# HPC Software II
# 55612

**Darach Golden**

January 20, 2025

# HPC Software II (55612)

Darach Golden

# Course Outline

# Poisson Eqn

Let $\Omega$ be the region $[0,1] \times [0,1]$.

$$\nabla^2 u(x,y) = \frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y)\,, \quad 0 \le x, y \le 1\,.$$

with $u(x,y) = \phi(x,y)$ on $\partial\Omega$ (Dirichlet boundary conditions).

## Example

Let $f \equiv 0$. Let the Dirichlet boundary conditions be: $u(x,0) \equiv 0$, $u(x,1) = \frac{1}{(1+x)^2+1}$ $(0 \le x \le 1)$, and $u(0,y) = \frac{y}{1+y^2}$, $u(1,y) = \frac{y}{4+y^2}$ $(0 \le y \le 1)$.
Then the solution is:

$$u(x,y) = \frac{y}{(1+x)^2 + y^2}\,, \quad 0 \le x, y \le 1\,.$$

# Poisson Eqn

## Another example

Let $\Omega$ be the region $[0,1]\times[0,1]$.

$$\nabla^2 u(x,y) = \frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y)\,, \quad 0 \le x, y \le 1\,.$$

with $u(x,y) = \phi(x,y)$ on $\partial\Omega$ (Dirichlet boundary conditions).

### Example

Let $f \equiv 0$. Let the Dirichlet boundary conditions be:

$$u(x,y) = \begin{cases} x, & 0 \le x \le \frac{1}{2}\,,\ \ y = 0 \\ 1 - x, & \frac{1}{2} < x \le 1\,,\ \ y = 0 \\ 0, & 0 \le x \le 1\,,\ \ y = 1 \\ 0, & x = 0\,,\ \ 0 \le y \le 1 \\ 0, & x = 1\,,\ \ 0 \le y \le 1 \end{cases}$$

Then the exact solution can be written in terms of the **infinite series**:

$$u(x,y) = \frac{4}{\pi^2}\sum_{i=0}^{\infty}\frac{(-1)^i \sin{(2i+1)\pi x}\sinh{(2i+1)\pi(1-y)}}{(2i+1)^2\sinh{(2i+1)\pi}}$$

# Finite Difference Approximation

$u(x_i, y_j) = u_{i,j}$.

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+\Delta x, y) - 2u(x,y) + u(x-\Delta x, y)}{\Delta x^2} + \mathcal{O}(\Delta x^2)\,,$$
$$\frac{\partial^2 u}{\partial y^2} = \frac{u(x, y+\Delta y) - 2u(x,y) + u(x, y-\Delta y)}{\Delta y^2} + \mathcal{O}(\Delta y^2)\,.$$

Therefore

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{u(x+\Delta x, y) - 2u(x,y) + u(x-\Delta x, y)}{\Delta x^2} + \frac{u(x, y+\Delta y) - 2u(x,y) + u(x, y-\Delta y)}{\Delta y^2} + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2)$$

If $\Delta x = \Delta y = h$,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{u(x+h, y) + u(x-h, y) + u(x, y+h) + u(x, y-h) - 4u(x,y)}{h^2} + \mathcal{O}(h^2)\,.$$

In terms of $u_{i,j}$, this becomes

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} + \mathcal{O}(h^2)\,.$$

# Finite Difference Approximation

The equation

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y)\,, \quad 0 \le x, y \le 1 \;,$$

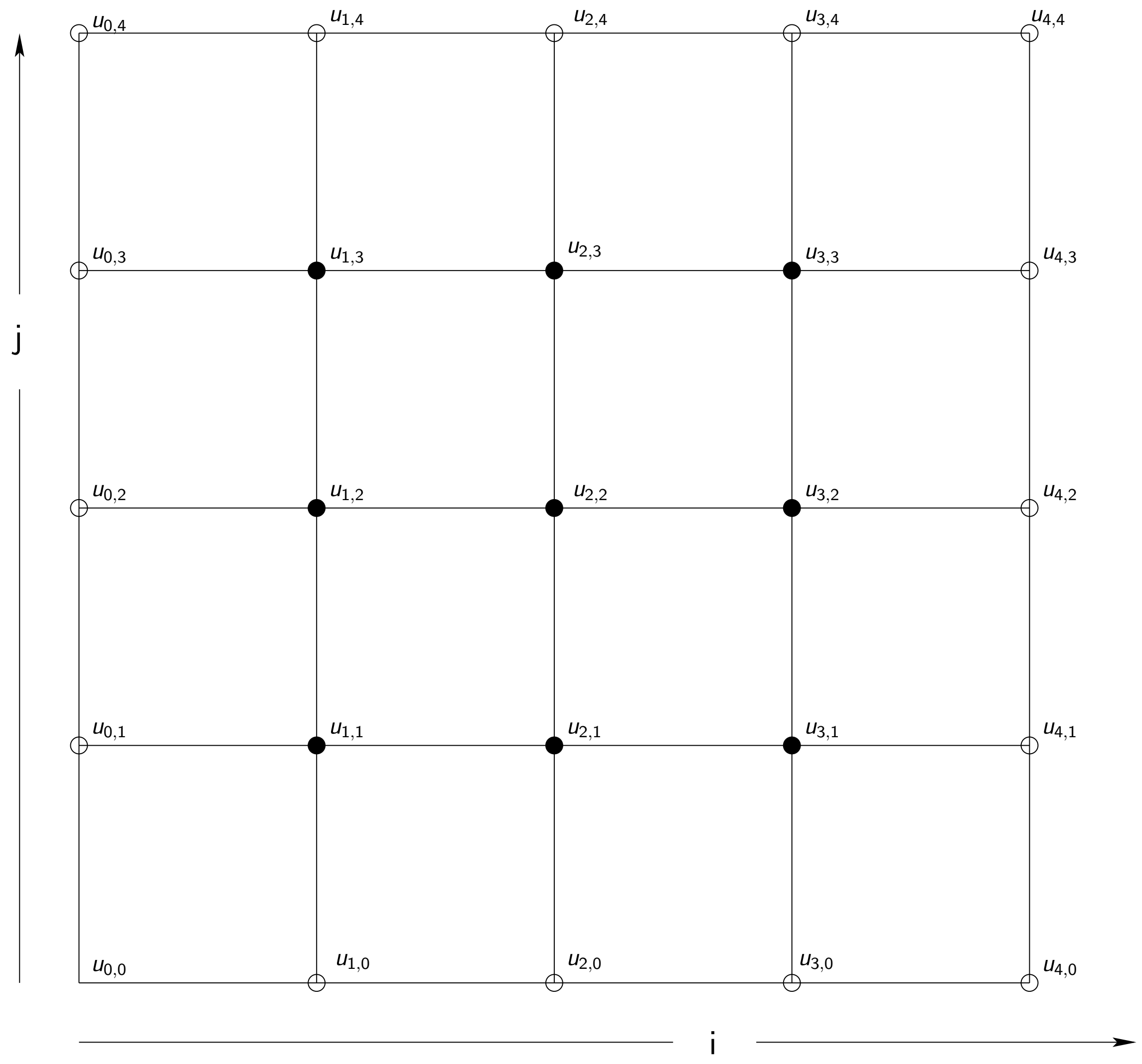
holds at every point in the domain $\Omega$.
The finite difference approximation to the equation should hold for each of the *internal* $u_{ij}$ (the boundary $u_{ij}$ are already specified). This means that the equation

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = h^2 f_{i,j}\;.$$

holds at every one of the internal points in the grid.

# Example: Approximation to $u$

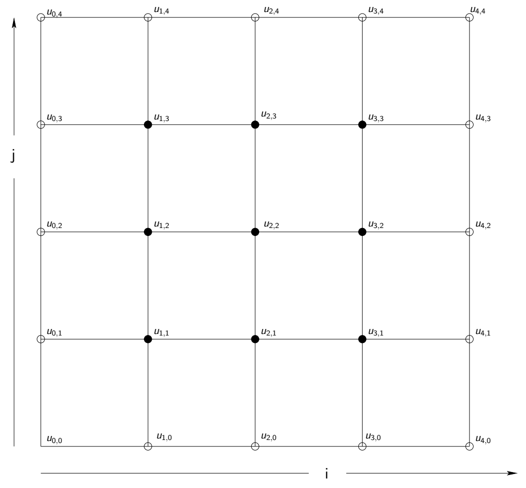
## Grid view ($n = 3$)



# Example: Approximation to $u$

## Grid view ($n = 3$)

$$
\begin{aligned}
u_{21} + u_{01} + u_{12} + u_{10} - 4u_{11} &= h^2 f_{11} \\
u_{31} + u_{11} + u_{22} + u_{20} - 4u_{21} &= h^2 f_{21} \\
u_{41} + u_{21} + u_{32} + u_{30} - 4u_{31} &= h^2 f_{31} \\
u_{22} + u_{02} + u_{13} + u_{11} - 4u_{12} &= h^2 f_{12} \\
u_{32} + u_{12} + u_{23} + u_{21} - 4u_{22} &= h^2 f_{22} \\
u_{42} + u_{22} + u_{33} + u_{31} - 4u_{32} &= h^2 f_{32} \\
u_{23} + u_{03} + u_{14} + u_{12} - 4u_{13} &= h^2 f_{13} \\
u_{33} + u_{13} + u_{24} + u_{22} - 4u_{23} &= h^2 f_{23} \\
u_{43} + u_{23} + u_{34} + u_{32} - 4u_{33} &= h^2 f_{33}
\end{aligned}
$$



# Example: Approximation to $u$

Grid view ($n = 3$)

$$
\begin{aligned}
u_{21} + u_{12} - 4u_{11} &= h^2 f_{11} - u_{01} - u_{10} \\
u_{31} + u_{11} + u_{22} - 4u_{21} &= h^2 f_{21} - u_{20} \\
u_{21} + u_{32} - 4u_{31} &= h^2 f_{31} - u_{41} - u_{30} \\
u_{22} + u_{13} + u_{11} - 4u_{12} &= h^2 f_{12} - u_{02} \\
u_{32} + u_{12} + u_{23} + u_{21} - 4u_{22} &= h^2 f_{22} \\
u_{22} + u_{33} + u_{31} - 4u_{32} &= h^2 f_{32} - u_{42} \\
u_{23} + u_{12} - 4u_{13} &= h^2 f_{13} - u_{03} - u_{14} \\
u_{33} + u_{13} + u_{22} - 4u_{23} &= h^2 f_{23} - u_{24} \\
u_{23} + u_{32} - 4u_{33} &= h^2 f_{33} - u_{43} - u_{34}
\end{aligned}
$$

# Example: Approximation to $u$
## Grid view ($n = 3$)

The above equation can be expressed in the form

$$Aw = \hat{f},$$

where

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \end{bmatrix} = \begin{bmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{13} \\ u_{23} \\ u_{33} \end{bmatrix}$$

and

$$\hat{f} = \begin{bmatrix} h^2 f_{11} - u_{01} - u_{10} \\ h^2 f_{21} - u_{20} \\ h^2 f_{31} - u_{41} - u_{30} \\ h^2 f_{12} - u_{02} \\ h^2 f_{22} \\ h^2 f_{32} - u_{42} \\ h^2 f_{13} - u_{03} - u_{14} \\ h^2 f_{23} - u_{24} \\ h^2 f_{33} - u_{43} - u_{34} \end{bmatrix}.$$

# Example: Approximation to $u$

## Grid view ($n = 3$)

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

which has the form

$$A = \begin{bmatrix} T & I & 0 \\ I & T & I \\ 0 & I & T \end{bmatrix},$$

where

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

and $I$ is the $3x3$ Identity matrix.

# Jacobi Method

Let a matrix $A \in \mathbb{R}^{n \times n}$.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \cdots & \vdots \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{bmatrix}.$$

We want to solve the linear system

$$Ax = b\,,$$

where $x, b \in \mathbb{R}^n$.

- This can be done in many ways.
- We are going to use the Jacobi method. This is an iterative method.
- We start with some initial $x^k$ and update $x$ multiple times until we are close the solution

## Jacobi Method

The Jacobi method moves from $x_i^k \to x_i^{k+1}$ by zeroing the residual component $(b - Ax^{k+1})_i$:

$$b_i - \sum_{j=1}^{n} a_{ij} x_j^{k+1} = 0\,,$$

The Jacobi method uses the following update $x_i^k \to x_i^{k+1}$:

$$a_{ii} x_i^{k+1} = b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij} x_j^{k}\,,$$

so

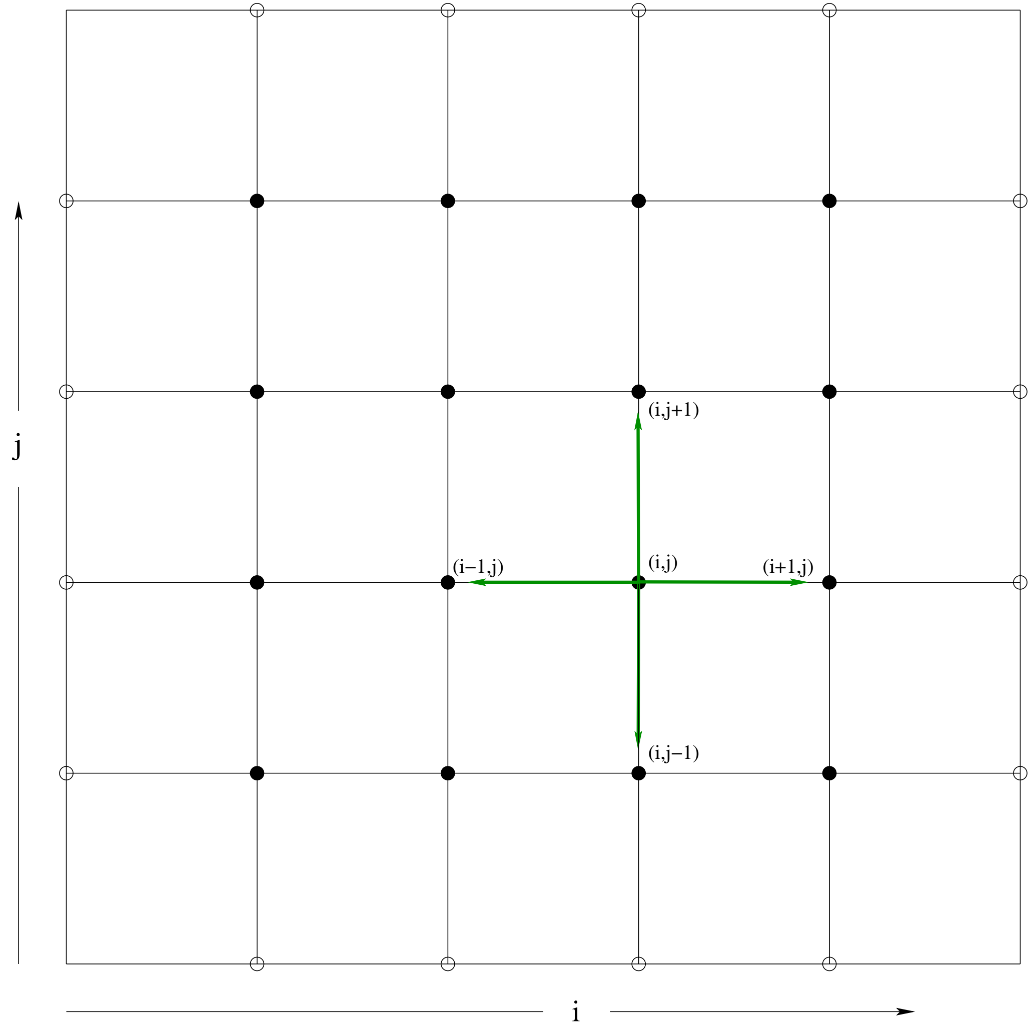
$$x_i^{k+1} = \frac{1}{a_{ii}} \left( b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij} x_j^{k} \right)\,.$$

When this formula is used in the special case of our finite difference matrix $A$, the formula becomes

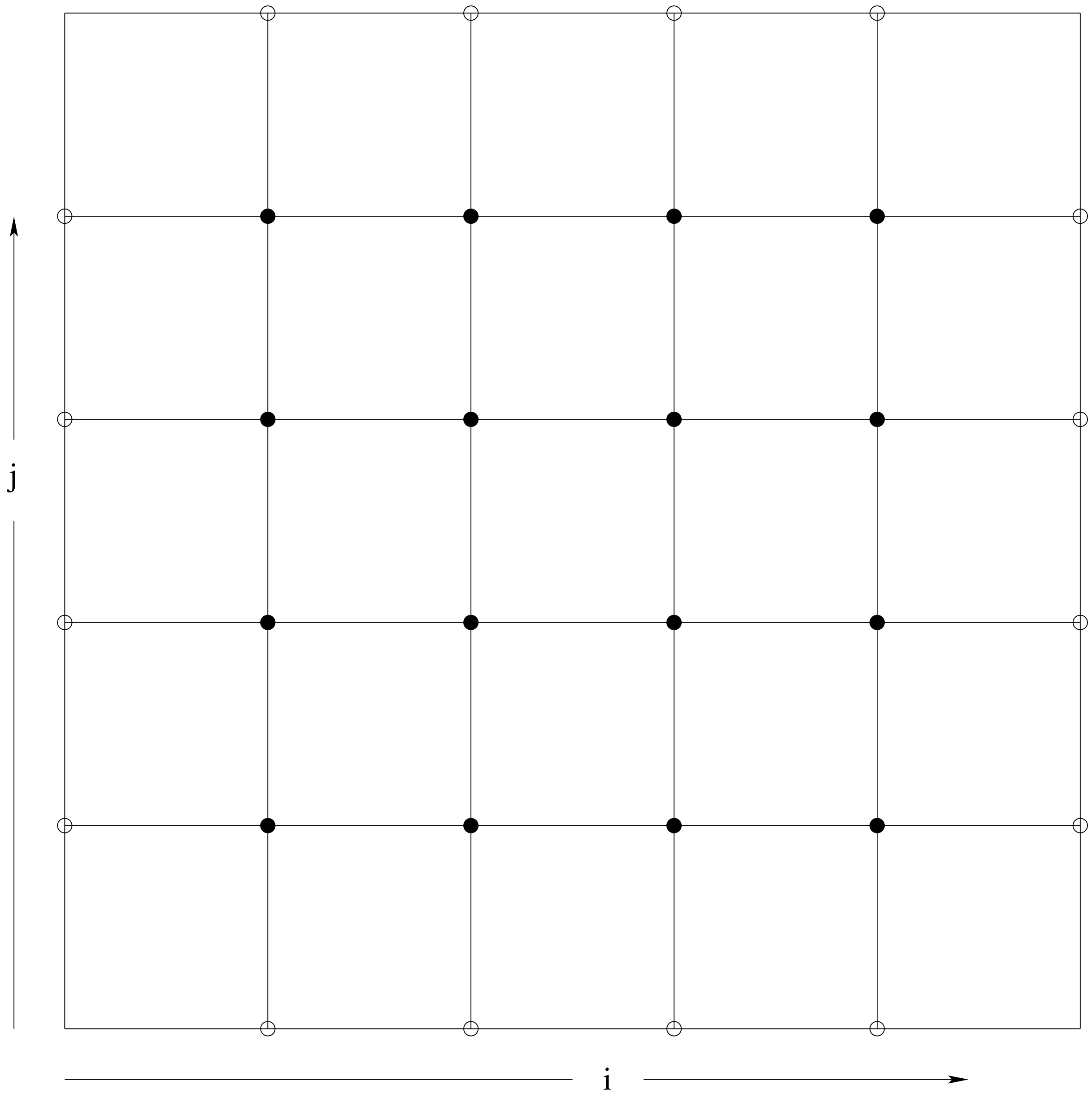
$$u_{ij}^{k+1} = \frac{1}{4}\left(u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k - h^2 f_{ij}\right)\,.$$

## Finite difference 5 point stencil

$$u_{ij}^{k+1} = \frac{1}{4}\left(u_{i+1,j}^{k} + u_{i-1,j}^{k} + u_{i,j+1}^{k} + u_{i,j-1}^{k} - h^2 f_{ij}\right) \; .$$

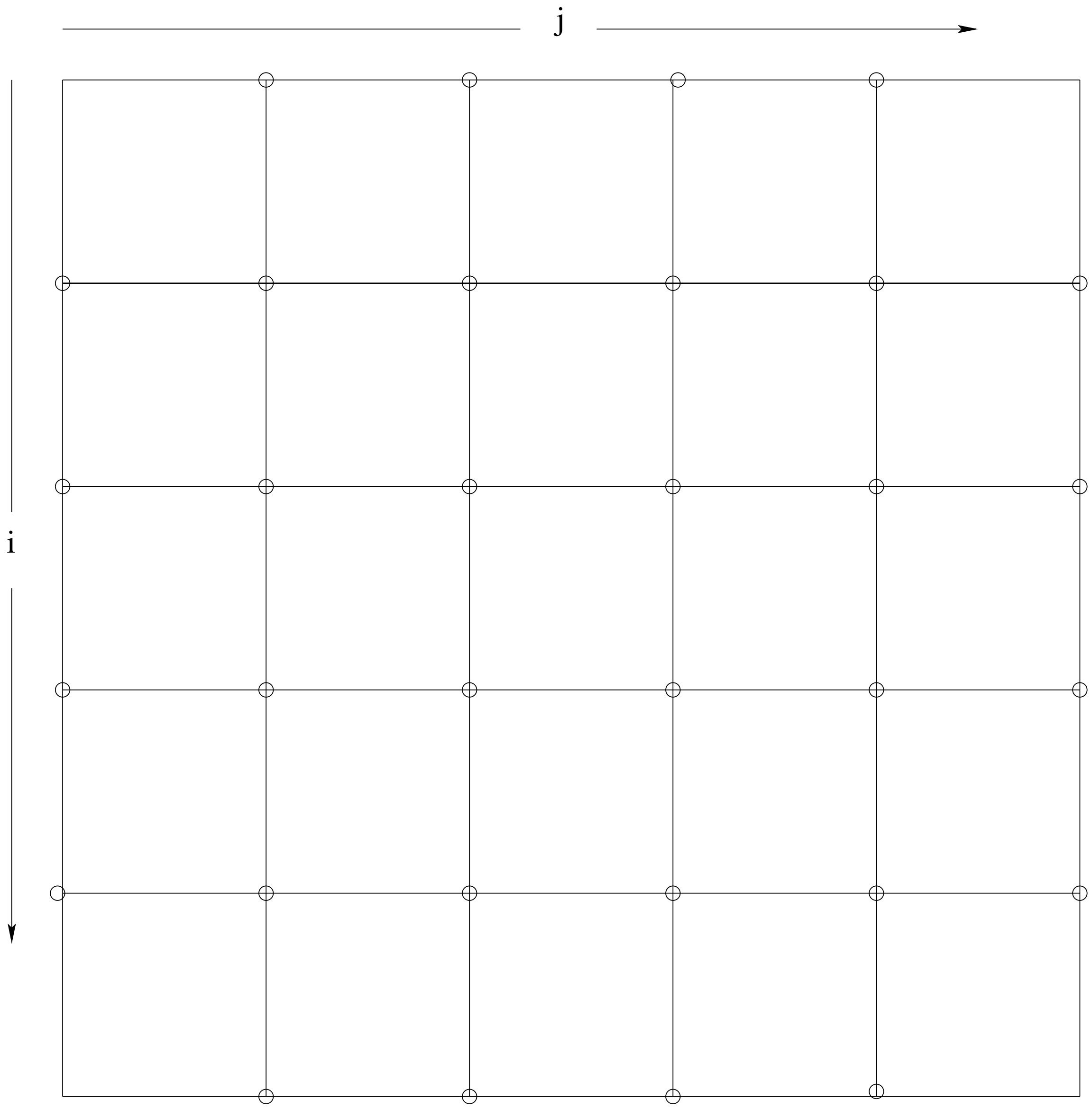


## Mesh/Grid View

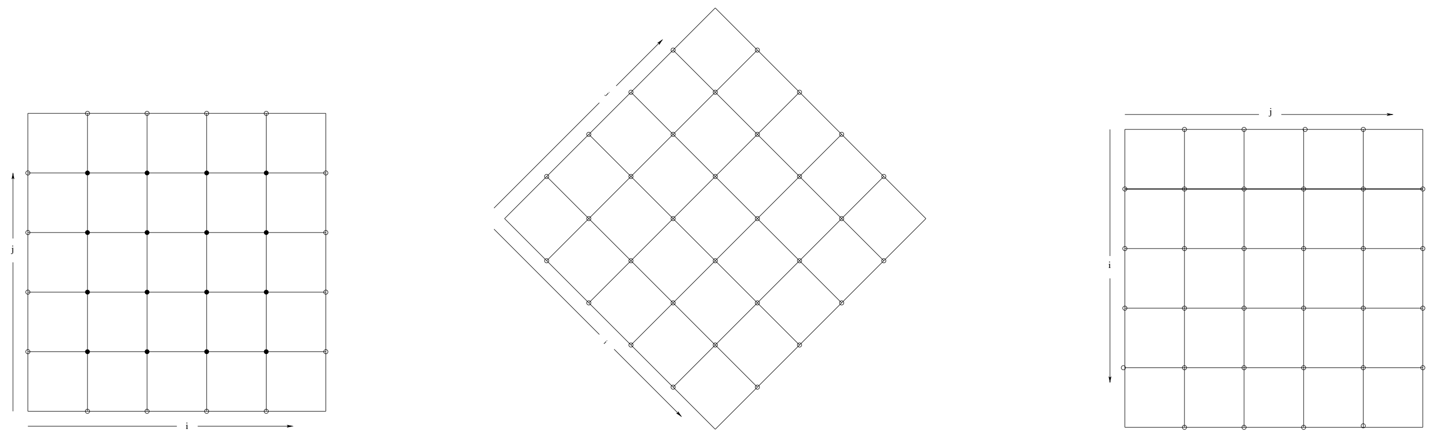


# Matrix View

## C and Fortran ordering



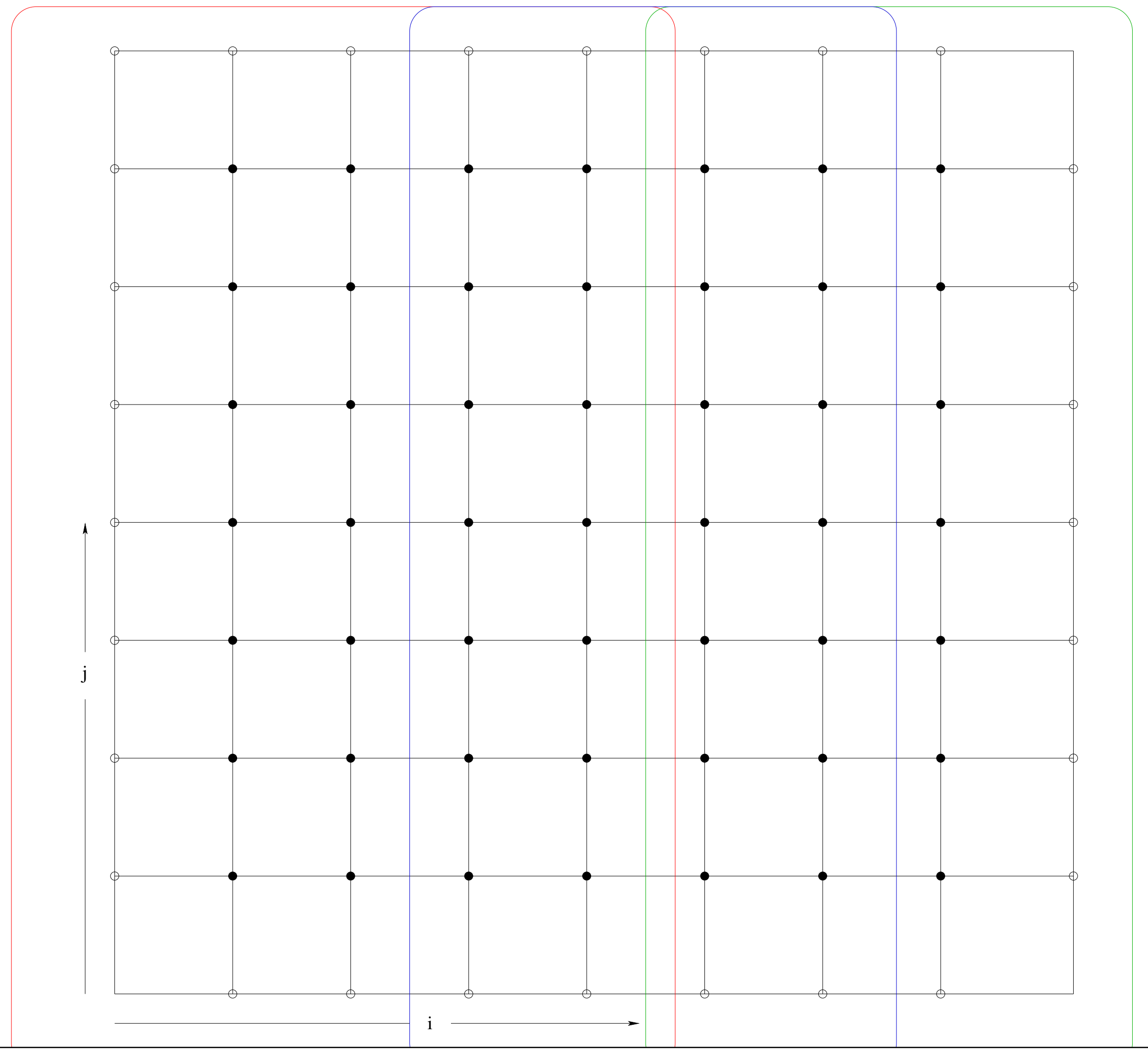
# Mesh/Grid → Matrix View



# Partitioning

## Grid view

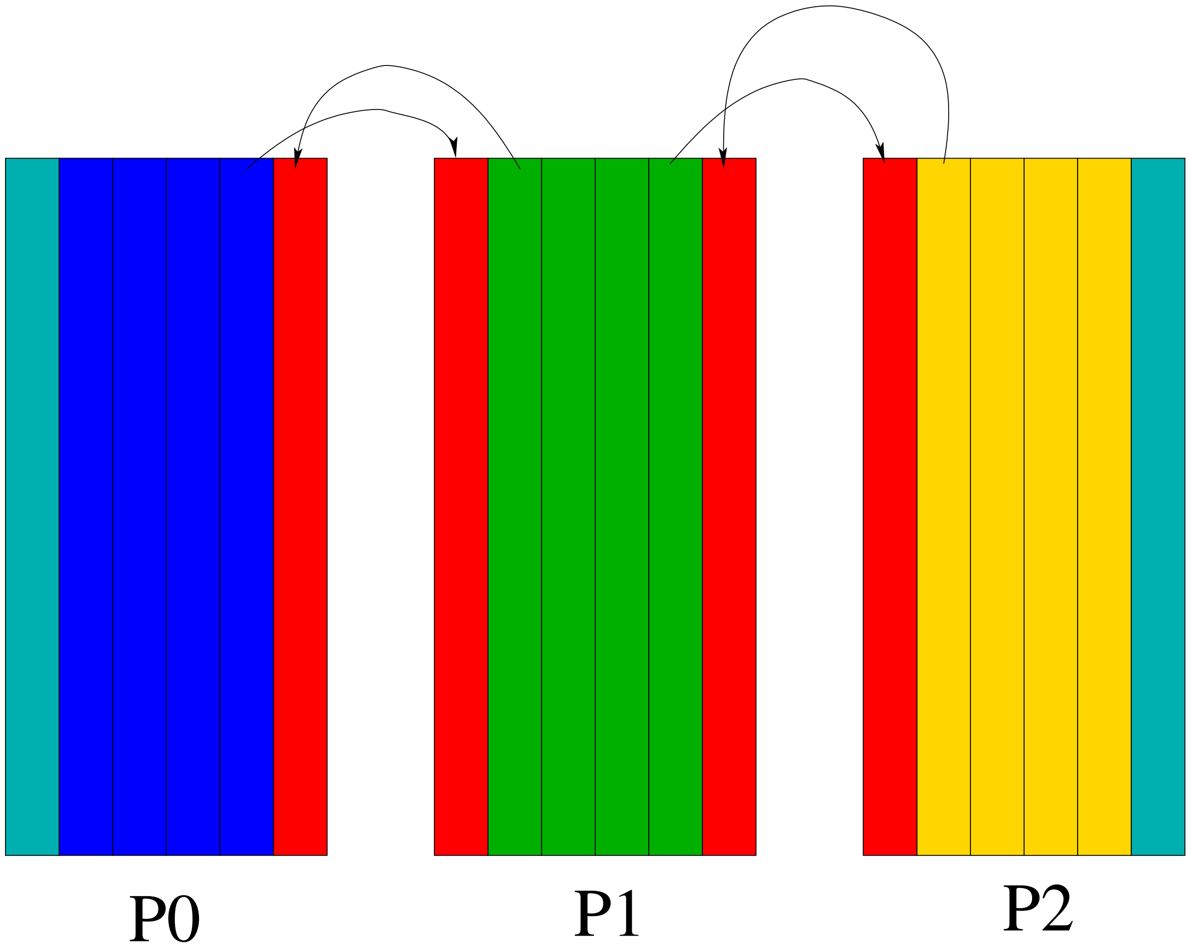
- 2D *Grid*
- **1D Partition** of grid



# Partitioning

## Grid view

| | |
|---|---|
| Red | Ghost or "halo" columns |
| dark blue, green, yellow | Columns assigned to each processor |
| light blue | Left/right boundaries |
| not shown | Top and bottom boundaries |



# Bad Ordering of Sends and Recvs

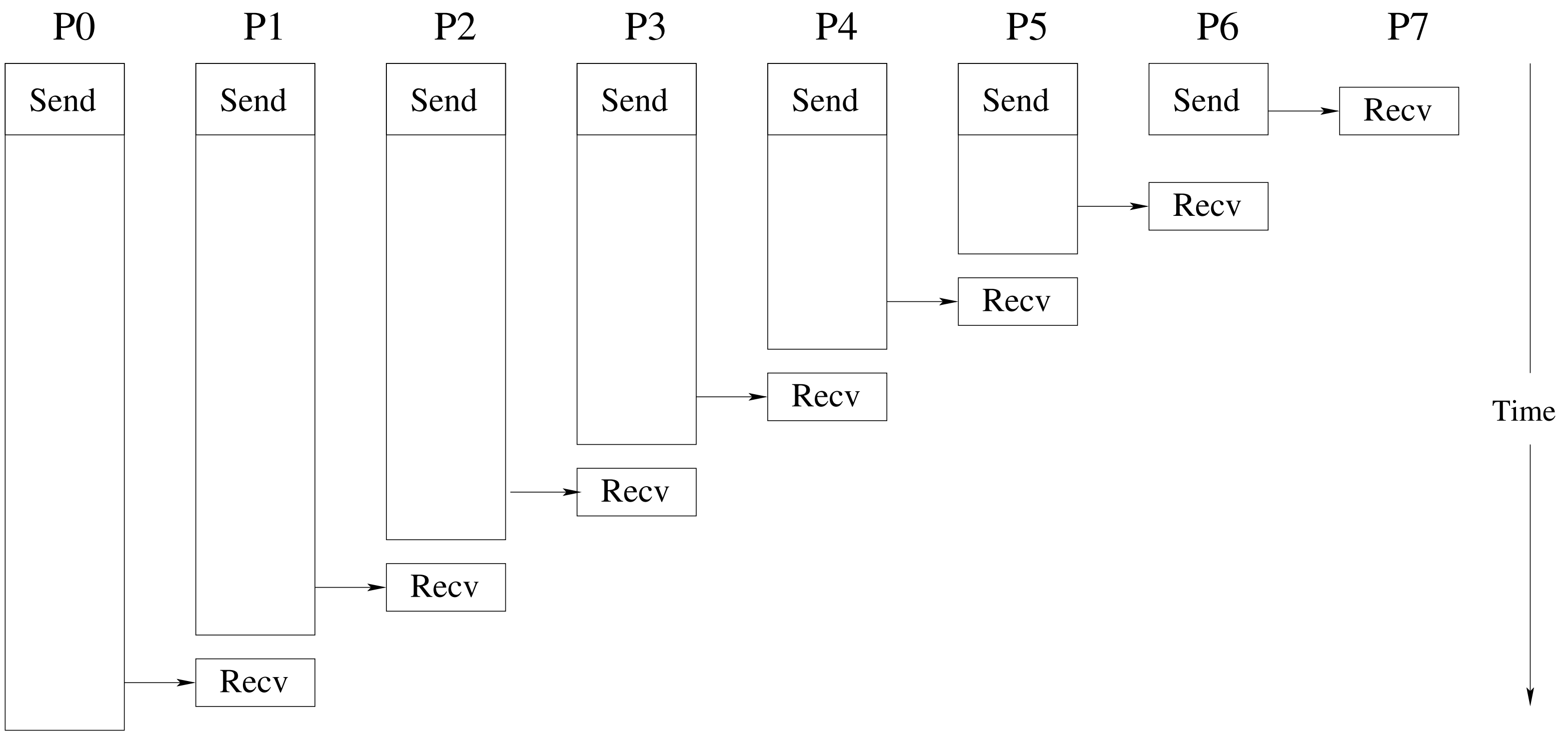
## Exchanging Ghost columns

```
MPI_Send(&x[e][1], ny, MPI_DOUBLE, nbrright, 0, comm);
MPI_Recv(&x[s-1][1], ny, MPI_DOUBLE, nbrleft, 0, comm,
         MPI_STATUS_IGNORE);

MPI_Send(&x[s][1], ny, MPI_DOUBLE, nbrleft, 1, comm);
MPI_Recv(&x[e+1][1], ny, MPI_DOUBLE, nbrright, 1, comm,
         MPI_STATUS_IGNORE);
```

# Sequentialization

## Sending to the right: case of no buffering



## Interleaved Sends and Recvs

```
coord = rank;
if(coord%2 == 0){

  MPI_Ssend(&x[e][1], nx, MPI_DOUBLE, nbrright, 0, comm);
  MPI_Recv(&x[s-1][1], nx, MPI_DOUBLE, nbrleft, 0, comm,
           MPI_STATUS_IGNORE);
  MPI_Ssend(&x[s][1], nx, MPI_DOUBLE, nbrleft, 1, comm);
  MPI_Recv(&x[e+1][1], nx, MPI_DOUBLE, nbrright, 1, comm,
           MPI_STATUS_IGNORE);

} else {

  MPI_Recv(&x[s-1][1], nx, MPI_DOUBLE, nbrleft, 0, comm,
           MPI_STATUS_IGNORE);
  MPI_Ssend(&x[e][1], nx, MPI_DOUBLE, nbrright, 0, comm);
  MPI_Recv(&x[e+1][1], nx, MPI_DOUBLE, nbrright, 1, comm,
           MPI_STATUS_IGNORE);
  MPI_Ssend(&x[s][1], nx, MPI_DOUBLE, nbrleft, 1, comm);
}
```

# MPI_Sendrecv

This type of operation:

- send to one processor
- receive from another processor

is so common that a special MPI function was created for it: `MPI_Sendrecv`

## MPI_Sendrecv

`MPI_SENDRECV(sendbuf, sendcount, sendtype, dest, sendtag, recvbuf, recvcount, recvtype, source, recvtag, comm, status)`

| | | |
|---|---|---|
| IN | sendbuf | initial address of send buffer (choice) |
| IN | sendcount | number of elements in send buffer (non-negative integer) |
| IN | sendtype | type of elements in send buffer (handle) |
| IN | dest | rank of destination (integer) |
| IN | sendtag | send tag (integer) |
| OUT | recvbuf | initial address of receive buffer (choice) |
| IN | recvcount | number of elements in receive buffer (non-negative integer) |
| IN | recvtype | type of elements in receive buffer (handle) |
| IN | source | rank of source or MPI_ANY_SOURCE (integer) |
| IN | recvtag | receive tag or MPI_ANY_TAG (integer) |
| IN | comm | communicator (handle) |
| OUT | status | status object (Status) |

### C Interface

```
int MPI_Sendrecv(const void *sendbuf, int sendcount, MPI_Datatype
sendtype, int dest, int sendtag, void *recvbuf, int recvcount,
MPI_Datatype recvtype, int source, int recvtag, MPI_Comm comm,
MPI_Status *status)
```

## Using Sendrecv

```
    MPI_Sendrecv(&x[e][1], nx, MPI_DOUBLE, nbrright, 0,
                 &x[s-1][1], nx, MPI_DOUBLE, nbrleft,
                 0, comm, MPI_STATUS_IGNORE);

    MPI_Sendrecv(&x[s][1], nx, MPI_DOUBLE, nbrleft, 1,
                 &x[e+1][1], nx, MPI_DOUBLE, nbrright,
                 1, comm, MPI_STATUS_IGNORE);
```

# Review of Topics

1. 55612
2. Parallise PDE Solution