High-performance Computing

Coursework Assignment

Deadline: 20th March 2024 – 23:00

Instructions

Please take note of the following when completing this assignment:

- Read all the tasks carefully and plan ahead before you start.
- 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 parallelise and optimise a 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 being 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.

A fully functional unoptimised serial solver is provided along with this document. The formulation provided below is intended to help you understand the problem and the provided code.

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).

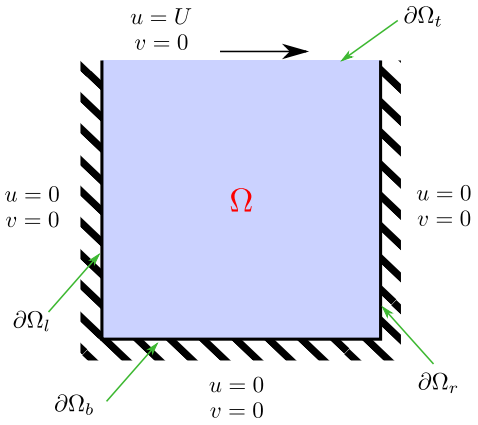

Figure 1: Diagram of the lid-driven cavity problem. The domain is bounded on three sides by walls, with fluid flowing at a uniform horizontal velocity across the top.

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.

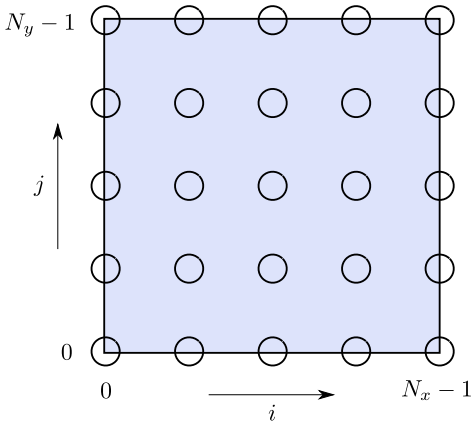

Figure 2: Illustration of the finite difference grid showing one possible indexing of nodes.

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 is solved using a linear solver.

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

Continues on next page ...

Tasks

Extend and optimise the lid-driven cavity code provided to support parallel execution using both distributed and shared-memory parallelism. Write a report (maximum 5 pages) as described in the tasks.

1. Add the initial code to a new Git repository and continue to use git appropriately throughout your code development.

   When submitting, provide evidence of your use of git by running the command

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

   and including the file `repository.log` in your submission. [5%]

2. Implement a suitable Makefile to compile the code.

   The executable generated should be called `solver`. [5%]

3. Add support for generating HTML code documentation using doxygen and document the source code appropriately. Add a `doc` target to the Makefile to compile the documentation. [5%]

4. Add at least two unit tests to provide evidence your code continues to execute correctly when tackling the subsequent tasks. As a minimum, you should have a test for:

   - the linear CG solver class;
   - the lid-driven cavity class.

   Add a `unittests` target to the Makefile to compile the unit tests executable.

   Report: Provide a brief description of the unit tests implemented. [5%]

5. Extend the code to execute in parallel using $P = p^2$ MPI ranks for integer $p > 0$. Other choices of P may be considered erroneous and your code should terminate. Parallelisation should be achieved using domain decomposition with each dimension of the domain partitioned p ways. [30%]

   Report: Provide a scaling plot showing the speedup of the code when run in parallel, relative to the case in serial, up to $P = 16$ ranks. [5%]

   Report: Concisely summarise your approach to parallelising the code with MPI and the changes you needed to make to convert the code to run in parallel. How did you try to ensure the work done by each process was as balanced as possible? Discuss the scaling plot and whether you consider it shows good scaling and, if not, why? [5%]

6. Add multi-threading to the code using OpenMP. The number of threads used by each process should respect the value of the `OMP_NUM_THREADS` environmental variable. [10%]

   Report: Provide a scaling plot showing the speed-up of your code when run in parallel, relative to the case in serial, using up to 16 threads. [5%]

   Report: Concisely summarise your approach to parallelising the code with OpenMP. Discuss the scaling plot and whether you consider it shows good scaling and, if not, why? [5%]

7. Optimise your code using a profiler. Use one MPI process and one OpenMP thread for this part. [10%]

   Report: Describe, with quantitative evidence, the most time-consuming parts of your code. List optimisations you have made to the code to improve performance, and quantify the change observed. [5%]

8. Use good coding practices (code layout, comments, etc) throughout your code. [5%]

Continues on next page ...

Code usage

The provided code accepts a number of command-line arguments to set the parameters of the problem. These can be seen using the `--help` option:

```
$ ./solver --help
Solver for the 2D lid-driven cavity incompressible flow problem:
--Lx arg (=1)          Length of the domain in the x-direction.
--Ly arg (=1)          Length of the domain in the y-direction.
--Nx arg (=9)          Number of grid points in x-direction.
--Ny arg (=9)          Number of grid points in y-direction.
--dt arg (=0.002)      Time step size.
--T arg (=1)           Final time.
--Re arg (=10)         Reynolds number.
--verbose              Be more verbose.
--help                 Print help message.
```

The default parameters provide a very coarse, but very fast solution.

For assessing performance and parallel scaling, you should use a larger problem. For example:

```
$ ./solver --Lx 1 --Ly 1 --Nx 201 --Ny 201 --Re 1000 --dt 0.005 --T 50
```

You are encouraged to explore performance using different combinations of parameters, but please state those used when presenting results in your report.

Submission and Assessment

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

- All the files needed to compile and run your C++ code which solves the Lid-driven cavity problem, modified as described above. i.e. All `.cpp` and `.h` files necessary to compile and run the code.
- The git log (`repository.log`) from Task 1.
- The `Makefile` from Task 2.
- The doxygen configuration from Task 3.
- The Unit Test code from Task 4.
- Your report (maximum 5 pages, in PDF format only).

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

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