Jacobi

Project 2016-17

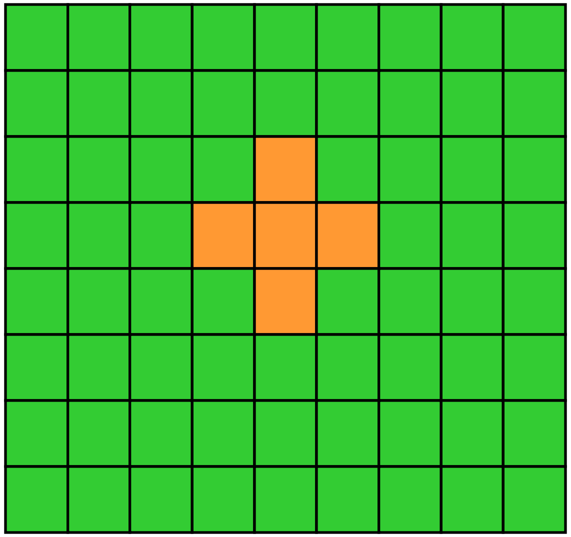
Salvatore Orlando

Regular computation, iterative, stencil-based

Loop until some condition is true

Perform computation which involves communicating with N,E,W,S neighbors of a point (5 point stencil)

[Convergence test?]



Jacobi

- **Jacobi is an iterative method to solve a system of n linear equations in n unknowns *Ax=b***
  - **where the i-th equations is the following:**

$$a_{i,1}x_1 + a_{i,1}x_1 + \cdots + a_{i,n}x_n = b_i$$

  - **alternatively, we can say that the i-th equations is:**

$$x_i = \frac{1}{a_{i,i}}\left[b_i - \sum_{j \neq i} a_{i,j}x_j \right]$$

- **The various *a*'s and *b*'s are constants, and we have to find the unkonwns *x***

Jacobi

- **The Jacobi method to solve the linear system is iterative, till we arrive at convergence:**

$$x^k{}_i = \frac{1}{a_{i,i}}\left[b_i - \sum_{j \neq i} a_{i,j} x^{k-1}{}_j \right]$$

where the values of the unknowns at the *(k-1)*-th iteration are used to compute the values at iteration *k*

- **The method converges for specific classes of matrixes**
  - **Usually the solution is approximated till a desired error threshold by computing the difference between the values of the unknowns at the *(k-1)*-th and *k*-th iterations**

Jacobi

- The linear system *Ax=b* can be easily and quickly solved with Jacobi when A is diagonally dominant
  - i.e., the magnitude of the diagonal entry in a row is larger than or equal to the sum of the magnitudes of all the other (non-diagonal) entries in that row

$$x^k{}_i = \frac{1}{a_{i,i}}\left[b_i - \sum_{j \neq i} a_{i,j} x^{k-1}{}_j\right]$$

Laplace and Jacobi

- **Jacobi can be used to solve the differential equation of Laplace in two variables (2D):**

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

- **The equation di Laplace models the *steady state* of a function *f* defined in a physical 2D space, where *f* is a given physical quantity**
- **For example, *f(x,y)* could represent *heat* as measured over a metal plate**
  - **Given a metal plate, <u>for which we know the temperature at the edges</u>, what is the temperature distribution inside the plate?**

Laplace and Jacobi

- For our purposes, it is enough to find the values of *f(x,y)* for a suitable 2D discretization of the space
- Let Δ be the uniform distance between the discrete grid along the two Cartesian dimensions
  - If Δ is enough small, we can approximate the 2nd order derivatives with the finite difference method. Using the Taylor series, we can thus approximate the 2nd order derivatives as follows:

$$\frac{\partial^2 f}{\partial x^2} = \frac{1}{\Delta^2}\left[f(x+\Delta,y) - 2f(x,y) + f(x-\Delta,y)\right]$$

$$\frac{\partial^2 f}{\partial y^2} = \frac{1}{\Delta^2}\left[f(x,y+\Delta) - 2f(x,y) + f(x,y-\Delta)\right]$$

- Substituting both in the Laplace equation:

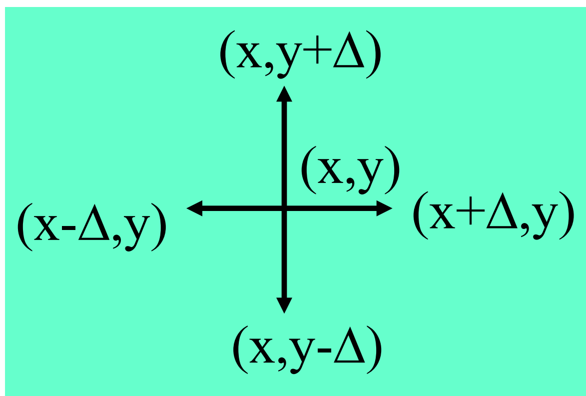
$$f(x,y) = \frac{\left[f(x-\Delta,y) + f(x,y-\Delta) + f(x+\Delta,y) + f(x,y+\Delta)\right]}{4}$$

Laplace and Jacobi

- **Note the relations between the values of the functions:**

$$f(x,y)=\frac{\left[f(x-\Delta,y)+f(x,y-\Delta)+f(x+\Delta,y)+f(x,y+\Delta)\right]}{4}$$

- **We can note a *cross stencil* of 4 points, which is the same for each (x,y)**



STENCIL: a thin sheet of cardboard, plastic, or metal with a pattern or letters cut out of it, used to produce the cut design on the surface below by the application of ink or paint through the holes.

Laplace and Jacobi

- To solve the Laplace equation we have to find all the values of f(x,y) (steady state) wrt a discrete grid with 2 indexes

- To apply Jacobi:
  - we need to solve a linear system, by finding the values of a 1D vector *x(i)* of unknowns.
  - we thus convert from f(x,y) to x(i)
    - We associate the variables x(i) of our linear system *Ax=b* with the discrete values of f(x,y) with respect to a discrete 2D grid of n^2 points
    - Row-major distribution of the 2D discrete matrix derived from f(x,y):

- We thus have n^2 unknowns x(i) with matrix A of size $n^2 \times n^2$

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix}$$

Laplace and Jacobi

- Since the vector of unknowns *x(i)* of our linear system *Ax=b* are distributed row-major over the discrete 2D grid of *f(x,y)*

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix}$$

then:

$$f(x,y) = \frac{[f(x-\Delta,y) + f(x,y-\Delta) + f(x+\Delta,y) + f(x,y+\Delta)]}{4}$$

becomes:

$$x_i = \frac{[x_{i-n} + x_{i-1} + x_{i+1} + x_{i+n}]}{4}$$

Laplace and Jacobi

- We have to find matrix A and vector b of the linear system, on which we have to apply Jacobi
- Our solution to the Laplace equation determined a linear system of equations of this form:

$$x_i = \frac{\left[x_{i-n} + x_{i-1} + x_{i+1} + x_{i+n}\right]}{4}$$

that we can re-write as follows:

$$-x_{i-n} - x_{i-1} + 4x_i - x_{i+1} - x_{i+n} = 0$$

- Then, the various *b(i)* are 0
- and matrix A?

Laplace and Jacobi

- Matrix A of the linear system is sparse and is diagonally dominant
  - Each row has at most 5 non-zero values
  - All the values on the diagonal are equal to 4, that is > (or >=) than the sum of all the values on the row

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

N=9 $\Rightarrow$ n=3

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

Jacobi to solve the generated linear system

- **Initialize the values of the unknowns x(i), e.g., x(i) = b(i)**
- **The new x(i) are computed iteratively by the Jacobi method as follows:**

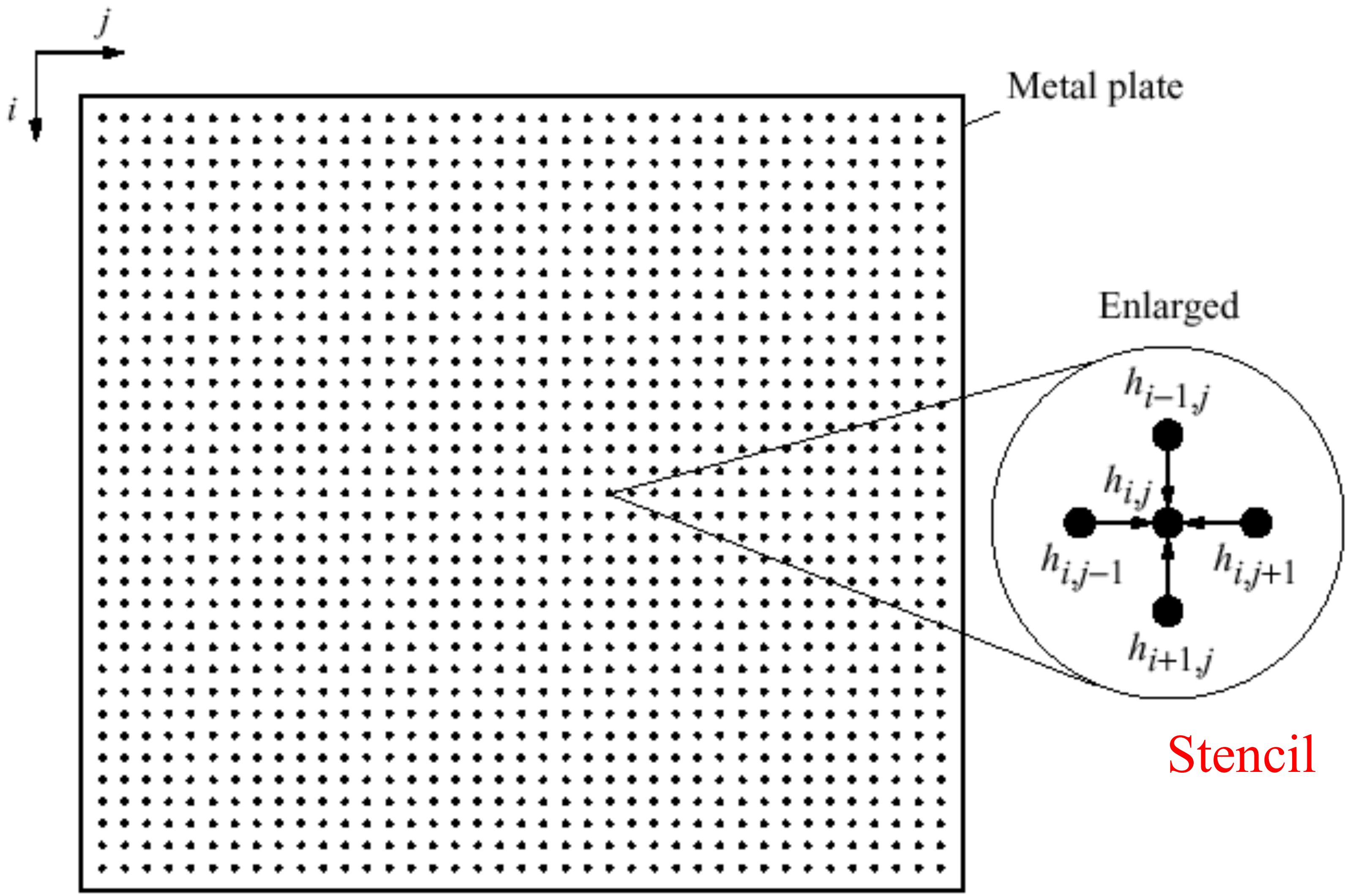
$$x^t{}_i = \frac{\left[x^{t-1}{}_{i-n} + x^{t-1}{}_{i-1} + x^{t-1}{}_{i+1} + x^{t-1}{}_{i+n}\right]}{4}$$

- **The new updated values of x(i) at iteration *t* are used at iteration *t+1***

- **Termination condition:**

$$\left|x^t{}_i - x^{t-1}{}_i\right| < \textit{error threshold}(\ \forall\, \boldsymbol{x_i}\ )$$

Example: Heat diffusion with finite differences



Pseudo-codice of Message Passing Jacobi

[Initialize boundary regions, read-only boundary condition]

```
for (i=1; i<=N; i++)
        x[0][i] = north[i];
        x[N+1][i] = south[i];
        x[i][0] = west[i];
        x[i][N+1] = east[i];
```

[Initialize matrix]

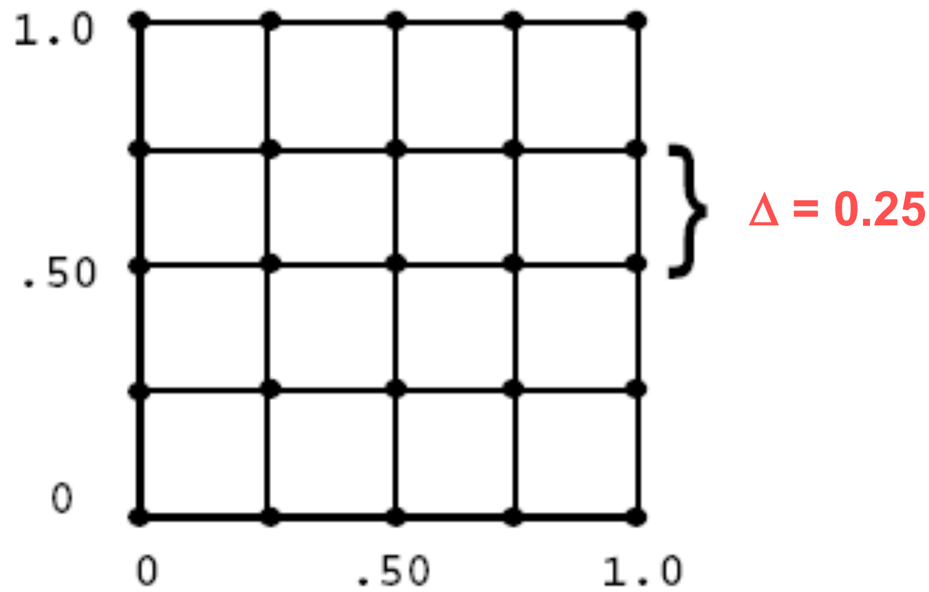
```
for (i=1; i<=N; i++)
        for (j=1; j<=N; j++)
                x[i][j] = initvalue;
```

[Iterative refinement of x until values converge]

```
while (maxdiff > CONVERG)
        [Update x array]
        for (i=1; i<=N; i++)
                for (j=1; j<=N; j++)
                        newx[i][j] = ¼ (x[i-1][j] + x[i][j+1] + x[i+1][j] + x[i][j-1]);
        [Convergence test]
        maxdiff = 0;
        for (i=1; i<=N; i++)
            for (j=1; j<=N; j++)
                maxdiff = max(maxdiff, |newx[i][j]-x[i][j]|);
                x[i][j] = newx[i][j]
```

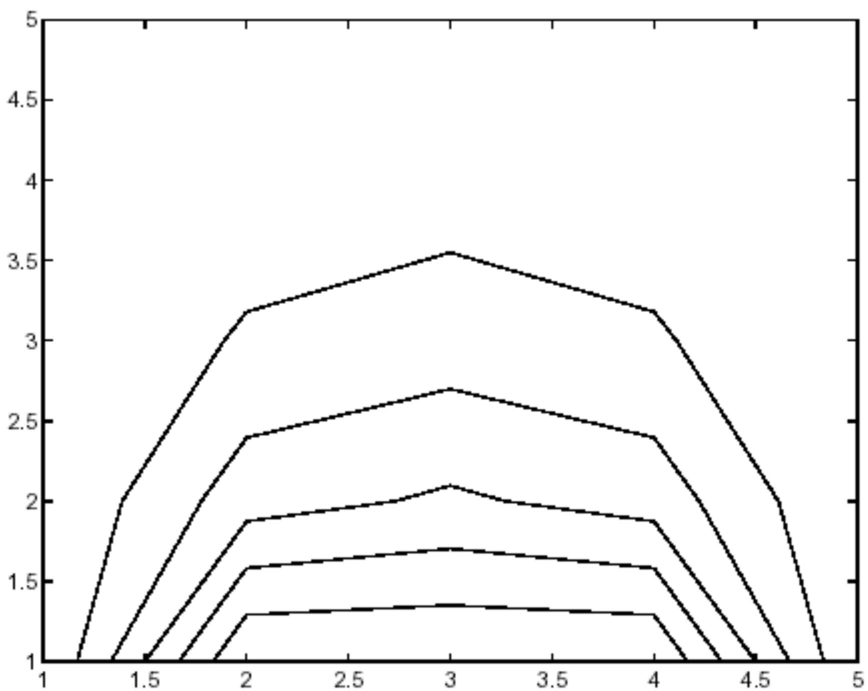
Example

- If we know the boundary conditions of *f(x,y)*, i.e., we know the values of f(x,y) on the edges of the grid and they are constant
  - We can compute the steady state by applying Laplace,
    and use Jacoby to compute the values of *f(x,y)* for all the internal points
- Heat diffusion on a metal plate, where the edge of the plate = 1
- Border conditions:
  - f(x, 0) = 300
  - f(x, 1) = f(0, y) = f(1, y) = 0
- Discretize using a uniform mesh where Δ = 0.25, we obtain the grid on the right-hand

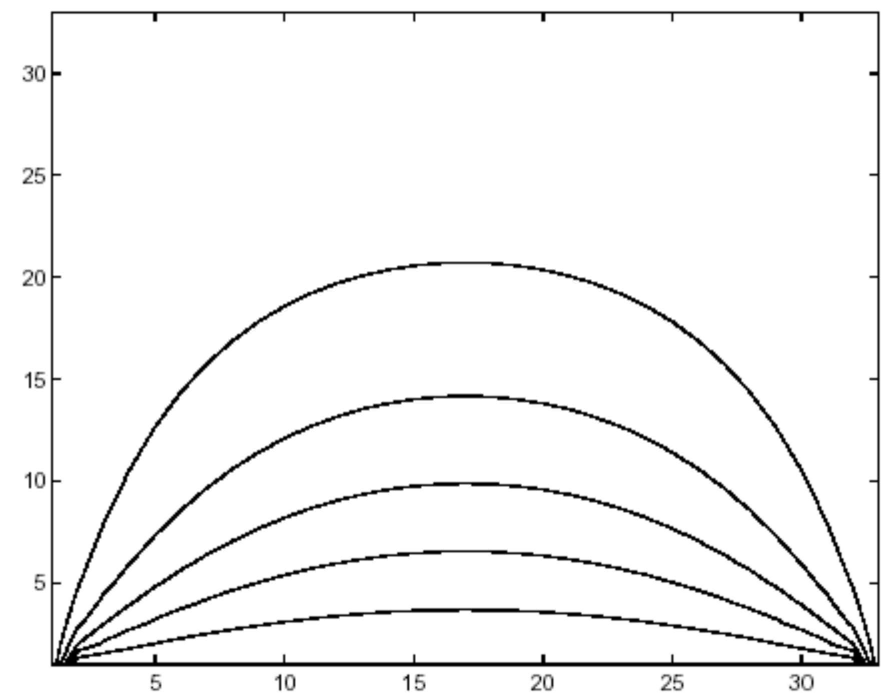


Example

- Once Jacoby solved the Laplace equation, for the discrete points of the 2D meah, we can plot some iso-temperature curves as follow
  - Remember that the temperature at the bottom edge is kept constant at a value of 300, while is 0 on all the other three edges
- The *isocurves* that connect all the points at the same temperature *t* are smoother for denser grids, but their computation is more complex in time and space



Δ = 0.25

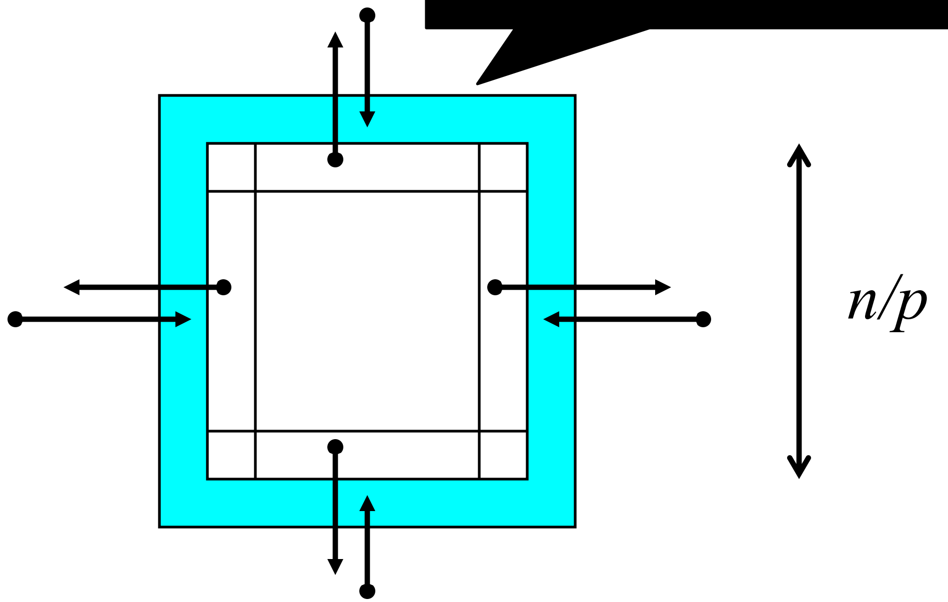


Δ = 0.03125

MPI parallelization: Block data distribution

- Given n^2 points and p^2 processors
  - Every processor is given n^2/p^2 points
  - The four borders, each of size n/p need to be exchanged with the neighbors
  - These four borders are calls *ghost area*
  - The communication volume is proportional to $4* n/p$
  - The computational cost is proportional to n^2/p^2
- What if we increase granularity ?

Allocate $(n/p +2)^2$ elements (ghost area)



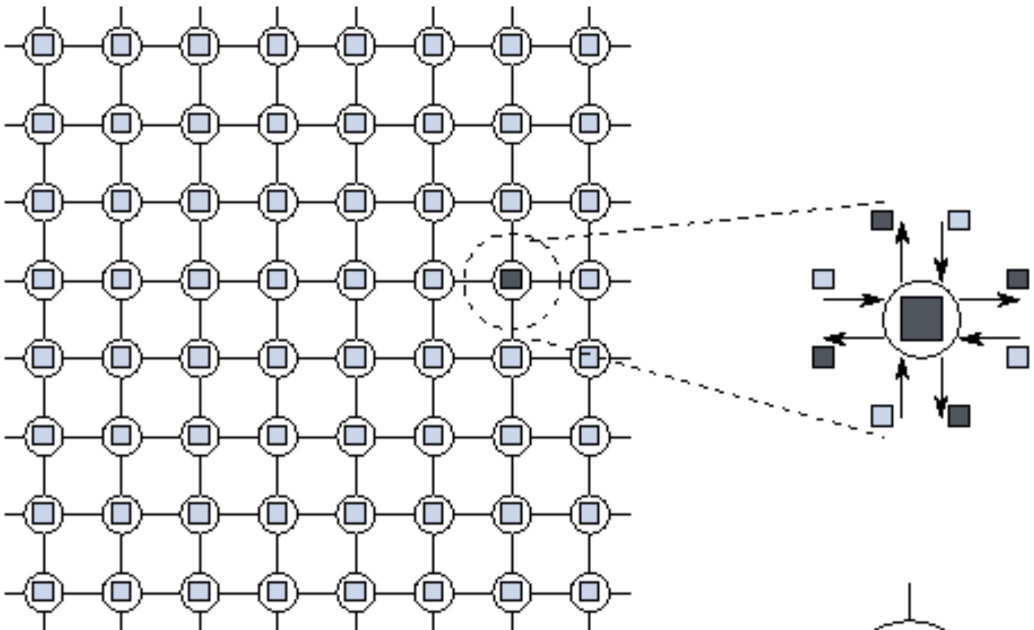
Heat diffusion: SPMD with block data distribution

- Matrix of n^2 elements distributed `(block,block)` on a grid of p^2 processes
- Each process $P_{i,j}$ is thus assigned a block of n^2/p^2 elements
- Code executed by a generic SPMD process $P_{i,j}$

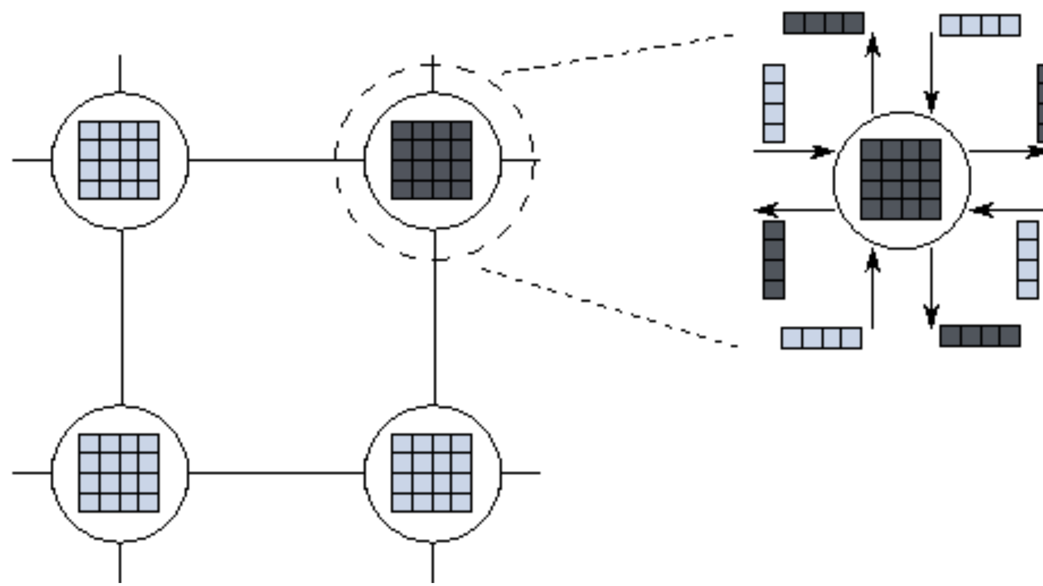
```
<declaration of array h(0:n/p+1, 0:n/p+1) with ghost area>
<declaration of array temp. g(1:n/p, 1:n/p) >
for (iteration = 0; iteration < limit; iteration++) {
  < border exchange of h[][] with
    the 4 neighbors Pi,j-1 Pi,j+1 Pi-1,j Pi+1,j >     | Local Barrier
  for (i = 1; i <= n/p; i++)
     for (j = 1; j <= n/p; j++)
        g[i][j] = 0.25*(h[i-1][j]+h[i+1][j]+h[i][j-1]+h[i][j+1]);
  for (i = 1; i <= n/p; i++) /* Update h[][] */
     for (j = 1; j <= n/p; j++)
        h[i][j] = g[i][j];
}
```

- We have to differentiate the code for blocks/processes on the grid borders
  - P_{ij} where `i,j=0,p-1`
  - less partners with which to exchange communications
- With asynchronous communications, we can postpone the receives
  - the receive must however be completed before accessing the ghost area

Increasing granularity



- 64 points
- 64 computations
- 4 points sent per processor
- 4 * n_procs = 4*64 = communication volume 256



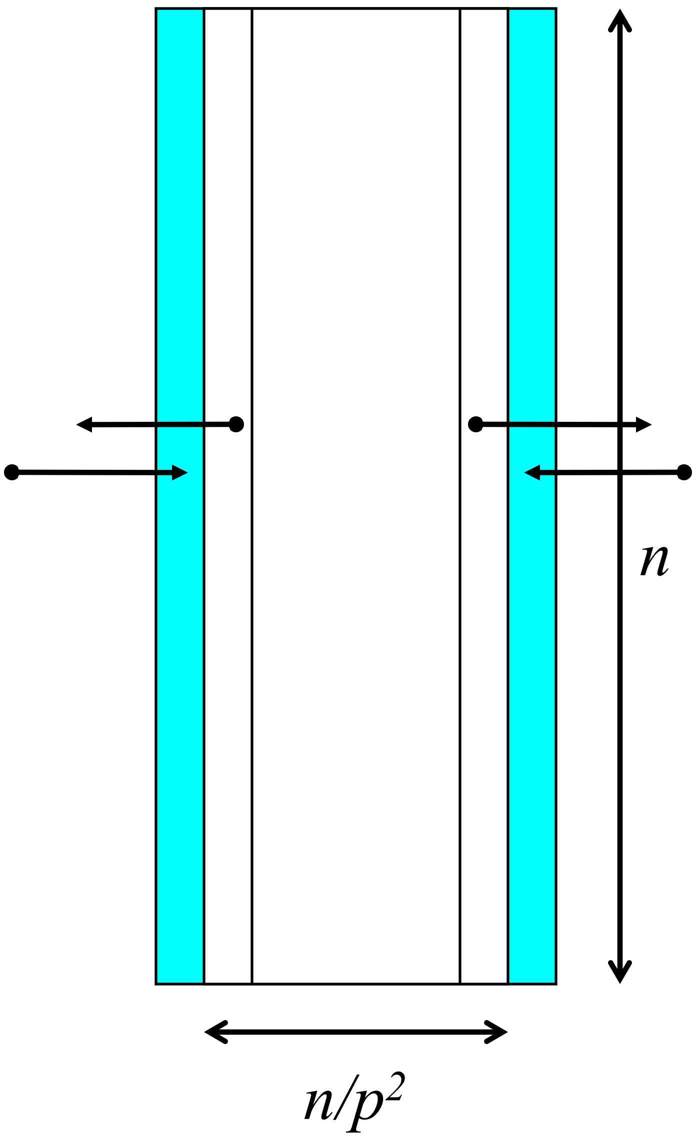
- 64 points
- 64 computations
- 16 points sent per processor
- 16 * n_procs = 16*4 = communication volume 64

Surface / Volume

- **The *communication cost* is proportional to the surface (perimeter) of the data partition**
- **The *computational cost is* proportional to the volume (area) of the data partition**
- **In two dimensions:**
  - **The surface increases as n**
  - **The volume increases as n^2**
- **In three dimensions:**
  - **The surface increase as n^2**
  - **The volume increases as n^3**
- **The ratio *communication/computation* decreases when increasing the partition size**
  - **Good !**
  - **This effect is less visible for high dimensional problems**

MPI parallelization: Stripe data distribution

- Given n^2 points and p^2 processors
  - Every processor is given n^2/p^2 points
  - Those points form a $n * n/p^2$ rectangle
  - Two borders, each of size n need to be exchanged with the neighbors
  - The data exchanged are proportional to n (2 sends and 2 receives)
- In the (block,block) distribution
  - The data exchanged are proportional to n/p (4 sends and 4 receives)
- To minimize the communication volume:
  - (block,block) is better



Project

- Design Jacobi in parallel and evaluate performance plots
  - Speedup, the same problem size, …
  - Scalability, how the execution time gets larger as we increases the size of the problem (e.g., by reducing Δ)
- MPI parallelization

- Choose some of these

  - OpenMP parallelization

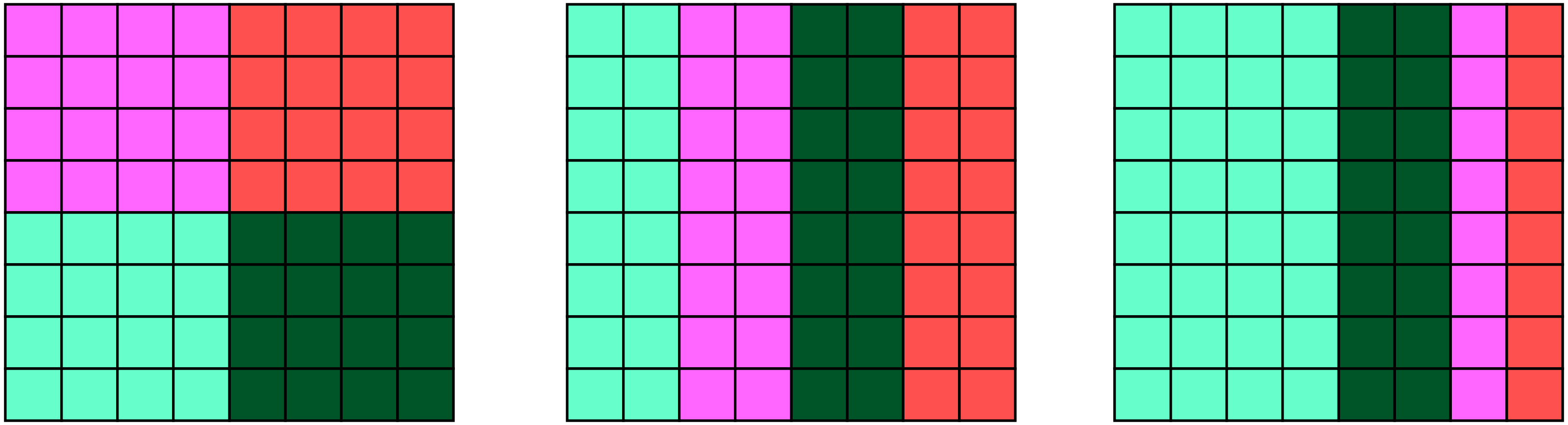
  - OpenMP & MPI parallelization

  - SIMD parallelization

  - GPU parallelization

Some issues in implementing Jacobi

- Syncronization
  - Can we avoid to synchronize at each iteration?
  - Measure the effect on convergence, and if this strategy save time
- Message passing (MPI) implementation
  - Can we avoid to update the borders owned by other tasks at each iteration? Effect on convergence and execution time.
- Data partitioning
  - Block or strip?



Block　　Uniform Strip　　Non-uniform Strip