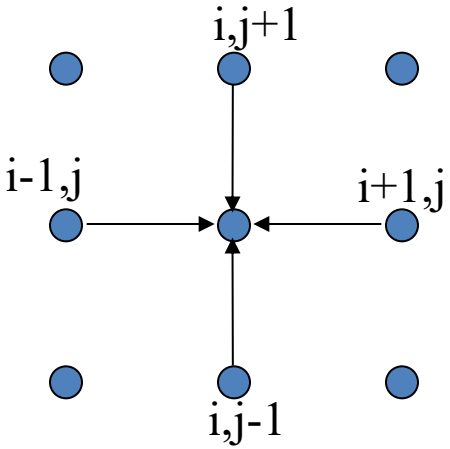
Homework 4

An implementation of Laplace's equation using MPI

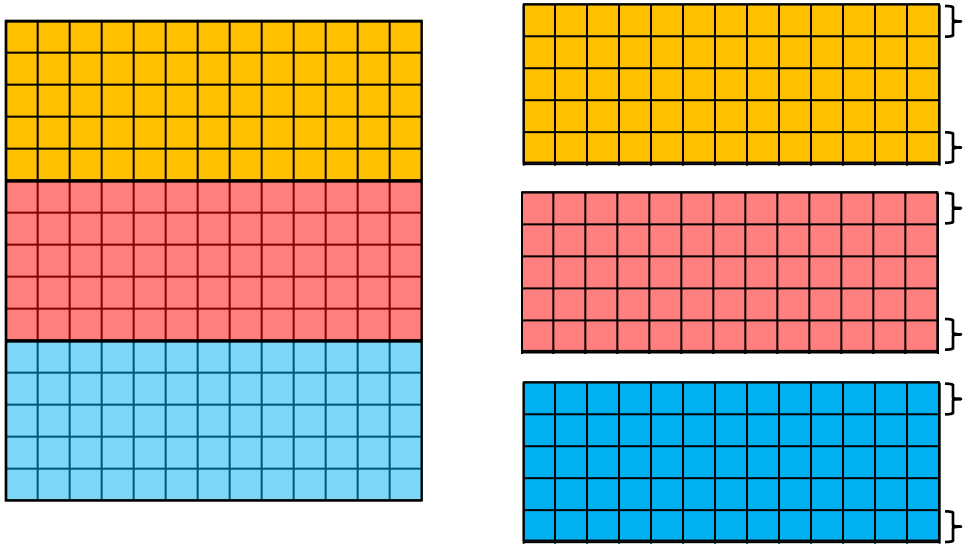
Deadline: February 12th 2020

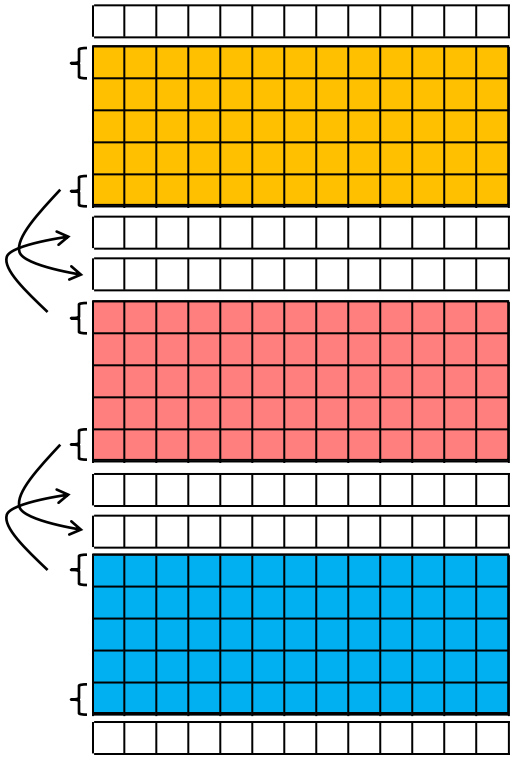
$$U_{i,j}^{n+1} = \frac{1}{4}\left(U_{i-1,j}^{n} + U_{i+1}^{n} + U_{i,j-1}^{n} + U_{i,j+1}^{n}\right)$$

Laplace's equation - MPI




```
for j = 1 to jmax
  for i = 1 to imax
    Unew(i,j) = 0.25 * ( U(i-1,j) + U(i+1,j)
                       +   U(i,j-1) + U(i,j+1))
  end for
end for
```





- Assuming you have a 2-dimensional matrix stored in row-major format, compute a well-defined number of iterations of the computation of the Laplace equation using multiple MPI processes.
  - Use as many MPI functions as possible (hint: define datatypes)
  - Minimize memory requirements
  - Use 1D block or 2D block/block data distributions. Justify your choice.
  - Implement support for ghost region
- Originally the matrix is initialized with 0 everywhere except the boundaries (first and last row and first and last column) which are initialized differently.
- Highlight the impact of using multiple MPI processes to execute this algorithm by doing an analysis of the algorithm's performance. Vary the number of processes and the problem size (weak and strong scaling) and comment on the results.
  - Present the average over multiple runs to account for any measuring error or outside effects

- Benchmarking of the Laplace algorithm should be measured excluding the initialization of the MPI environment.
  - Keep everything related to setting up the problem outside of the timed section
- A skeleton code is available on the class website (or can be obtained by email from the TA).
- You are strongly encouraged to explore different MPI programming technique, such as blocking, non-blocking, persistent communications, in order to improve the scalability of your solution.
  - Detail your approach on the final document.
- A skeleton code will be provided, you need to add MPI to it.