Module Code
COM00036H

BSc, BEng and MEng Degree Examinations 2022–3

DEPARTMENT OF COMPUTER SCIENCE

High-Performance Parallel and Distributed Systems

Open Individual Assessment

Issued: 8th March 2023, 12:00 noon
Submission due: 19th April 2023, 12:00 noon
Feedback and marks due: 24th May 2023, 12:00 noon

All students should submit their answers through the electronic submission system: http://www.cs.york.ac.uk/student/assessment/submit/ by 19th April 2023, 12:00 noon. An assessment that has been submitted after this deadline will be marked initially as if it had been handed in on time, but the Board of Examiners will normally apply a lateness penalty.

Your attention is drawn to the section about Academic Misconduct in your Departmental Handbook: https://www.cs.york.ac.uk/student/handbook/.

Any queries on this assessment should be addressed by email to Steven Wright at steven.wright@york.ac.uk. Answers that apply to all students will be posted on the VLE.

Rubric:

Carry out the whole task as described in the following pages. Your report must be formatted according to the provided template. The word limit provided for each section should be strictly observed, and failure to do so will result in a mark of 0 for any offending section.

References may be listed at the end of the document and will not count towards word counts.

Your exam number should be on the front cover of your assessment. You should not be otherwise identified anywhere on your submission.

Computational Fluid Dynamics

Fluid dynamics is the subdiscipline of fluid mechanics that is concerned with the flow of fluids – both liquids and gases. It has a wide range of applications across the science and engineering disciplines, from describing the flow of liquids through pipes, to the description of airflow around vehicles and other structures. *Computational fluid dynamics* (or CFD) is the application of computational and numerical methods to solve problems that involve fluid flows. High Performance Computing (or Supercomputing) is widely applied to CFD problems to improve the accuracy of solutions and to accelerate the time-to-solution.

Perhaps the most famous equations in CFD are the *Navier-Stokes* equations, used to treat laminar flows of viscous, incompressible fluids[1]. The equations can be presented in numerous forms beside the derivation below, and are named after French engineer and physicist Claude-Louis Navier and Anglo-Irish physicist and mathematician George Gabriel Stokes. The flow is described by a system of partial differential equations whose dimensionless form is given by:

$$\frac{\partial}{\partial t}\vec{u} + (\vec{u} \cdot \nabla)\vec{u} + \nabla p = \frac{1}{Re}\Delta\vec{u} + \vec{g} \qquad \textit{(momentum equation)} \qquad (1)$$

$$\nabla \cdot \vec{u} = 0 \qquad \textit{(continuity equation)} \qquad (2)$$

where $\vec{u}$ is flow velocity, t is time, p is pressure, Re is the dimensionless Reynolds number and $\vec{g}$ denotes body forces such as gravity[2].

For this assignment, you are provided with a simple solver for the Navier-Stokes equations. The solver operates according to the computational scheme described in Chapter 3 of the book by Griebel et al. [1]. In this assignment, we discretise our space in two dimensions, and we solve the equations using a finite-differencing scheme. We use a *staggered grid*, where the horizontal velocity u is stored on the vertical cell edges, the vertical velocity v is stored on the horizontal cell edges, and the pressure p is stored in the cell centres.

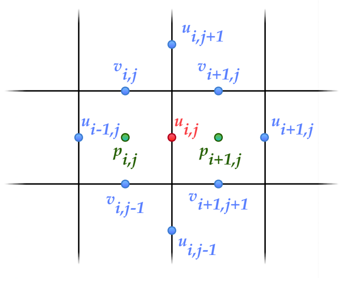

Figure 1: A staggered grid

[1] Fear not, you do not need to understand these equations. An implementation is provided.

[2] ∇ and Δ represent operations from vector calculus. You can read more about them in books such as “div, grad, curl and all that”, or from Wikipedia. Though again, understanding these equations is not required to solve this assessment.

Beginning at time $t = 0$, with given initial values for the u and v vector fields, we increment time by δt each step until t_{end} is reached. At any time step n, the values of all variables are known, and those at time t_{n+1} are to be computed.

We start by providing the discretisation of the momentum equation (Eq. (1)):

$$\begin{aligned} u^{(n+1)} &= u^{(n)} + \delta t \left[\frac{1}{Re} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} + g_x - \frac{\partial p}{\partial x} \right] \\ v^{(n+1)} &= v^{(n)} + \delta t \left[\frac{1}{Re} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) - \frac{\partial (uv)}{\partial x} - \frac{\partial (v^2)}{\partial y} + g_y - \frac{\partial p}{\partial y} \right] \end{aligned} \tag{3}$$

We then introduce the abbreviations, F and G:

$$\begin{aligned} F &:= u^{(n)} + \delta t \left[\frac{1}{Re} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} + g_x \right] \\ G &:= v^{(n)} + \delta t \left[\frac{1}{Re} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) - \frac{\partial (uv)}{\partial x} - \frac{\partial (v^2)}{\partial y} + g_y \right] \end{aligned} \tag{4}$$

Substituting (4) into (3), we obtain the form:

$$\begin{aligned} u^{(n+1)} &= F^{(n)} - \delta t \frac{\partial p^{(n+1)}}{\partial x} \\ v^{(n+1)} &= G^{(n)} - \delta t \frac{\partial p^{(n+1)}}{\partial y} \end{aligned} \tag{5}$$

This manner of discretisation may be characterised as being *explicit* in the velocities and *implicit* in the pressure, i.e., the velocity field at time step t_{n+1} can be computed once the corresponding pressure is known.

We can then evaluate the continuity equation, Eq. (2), at time t_{n+1}. We substitute Eq. (5) in for the velocity field and obtain:

$$0 = \frac{\partial u^{(n+1)}}{\partial x} + \frac{\partial v^{(n+1)}}{\partial y} = \frac{\partial F^{(n)}}{\partial x} - \delta t \frac{\partial^2 p^{(n+1)}}{\partial x^2} + \frac{\partial G^{(n)}}{\partial y} - \delta t \frac{\partial^2 p^{(n+1)}}{\partial y^2} \tag{6}$$

We can rearrange this to become a *Poisson equation for the pressure* $p^{(n+1)}$ at time t_{n+1}:

$$\frac{\partial^2 p^{(n+1)}}{\partial x^2} + \frac{\partial^2 p^{(n+1)}}{\partial y^2} = \frac{1}{\delta t} \left(\frac{\partial F^{(n)}}{\partial x} + \frac{\partial F^{(n)}}{\partial y} \right) \tag{7}$$

The computation for the velocity field at the $n + 1^{\text{th}}$ time step therefore consists of the following parts:

1. Compute the tentative velocities, $F^{(n)}$, $G^{(n)}$, using the velocity fields $u^{(n)}$, $v^{(n)}$.
2. Compute the right hand side of the Poisson Equation, Eq. 7.
3. Solve the Poisson equation for the pressure $p^{(n+1)}$.
4. Compute the new velocity field ($u^{(n+1)}$, $v^{(n+1)}$) with the pressure value $p^{(n+1)}$.

The Problem

For this assignment you are provided with a simple CFD application, implemented according to the computational scheme by Griebel et al. [1]. It is set up to solve a problem often studied using CFD, known as a Kármán vortex street.

Named after the engineer and fluid dynamicist Theodore von Kármán, a vortex street is a repeating pattern of swirling vortices, caused by a process known as vortex shedding. A vortex street will only form at a certain range of flow velocities, specified by a range of Reynolds numbers, and they can be found when atmospheric air flows over obstacles such as buildings or even islands and isolated mountains. Figure 2 shows one such example.

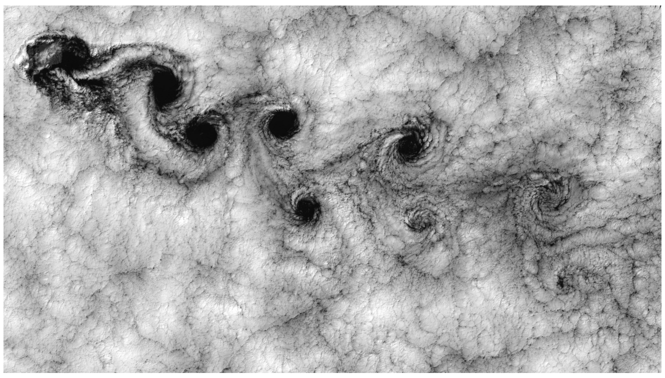

Figure 2: Kármán vortex street caused by wind flowing around the Juan Fernández Islands

These turbulent flows, caused by a Kármán street, have even been known to damage the structure of tall buildings, and thus it is important for engineers to account for the effects of vortex shedding when designing structures. One particularly notable example is the Ferrybridge Power Station C cooling towers. In 1965, during high winds, three of the cooling towers collapsed, due to the vibrations caused by Kármán turbulence.

The application provided, `vortex`, sets up a problem where there is a circular object to the left of the domain, and the initial fluid velocity is such that it is flowing from left to right. Figure 3 shows the starting state of the simulation, while Figure 4 shows the state after approximately 5 seconds of simulation time.

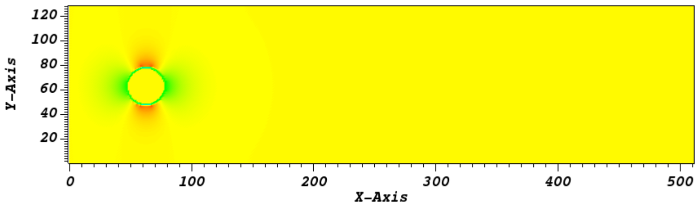

Figure 3: Starting state of the u velocity field in the `vortex` application.

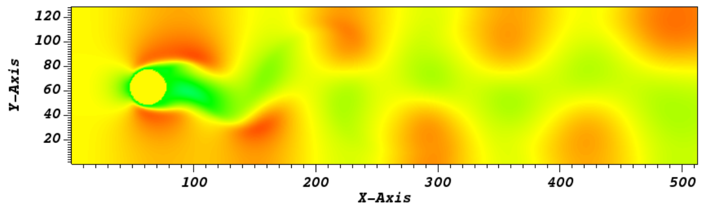

Figure 4: A visualisation of the u velocity field after 4.98s of simulation time.

The `vortex` application can be downloaded from the VLE, and built and run using:

```
$ make
$ ./vortex
```

At the end of execution, a VTK file will be written that can be loaded into a visualisation application such as VisIt [3]. The output file will contain information about the geometry of the problem, the values of the u and v velocity fields and the values of the pressure field p. The final simulation state will be based upon the default configuration options, but you can customise the parameters by changing the runtime arguments passed to the application.

```
$ ./vortex --help
```

The `vortex` application consists of 6 C files and 5 associated header files.

`args.c` contains functionality to handle the command line options, and can override some simulation parameters based on these options.

`boundary.c` contains logic to enforce the boundary conditions at the edge of the domain.

`data.c` contains the definitions for the shared storage and simulation parameters that are used in the application, and also a function to allocate addressable, contiguous 2D arrays.

`setup.c` has functionality to set up a simulation based on the inputs provided, and assigns and allocates the required variables and storage.

`vtk.c` handles the file input and output for the application.

`vortex.c` contains the `main` method, and the methods required to progress the simulation.

References

[1] Michael Griebel, Thomas Dornseifer, Tilman Neunhoeffer, “Numerical Simulation in Fluid Dynamics”, SIAM, 1998. https://people.math.sc.edu/Burkardt/cpp_src/nast2d/nast2d.html

[3] https://visit-dav.github.io/visit-website/

Assignment

The `vortex` application is currently a simple single-threaded implementation of the Navier-Stokes equations as described above. Your task is to produce **three** parallelisations of the `vortex` application and complete a report detailing the parallelisation process, the performance of your three parallelisations, and a comparative study of these parallelisations.

You are expected to produce parallelisations using **OpenMP**, **MPI** and **CUDA**.

Your parallelisations should produce the same result (within some small tolerance due to floating point arithmetic) as the original single-threaded application for equivalent starting parameters. You are expected to fully evaluate your applications using the HPC hardware that is available within the University (e.g. CUDA-capable workstations within the Department, the Viking cluster, etc.), and that is appropriate for each particular programming model.

Submission

Your submission should be a **.zip** file that will contain your three application "ports" and a completed report document (as a **.pdf**). The report document should be formatted according to the provided template, and word counts should be strictly observed. Failure to observe a word count will result in a mark of 0 for the section in question. Your report will cover:

- **Parallelisation Approach** [35%]
  The approach taken to parallelise the application for each programming model.
- **Validation** [15%]
  The validation process used to verify the correctness of your applications.
- **Performance Evaluation** [25%]
  Results demonstrating the performance and scaling behaviour of your applications.
- **Comparative Analysis** [25%]
  A comparative analysis of your three applications.

You may use figures when necessary, and text in figures and captions will not be considered in the word count. References may be listed at the end of your report.

End of examination paper