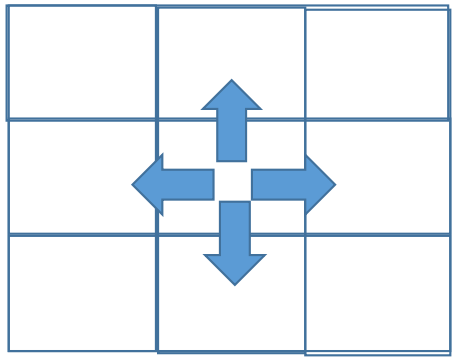
MPI Tutorial

MPI Course February 2017

Jacobi Solver

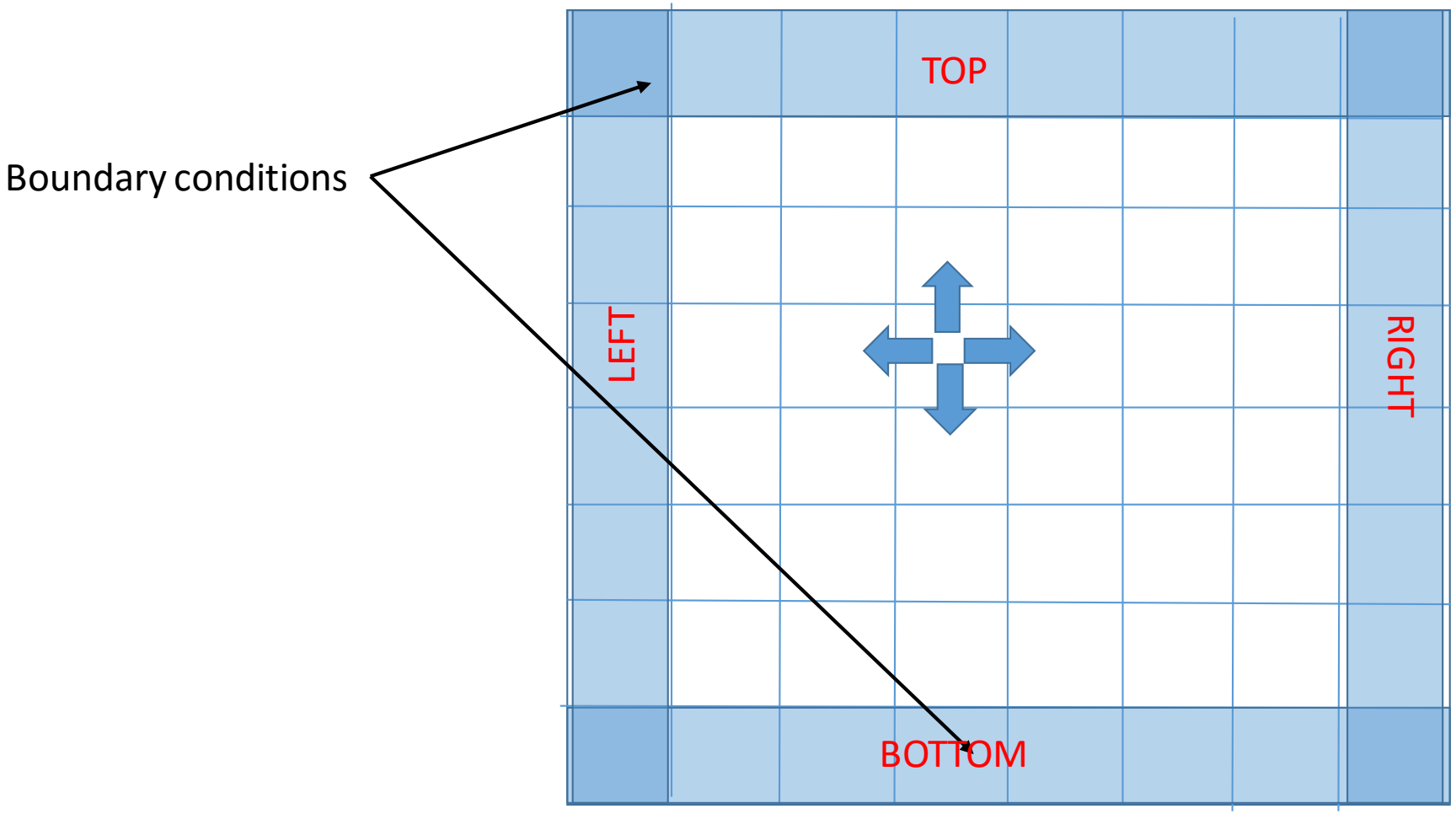
- The Jacobi method is an iterative algorithm for solving a system of linear equations.
- In the 2D model an approximation can be made by taking the average of the 4 neighbouring values (4 point stencil).



```
REAL A(0:n+1,0:n+1), B(1:n,1:n)
...
!Main Loop
DO WHILE(.NOT.converged)
   ! perform 4 point stencil
   DO j=1, n
      DO i=1, n
         B(i,j)=0.25*(A(i-1,j)+A(i+1,j)+A(i,j-
1)+A(i,j+1))
      END DO
   END DO

   ! copy result back into array A
   DO j=1, n
      DO i=1, n
         A(i,j) = B(i,j)
      END DO
   END DO
...
! convergence test
END DO
```

Jacobi Grid



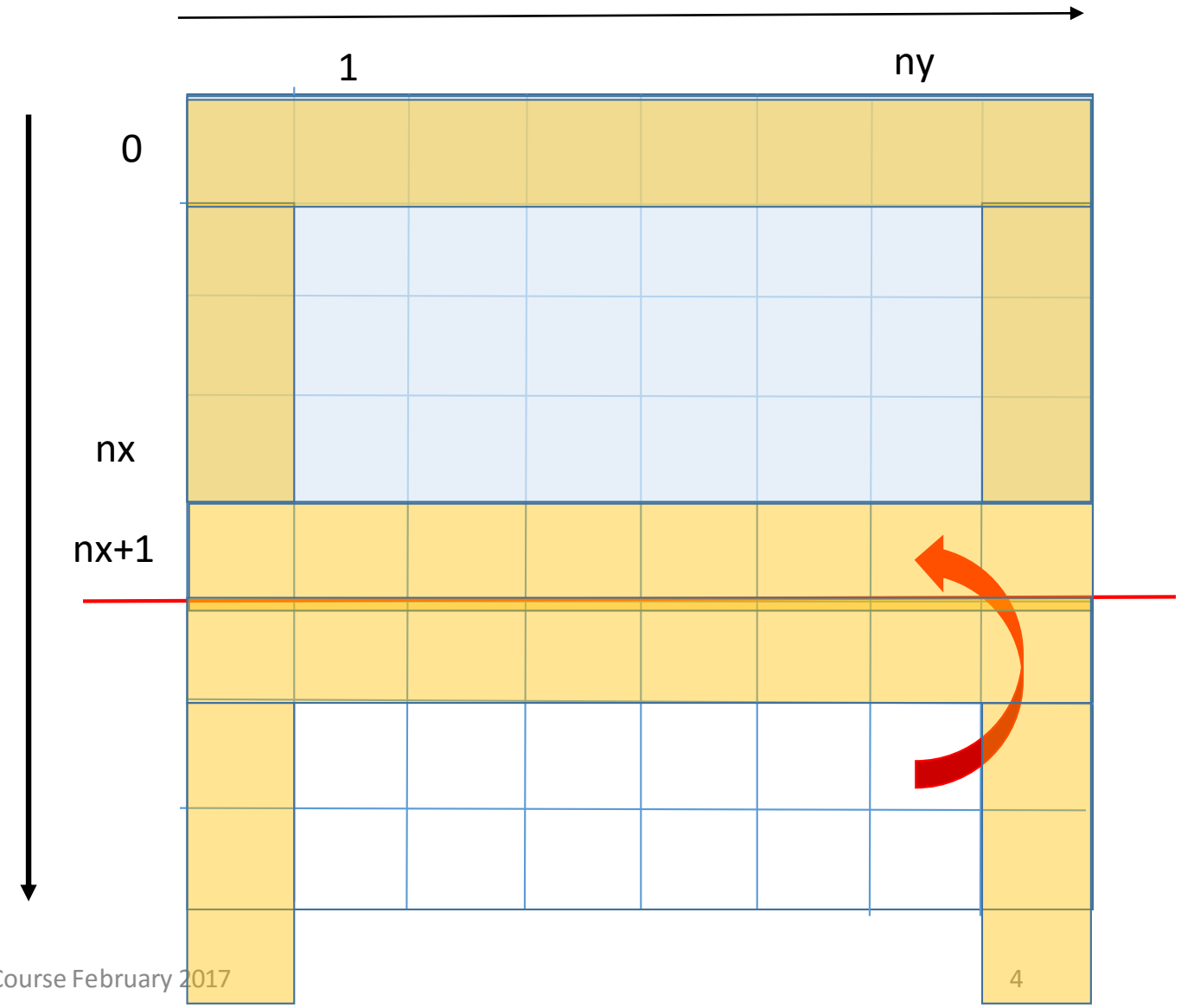
Parallel Jacobi solver- How to do it

Divide up the grid into sub domains, one for each rank (1..ny,1..nx), and add "halo" regions to store values from neighbouring domains

(0..ny+1,0..nx+1)

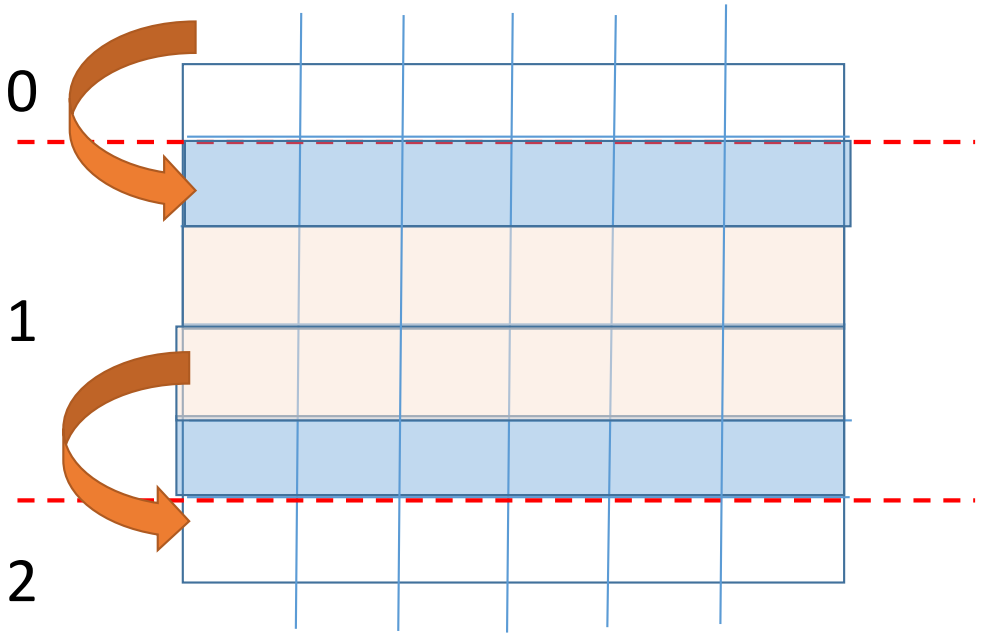
(For the top and bottom domains you need only one halo region.)

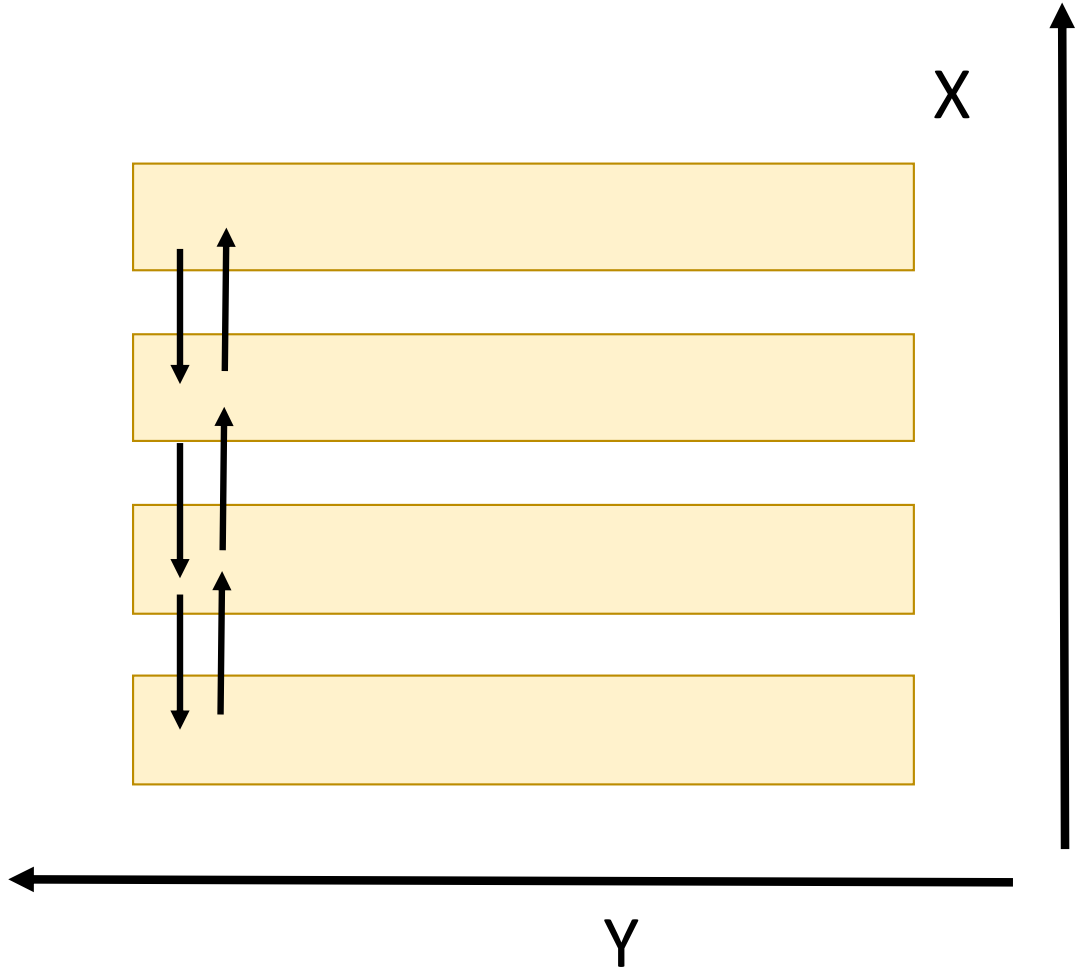
You can use MPI P2P commands to transfer data from one domain to another.



Jacobi Solver - parallel

The parallel version can use a simple 1D decomposition and MPI P2P commands for the halo communications.





Practical – MPI Point-to-Point

1. Download the MPI C or Fortran source code:
   - git clone https://gitlab.hpc.cineca.it/training/mpi-feb-2107.git
2. You will find the serial versions of the Jacobi solver in C and Fortran. Compile and run in the batch system for various grid sizes.
3. You will also find “templates” for the MPI versions. Your goal is to insert the MPI commands in the program where indicated. Try:
   1. MPI Blocking sends and receives
   2. MPI Non-blocking sends and receives.
4. Run the MPI program for various grid sizes and number of ranks.

Based on the example at http://www.archer.ac.uk/training/course-material/2016/09/160929_AdvMPI_EPCC/index.php

Practical - advanced

The MPI P2P Sends and Receives can be replaced with MPI Remote Access Calls.

Procedure:

- Add and MPI_Win_Create command to create an RMA Window.
- Choose to either use MPI_Put or MPI_Get.
- In the case of MPI_Get, replace the MPI_Irecv with an MPI_Get, using the information contained in the MPI_send. Reverse logic for MPI_Put.
- Surround the MPI_Get/MPI_Put calls with MPI_Win_fence and add an MPI_Win_Free command when finished.

Compare the RMA performance with that of the MPI P2P version.

Practical (optional) – Using PMPI

- Objective: Print out the number of MPI Send and Receive calls *without* modifying the Jacobi source code.
- Method:
  - In a separate file, write alternative versions of MPI_send and MPI_receive using PMPI (profiling MPI) to keep a count of the number of times the commands have been called.
  - In the same file re-write MPI_Finalize to print out the counters.
  - Compile (with –c) and link into the MPI Jacobi program.