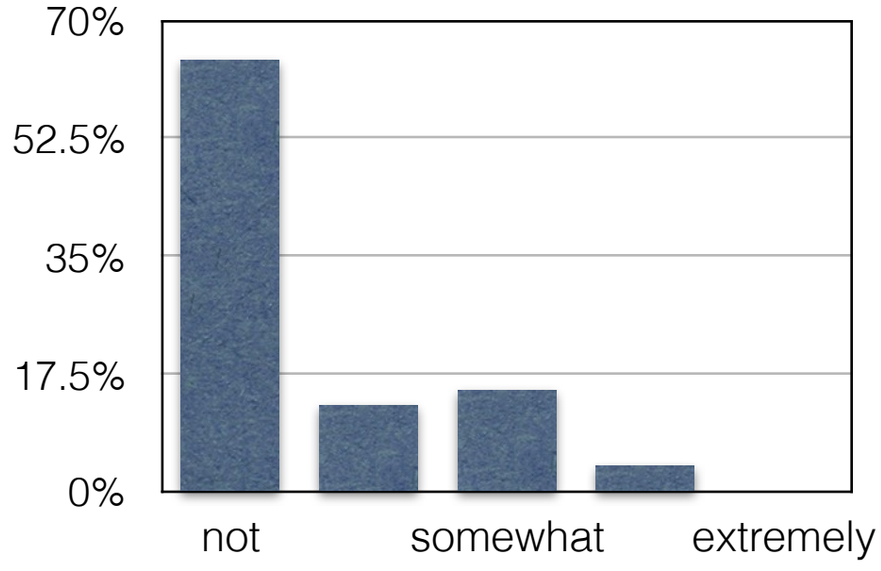
Muddiest Points from Class 01/30

- *"I have only briefly seen Fourier transforms used but have never had any experience applying them. Will we be using them in future assignments?"*
  - no

- *"[…] Also, the methods you gave in class, both include to refine the mesh if for method A the wave number is greater than the threshold and for method b if the estimated error is greater than the threshold, but what is the threshold in both these cases?*
  - the threshold is related to an acceptable error level for the case analyzed

- *"Are non-uniform grids universally less accurate or are there cases to use them?"*
  - mesh size transitions are always less accurate than uniform meshes using the finer spacing
  - in practice one tries to use uniform, fine enough mesh patches wherever possible, and smooth transitions (small changes) to connect uniform patches

- *"Will we always be using uniform meshes for class assignments?"*
  - uniform meshes are not a focus of class assignments



First Model Problem

- Poisson Equation in 2D: $\nabla^2\varphi = f$ or $\dfrac{\partial^2\varphi}{\partial x^2} + \dfrac{\partial^2\varphi}{\partial y^2} = f(x,y)$
  - variations of this are

    Laplace equation: $\nabla^2\varphi = 0$

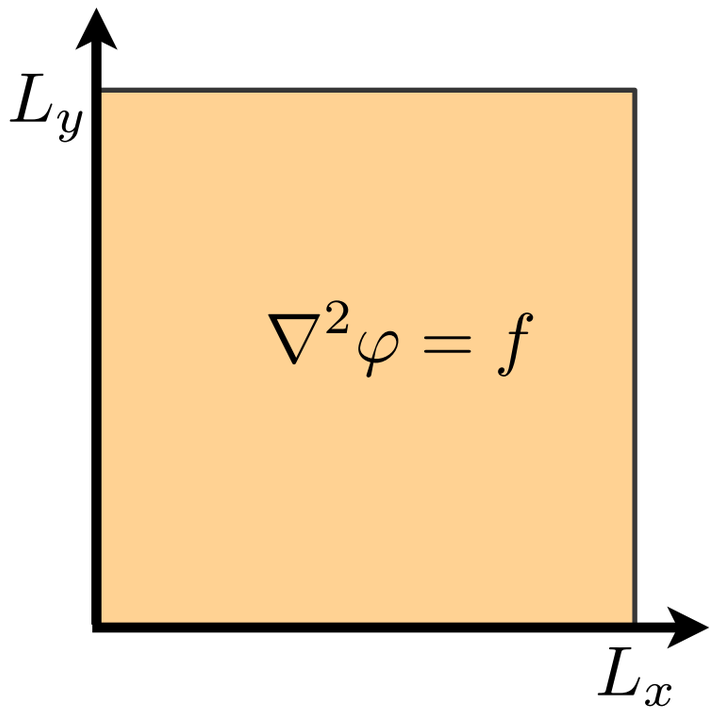
    Helmholtz equation: $\nabla^2\varphi + \alpha^2\varphi = 0$

  - elliptic equation!
    - closed solution domain
    - instantaneous propagation of information
    - requires boundary conditions:

      $c_1\varphi + c_2\dfrac{\partial\varphi}{\partial n} = g$ n: coordinate normal to boundary

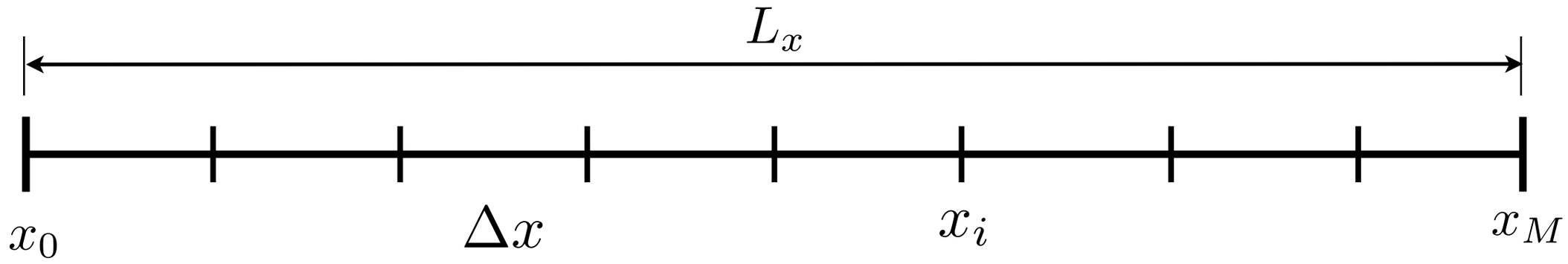
Step 1: Define Solution Domain

$$0 \leq x \leq L_x \quad \text{and} \quad 0 \leq y \leq L_y$$



Step 2: Define Mesh

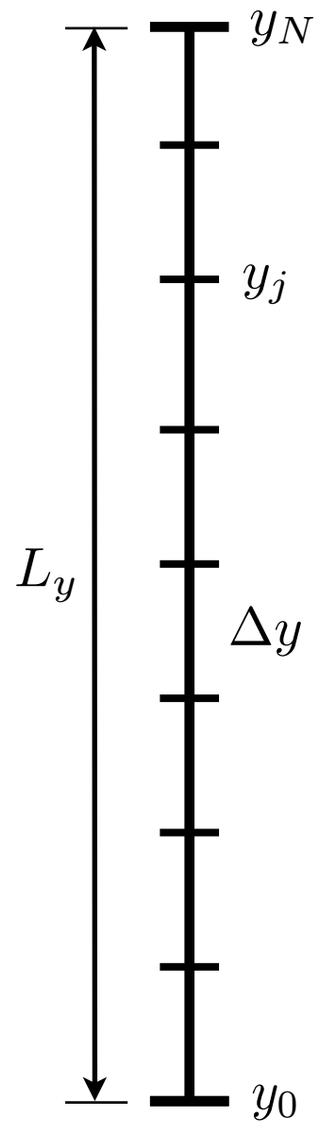
- here: for simplicity: Cartesian, equidistant in each direction
- x-direction:



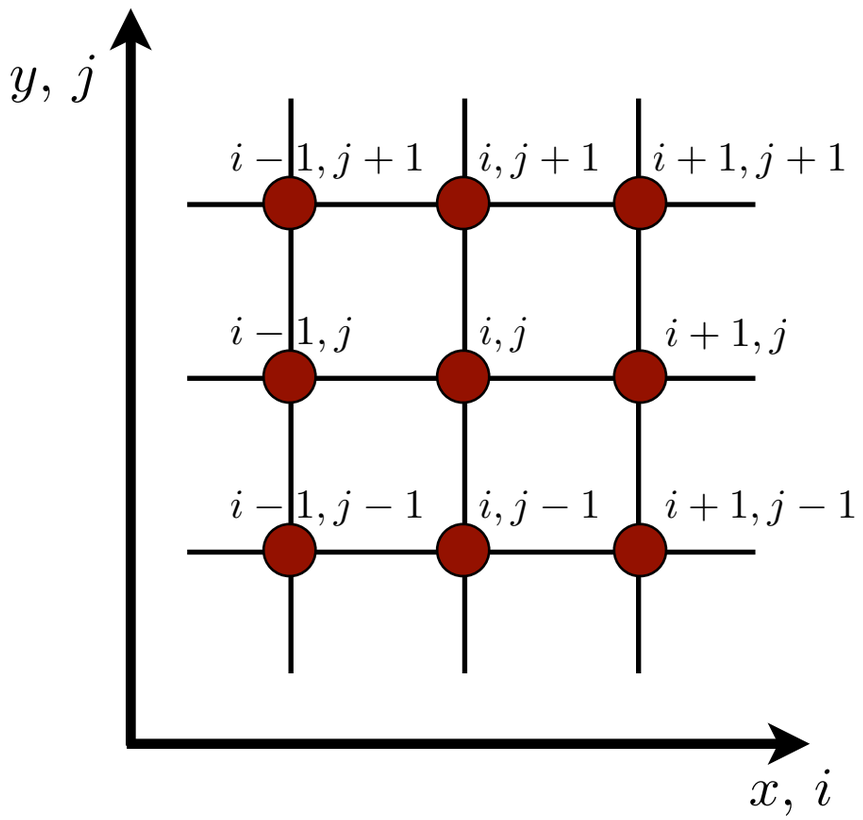
- $M+1$ points: $x_0, x_1, x_2, \ldots x_{M-1}, x_M$
- points at boundary: x_0 and x_M
- $M-1$ interior points: $x_1, x_2, \ldots x_{M-1}$
- grid spacing: $\Delta x = \frac{L_x}{M}$

Step 2: Define Mesh

- y-direction:
  - $N+1$ points: $y_0, y_1, y_2, \ldots y_{N-1}, y_N$
  - points at boundary: y_0 and y_N
  - $N-1$ interior points: $y_1, y_2, \ldots y_{N-1}$
  - grid spacing: $\Delta y = \dfrac{L_y}{N}$



- 2D Mesh:



Step 3: Approximate Spatial Derivatives

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = f(x,y)$$

- using Taylor tables for 3-point centered stencil:

$$\left.\frac{\partial^2 \varphi}{\partial x^2}\right|_{i,j} = \frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{\Delta x^2} + O(\Delta x^2)$$

$$\left.\frac{\partial^2 \varphi}{\partial y^2}\right|_{i,j} = \frac{\varphi_{i,j+1} - 2\varphi_{i,j} + \varphi_{i,j-1}}{\Delta y^2} + O(\Delta y^2)$$

Step 4: Substitute into PDE

$$\left.\frac{\partial^2 \varphi}{\partial x^2}\right|_{i,j} = \frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{\Delta x^2} + O(\Delta x^2)$$

$$\left.\frac{\partial^2 \varphi}{\partial y^2}\right|_{i,j} = \frac{\varphi_{i,j+1} - 2\varphi_{i,j} + \varphi_{i,j-1}}{\Delta y^2} + O(\Delta y^2)$$

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = f(x,y)$$

$$\frac{\varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}}{\Delta x^2} + \frac{\varphi_{i,j+1} - 2\varphi_{i,j} + \varphi_{i,j-1}}{\Delta y^2} = f_{i,j}$$

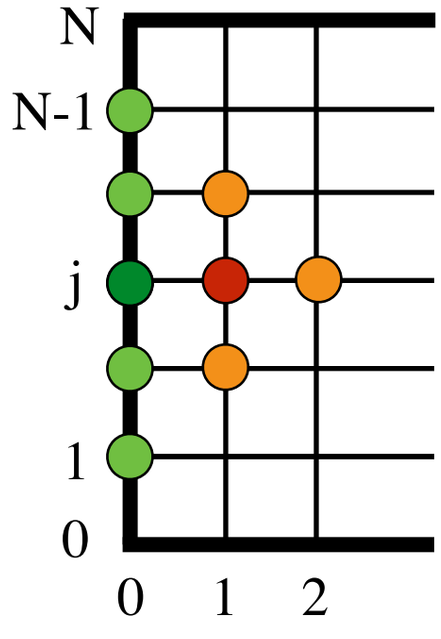
➡ this is now a difference equation and no longer a differential equation!

- let's assume for simplicity: $\Delta x = \Delta y = \Delta$

$$\varphi_{i+1,j} - 4\varphi_{i,j} + \varphi_{i-1,j} + \varphi_{i,j+1} + \varphi_{i,j-1} = \Delta^2 f_{i,j}$$

  - valid only for interior points, i.e. for $i=1,..,M-1$ and $j=1,..,N-1$
  - **not valid** for boundary points, i.e. for $i=0,M$ or $j=0,N$

Step 5: Incorporate Boundary Conditions



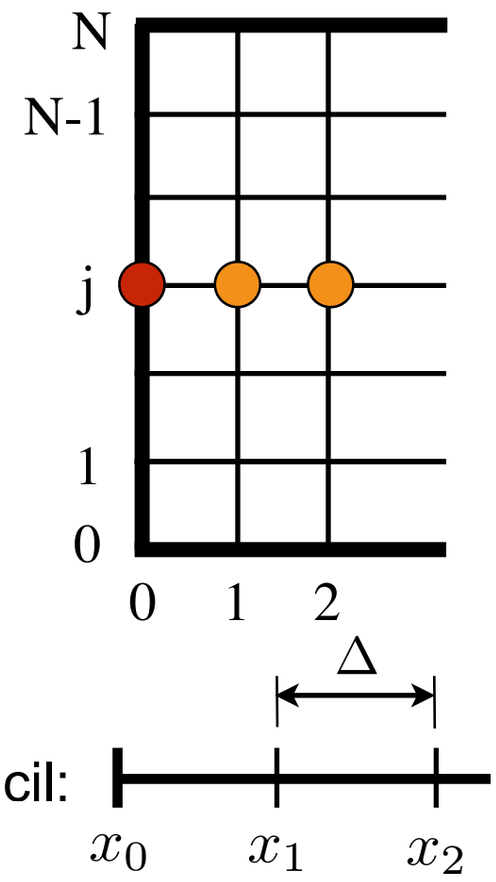
- Example #1:
  - Dirichlet at x_0: $\varphi_{0,j}$ given: $j=1, N-1$
    - finite difference formula at $i=1,j$:

$$\varphi_{2,j} - 4\varphi_{1,j} + \varphi_{0,j} + \varphi_{1,j+1} + \varphi_{1,j-1} = \Delta^2 f_{1,j}$$

$$\varphi_{2,j} - 4\varphi_{1,j} \quad + \varphi_{1,j+1} + \varphi_{1,j-1} = \Delta^2 f_{1,j} - \varphi_{0,j} \qquad \text{for } j = 1,N-1$$

➡ Dirichlet boundary conditions modify right-hand side only!

Step 5: Incorporate Boundary Conditions



- Example #2:
  - Neumann at x_0: $\left.\frac{\partial \varphi}{\partial x}\right|_{x_0} = g(y)$

    - approximate derivative by one-sided finite difference formula

$$\left.\frac{\partial \varphi}{\partial x}\right|_{0,j} = \frac{-3\varphi_{0,j} + 4\varphi_{1,j} - \varphi_{2,j}}{2\Delta}$$

    - solve for $\varphi_{0,j}$

$$\varphi_{0,j} = \frac{1}{3}\left(-2\Delta g_j + 4\varphi_{1,j} - \varphi_{2,j}\right)$$

$$\varphi_{2,j} - 4\varphi_{1,j} + \varphi_{0,j} + \varphi_{1,j+1} + \varphi_{1,j-1} = \Delta^2 f_{1,j}$$

    - substitute into finite difference formula at *i=1*

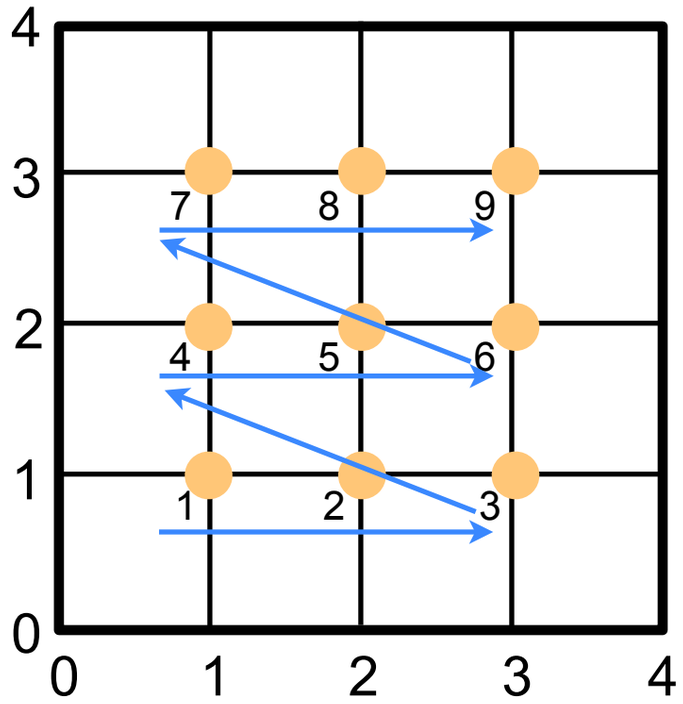
$$\frac{2}{3}\varphi_{2,j} - \frac{8}{3}\varphi_{1,j} + \varphi_{1,j+1} + \varphi_{1,j-1} = \Delta^2 f_{i,j} + \frac{2}{3}\Delta g_j$$

➡ Neumann boundary conditions modify both sides!

Step 6: Matrix Form (only for illustration, never code!)

- Example:
  - use Dirichlet boundary conditions
  - $M=N=4$ ⇒ 3x3 interior points

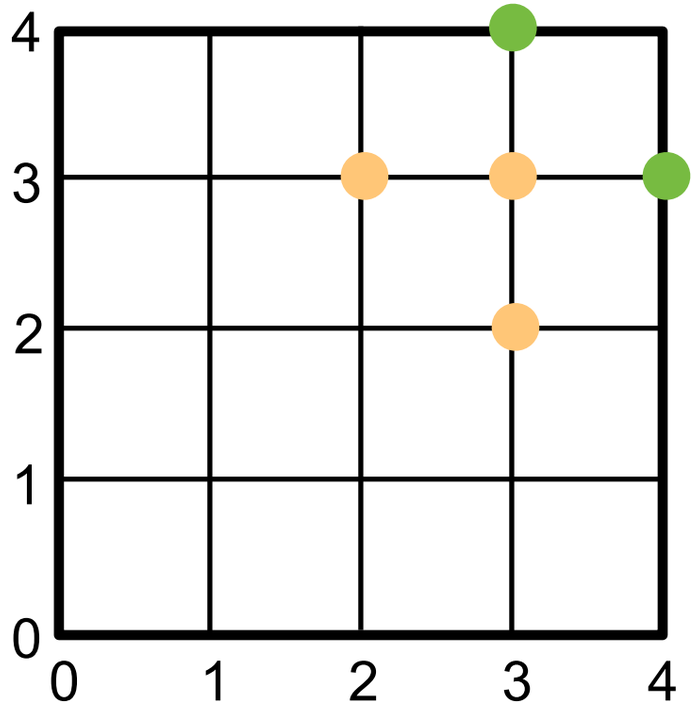
➡ the solution vector φ covers 2D space
⇒ need to define a sorting



Step 6: Matrix Form (only for illustration, never code!)

- Example:
  - use Dirichlet boundary conditions
  - $M=N=4$ ⇒ 3x3 interior points

➡ the solution vector φ covers 2D space
⇒ need to define a sorting



(1,1): $\varphi_{2,1} - 4\varphi_{1,1} + \varphi_{1,2} = \Delta^2 f_{1,1} - \varphi_{0,1} - \varphi_{1,0}$

(2,1): $\varphi_{3,1} - 4\varphi_{2,1} + \varphi_{1,1} + \varphi_{2,2} = \Delta^2 f_{2,1} - \varphi_{2,0}$

(3,1): $-4\varphi_{3,1} + \varphi_{2,1} + \varphi_{3,2} = \Delta^2 f_{3,1} - \varphi_{4,1} - \varphi_{3,0}$

(1,2): $\varphi_{2,2} - 4\varphi_{1,2} + \varphi_{1,3} + \varphi_{1,1} = \Delta^2 f_{1,2} - \varphi_{0,2}$

(2,2): $\varphi_{3,2} - 4\varphi_{2,2} + \varphi_{1,2} + \varphi_{2,3} + \varphi_{2,1} = \Delta^2 f_{2,2}$

(3,2): $-4\varphi_{3,2} + \varphi_{2,2} + \varphi_{3,3} + \varphi_{3,1} = \Delta^2 f_{3,2} - \varphi_{4,2}$

(1,3): $\varphi_{2,3} - 4\varphi_{1,3} + \varphi_{1,2} = \Delta^2 f_{1,3} - \varphi_{0,3} - \varphi_{1,4}$

(2,3): $\varphi_{3,3} - 4\varphi_{2,3} + \varphi_{1,3} + \varphi_{2,2} = \Delta^2 f_{2,3} - \varphi_{2,4}$

(3,3): $-4\varphi_{3,3} + \varphi_{2,3} + \varphi_{3,2} = \Delta^2 f_{3,3} - \varphi_{4,3} - \varphi_{3,4}$

Step 6: Matrix Form (only for illustration, never code!)

- Example:

$$\begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} \varphi_{1,1} \\ \varphi_{2,1} \\ \varphi_{3,1} \\ \varphi_{1,2} \\ \varphi_{2,2} \\ \varphi_{3,2} \\ \varphi_{1,3} \\ \varphi_{2,3} \\ \varphi_{3,3} \end{bmatrix} = \begin{bmatrix} \Delta^2 f_{1,1} - \varphi_{0,1} - \varphi_{1,0} \\ \Delta^2 f_{2,1} - \varphi_{2,0} \\ \Delta^2 f_{3,1} - \varphi_{4,1} - \varphi_{3,0} \\ \Delta^2 f_{1,2} - \varphi_{0,2} \\ \Delta^2 f_{2,2} \\ \Delta^2 f_{3,2} - \varphi_{4,2} \\ \Delta^2 f_{1,3} - \varphi_{0,3} - \varphi_{1,4} \\ \Delta^2 f_{2,3} - \varphi_{2,4} \\ \Delta^2 f_{3,3} - \varphi_{4,3} - \varphi_{3,4} \end{bmatrix}$$

➡ Symmetric block-tridiagonal matrix!

$$A = \begin{bmatrix} B & C & 0 \\ C & B & C \\ 0 & C & B \end{bmatrix} \qquad B = \begin{bmatrix} -4 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & -4 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

➡ next: we need to solve $A\vec{\varphi} = \vec{b}$

Step 7: Solve $A\vec{\varphi} = \vec{b}$

- Two options:
  1) Direct method: Gaussian elimination
  2) Iterative methods

- Gaussian elimination usually not efficient
  - operation count is $O(P^3)$
  - solution only available after $O(P^3)$ operations at the very end!

➡ Iterative methods are typically preferable

- Review of Iterative Methods

Reference: any textbook on linear algebra: eigenvalues & eigenvectors: difference equation

Difference equation example:

Suppose you invest $1,000 at 6% interest in year 0.

If you have P(k) in year k, how much money would you have in year k+1?

$$P^{(k+1)} = 1.06 P^{(k)}$$

Suppose you invest P(0) in year 0, how much would you have in year k?

$$P^{(k)} = 1.06^k P^{(0)}$$

Same applies to matrix equations:

$$\vec{u}^{(k+1)} = A\vec{u}^{(k)} \qquad \Rightarrow \qquad \vec{u}^{(k)} = A^k \vec{u}^{(0)} = S\Lambda^k S^{-1}\vec{u}^{(0)}$$

S: eigenvector matrix

Λ: eigenvalue matrix

Let's say we want u(k) to go to the zero vector:

$$\lim_{k\to\infty} \vec{u}^{(k)} = \vec{0} \qquad \text{only if} \qquad |\lambda_i| < 1 \qquad \text{(spectral radius } \rho = \max|\lambda_i|\text{)}$$

how quickly u(k) approaches the zero vector depends **only** on the **largest value of λ**, i.e. the **spectral radius**

In class we will discuss two different ways to describe iterative methods

- Matrix form: **used only for theoretical derivations, spectral radius**
- Index form: **used to code the method**

Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

- Idea: Split A into $A = A_1 - A_2$

$$A\vec{\varphi} = \vec{b} \quad \Leftrightarrow \quad (A_1 - A_2)\vec{\varphi} = \vec{b} \quad \Leftrightarrow \quad A_1\vec{\varphi} = A_2\vec{\varphi} + \vec{b}$$

➡ to find the solution, iteratively solve this

$$A_1\vec{\varphi}^{(k+1)} = A_2\vec{\varphi}^{(k)} + \vec{b}$$

iteration counter/index

- This works, if

  1) A_1 is easily invertible

$$\vec{\varphi}^{(k+1)} = A_1^{-1}A_2\vec{\varphi}^{(k)} + A_1^{-1}\vec{b}$$

  2) Iterations converge to true solution

$$\lim_{k\to\infty} \vec{\varphi}^{(k)} = \vec{\varphi}$$

How can we ensure this? let's define the error:

$$\vec{\varepsilon}^{(k)} = \vec{\varphi} - \vec{\varphi}^{(k)} \quad \Rightarrow \quad \lim_{k\to\infty} \vec{\varepsilon}^{(k)} = \vec{0}$$

$$A_1\vec{\varphi} = A_2\vec{\varphi} + \vec{b}$$

$$\vec{\varepsilon}^{(k)} = \vec{\varphi} - \vec{\varphi}^{(k)}$$

$$- \quad A_1\vec{\varphi}^{(k+1)} = A_2\vec{\varphi}^{(k)} + \vec{b}$$

$$A_1\left(\vec{\varphi} - \vec{\varphi}^{(k+1)}\right) = A_2\left(\vec{\varphi} - \vec{\varphi}^{(k)}\right)$$

$$A_1\vec{\varepsilon}^{(k+1)} = A_2\,\vec{\varepsilon}^{(k)}$$

$$\vec{\varepsilon}^{(k+1)} = A_1^{-1}A_2\vec{\varepsilon}^{(k)}$$

now use linear algebra

$$\vec{\varepsilon}^{(k)} = \left(A_1^{-1}A_2\right)^k\vec{\varepsilon}^{(0)}$$

from linear algebra we also know:

$$\lim_{k\to\infty}\vec{\varepsilon}^{(k)} = 0 \quad \text{if} \quad \rho\left(A_1^{-1}A_2\right) = |\lambda_i|_{max} < 1$$

ρ : spectral radius

λ_i : eigenvalues of $\left(A_1^{-1}A_2\right)$

⇒ the smaller the spectral radius ρ, the faster the convergence!

- We thus need
  - split of A into $A = A_1 - A_2$ with
    - A_1 easily invertible
    - $\max|\lambda_i| < 1$

Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

Point Jacobi Method

- Let $A_1 = D$: diagonal entries of A
  - in our example:

$$A_1 = -4I$$

  - A_1^{-1} is easy: $A_1^{-1} = -\frac{1}{4}I$

$$A = \begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 \end{bmatrix}$$

we had: $\vec{\varphi}^{(k+1)} = A_1^{-1}A_2\vec{\varphi}^{(k)} + A_1^{-1}\vec{b}$

$$\Rightarrow \quad \vec{\varphi}^{(k+1)} = -\frac{1}{4}A_2\vec{\varphi}^{(k)} - \frac{1}{4}\vec{b} \qquad \text{with} \quad A_2 = A_1 - A = -4I - A$$

$$\Rightarrow \quad \varphi_{i,j}^{(k+1)} = \frac{1}{4}\left(\varphi_{i-1,j}^{(k)} + \varphi_{i+1,j}^{(k)} + \varphi_{i,j-1}^{(k)} + \varphi_{i,j+1}^{(k)}\right) - \frac{1}{4}b_{i,j}$$

Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

Point Jacobi Method

- Let $A_1 = D$: diagonal entries of A

  - What are the eigenvalues of $\left(A_1^{-1} A_2\right)$?

$$\lambda_{m,n} = \frac{1}{2}\left[\cos\left(\frac{m\pi}{M}\right) + \cos\left(\frac{n\pi}{N}\right)\right] \qquad \begin{matrix} m = 1, \ldots, M-1 \\ n = 1, \ldots, N-1 \end{matrix}$$

  - expand cosines [...]:

$$|\lambda|_{\max} = 1 - \frac{1}{4}\left(\frac{\pi^2}{M^2} + \frac{\pi^2}{N^2}\right) + \cdots \qquad \text{in 1D: } |\lambda|_{\max} = \cos\left(\frac{\pi}{N}\right)$$

➡ if $M,N \uparrow \Rightarrow \rho \rightarrow 1$ (but always $\rho < 1$)

➡ slow convergence ⇒ Point Jacobi seldom used in practice

Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

Point Jacobi Method

- How to code?
  - use 2 storage vectors, one for $\vec{\varphi}^{(k+1)}$ and one for $\vec{\varphi}^{(k)}$

```
while not converged (or loop from k=1 to #requested iterations)
  loop from j = 1 to N-1
    loop from i = 1 to M-1
      phinew(i,j) = 0.25*(phi(i-1,j)+phi(i+1,j)+...
    end loop i
  end loop j
  phi = phinew
end while
```

$$\varphi_{i,j}^{(k+1)} = \frac{1}{4}\left(\varphi_{i-1,j}^{(k)} + \varphi_{i+1,j}^{(k)} + \varphi_{i,j-1}^{(k)} + \varphi_{i,j+1}^{(k)}\right) - \frac{1}{4}b_{i,j}$$