First Model Problem

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

Step 1: Define Solution Domain

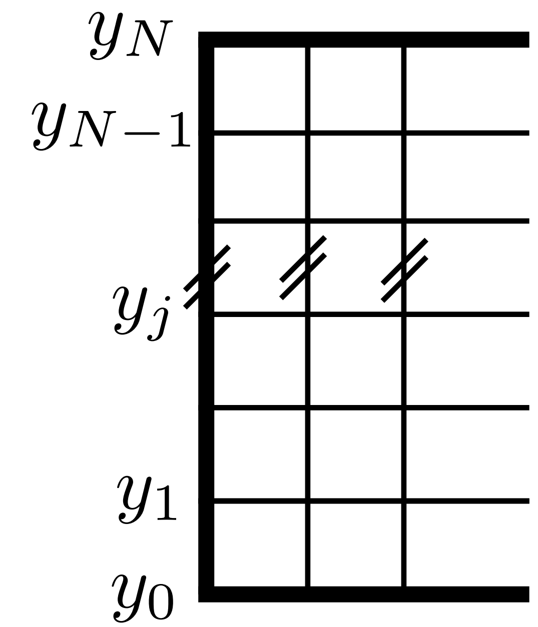
Step 2: Define Mesh

Step 3: Approximate Spatial Derivatives

Step 4: Substitute into PDE

Step 5: Incorporate Boundary Conditions

Step 5: Incorporate Boundary Conditions

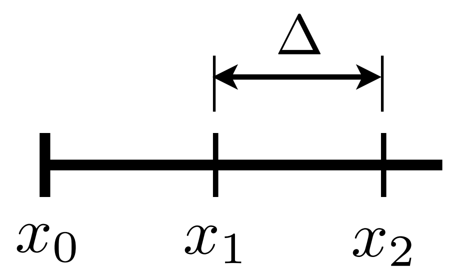


- Example #2:
  - Neumann at x_0: $\left.\dfrac{\partial\varphi}{\partial x}\right|_{x_0} = g(y)$

    - approximate derivative by one-sided finite difference formula

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

stencil:



    - 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*

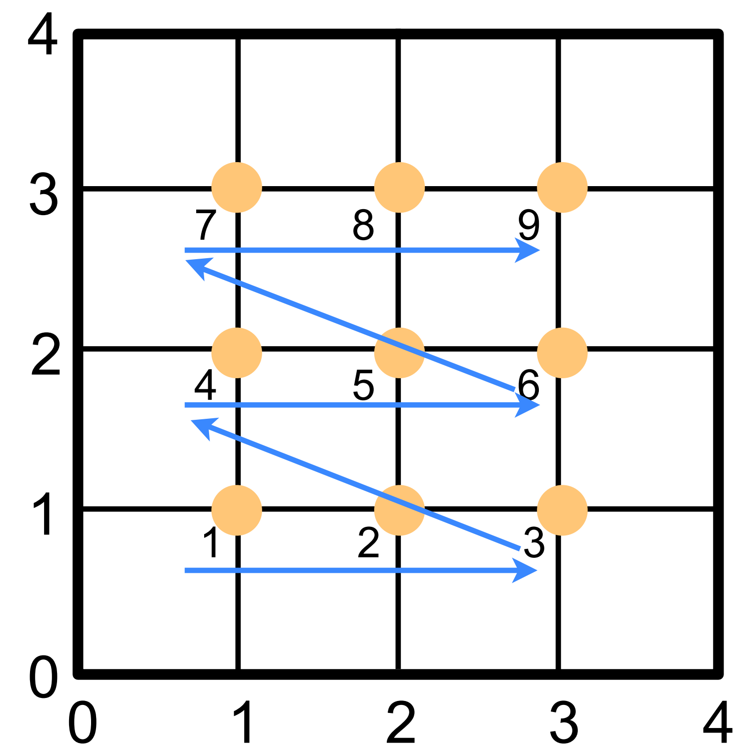
$$\frac{2}{3}\varphi_{2,j} - \frac{8}{3}\varphi_{1,j} + \varphi_{1,j+1} + \varphi_{1,j-1} = \Delta^2 f_{i,j} + \frac{2}{3}\Delta g_j$$

➡ Neumann boundary conditions modify both sides!

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

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

➡ the solution vector φ covers 2D space
$\Rightarrow$ need to define a sorting

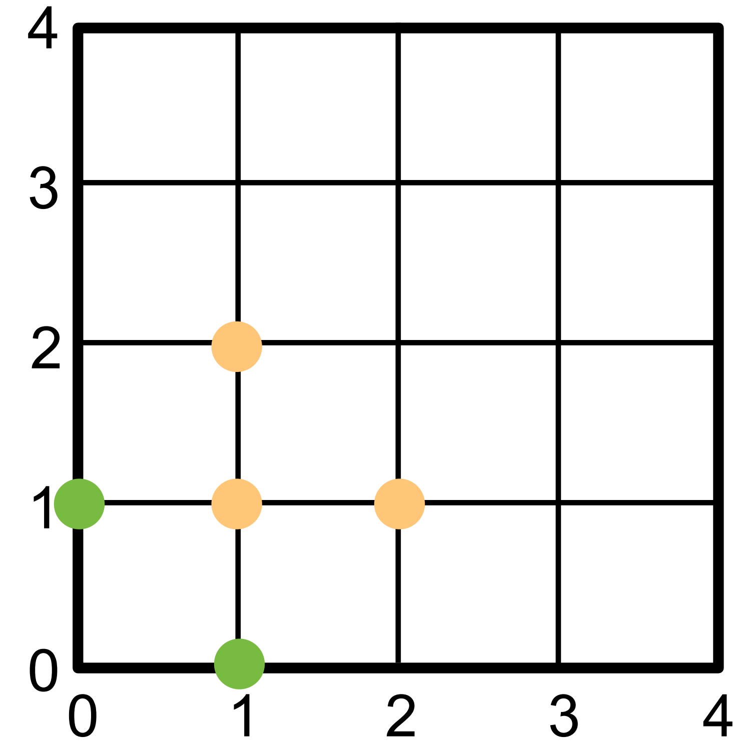


Step 6: Matrix Form (only for illustration, never code!)

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



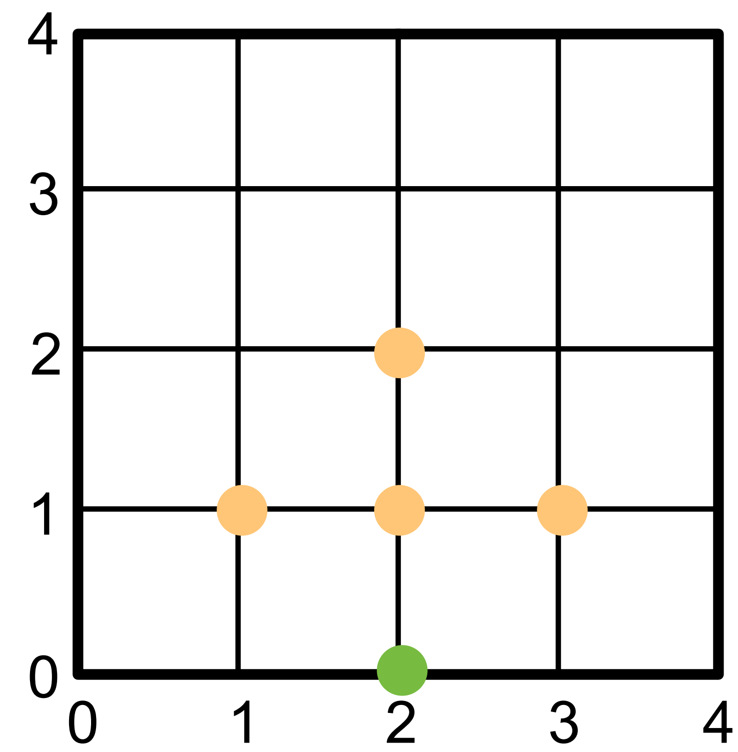
Step 6: Matrix Form (only for illustration, never code!)

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

➡ the solution vector φ covers 2D space
$\Rightarrow$ need to define a sorting

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

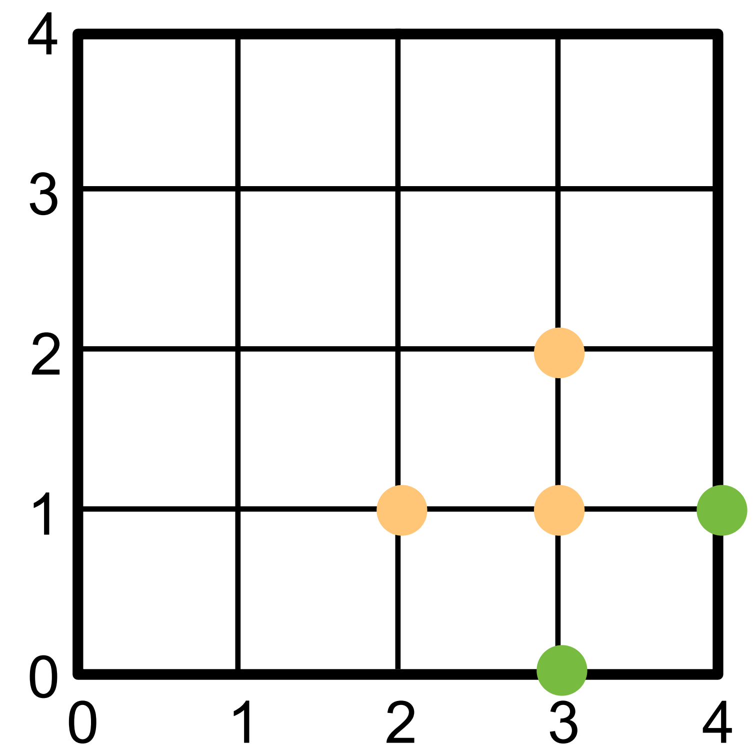
(2,1): $\varphi_{3,1} - 4\varphi_{2,1} + \varphi_{1,1} + \varphi_{2,2} = \Delta^2 f_{2,1} - \varphi_{2,0}$



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

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

➡ the solution vector φ covers 2D space
$\Rightarrow$ need to define a sorting

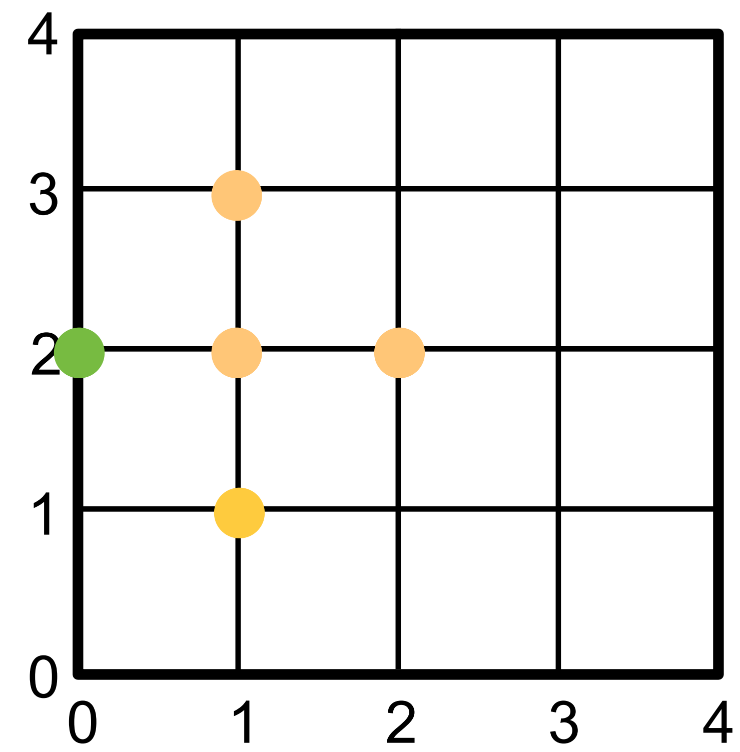


(1,1): $\varphi_{2,1} - 4\varphi_{1,1} + \varphi_{1,2} = \Delta^2 f_{1,1} - \varphi_{0,1} - \varphi_{1,0}$

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

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



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

➡ the solution vector φ covers 2D space
$\Rightarrow$ need to define a sorting

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

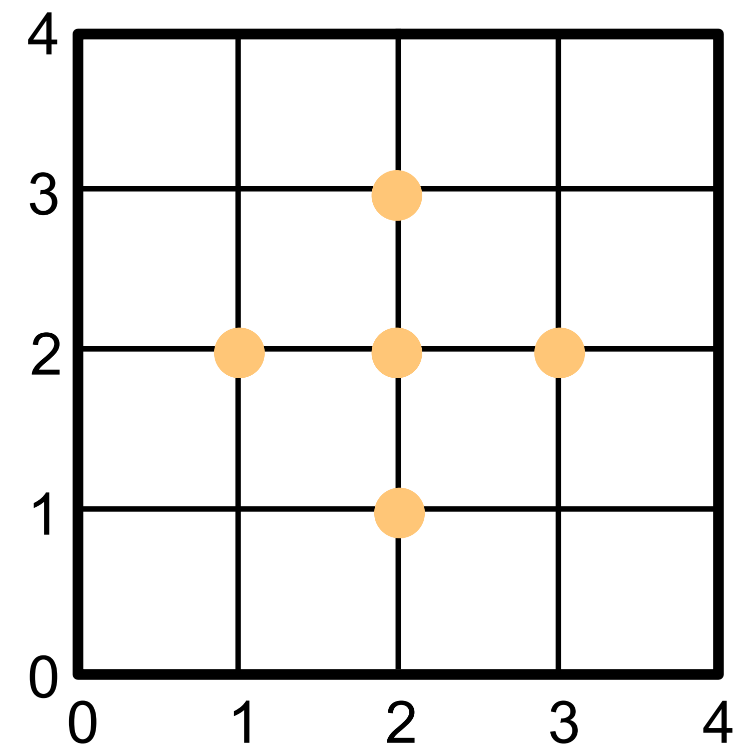
(2,1): $\varphi_{3,1} - 4\varphi_{2,1} + \varphi_{1,1} + \varphi_{2,2} = \Delta^2 f_{2,1} - \varphi_{2,0}$

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

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

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



➡ the solution vector φ covers 2D space
$\Rightarrow$ 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): $-4\varphi_{3,1} + \varphi_{2,1} + \varphi_{3,2} = \Delta^2 f_{3,1} - \varphi_{4,1} - \varphi_{3,0}$

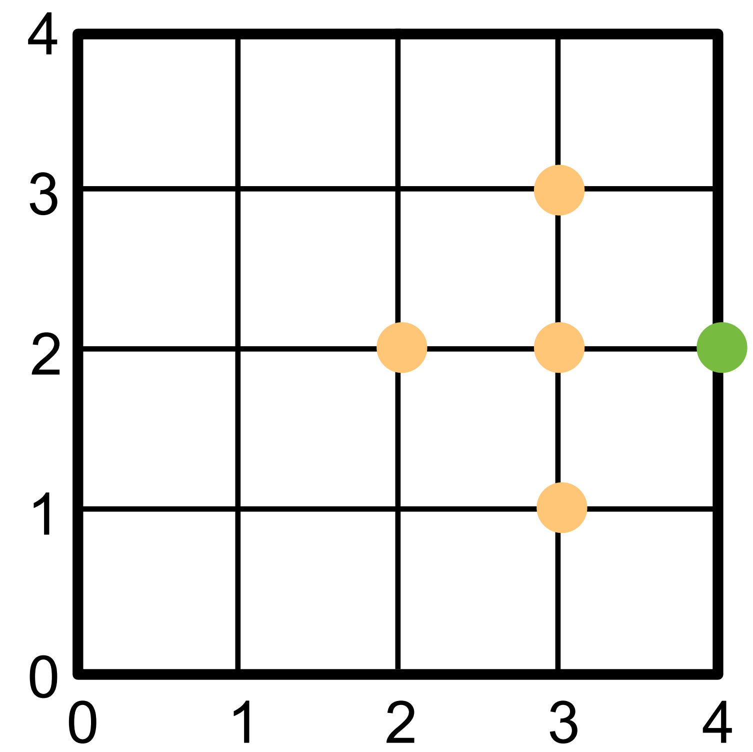
(1,2): $\varphi_{2,2} - 4\varphi_{1,2} + \varphi_{1,3} + \varphi_{1,1} = \Delta^2 f_{1,2} - \varphi_{0,2}$

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

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

- Example:
  - use Dirichlet boundary conditions
  - $M{=}N{=}4 \Rightarrow$ 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): $-4\varphi_{3,1} + \varphi_{2,1} + \varphi_{3,2} = \Delta^2 f_{3,1} - \varphi_{4,1} - \varphi_{3,0}$

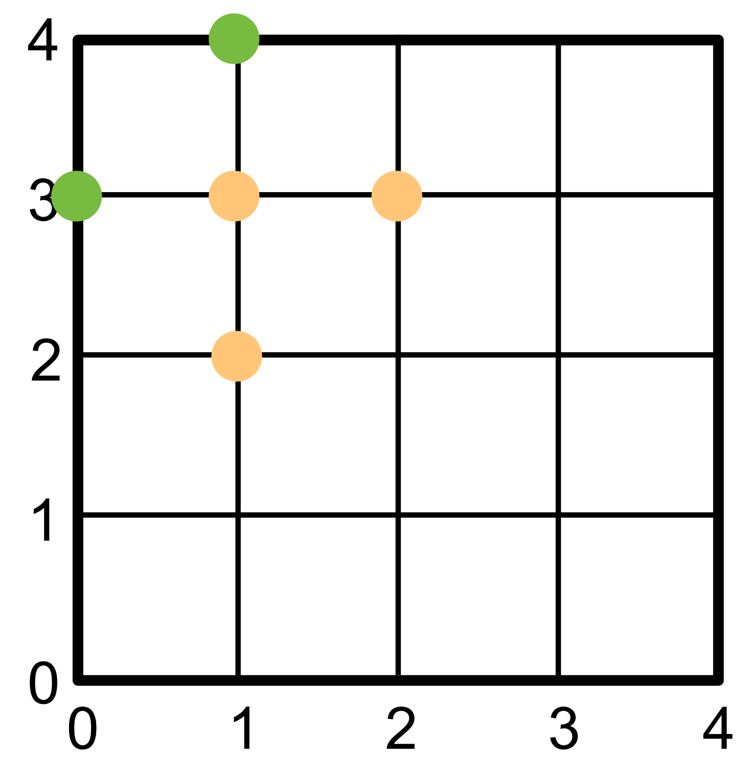
(1,2): $\varphi_{2,2} - 4\varphi_{1,2} + \varphi_{1,3} + \varphi_{1,1} = \Delta^2 f_{1,2} - \varphi_{0,2}$

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

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

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



➡ the solution vector φ covers 2D space
$\Rightarrow$ 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): $\varphi_{3,2} - 4\varphi_{2,2} + \varphi_{1,2} + \varphi_{2,3} + \varphi_{2,1} = \Delta^2 f_{2,2}$

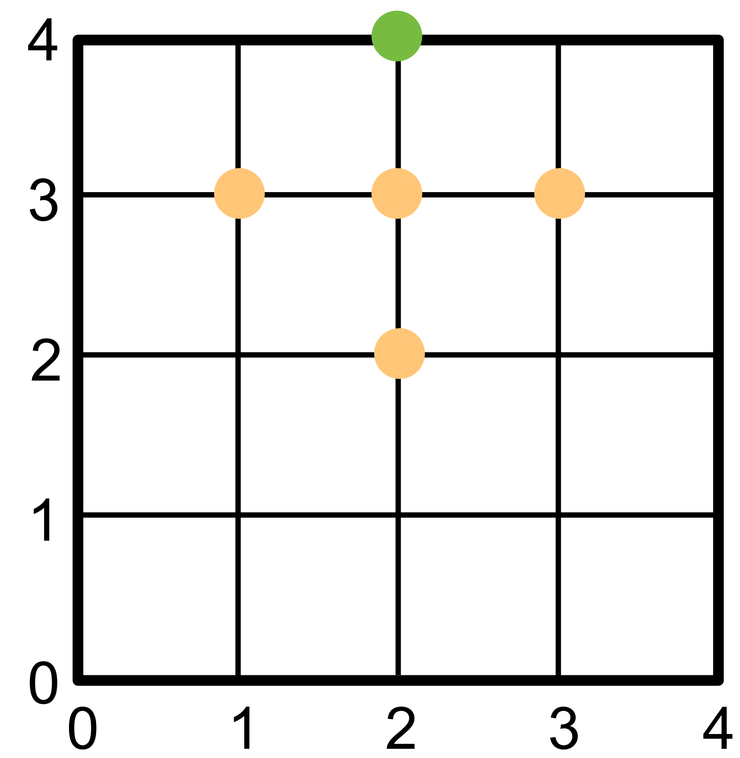
(3,2): $-4\varphi_{3,2} + \varphi_{2,2} + \varphi_{3,3} + \varphi_{3,1} = \Delta^2 f_{3,2} - \varphi_{4,2}$

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

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

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

➡ the solution vector φ covers 2D space
$\Rightarrow$ 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): $-4\varphi_{3,2} + \varphi_{2,2} + \varphi_{3,3} + \varphi_{3,1} = \Delta^2 f_{3,2} - \varphi_{4,2}$

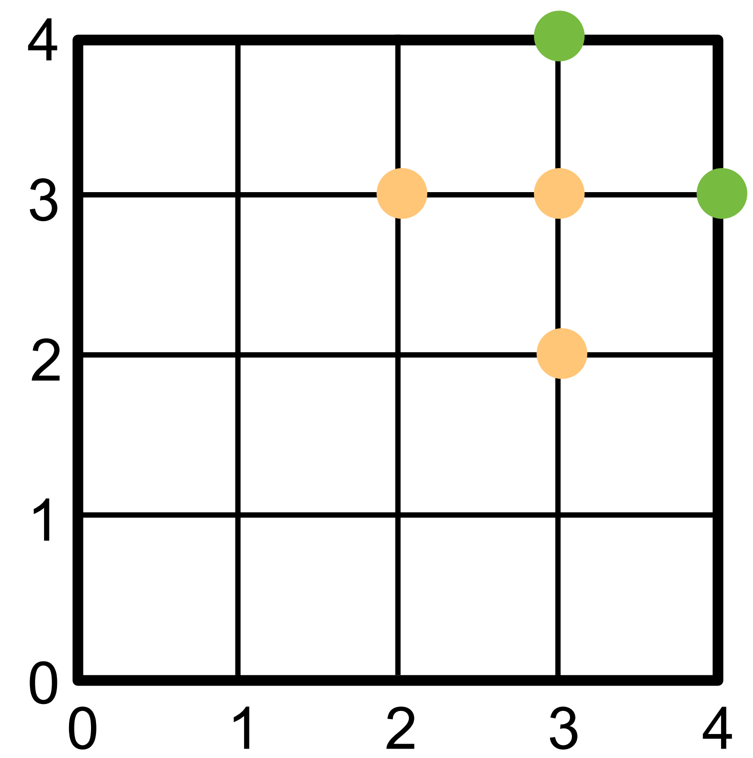
(1,3): $\varphi_{2,3} - 4\varphi_{1,3} + \varphi_{1,2} = \Delta^2 f_{1,3} - \varphi_{0,3} - \varphi_{1,4}$

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

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

- Example:
  - use Dirichlet boundary conditions
  - $M=N=4 \Rightarrow$ 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:

$$\left[\begin{array}{ccc|ccc|ccc}
-4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4
\end{array}\right]
\begin{bmatrix} \varphi_{1,1} \\ \varphi_{2,1} \\ \varphi_{3,1} \\ \varphi_{1,2} \\ \varphi_{2,2} \\ \varphi_{3,2} \\ \varphi_{1,3} \\ \varphi_{2,3} \\ \varphi_{3,3} \end{bmatrix}
=
\begin{bmatrix}
\Delta^2 f_{1,1} - \varphi_{0,1} - \varphi_{1,0} \\
\Delta^2 f_{2,1} - \varphi_{2,0} \\
\Delta^2 f_{3,1} - \varphi_{4,1} - \varphi_{3,0} \\
\Delta^2 f_{1,2} - \varphi_{0,2} \\
\Delta^2 f_{2,2} \\
\Delta^2 f_{3,2} - \varphi_{4,2} \\
\Delta^2 f_{1,3} - \varphi_{0,3} - \varphi_{1,4} \\
\Delta^2 f_{2,3} - \varphi_{2,4} \\
\Delta^2 f_{3,3} - \varphi_{4,3} - \varphi_{3,4}
\end{bmatrix}$$

➡ 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(N^3)
  - solution only available after O(N^3) operations at the very end!

➡ Iterative methods are typically preferable

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

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

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

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 $\left(A_1^{-1} A_2\right)$?

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

  - expand cosines [...]:

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

in 1D: $|\lambda|_{\max} = \cos\left(\frac{\pi}{N}\right)$

➡ if $M,N \uparrow \Rightarrow \rho \to 1$ (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
```

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

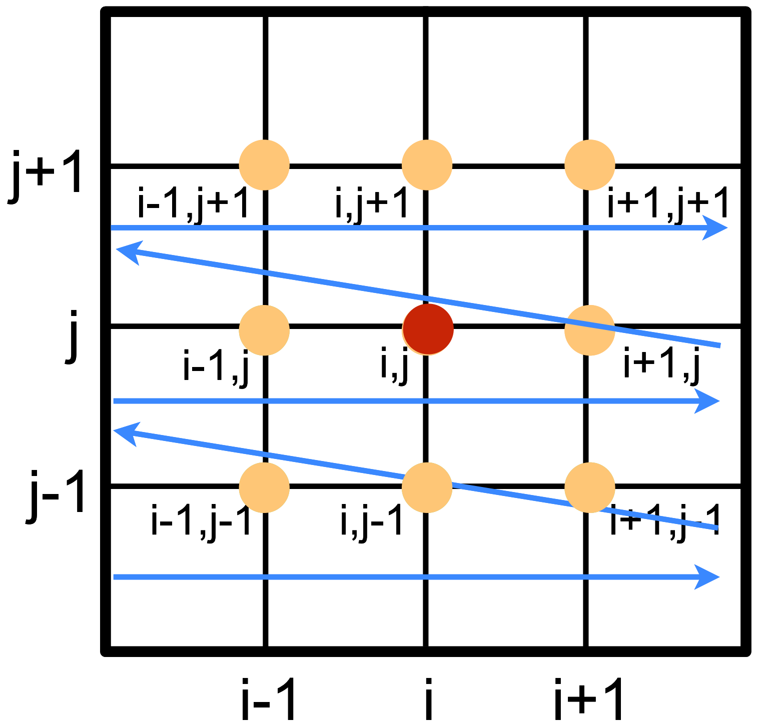
Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

Gauss-Seidel Method

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

- In Point-Jacobi, we use only $\vec{\varphi}^{(k)}$ to calculate $\vec{\varphi}^{(k+1)}$
- Idea: Why don't we use $\vec{\varphi}^{(k+1)}$ instead of $\vec{\varphi}^{(k)}$ for nodes we have already calculated?

  for our sorting, for $\vec{\varphi}_{i,j}^{(k+1)}$ this would be $\vec{\varphi}_{i,j-1}^{(k+1)}$ and $\vec{\varphi}_{i-1,j}^{(k+1)}$



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

➡ to code: use just one storage vector!

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

Gauss-Seidel Method

- How to code?
  - use 1 storage vector that gets overwritten

```
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
       phi(i,j) = 0.25*(phi(i-1,j)+phi(i+1,j)+...
     end loop i
  end loop j
end while
```

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

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

Gauss-Seidel Method

L: - lower triangular part of A
U: - upper triangular part of A
Note: $A \neq LU$

- in matrix notation: $A_1 = D - L$ and $A_2 = U$
  - What are the eigenvalues of $\left(A_1^{-1} A_2\right)$?

$$\lambda_{m,n} = \frac{1}{4}\left[\cos\left(\frac{m\pi}{M}\right) + \cos\left(\frac{n\pi}{N}\right)\right]$$

$$m = 1, \ldots, M-1$$
$$n = 1, \ldots, N-1$$

  - factor 2 faster than Point Jacobi (and need only half the storage!)

PJ: $\lambda_{m,n} = \frac{1}{2}\left[\cos\left(\frac{m\pi}{M}\right) + \cos\left(\frac{n\pi}{N}\right)\right]$