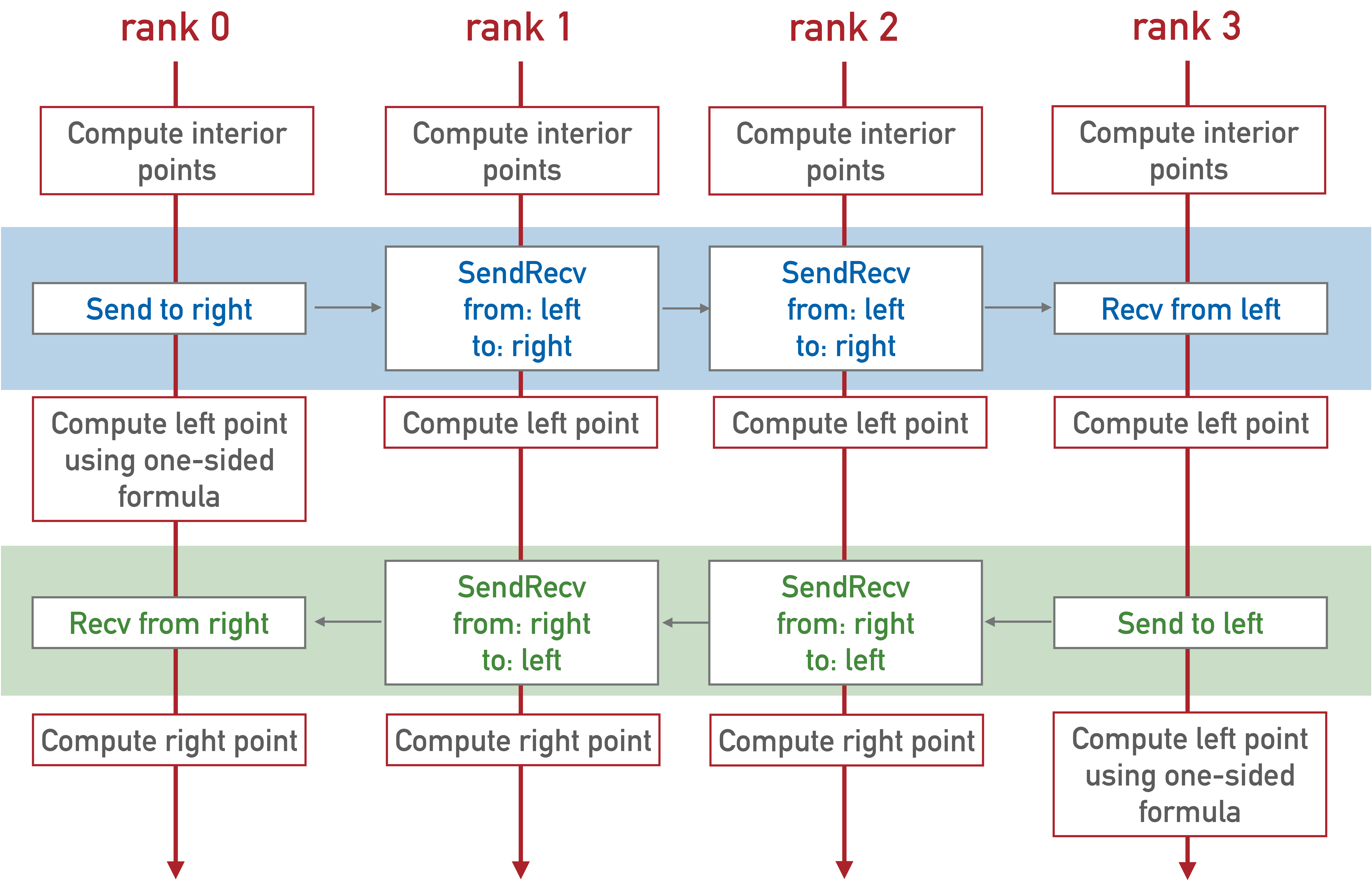
ME 471/571

Non-blocking communication

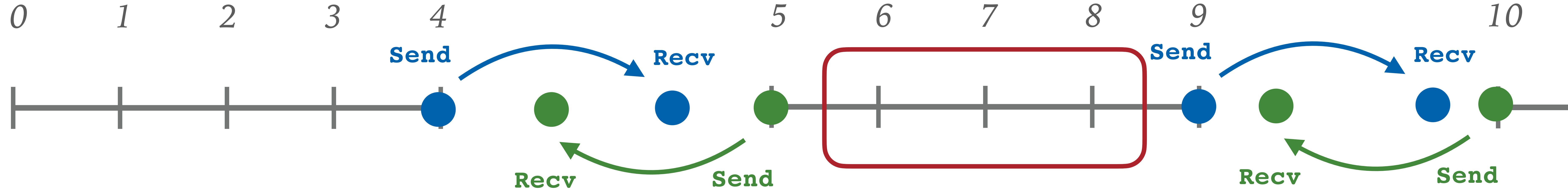
TOWARD SOLVING PDE'S




```
MPI_Sendrecv(&ghost_right,
             1, MPI_DOUBLE,
             irank+1, 101,
             &ghost_left,
             1, MPI_DOUBLE,
             irank-1, 101,
             MPI_COMM_WORLD,
             &status)
```

```
MPI_Sendrecv(&ghost_left,
             1, MPI_DOUBLE,
             irank-1, 102,
             &ghost_right,
             1, MPI_DOUBLE,
             irank+1, 102,
             MPI_COMM_WORLD,
             &status)
```

SOME EFFICIENCY CONSIDERATIONS



$$T_1 = \gamma a N$$

$$T_p = \gamma \frac{aN}{p} + 2\alpha + 2\beta$$

$$S_p = \frac{T_1}{T_p} = \frac{\gamma a N}{\gamma a \frac{N}{p} + 2\alpha + 2\beta}$$

$$E_p = \frac{1}{1 + 2\frac{\alpha+\beta}{\gamma}\frac{p}{aN}}$$

bad for efficiency:

- large $\frac{\alpha+\beta}{\gamma}$
- large p/N

good for efficiency:

- small p/N
- large a

THE POISSON PROBLEM

Consider the following equation

$$\nabla^2 u = f(x, y)$$

defined on a unit square

$$(x, y) \in [0,\ 1] \times [0,\ 1]$$

with Dirichlet boundary conditions $u(x, y) = g(x, y)$ *at the boundary.*

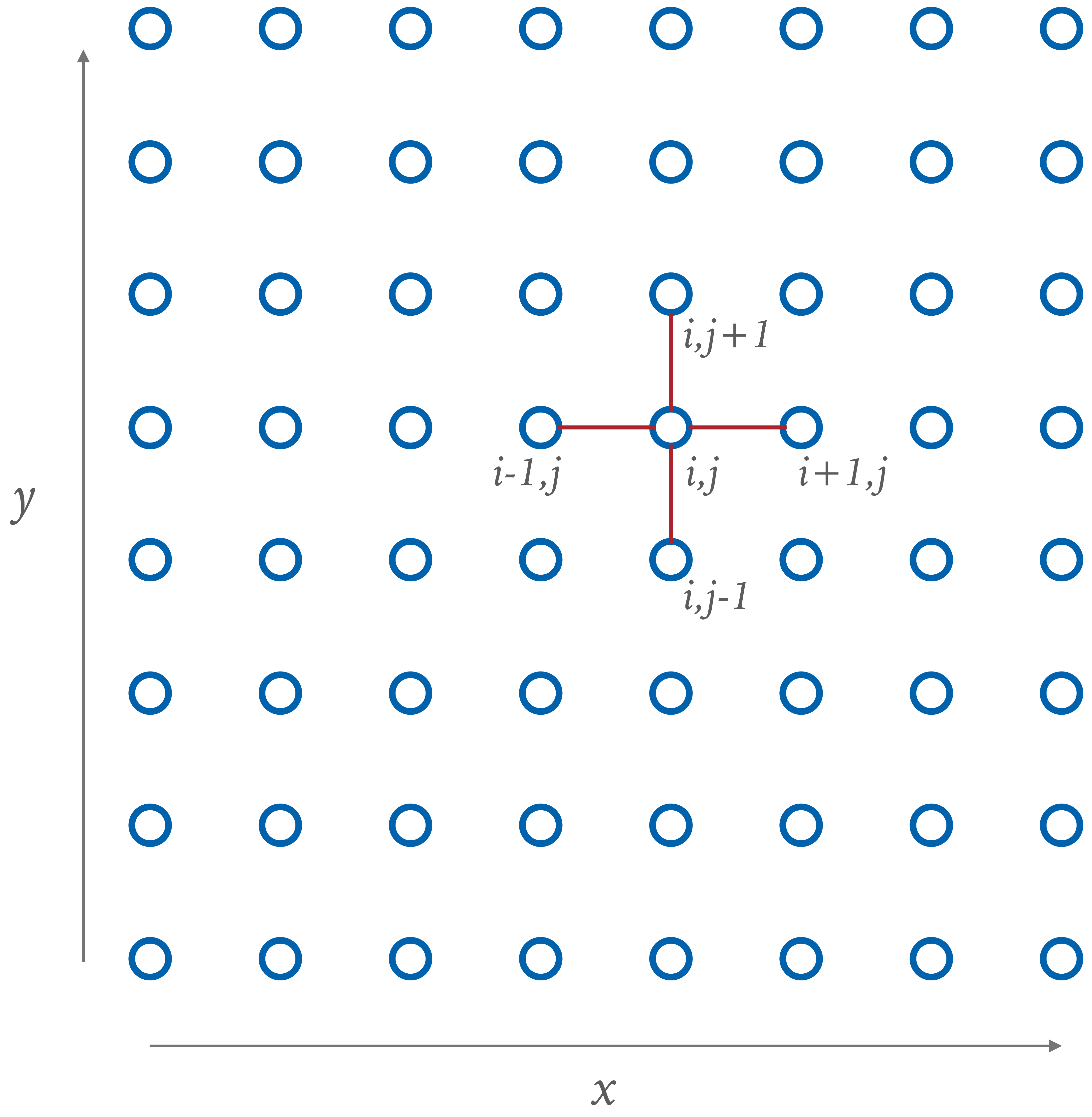
We can approximate the solution on a square grid of points

and use a finite difference method to approximate the diffusion operator:

$$\nabla^2 u \approx \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i.j}}{h^2}$$

We get the approximation to Poisson equation:

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



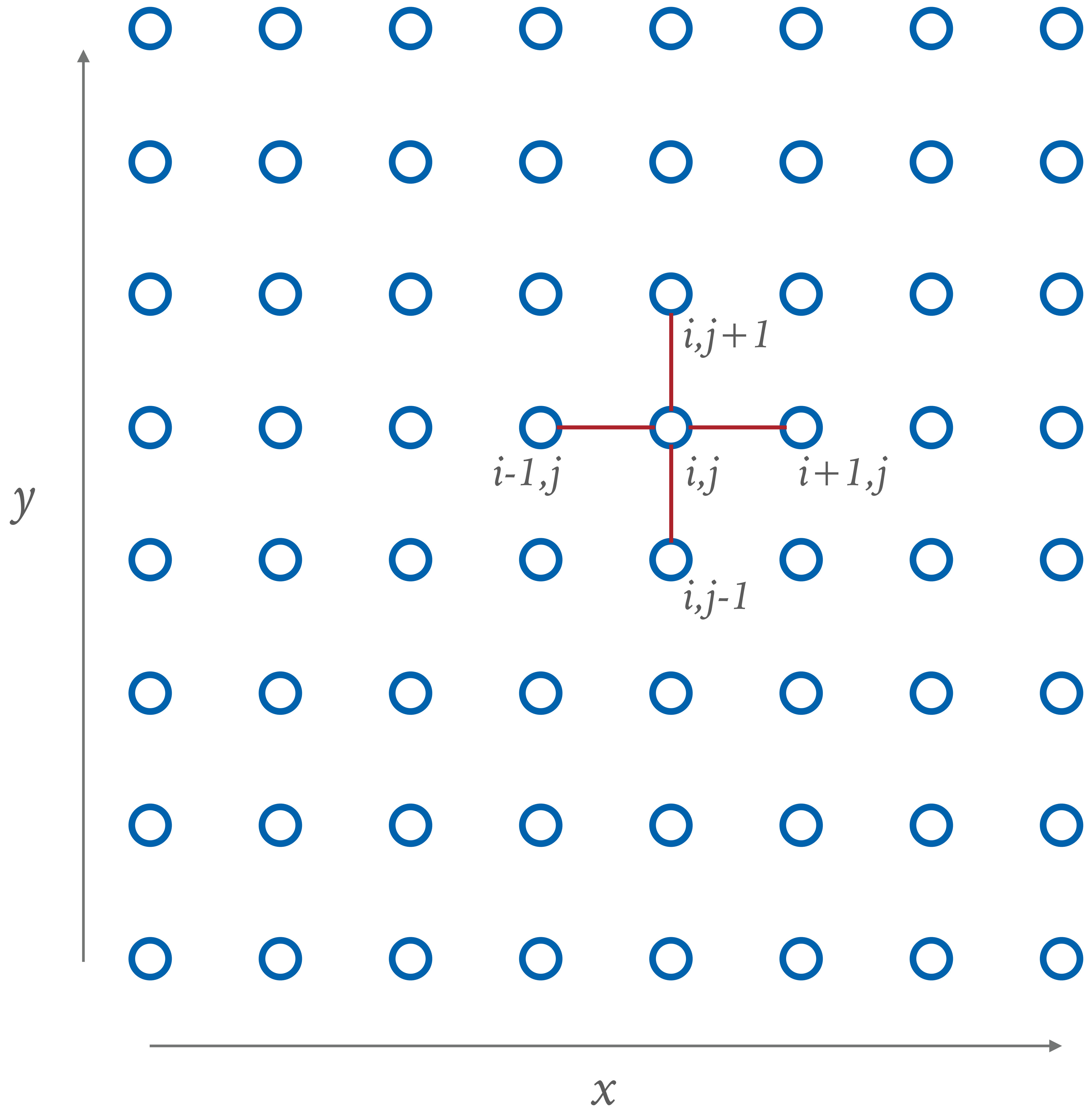
$$\frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}}{h^2} = f_{i,j}$$

We can solve this using Jacobi iteration:

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

We will repeat this iteration until the solution does not change much:

$$||u^{k+1} - u^{k}||_2 < \epsilon$$

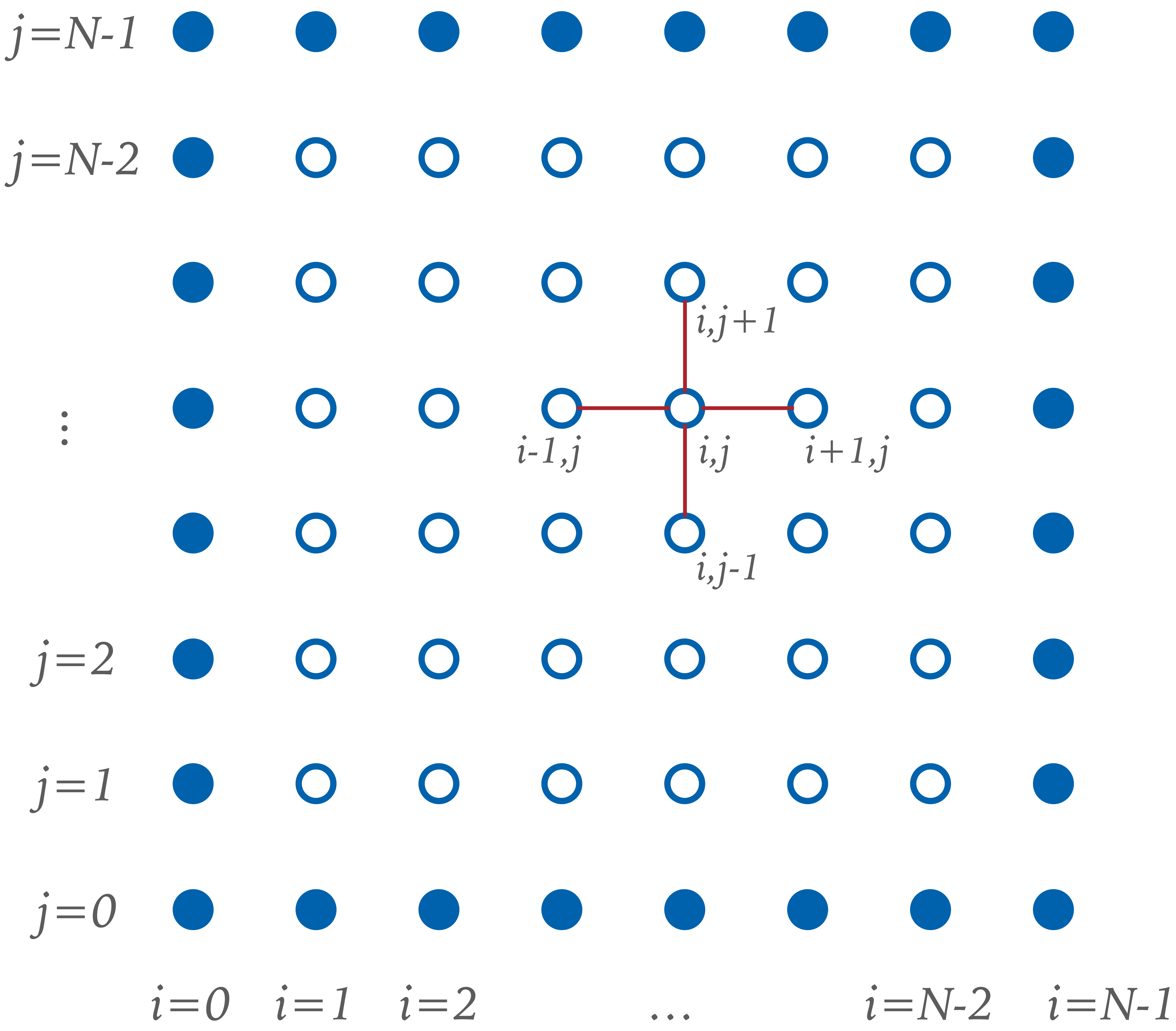


$$u_{i,j}^{k+1} = \frac{1}{4}\left(u_{i-1,j}^{k} + u_{i+1,j}^{k} + u_{i,j-1}^{k} + u_{i,j+1}^{k} - h^2 f_{i,j}\right)$$

Because we prescribe Dirichlet conditions, we know the values at the boundaries.

This means we need to solve only in the interior of the domain:

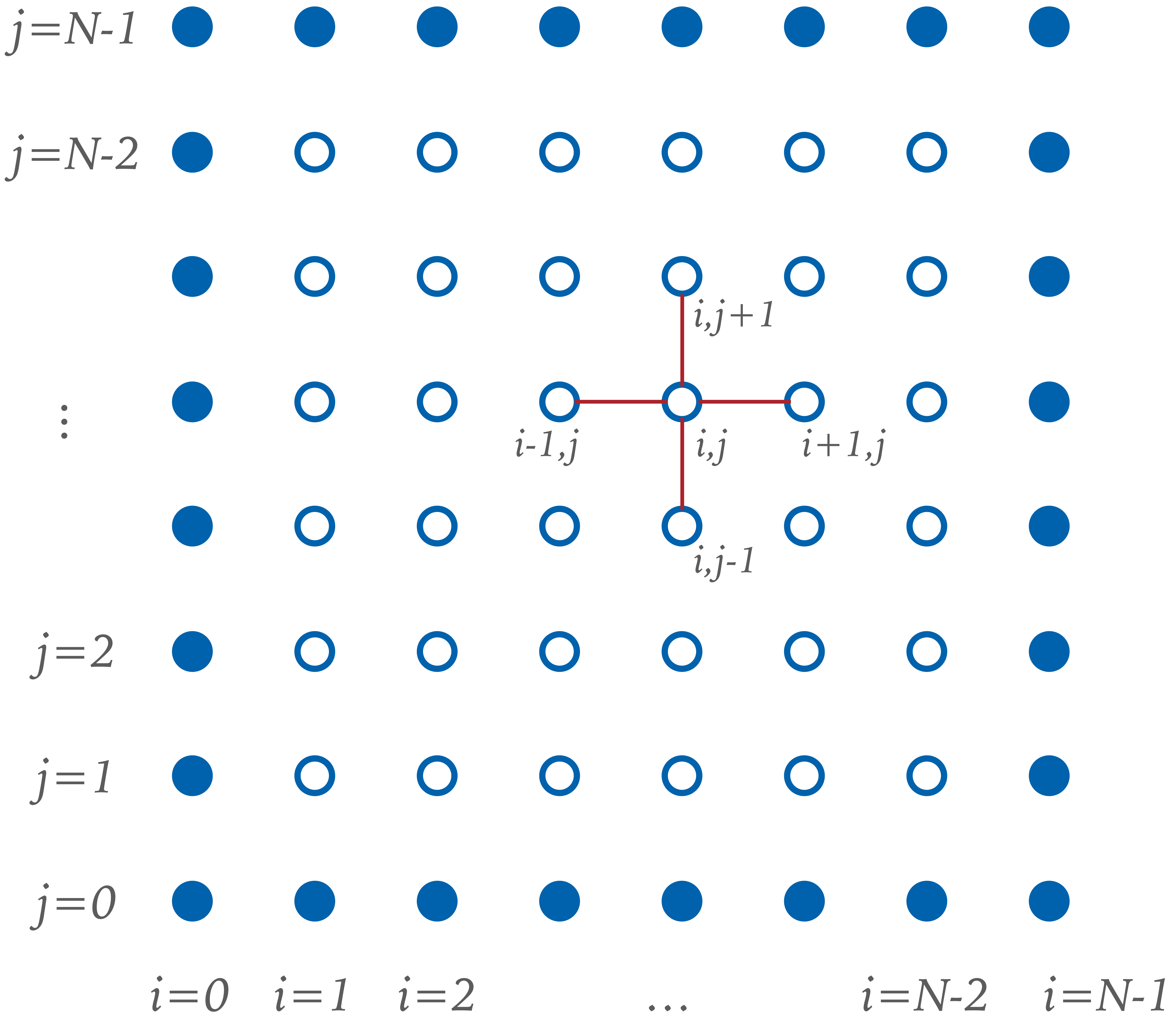
$$i = 1,\ 2,\ \ \ldots,\ N-2$$

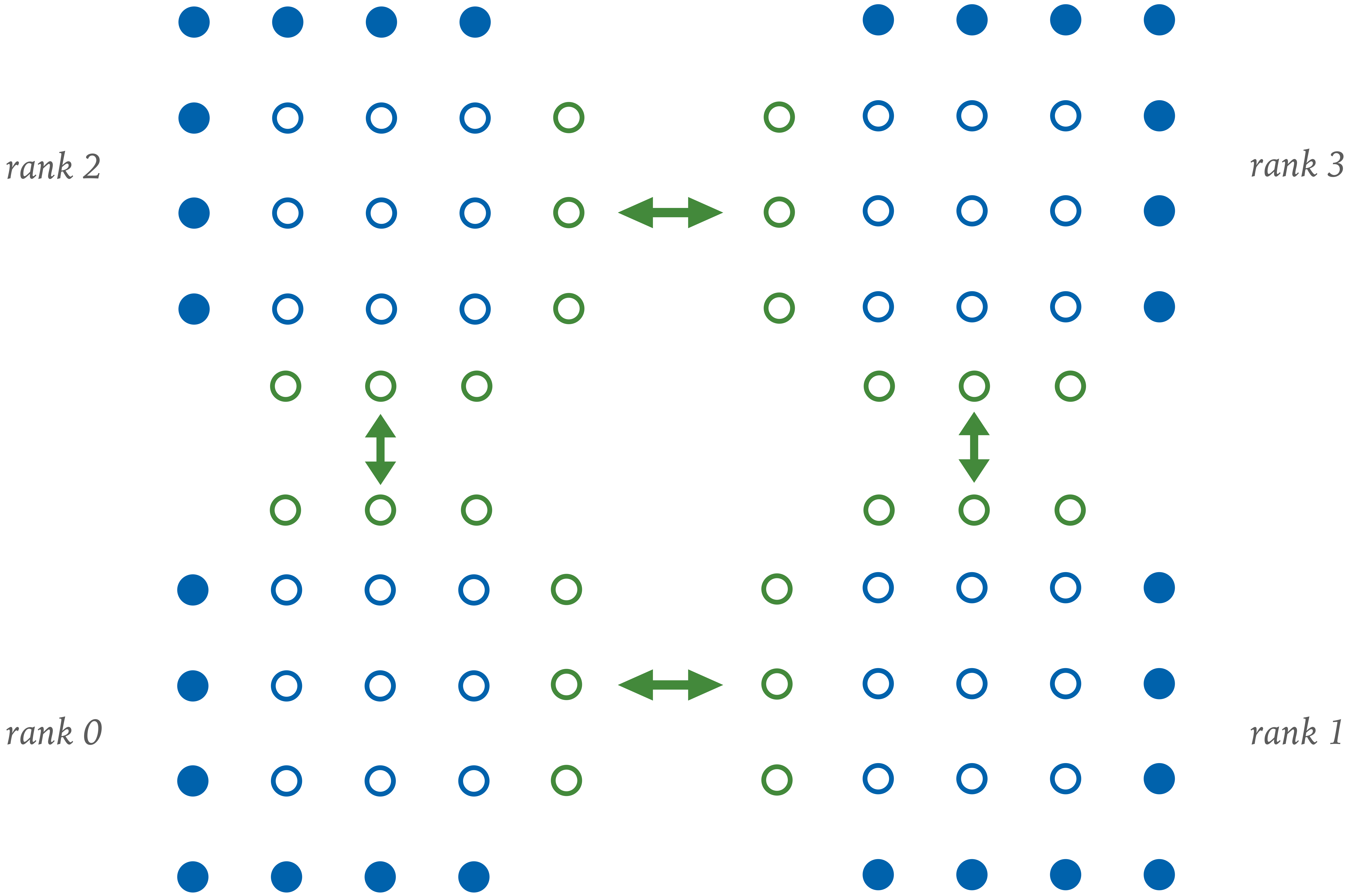


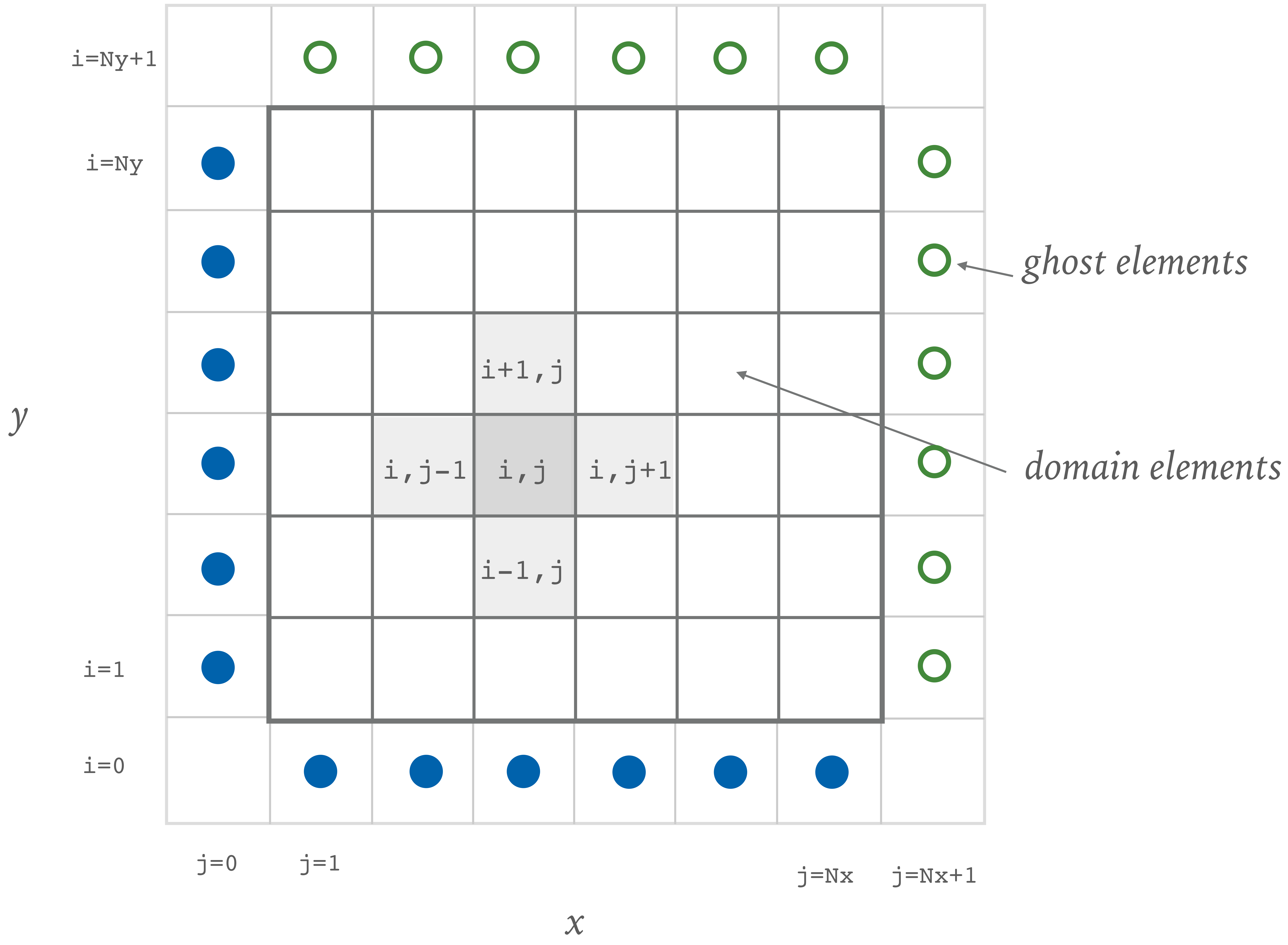
Serial code could look like:

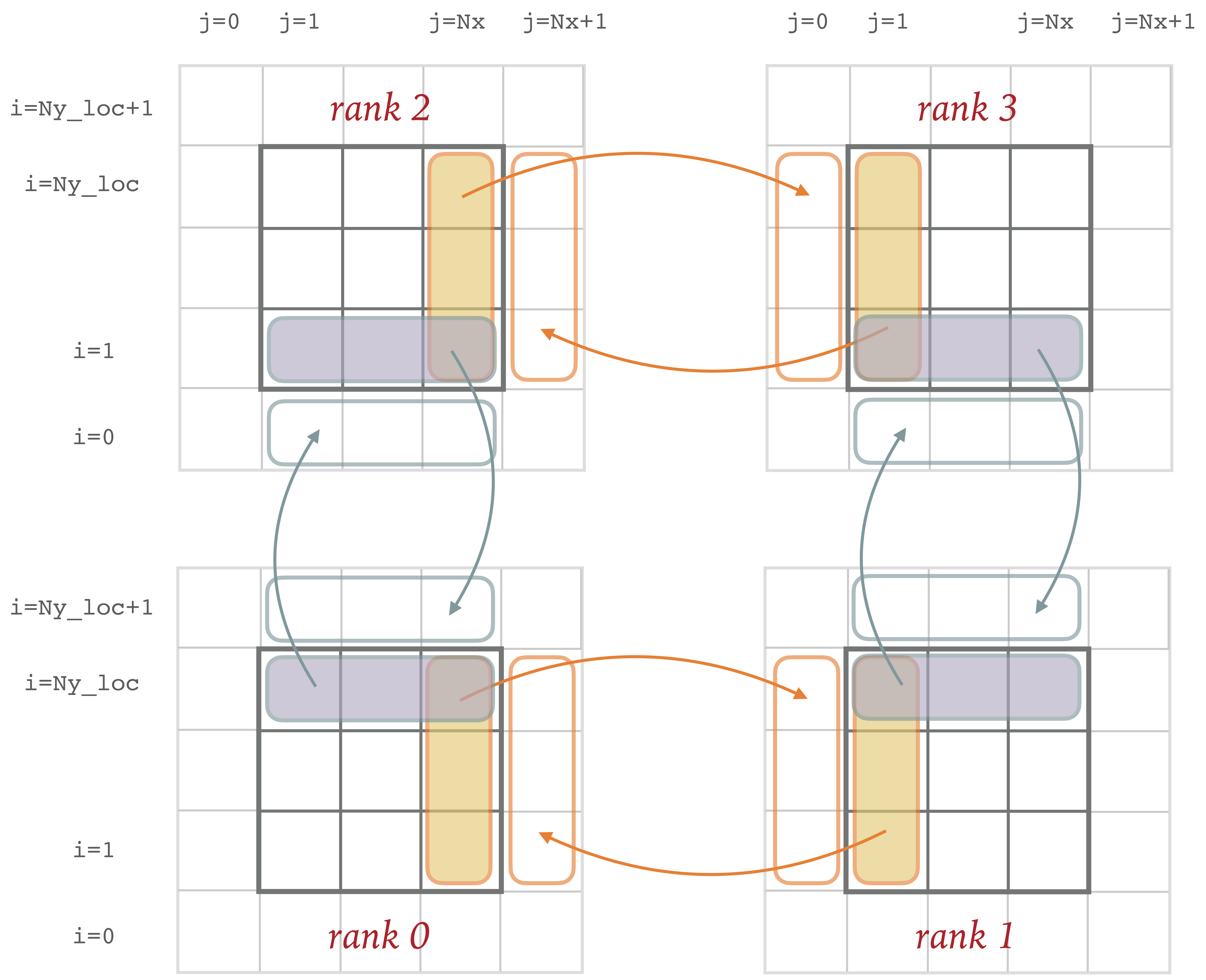
```
for(i=1; i<N-2; i++){
  for(j=1; j<N-2; j++){
    unew[i][j] = 0.25*(
       u[i-1][j]+u[i+1][j]+u[i][j-1]+u[i][j+1]
       -h*h*f[i][j]);
  }
}
```

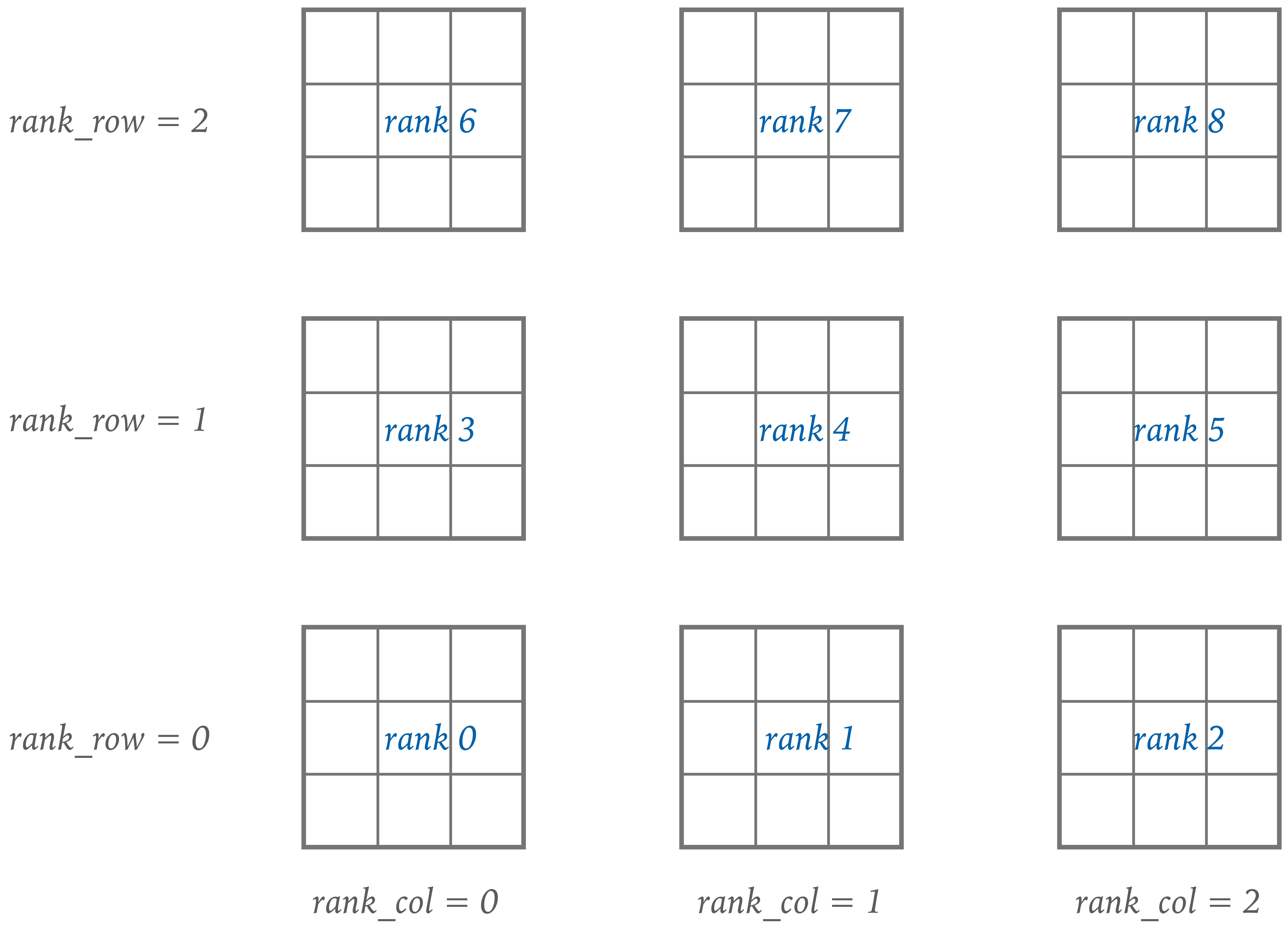
and we repeat that iteration until convergence.

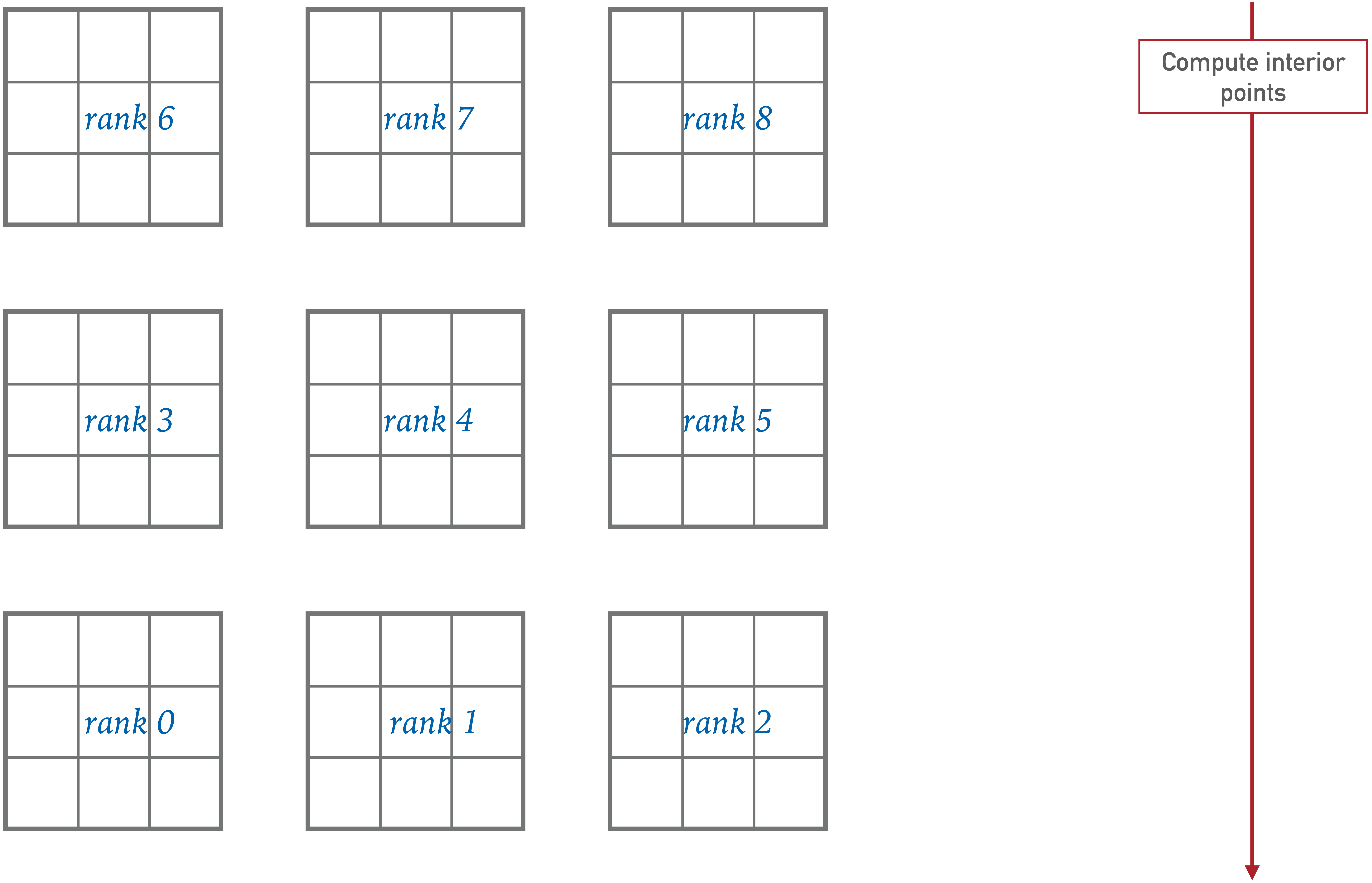


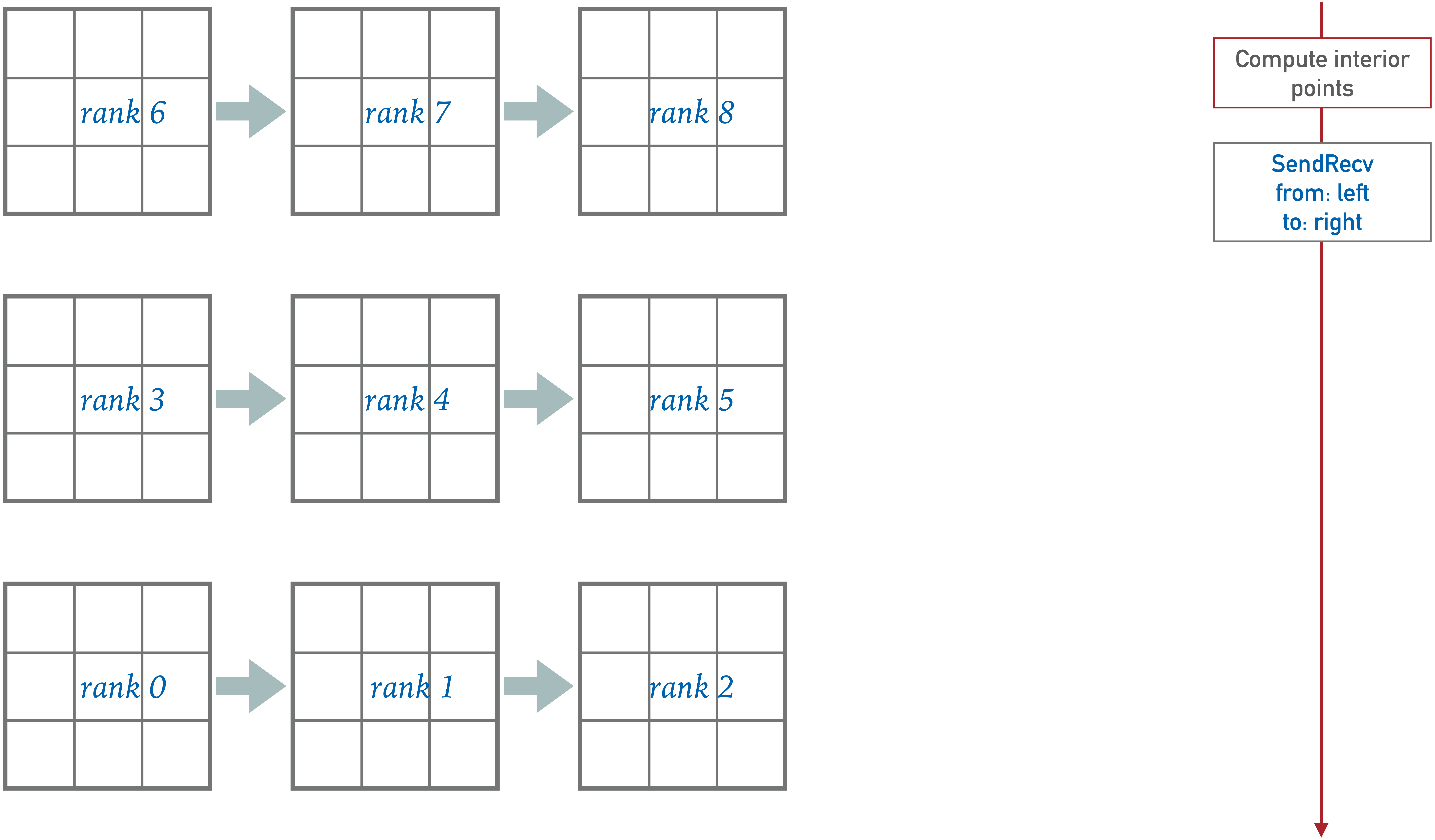

rank 2
rank 3
rank 0
rank 1


i=Ny+1
i=Ny
i=1
i=0
y
i+1,j
i,j-1
i,j
i,j+1
i-1,j
ghost elements
domain elements
j=0
j=1
j=Nx
j=Nx+1
x

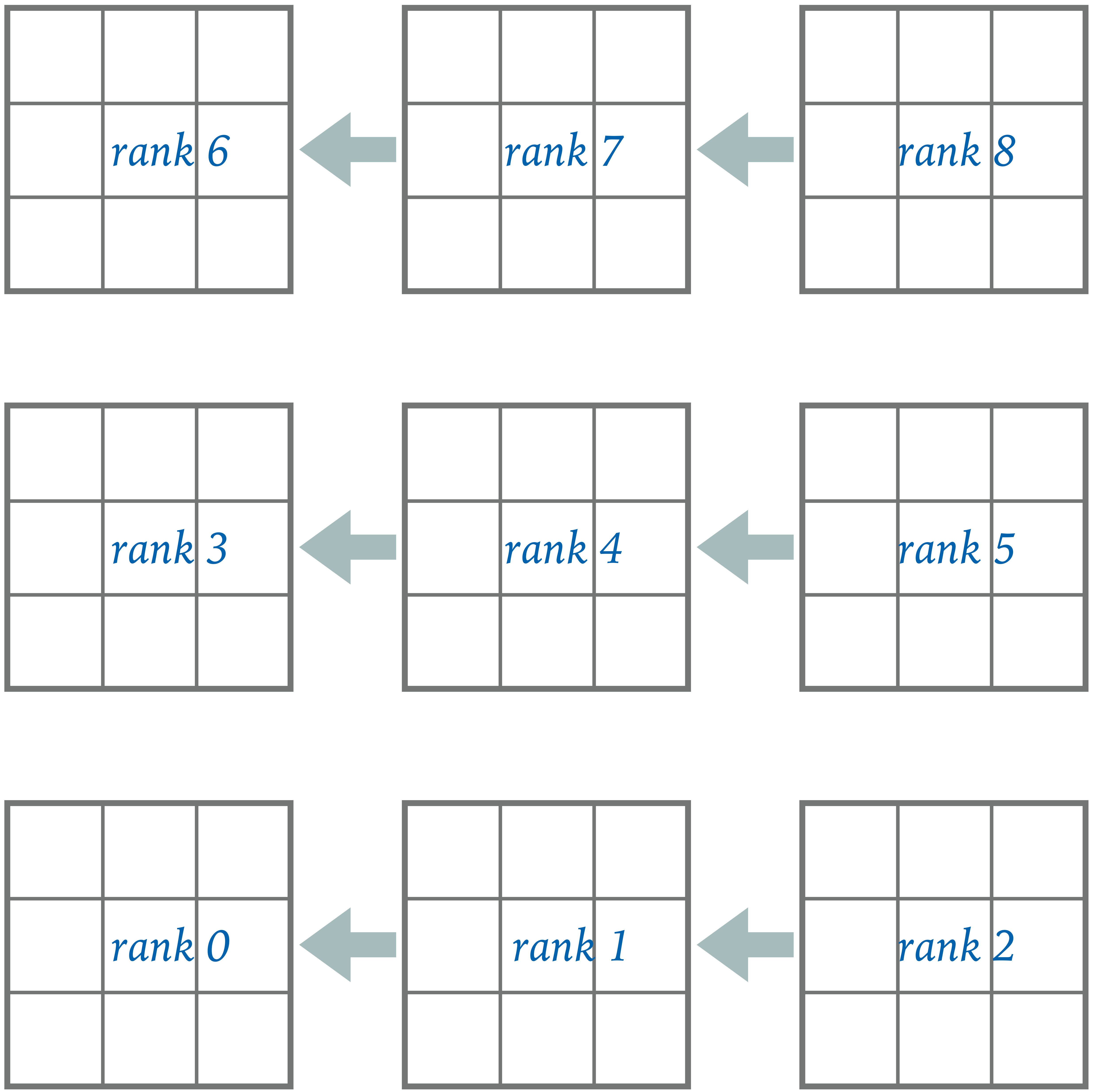

j=0
j=1
j=Nx
j=Nx+1
j=0
j=1
j=Nx
j=Nx+1
i=Ny_loc+1
i=Ny_loc
i=1
i=0
rank 2
rank 3
i=Ny_loc+1
i=Ny_loc
i=1
i=0
rank 0
rank 1


rank_row = 2
rank 6
rank 7
rank 8
rank_row = 1
rank 3
rank 4
rank 5
rank_row = 0
rank 0
rank 1
rank 2
rank_col = 0
rank_col = 1
rank_col = 2

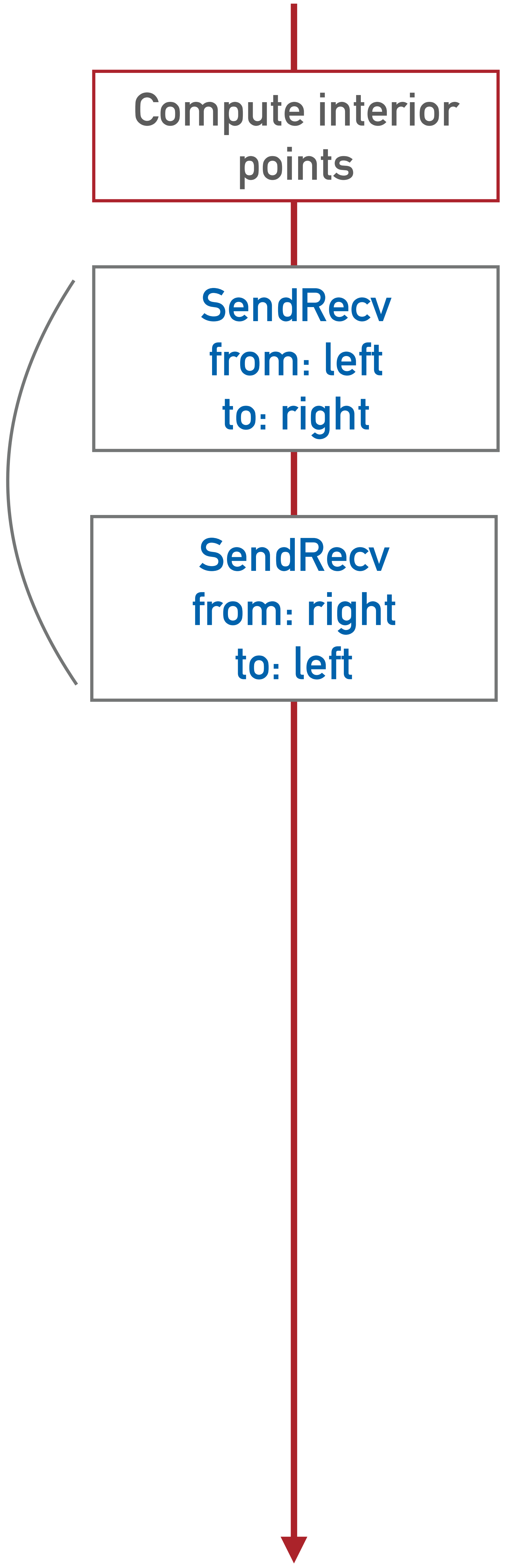

rank 6
rank 7
rank 8
rank 3
rank 4
rank 5
rank 0
rank 1
rank 2
Compute interior points

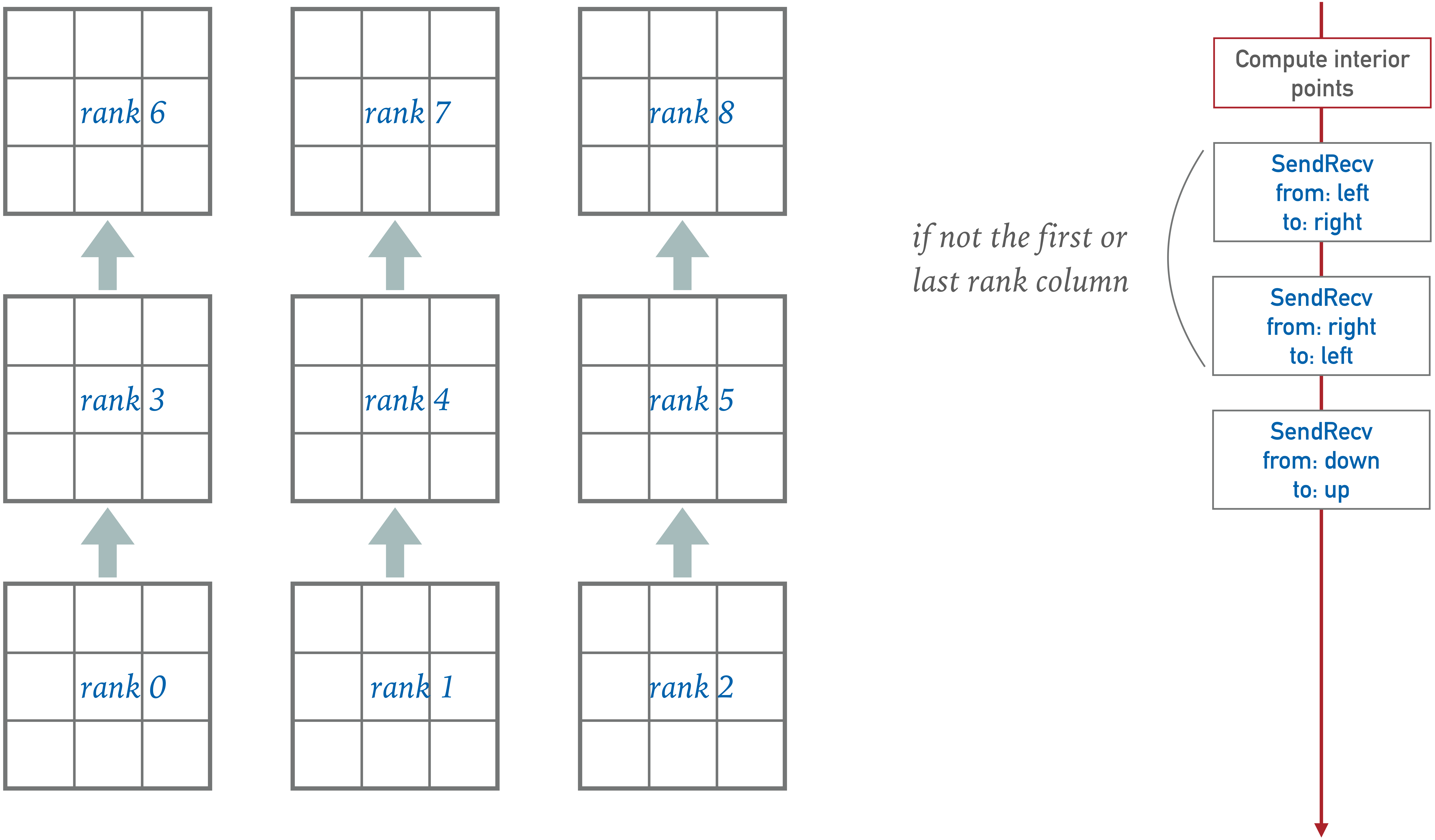


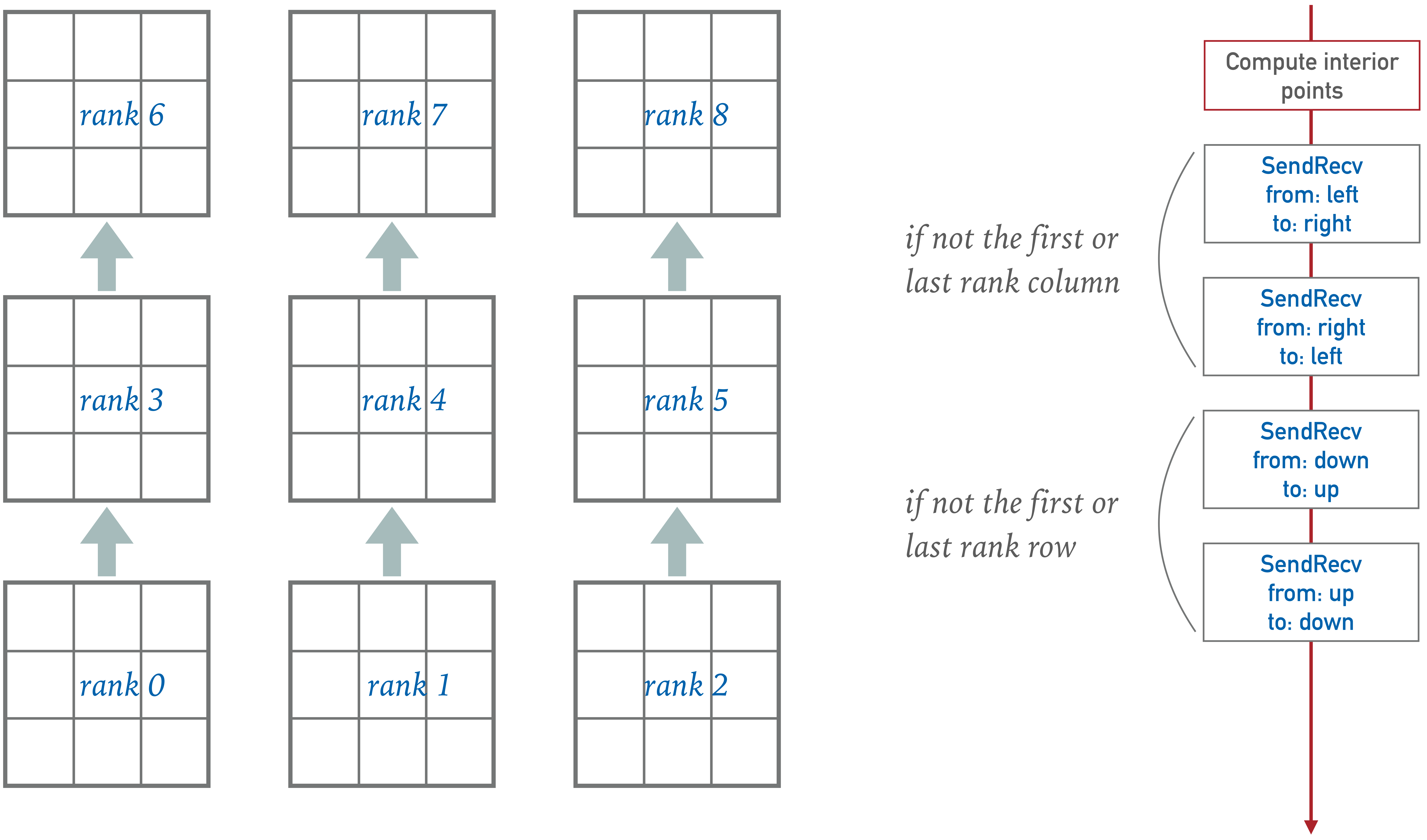
rank 6
rank 7
rank 8
rank 3
rank 4
rank 5
rank 0
rank 1
rank 2
Compute interior points
SendRecv
from: left
to: right

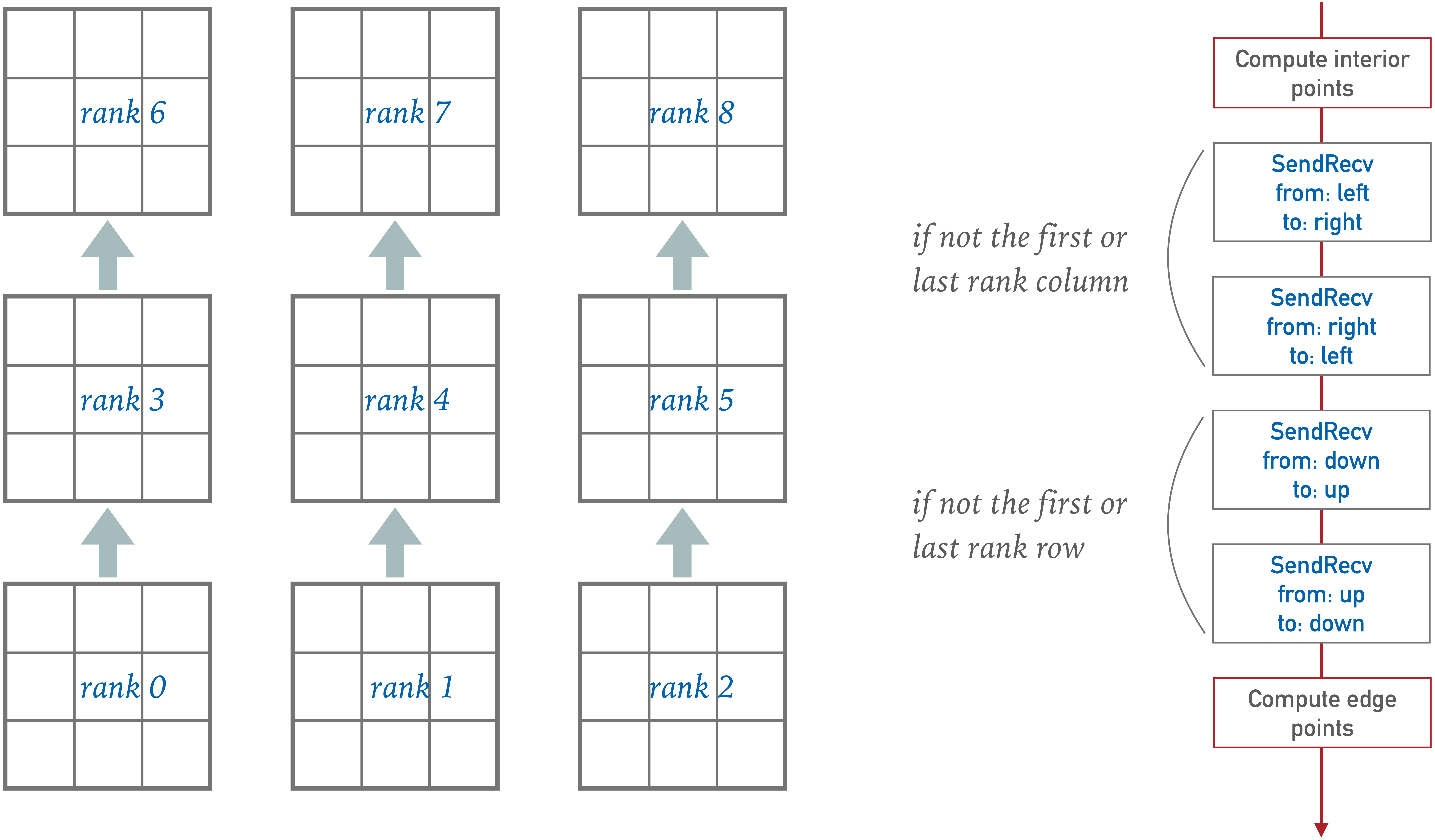


if not the first or last rank column

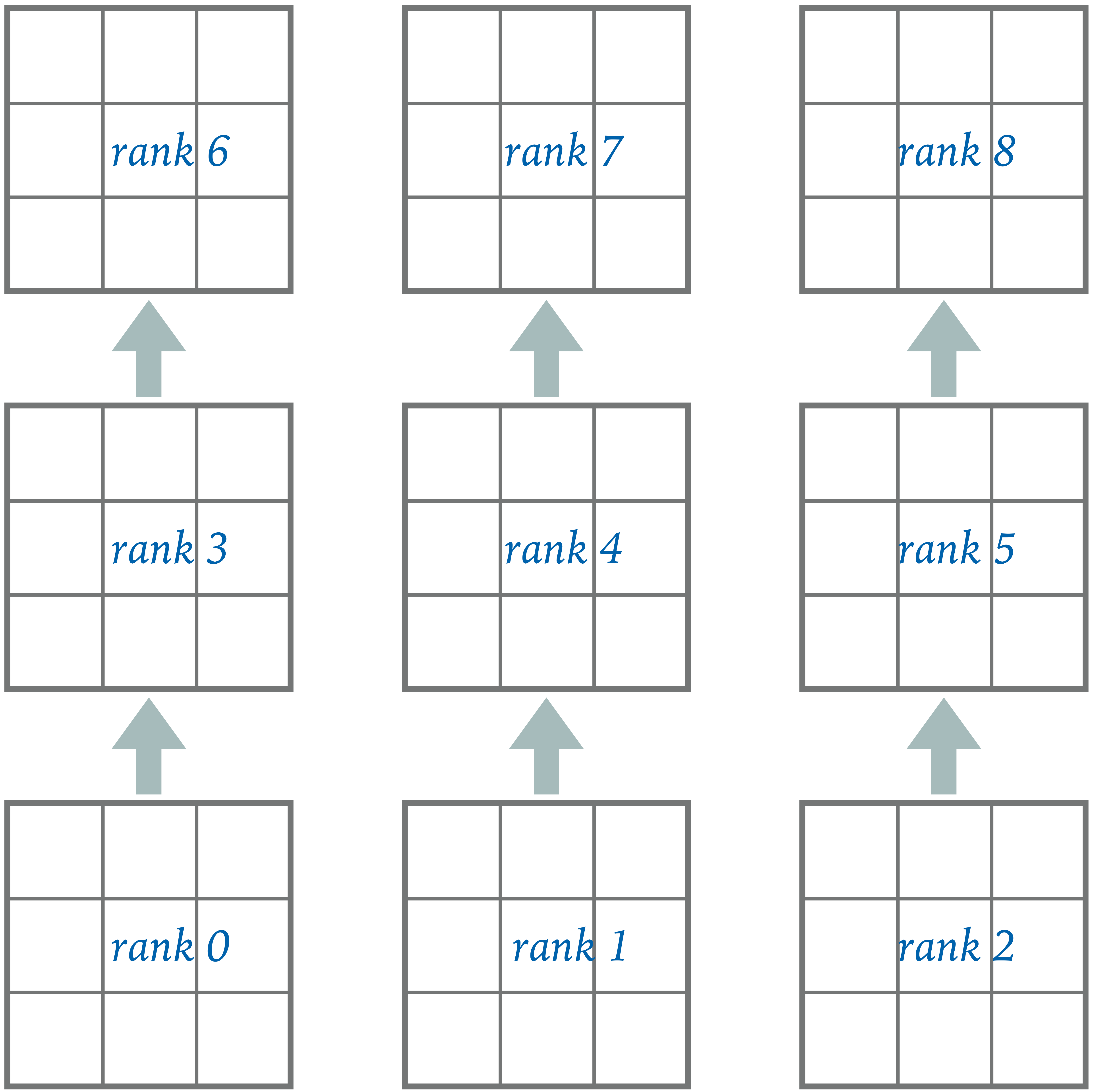



rank 6
rank 7
rank 8
rank 3
rank 4
rank 5
rank 0
rank 1
rank 2
Compute interior points
if not the first or last rank column
SendRecv from: left to: right
SendRecv from: right to: left
SendRecv from: down to: up


rank 6
rank 7
rank 8
rank 3
rank 4
rank 5
rank 0
rank 1
rank 2
Compute interior points
if not the first or last rank column
SendRecv from: left to: right
SendRecv from: right to: left
if not the first or last rank row
SendRecv from: down to: up
SendRecv from: up to: down



rank 6
rank 7
rank 8
rank 3
rank 4
rank 5
rank 0
rank 1
rank 2
if not the first or last rank column
if not the first or last rank row
Compute interior points
SendRecv from: left to: right
SendRecv from: right to: left
SendRecv from: down to: up
SendRecv from: up to: down
Compute edge points



$$q = \sqrt{p}$$

$$T_1 = \gamma a N^2$$

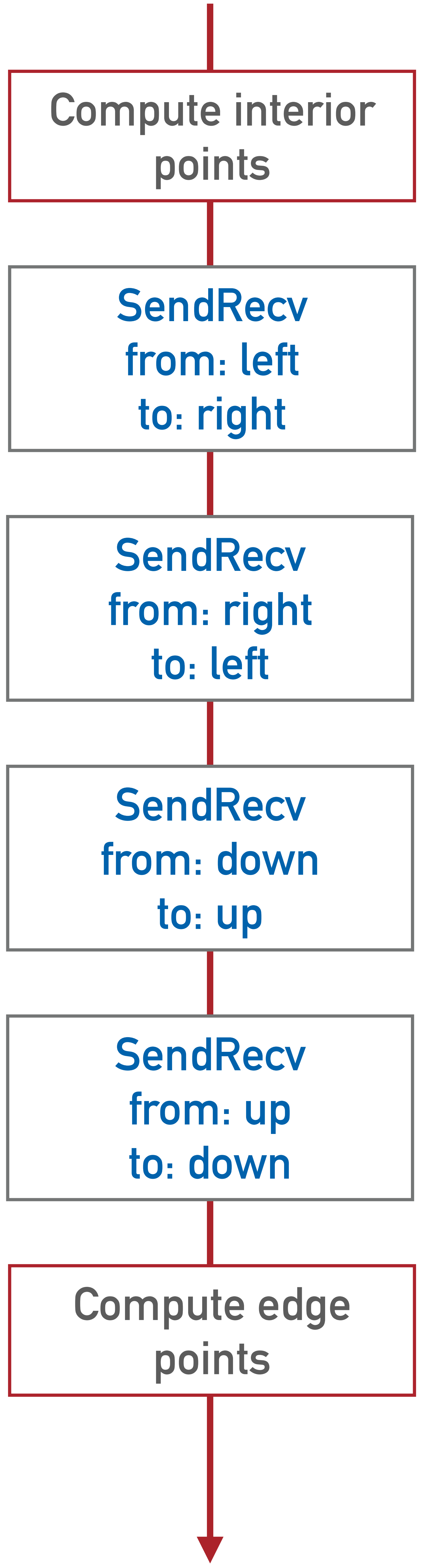
$$T_p = \gamma a \left(\frac{N}{q}\right)^2 + 4\alpha + 4\beta\frac{N}{q}$$

$$E_p = \frac{1}{1 + 4\frac{\alpha}{\gamma}\frac{p}{aN^2} + \boxed{4\frac{\beta}{\gamma}\frac{1}{a}\sqrt{\frac{p}{N^2}}}}$$

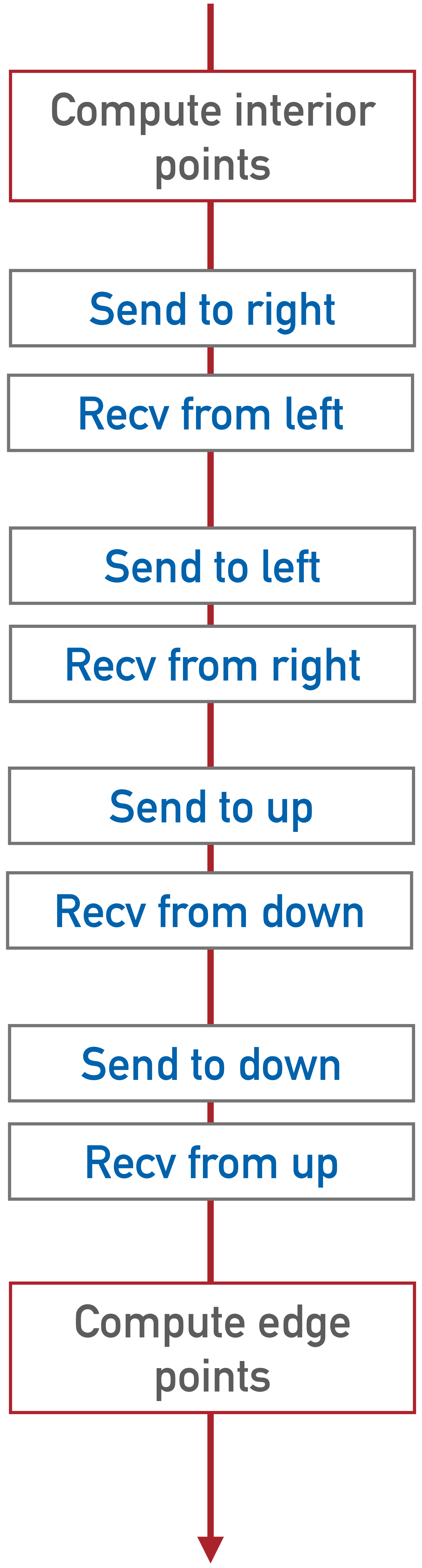
compare to 1D efficiency:

$$E_p = \frac{1}{1 + 2\frac{\alpha + \beta}{\gamma}\frac{p}{aN}}$$

NON-BLOCKING COMMUNICATION



NON-BLOCKING COMMUNICATION



NON-BLOCKING COMMUNICATION

$$T_1 = p(T_{comp} + T_{edge})$$

$$T_p = max(T_{comp}\ ,\ T_{comm}) + T_{edge}$$

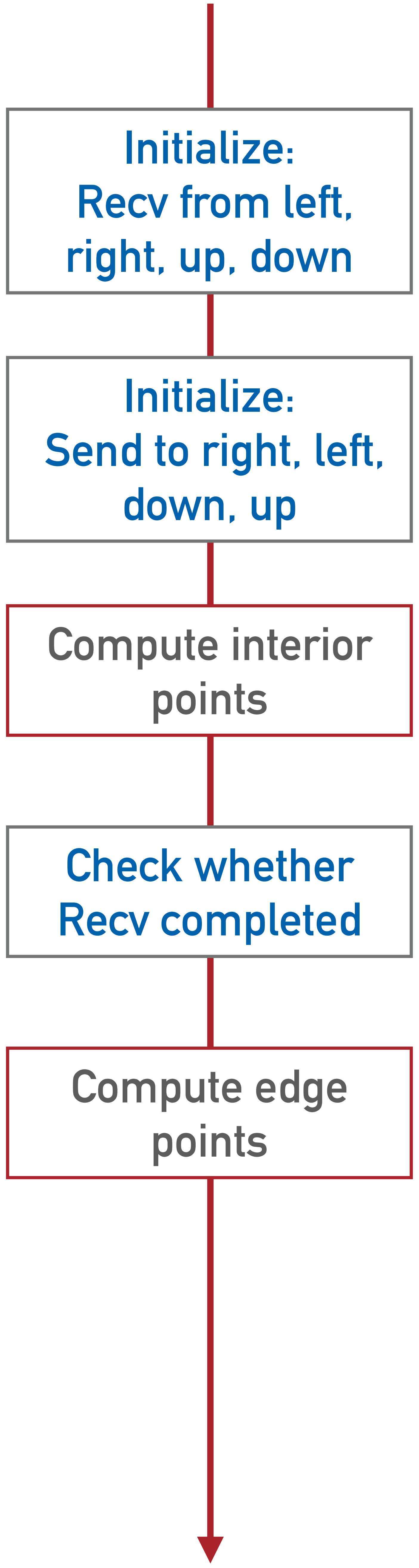
$$E_p = \frac{T_1}{pT_p} = \frac{p(T_{comp} + T_{edge})}{p\left(\ max(T_{comp}\ ,\ T_{comm}) + T_{edge}\right)}$$

$$if \quad T_{comp} > T_{comm} \quad then \quad E_p = 1$$

$$\gamma a \frac{N^2}{p} > 4\beta \frac{N}{\sqrt{p}} \qquad ignoring\ \alpha$$

$$p < \left(\frac{\gamma}{4\beta}\right)^2 a^2 N^2$$

$$T_{comm} = 4\alpha + 4\beta \frac{N}{q}$$



$$T_{comp} = \gamma a \left(\frac{N}{q}\right)^2$$

$$T_{edge} = 4\gamma a \frac{N}{q}$$

NON-BLOCKING COMMUNICATION

```
int MPI_Irecv(void *buf, int count, MPI_Datatype datatype, int source, int tag,
              MPI_Comm comm, MPI_Request *request)

int MPI_Isend(void *buf, int count, MPI_Datatype datatype, int dest, int tag,
              MPI_Comm comm, MPI_Request *request)


int MPI_Wait(MPI_Request *request, MPI_Status *status)

int MPI_Test(MPI_Request *request, int *ready, MPI_Status *status)
```