi(x,y+a) + \phi(x,y-a) - a^2\rho(x,y)/\epsilon_0\right)$$

Matching

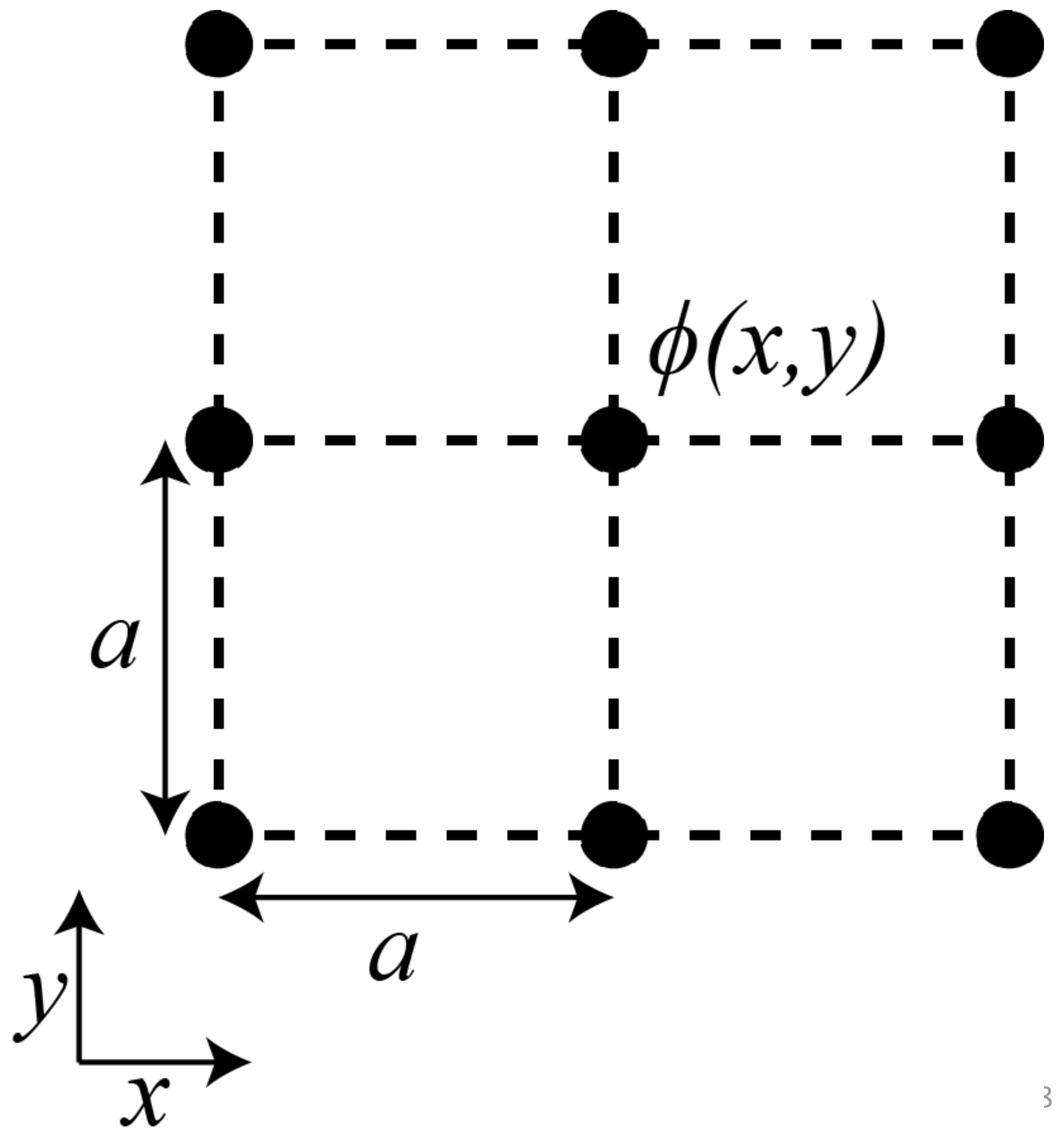
- Match the points

$$\phi(x+a,y)$$

$$\phi(x,y+a)$$

$$\phi(x-a,y)$$

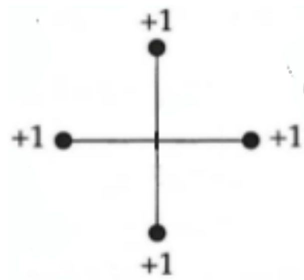
$$\phi(x,y-a)$$



Relaxation Method for solving 2D Laplace Equation

MINNESOTA STATE UNIVERSITY MOORHEAD

$$\phi_{new} = \frac{1}{4}\sum_{old}$$



Interior Points

$$\phi_{new} = \phi_{old}$$

Boundary Points

Cheat sheet…

$$f''(x) \approx \frac{f'\left(x+\frac{h}{2}\right) - f'\left(x-\frac{h}{2}\right)}{h}$$

$$\approx \frac{\frac{f(x+h)-f(x)}{h} - \frac{f(x)-f(x-h)}{h}}{h}$$

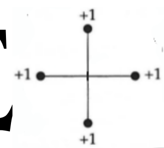
$$\approx \frac{f(x+h)+f(x-h)-2f(x)}{h^2}$$

The Relaxation Method Loop: Step By Step

1. Initialize the $\phi_{\text{old}}(x,y)$ array.
2. Relaxation Method Loop:

a) Compute new solution

$$\phi_{new} = \frac{1}{4}\sum_{old}$$



(interior points)

$$\phi_{new} = \phi_{old}$$

(boundary points)

note: first line can include boundary

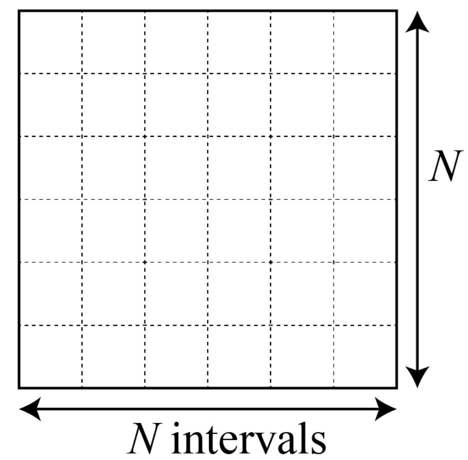
b) Check if the difference between $\phi_{\text{new}}(x,y)$ and $\phi_{\text{old}}(x,y)$ is small enough to call it final.

- If so: Exit loop since $\phi_{\text{new}}(x,y)$ is solution
- If not: Set $\phi_{\text{old}}(x,y) = \phi_{\text{new}}(x,y)$ and start again.

Initializing the ϕ Array

- If we have $N \times N$ <u>intervals</u>, how many points do we need for the ϕ array in Python?
  a) $N \times N$
  b) $(N-1) \times (N-1)$
  c) $N \times (N-1)$
  d) None of the above

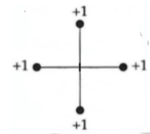
$\phi =$



- Which of the following commands sets up this ϕ array properly?
  a) `phi = np.zeros([N, N], float)`
  b) `phi = np.zeros([N + 1, N + 1], float)`
  c) `phi = np.empty([N, N], float)`
  d) None of the above

Calculating the New Values

$$\phi_{new} = \frac{1}{4}\sum_{old}$$



(interior points)

$$\phi_{new} = \phi_{old}$$

(boundary points)

- You could compute the new interior values for $\phi_{\text{new}}(x,y)$ using $\phi_{\text{old}}(x,y)$ by doing something like:

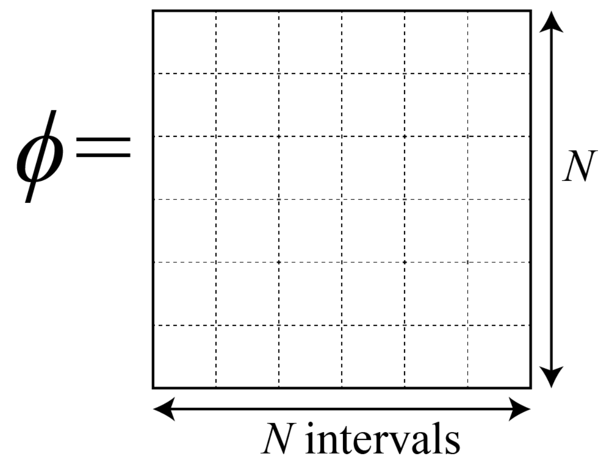
```
for i in range(N + 1):
    for j in range(N + 1):
        phiNew[i, j] = .25 * (phi[i, j - 1] + phi[i, j + 1]
                                + phi[i - 1, j] + phi[i + 1, j])
```

- There are 2 problems with this:
  1. It includes the boundaries, which is fine if you then handle them after doing this.
  2. It is SLOW for big arrays. Why?

Initializing the ϕ Array

MINNESOTA STATE UNIVERSITY MOORHEAD

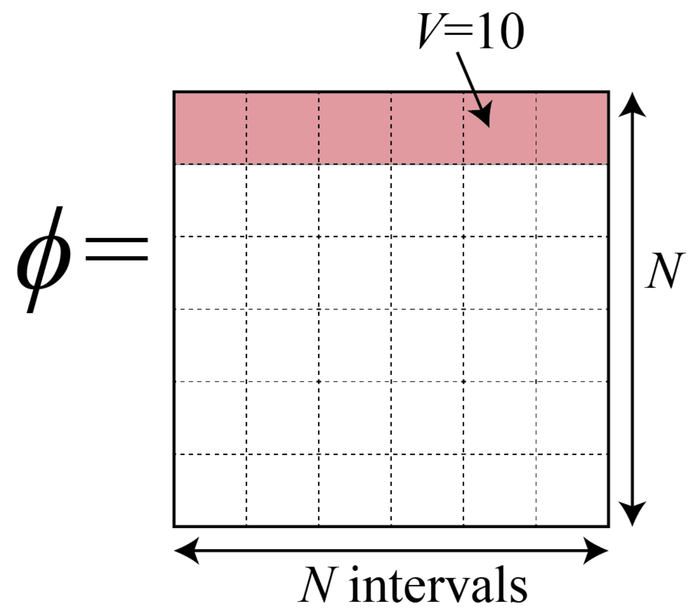
- If we have $N \times N$ intervals, how many points do we need for the ϕ array in Python?
  - a) $N \times N$
  - b) $(N-1) \times (N-1)$
  - c) $N \times (N-1)$
  - d) None of the above



- Which of the following commands sets up this ϕ array properly?
  - a) `phi = np.zeros([N, N], float)`
  - b) `phi = np.zeros([N + 1, N + 1], float)`
  - c) `phi = np.empty([N, N], float)`
  - d) None of the above

Setting Up the Bounds on the ϕ Array

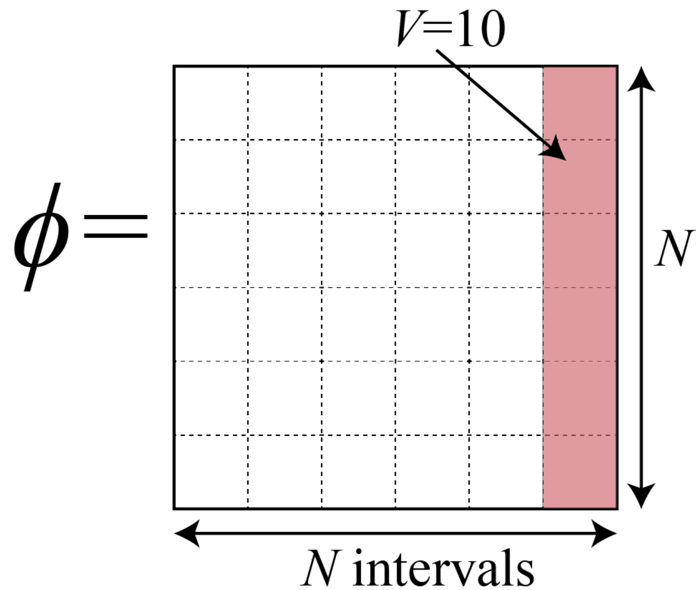
- To set the voltage on the top bound of this array to V=10, define
  `V = 10`
  and then what command should be used?




```
phi[     ] = V
```

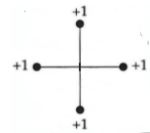
Setting Up the Bounds on the ϕ Array

- To set the voltage on the right bound of this array to V=10, define
  `V = 10`
  and then what command should be used?




```
phi[      ] = V
```

Calculating the New Values

$$\phi_{new} = \frac{1}{4}\sum_{old}$$ 

(interior points)

$$\phi_{new} = \phi_{old}$$ (boundary points)

- You could compute the new interior values for $\phi_{\text{new}}(x,y)$ using $\phi_{\text{old}}(x,y)$ by doing something like:

```
for i in range(N + 1):
    for j in range(N + 1):
        phiNew[i, j] = .25 * (phi[i, j - 1] + phi[i, j + 1]
                              + phi[i - 1, j] + phi[i + 1, j])
```

- There are 2 problems with this:
  1. It includes the boundaries, which is fine if you then handle them after doing this.
  2. It is SLOW for big arrays. Why?

Calculating `PhiNew` the NumPy Way

- In Spyder, search for help on the command **`numpy.roll()`** and try the following:

```
import numpy as np
a = np.array([[1,2,3],[4,5,6],[7,8,9]],
float)
```

- What do each of the following return?
  - **`np.roll(a, 1, axis=0)`**
  - **`np.roll(a, 1, axis=1)`**
  - **`np.roll(a, -1, axis=0)`**

- Explain what **`np.roll`** does and how this can help you rework the updating of **`phiNew`** using array arithmetic?