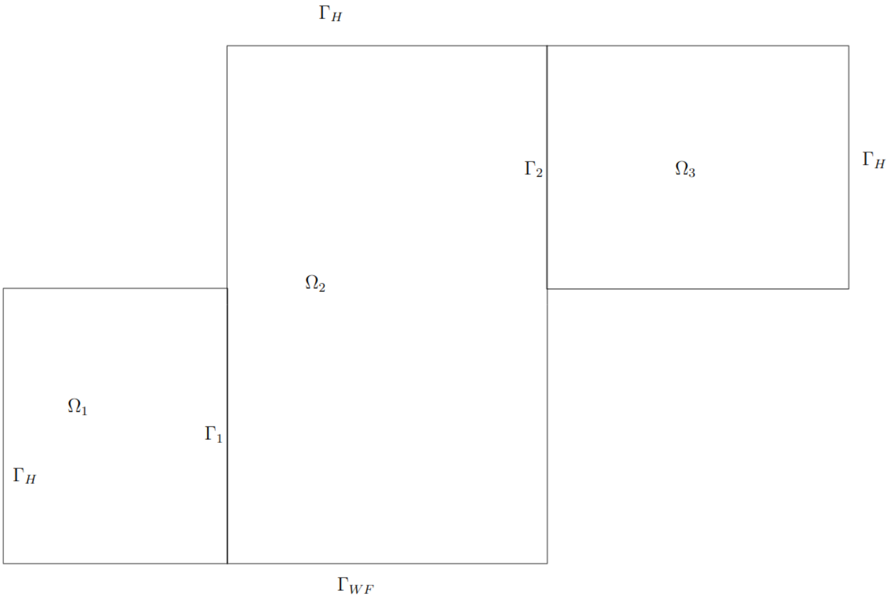
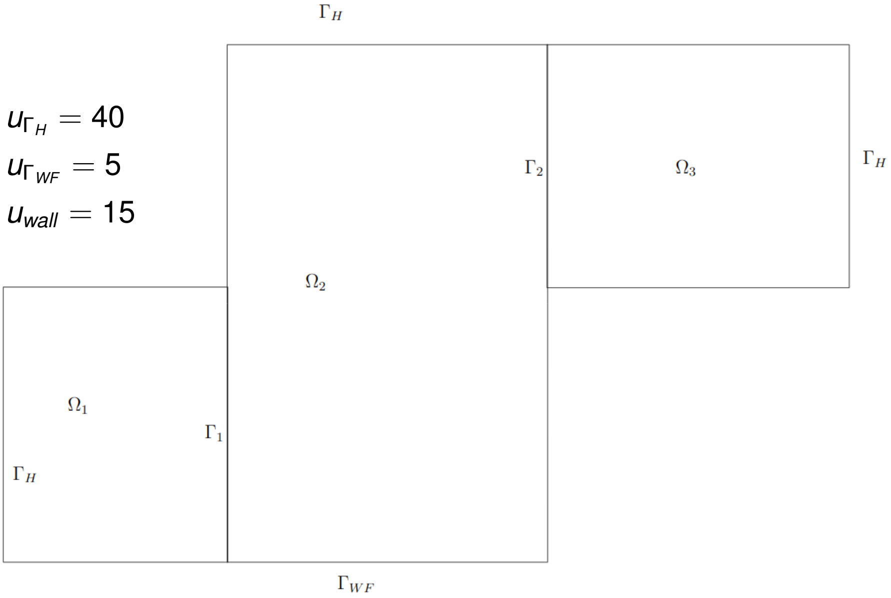
LUND UNIVERSITY

Advanced Numerical Algorithms with Python

PROJECT 3


Γ_H
Γ_2
Ω_3
Γ_H
Ω_2
Ω_1
Γ_1
Γ_H
Γ_{WF}

$$u_{\Gamma_H} = 40$$
$$u_{\Gamma_{WF}} = 5$$
$$u_{wall} = 15$$



Laplace equation

$$\Delta u(\mathbf{x}) = u_{xx} + u_{yy} = 0, \qquad \mathbf{x} = (x, y) \in \Omega \subset \mathbb{R}^2$$

LUND UNIVERSITY

Laplace equation

$$\Delta u(\mathbf{x}) = u_{xx} + u_{yy} = 0, \qquad \mathbf{x} = (x, y) \in \Omega \subset \mathbb{R}^2$$

- Discretize

LUND UNIVERSITY

Laplace equation

$$\Delta u(\mathbf{x}) = u_{xx} + u_{yy} = 0, \qquad \mathbf{x} = (x, y) \in \Omega \subset \mathbb{R}^2$$

- Discretize
- Parallelize

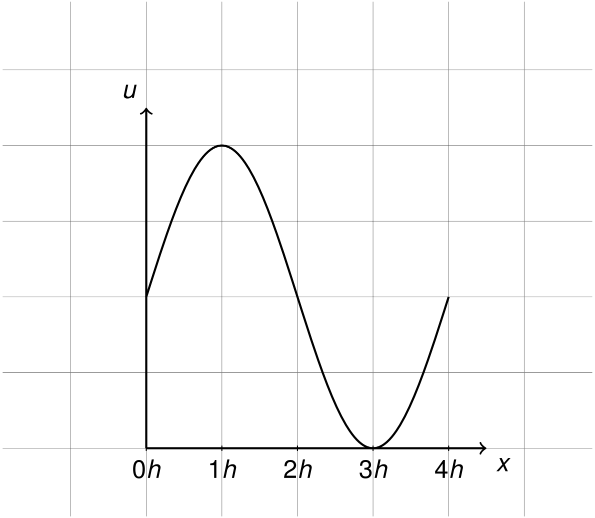
Laplace equation

$$\Delta u(\mathbf{x}) = u_{xx} + u_{yy} = 0, \qquad \mathbf{x} = (x, y) \in \Omega \subset \mathbb{R}^2$$

- Discretize
- Parallelize
- Solve

LUND UNIVERSITY

Discretization: 1D



Discretization: Taylor expansion

Taylor expand $u(x_{i+1})$ around $u(x_i)$:

$$u(x_{i+1}) = u(x_i) + hu'(x_i) + \frac{h^2}{2}u''(x_i) + \dots$$

The problem

Discretization

Parallelization

Discretization: Taylor expansion

Taylor expand $u(x_{i+1})$ around $u(x_i)$:

$$u(x_{i+1}) = u(x_i) + hu'(x_i) + \frac{h^2}{2}u''(x_i) + \dots$$

Solve for $u'(x_i)$:

$$u'(x_i) = \frac{u(x_{i+1}) - u(x_i)}{h} + \mathcal{O}(h)$$

LUND UNIVERSITY

Discretization: Taylor expansion

Taylor expand $u(x_{i+1})$ around $u(x_i)$:

$$u(x_{i+1}) = u(x_i) + hu'(x_i) + \frac{h^2}{2}u''(x_i) + \dots$$

Solve for $u'(x_i)$:

$$u'(x_i) = \frac{u(x_{i+1}) - u(x_i)}{h} + \mathcal{O}(h)$$

Choose discretization:

$$v_i' = \frac{v_{i+1} - v_i}{h}$$

Here, v denotes the discrete solution variable.

2nd derivative

Use the finite difference approximation applied the first derivative in order to obtain an approximation for the second derivative:

$$v_i'' = \frac{v_i' - v_{i-1}'}{h} = \frac{\frac{v_{i+1}-v_i}{h} - \frac{v_i - v_{i-1}}{h}}{h} = \frac{v_{i+1} - 2v_i + v_{i-1}}{h^2}$$

2nd derivative

Use the finite difference approximation applied the first derivative in order to obtain an approximation for the second derivative:

$$v_i'' = \frac{v_i' - v_{i-1}'}{h} = \frac{\frac{v_{i+1} - v_i}{h} - \frac{v_i - v_{i-1}}{h}}{h} = \frac{v_{i+1} - 2v_i + v_{i-1}}{h^2}$$

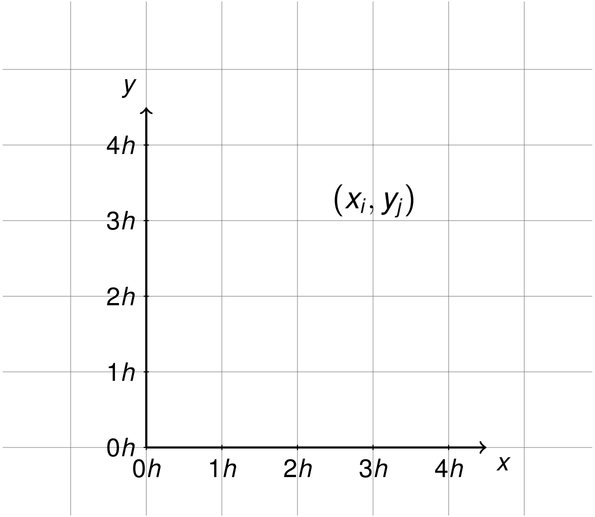
<u>Exercise</u>: What is the accuracy of the approximation?

Discretization: 2D

The problem

Discretization

Parallelization



LUND UNIVERSITY

Discretization: 2D

Simplifying assumption: Use grid spacing h in both dimensions.
Repeating the same execrise in x and y direction gives approximation

$$\frac{v_{i+1,j} + v_{i-1,j} - 4v_{i,j} + v_{i,j+1} + v_{i,j-1}}{h^2}$$

Discretization: 2D

Simplifying assumption: Use grid spacing h in both dimensions.
Repeating the same execrise in x and y direction gives approximation

$$\frac{v_{i+1,j} + v_{i-1,j} - 4v_{i,j} + v_{i,j+1} + v_{i,j-1}}{h^2}$$

Collect in a matrix: $\Delta u \approx A\mathbf{v}$

$$A = \frac{1}{h^2} \left[\begin{array}{ccccccccc}
 & & & \ddots & & & & & \\
1 & \dots 0 \dots & 1 & -4 & 1 & \dots 0 \dots & 1 & & \\
 & 1 & \dots 0 \dots & 1 & -4 & 1 & \dots 0 \dots & 1 & \\
 & & 1 & \dots 0 \dots & 1 & -4 & 1 & \dots 0 \dots & 1 \\
 & & & & & \ddots & & &
\end{array} \right]$$

Discretization: 2D

Simplifying assumption: Use grid spacing h in both dimensions.
Repeating the same execrise in x and y direction gives approximation

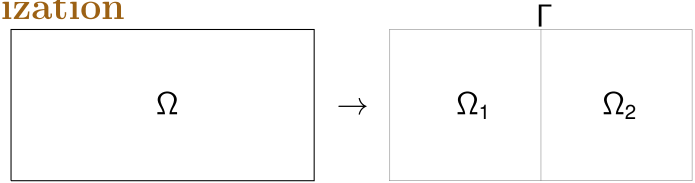
$$\frac{v_{i+1,j} + v_{i-1,j} - 4v_{i,j} + v_{i,j+1} + v_{i,j-1}}{h^2}$$

Collect in a matrix: $\Delta u \approx A\mathbf{v}$

$$A = \frac{1}{h^2}\begin{bmatrix} & & & & \ddots & & & & & \\ 1 & \ldots 0 \ldots & 1 & & -4 & 1 & \ldots 0 \ldots & 1 & & \\ & 1 & \ldots 0 \ldots & & 1 & -4 & 1 & \ldots 0 \ldots & 1 & \\ & & 1 & & \ldots 0 \ldots & 1 & -4 & 1 & \ldots 0 \ldots & 1 \\ & & & & & & \ddots & & & \end{bmatrix}$$

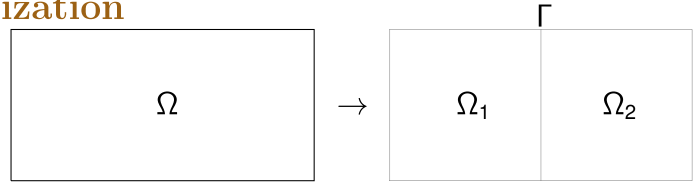
<u>Exercise</u>: What changes if $h_x \neq h_y$?

Parallelization



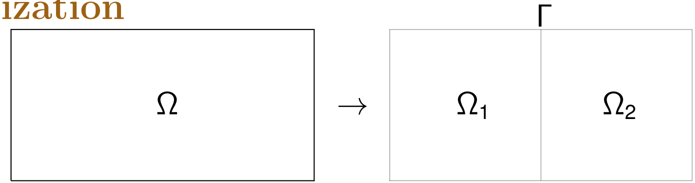
- We could solve the problem on a single processor (left)
  → Slow if h is small.

Parallelization



- We could solve the problem on a single processor (left)
  $\rightarrow$ Slow if h is small.
- Use two (or more) processors (right)
  $\rightarrow$ **Domain decomposition**

Parallelization



- We could solve the problem on a single processor (left)
  → Slow if h is small.
- Use two (or more) processors (right)
  → **Domain decomposition**
- The interface Γ is artificial
  → No information available from the other side of Γ
  → The problem has fundamentally changed

Dirichlet-Neumann Iteration

- We need a method to solve the decomposed problem such that the solution converges to that of the original problem.

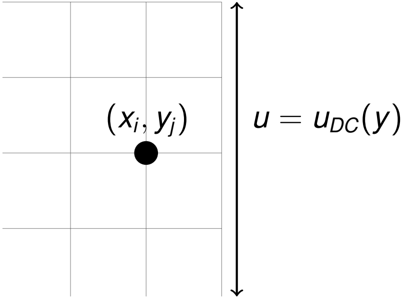
Dirichlet-Neumann Iteration

- We need a method to solve the decomposed problem such that the solution converges to that of the original problem.
- Communication between processor 1 and 2 should only happen through Γ.

Dirichlet-Neumann Iteration

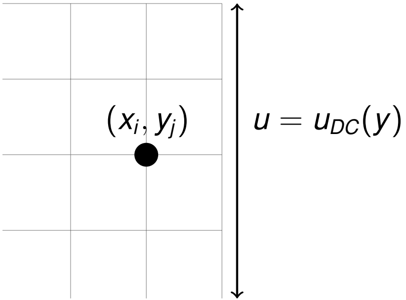
- We need a method to solve the decomposed problem such that the solution converges to that of the original problem.
- Communication between processor 1 and 2 should only happen through Γ.
- Solution: Alternate between using Dirichlet and Neumann conditions.

Dirichlet condition



$$\Delta u(x_i, y_j) \approx \frac{v_{i,j-1} + v_{i,j+1} - 4v_{i,j} + v_{i-1,j} + u_{DC}(y_j)}{h^2} = 0$$

Dirichlet condition

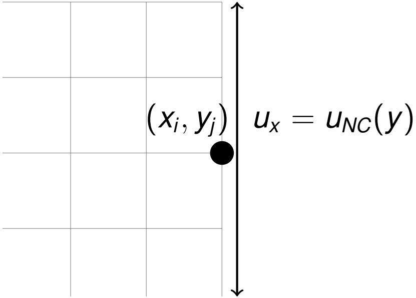


$$\Delta u(x_i, y_j) \approx \frac{v_{i,j-1} + v_{i,j+1} - 4v_{i,j} + v_{i-1,j} + u_{DC}(y_j)}{h^2} = 0$$

Move data over to right-hand side:

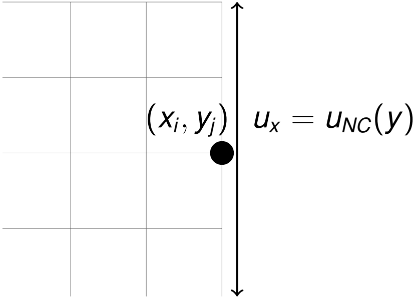
$$\frac{v_{i,j-1} + v_{i,j+1} - 4v_{i,j} + v_{i-1,j}}{h^2} = -\frac{u_{DC}(y_j)}{h^2}$$

Neumann condition



$$\Delta u(x_i, y_j) \approx \frac{\frac{v_{i,j+1} - v_{i,j}}{h} - \frac{v_{i,j} - v_{i,j-1}}{h} + u_{NC}(y_j) - \frac{v_{i,j} - v_{i-1,j}}{h}}{h} = 0$$

Neumann condition

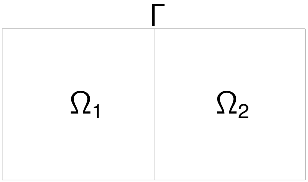


$$\Delta u(x_i, y_j) \approx \frac{\frac{v_{i,j+1} - v_{i,j}}{h} - \frac{v_{i,j} - v_{i,j-1}}{h} + u_{NC}(y_j) - \frac{v_{i,j} - v_{i-1,j}}{h}}{h} = 0$$

Move data over to right-hand side:

$$\frac{v_{i,j+1} + v_{i,j-1} - 3v_{i,j} + v_{i-1,j}}{h^2} = -\frac{u_{NC}(y_j)}{h}$$

Dirichlet-Neumann Iteration



- Step 1: On Ω_2 - Obtain $\mathbf{v}_\Gamma^k$. Send to Ω_1.
- Step 2: On Ω_1 - Receive $\mathbf{v}_\Gamma^k$ from Ω_2. Solve left system with $\mathbf{v}_\Gamma^k$ as Dirichlet condition. From $\mathbf{v}_{\Omega_1}^{k+1}$, compute Neumann condition at Γ. Send to Ω_2.
- Step 3: On Ω_2 - Receive data from Ω_1. Solve right system with Neumann condition. Obtain $\mathbf{v}_{\Omega_2}^{k+1}$.
- Step 4: Relax - $\mathbf{v}_{\Omega_{1,2}}^{k+1} \leftarrow \omega \mathbf{v}_{\Omega_{1,2}}^{k+1} + (1-\omega)\mathbf{v}_{\Omega_{1,2}}^{k}$.