High-performance Computing

Coursework Assignment

Deadline: 18th March 2020 - 17:00

Instructions

Please take note of the following when completing this assignment:

- Read all the tasks carefully and plan ahead before you start designing and implementing your code.
- You may use any of the tools and libraries available on the provided Linux environment.
- Your submitted code **must** compile and run correctly on the provided Linux environment.

Make regular backups of your code onto a separate computer system; no allowance will be made for data loss resulting from human or computer error.

1 Introduction

The objective of this coursework is to write a parallel numerical code for solving the vorticity-stream function formulation of the incompressible Navier-Stokes equations in 2D using the finite difference method. The specific problem to be solved is the lid-driven cavity, illustrated in Figure 1. Fluid within the cavity is driven by the lateral flow across the top of the cavity and is a common test-case for numerical flow solvers.

The incompressible Navier-Stokes equations are given by:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \frac{1}{Re}\nabla^2 u + \mathbf{g} \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

where $\mathbf{u} = (u, v)$ is the velocity field in the directions (x, y) respectively, p is the pressure, ρ is the density of the fluid and Re is the Reynolds number, a non-dimensional constant. This is subject to initial and boundary conditions,

$$\begin{aligned} \mathbf{u}(x, y, 0) &= \mathbf{u_0}(x, y), \quad \text{on} \quad \Omega \\ \mathbf{u}(x, y, t) &= \mathbf{g}(x, y), \quad \text{on} \quad \partial\Omega \end{aligned}$$

1.1 Vorticity stream function formulation

One approach to solving Equations (1) and (2) is to express them in terms of vorticity and a corresponding stream function. Recall the stream function Ψ for a two-dimensional incompressible flow satisfies:

$$\frac{\partial}{\partial x}\left(\frac{\partial \psi}{\partial y}\right) + \frac{\partial}{\partial y}\left(-\frac{\partial \psi}{\partial x}\right) = 0,$$

and so naturally satisfies the incompressibility constraint given in Equation 2. ψ is therefore related to the velocity as:

$$\frac{\partial \psi}{\partial y} = u \qquad \frac{\partial \psi}{\partial x} = -v \tag{3}$$

We next assume that the gravity term is expressed as a potential function $\boldsymbol{g} = \nabla\phi$. Taking the curl of Equation (1) the pressure and gravity terms disappear. After some rearranging, we arrive at:

$$\frac{\partial \omega}{\partial t} + \frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right) \tag{4}$$

where ω is the vorticity which – using Equation (3) – is related to the stream-function as

$$\omega = \nabla \times \boldsymbol{V} = -\left(\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}\right)\hat{\mathbf{k}} = -\nabla^2 \psi \hat{\mathbf{k}} \tag{5}$$

Using Equations (4) and (5) we can solve for both ω and ψ. From these we can easily recover the velocity from Equation (3).

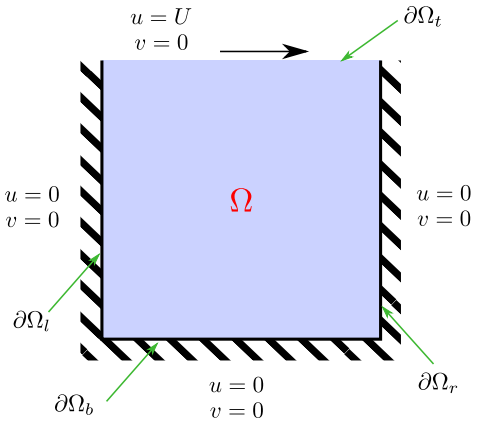

Figure 1: Diagram of the lid-driven cavity problem. The domain is bounded on three sides by walls, with fluid flowing at a uniform horizontal velocity across the top.

1.2 Boundary conditions

No-slip conditions $(u, v) = (0, 0)$ are prescribed on the lower and side walls, $\partial\Omega_l$, $\partial\Omega_b$ and $\partial\Omega_r$ (indicated in Figure 1. For the top surface, $\partial\Omega_t$, there is a prescribed horizontal velocity $(u, v) = (U, 0)$. However, in the vorticity streamfunction formulation we need boundary conditions for vorticity and the streamfunction.

To derive boundary conditions for the vorticity on the top surface $\partial\Omega_t$, we can expand the stream-function in terms of a Taylor series about a boundary point and evaluate a short distance dy from the wall to obtain:

$$\psi(x, L_y - dy) = \psi(x, L_y) + \frac{\partial \psi(x, L_y)}{\partial y}(-dy) + \frac{\partial^2 \psi(x, L_y)}{\partial y^2}\frac{dy^2}{2} + \mathcal{O}(dy^3).$$

We plug in the definitions of the stream-function (Equation 3) and vorticity (Equation 5) to derive a condition at the boundary. After dropping the higher order terms and rearranging we arrive at

$$\omega(x, L_y) = (\psi(x, L_y) - \psi(x, L_y - dy))\frac{2}{dy^2} - \frac{2U}{dy}.$$

Similar expressions can be derived for the solid walls in which the last term will, of course, be absent.

The velocity in the direction perpendicular to each surface is zero. Therefore, from Equation (3) the derivative of the stream function in the direction parallel to the wall is zero. Consequently we must have that

$$\psi|_{\partial\Omega} = c$$

where c is an arbitrary constant, which we can assume to be zero.

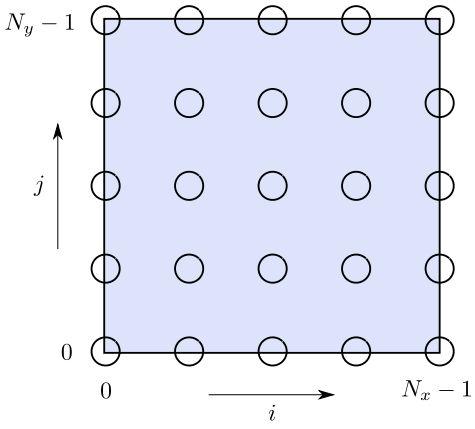

Figure 2: Illustration of the finite difference grid showing one possible indexing of nodes.

1.3 Initial Conditions

The fluid is initially at rest. Therefore, initial conditions at $t = 0$ are $\omega(x, y) = 0$ and $\psi(x, y) = 0$.

Continues on next page ...

2 Finite Difference Method Discretisation

In order to solve Equations (4) and (5) numerically, space and time need to be discretized. We suppose the domain is $[0, L_x] \times [0, L_y]$ and is represented by a grid of $N_x \times N_y$ points. The grid spacing is therefore $\Delta x = L_x/(N_x - 1)$ and $\Delta y = L_y/(N_y - 1)$. You can assume the lid velocity $U = 1.0$ throughout this assignment.

We will use an explicit (forward) time-integration scheme when discretising the time derivative in Equation (4) to compute the new vorticity. Spatial derivatives will use second-order central-difference. A restriction on the time-step is given by $\Delta t < \frac{Re\,\Delta x \Delta y}{4}$.

2.1 Algorithm

The vorticity and stream function are time integrated iteratively. At each time step (from time t to $t + \Delta t$):

1. the current vorticity at time t is updated on the boundaries
2. the current vorticity at time t is updated in the interior
3. the new vorticity at time $t + \Delta t$ is computed in the interior
4. the new stream function at time $t + \Delta t$ is computed by solving the Poisson equation given by Equation (5).

The details of these steps are given below.

Calculation of vorticity boundary conditions at time t:

Top:
$$\omega^n_{i,N_y-1} = (\psi_{i,N_y-1} - \psi_{i,N_y-2})\frac{2}{(\Delta y)^2} - \frac{2U}{\Delta y} \tag{6}$$

Bottom:
$$\omega^n_{i,0} = (\psi_{i,0} - \psi_{i,1})\frac{2}{(\Delta y)^2} \tag{7}$$

Left:
$$\omega^n_{0,j} = (\psi_{0,j} - \psi_{1,j})\frac{2}{(\Delta x)^2} \tag{8}$$

Right:
$$\omega^n_{N_x-1,j} = (\psi_{N_x-1,j} - \psi_{N_x-2,j})\frac{2}{(\Delta x)^2} \tag{9}$$

Calculation of interior vorticity at time t:

$$\omega^n_{i,j} = -\left(\frac{\psi^n_{i+1,j} - 2\psi^n_{i,j} + \psi^n_{i-1,j}}{(\Delta x)^2} + \frac{\psi^n_{i,j+1} - 2\psi^n_{i,j} + \psi^n_{i,j-1}}{(\Delta y)^2}\right) \tag{10}$$

Calculation of interior vorticity at time $t + \Delta t$:

$$\frac{\omega^{n+1}_{i,j} - \omega^n_{i,j}}{\Delta t} + \left(\frac{\psi^n_{i,j+1} - \psi^n_{i,j-1}}{2\Delta y}\right)\left(\frac{\omega^n_{i+1,j} - \omega^n_{i-1,j}}{2\Delta x}\right) - \left(\frac{\psi^n_{i+1,j} - \psi^n_{i-1,j}}{2\Delta x}\right)\left(\frac{\omega^n_{i,j+1} - \omega^n_{i,j-1}}{2\Delta y}\right) = \frac{1}{Re}\left(\frac{\omega^n_{i+1,j} - 2\omega^n_{i,j} + \omega^n_{i-1,j}}{(\Delta x)^2} + \frac{\omega^n_{i,j+1} - 2\omega^n_{i,j} + \omega^n_{i,j-1}}{(\Delta y)^2}\right) \tag{11}$$

Solution of a Poisson problem to compute the stream-function at time $t + \Delta t$:
This system of equations can be solved using a linear solver of your choice.

$$\omega^{n+1}_{i,j} = -\left(\frac{\psi^{n+1}_{i+1,j} - 2\psi^{n+1}_{i,j} + \psi^{n+1}_{i-1,j}}{(\Delta x)^2} + \frac{\psi^{n+1}_{i,j+1} - 2\psi^{n+1}_{i,j} + \psi^{n+1}_{i,j-1}}{(\Delta y)^2}\right) \tag{12}$$

Continues on next page ...

Tasks

The objective of this coursework is to write a high-performance parallel C++ code which will solve the lid-driven cavity problem as described by the algorithm above. A template code is provided to get you started. Subject to the constraints outlined below, you are free to change this as you wish (e.g. add/remove member functions, classes and parameters to those functions already defined, etc).

In completing this assignment you should write a **single** code which satisfies the following requirements:

- Reads the parameters of the problem **from the command line** (do not prompt the user for input). Please use the **exact** syntax given for the parameters below as these will be used when testing your code.

```
--Lx arg          Length of the domain in the x-direction.
--Ly arg          Length of the domain in the y-direction.
--Nx arg          Number of grid points in x-direction.
--Ny arg          Number of grid points in y-direction.
--Px arg          Number of partitions in the x-direction (parallel)
--Py arg          Number of partitions in the y-direction (parallel)
--dt arg          Time step size.
--T arg           Final time.
--Re arg          Reynolds number.
```

  [5%]
- Implements a class `LidDrivenCavity` given in the template to solve the lid-driven cavity problem using the algorithm described above. You are free to add any additional functions or member variables as required. [30%]
- Implements a separate class for the Poisson solver. [10%]
- Is parallelised using MPI through partitioning of the domain Ω into sub-domains. It should be able to run on a user-selected number of processes (as specified by the `-np` parameter to `mpiexec`). You should check the value given to `-np` is compatible with the command-line arguments `--Px` and `--Py`. [30%]
- Uses a build system (Make or CMake) to compile your code and uses git for version control. Provide evidence of the latter by running the command

```
git log --name-status > repository.log
```

  and including the file `repository.log` in your submission. [5%]
- Uses good coding practices (code layout, comments, documentation) [5%]
- Write a short report (max 3 pages) as detailed below. [15%]

2.2 Report (3 pages)

Write a short and concise report which includes:

- results of running your code:
  - a plot showing horizontal velocity u along the line $y = 0.5$ of the steady-state solution for Reynolds numbers of 100, 400, 1000 and 3200 (all on the same plot please) using a 161x161 grid and a domain size of $L_x = L_y = 1$;
  - an equivalent plot showing vertical velocity v along the line $x = 0.5$ for the same parameters;
  - a contour plot of the vorticity and streamfunction at $Re = 100$ on the same domain as above;
  - a table listing the coordinates of the streamfunction minimum at the above Reynolds numbers.
  - streamfunction contour plots of steady-state solutions with $(L_x, L_y) = (1, 2)$ and $(L_x, L_y) = (2, 1)$ at $Re = 100$.
- a discussion of how you chose to parallelise your code (and why) as well as any design decisions or optimisations you explicitly made to improve the serial and/or parallel performance of your code;
- analysis of the parallel performance of your code (typically in the form of a scaling plot).

Submission and Assessment

When submitting your assignment, make sure you include the following:

- All the files needed to compile and run your C++ code:
  - Source files for a single C++ program which performs all the requirements. i.e. All `.cpp` and `.h` files necessary to compile and run the code.
  - The `Makefile` or `CMakeLists.txt` file (as appropriate) used for compiling your code and linking with any necessary libraries.
- Your three-page report (in PDF format only).
- The git log (`repository.log`).

These files should be submitted in a **single `tar.gz` archive file** to Blackboard Learn.

Preparing your submission

To generate your `tar.gz` archive in the Linux environment, put all files to be submitted in a new directory (e.g. `hpc-assessment`) and run the following command **from the directory above it**:

```
tar -cvzf submission.tar.gz hpc-assessment
```

Please copy your archive to a temporary directory elsewhere and unpack it to ensure you have included all the files.

```
tar -xvf submission.tar.gz
```

It is your responsibility to ensure all necessary files are submitted.

You may make unlimited submissions and the last submission before the deadline will be assessed.

END OF ASSIGNMENT

Competition! (entirely optional)

Maximising the efficiency and execution speed of a code is critical for High-performance Computing applications. Many aspects of a code's design affect performance, such as the ordering of operations, memory layout and loop ordering. As an additional challenge, you can **opt-in** to your code being included in the *High-performance Computing performance challenge*! Your code will be run both in serial and in parallel on a large parallel computer system to measure its performance.

Note: Submissions must correctly solve the problem in parallel in order to be included in the parallel competition.

Choosing to enter the competition is entirely optional and participation (or lack of) will have no bearing on your mark for this module.

Entries for the competition will be assessed and ranked after your assignments are marked and feedback has been returned.