universidade de aveiro

Computação Paralela
Módulo MPI

2021/2022

Rui Costa

Email: americo.costa@ua.pt

Recall Jacobi algorithm

- Jacobi method finds solutions of *diagonally dominant* systems of linear equations of the type $\hat{A}\vec{x} = \vec{b}$, using an iterative procedure.
- Decomposing $\hat{A} = \left(\hat{A} - \widehat{D}\right) + \widehat{D}$, and rearranging the matrix eq. we write $\vec{x} = (\hat{I} - \widehat{D}^{-1}\hat{A})\vec{x} + \widehat{D}^{-1}\vec{b}$.
- Computing the right-hand side for arbitrary vector, say $\vec{x}^{(i)}$, results in a vector $\vec{x}^{(i+1)}$ that better approximates the desired solution. In other words, by iterating enough times

$$\vec{x}^{(i+1)} = (\hat{I} - \widehat{D}^{-1}\hat{A})\vec{x}^{(i)} + \widehat{D}^{-1}\vec{b}$$

the vectors $\vec{x}^{(i)}$ converge to solution of the original system.

Recall 2D Poisson equation

universidade de aveiro

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) V(x, y) = f(x, y)$$

Approximate 2nd derivatives with finite differences

$$\frac{\partial^2}{\partial x^2} V(x, y) \approx \frac{1}{h^2}[V(x + h, y) - 2V(x, y) + V(x - h, y)],$$

$$\frac{\partial^2}{\partial y^2} V(x, y) \approx \frac{1}{h^2}[V(x, y + h) - 2V(x, y) + V(x, y - h)],$$

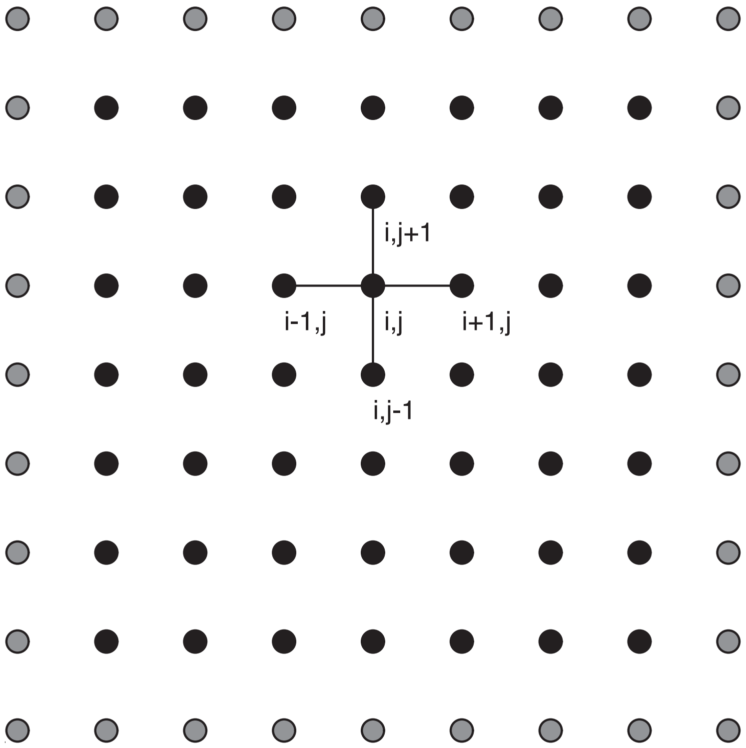
and write the (Jacobi) iterative discretized Poisson equation as

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

Boundary condition is needed to stop the recursion at domain border.

Recall 2D Poisson equation

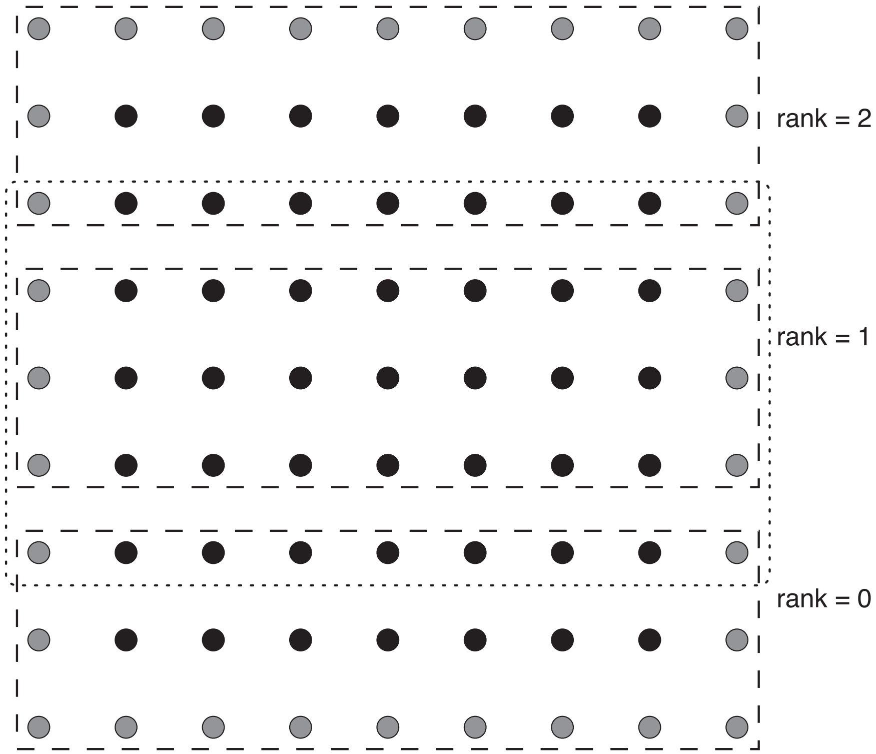
```
integer i, j, n
double precision u(0:n+1,0:n+1), unew(0:n+1,0:n+1)
do j=1, n
   do i=1, n
      unew(i,j) = &
          0.25*(u(i-1,j)+u(i,j+1)+u(i,j-1)+u(i+1,j) - &
          h * h * f(i,j))
   enddo
enddo
```



2D Jacobi parallel algorithm
Domain decompositions

universidade de aveiro

```
integer i, j, n
double precision u(0:n+1,s:e), unew(0:n+1,s:e)
do j=s, e
   do i=1, n
      unew(i,j) = &
           0.25*(u(i-1,j)+u(i,j+1)+u(i,j-1)+u(i+1,j) - &
           h * h * f(i,j))
   enddo
enddo
```



Virtual Topologies

universidade de aveiro

Creating a new Cartesian communicator:

```
MPI_Cart_create(old_comm, ndims, dims[],
isperiodic[], reorder, &new_cart_comm)
```

Getting the rank of neighbours (in cartesian decomposition) with whom there will be communications:

```
MPI_Cart_shift(new_cart_comm, direction, displ,
&src, &dest)
```

Avoiding communications deadlock with MPI_Sendrecv

universidade de aveiro

```
MPI_Sendrecv(&sendbuf, sendcount, sendtype,
dest, sendtag, &recvbuf, recvcount, recvtype,
source, recvtag, comm, MPI_STATUS_IGNORE)
```

Collective data movements: MPI_Scatter & MPI_Gather

```
MPI_Scatter(&sendbuf, sendcount, sendtype,
&recvbuf, recvcount, recvtype, root, comm)
```

```
MPI_Gather(&sendbuf, sendcount, sendtype,
&recvbuf, recvcount, recvtype, root, comm)
```

Bibliography

Using MPI: portable parallel programming with the message-passing interface, 3rd edition, William Gropp, Ewing Lusk, and Anthony Skjellum, MIT press (2014).