OCEN 689
Fall 2019

Final project, due Wed 11 Dec

In this project, you will experiment with the classical lid-driven cavity problem and can create lovely videos like this one at `http://www.youtube.com/watch?v=rBKcYbFpcrk`. The submission should take the form of a report that is at most five pages (seven if extra credit problems are attempted). This page count must include all plots and figures (but not source code).

Consider a rectangular body of fluid (see Figure 1) of length L and depth D filled with a viscous fluid. The bottom, left and right boundaries are fixed. The upper boundary of the domain, however, slides at a speed U, and thanks to the viscous nature of the fluid, drives the flow underneath. The behavior of the fluid is described by the Navier–Stokes equations, discussed below. The lid-driven cavity problem seems simple at first, but the unsteady flow is tough on numerical simulators. As you can see from the video, the flow may form eddies in several corners of the domain, and the pressure behaviour is extreme with a pressure singularity developing in the upper right corner.

Although this is a tough problem and often used as a benchmark for sophisticated solvers, you can still get nice results out just using the relatively simple methods you learned about in this course. An important part of applying numerical methods to PDEs is to understand when a numerical solution is behaving poorly and addressing these issues by reconsidering the numerical scheme.

This system is governed by the Navier–Stokes equations a set of partial differential equations that model the conservation of momentum in a viscous fluid. We will assume that the viscosity, μ, of the fluid is constant and that the flow is incompressible, that is, the density, ρ, of the fluid is constant. Then, the vector velocity, $\mathbf{v}$, and pressure, p, fields satisfy the

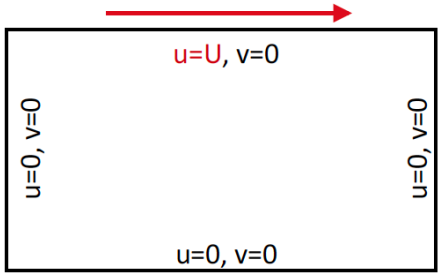

Figure 1: Computational domain for the driven cavity problem.

equations

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} = -\frac{\nabla p}{\rho} + \frac{\mu}{\rho} \nabla^2 \mathbf{v}, \tag{1}$$

$$\nabla \cdot \mathbf{v} = 0. \tag{2}$$

The first equation enforces conservation of momentum and second equation enforces the incompressibility of the fluid (i.e. a divergence-free velocity field).

Part I (10 marks)

The Navier–Stokes equations are a coupled system of PDEs. Before applying numerical methods to a set of equations, it is helpful to first understand the form and gain a physical intuition of the underlying PDEs. Explain what each term in the Navier–Stokes equations means physically.

It is also an important practice to non-dimensionalise equations to understand the underlying system parameters that influence the behaviour of the PDEs. When non-dimensionalising equations, dimensionless numbers will appear. In the case of the Navier–Stokes equations, the primary number of interest is the Reynolds (Re) number. Your first task, therefore, is to non-dimensionalise the Navier–Stokes equations. The general process of non-dimensionalising an equation is:

1. Identify characteristic parameters that represent the system. In this case, the length of the body, L, is a characteristic length scale, the final time, T, is a characteristic timescale, and the driving velocity, U, is a characteristic velocity scale.

2. Non-dimensionalise each variable with respect to these characteristic parameters and plug the non-dimensional variables into the equation.

3. Gather the characteristic parameters into non-dimensional numbers that provide a physical intuition of the system.

Non-dimensionalisation takes practice to gather the resulting parameters into useful non-dimensional quantities. In the Navier–Stokes equations, what do the non-dimensionalised numbers represent for the system?

Part II (14 marks)

To enforce the incompressibility constraint, we will approximate the incompressible Navier–Stokes equations numerically through the projection method. In this predictor-corrector-type approach, the pressure gradient term is initially neglected to obtain an intermediate approximate velocity prediction, $\mathbf{v}^*$. Then, pressure is computed to enforce incompressibility, and in the last step the initial velocity guess is corrected by applying the estimated pressure

field. The Navier–Stokes equations are hence decoupled into the three computational steps in each time iteration,

$$\begin{aligned}
\frac{\mathbf{v}^* - \mathbf{v}^\mathbf{n}}{\Delta t} &= -(\mathbf{v}^\mathbf{n} \cdot \nabla)\mathbf{v}^\mathbf{n} + \frac{\mu}{\rho}\nabla^\mathbf{2}\mathbf{v}^\mathbf{n}, \\
\nabla^2 p^{n+1} &= \frac{\rho}{\Delta t}\nabla \cdot \mathbf{v}^*, \\
\mathbf{v}^{\mathbf{n+1}} &= \mathbf{v}^* - \frac{\Delta t}{\rho}\nabla p^{n+1}, \text{ or } \frac{\mathbf{v}^{\mathbf{n+1}} - \mathbf{v}^*}{\Delta t} = -\frac{1}{\rho}\nabla p^{n+1}\,.
\end{aligned}$$

As a first step, study these equations carefully and explain in your own words how the incompressibility of the fluid is enforced. Note that the intermediate velocity is not divergence-free.

From concepts in class, we can discretise each of the terms with a finite difference method. In space, choose between a first or second order discretisation, whichever you believe is most appropriate, and give your motivation.

Write down the predictor-corrector equations using your discretisation method and group terms. The resulting set of equations should consist of five equations, two solving for the x and y-components of the velocity prediction, $\mathbf{v}^*$, one solving for the pressure, p, and two solving for the x and y-components of the velocity at the next timestep, $\mathbf{v}^{\mathbf{n+1}}$.

Write down the boundary and initial conditions for the system. What are the boundary conditions on pressure, p, and how can we obtain them from our initial and boundary conditions on velocity, $\mathbf{v}$? How will you enforce these boundary conditions in the discretised equations?

Part III (40 marks)

Implement the method that you outlined in Part II, using the forward Euler discretisation in time. Before you start coding, we suggest that you discuss your suggested approach with your colleagues. Key discussion points include: how you plan on handling boundary conditions, how you will store unknowns, and how you will construct and solve the resulting linear system. Your code should be general so that you can run it for any spatial grid size and time step.

For a spatial grid spacing of $h = 1/40$ in the x and y-direction, time step of $\Delta t = h^2/2U$ and driving velocity, $U = 10$, run the code for four Reynolds numbers, $\mathrm{Re} = [10, 100, 1000, 10000]$. Plot the velocity and pressure fields at $t = 10$ for each of these cases and comment on the behaviour of the system at different Reynold's numbers. Are these results physical or are there numerical artefacts?

If the simulations becomes unstable look for the first signs of instability and plot the solution fields at this timestep. What seems to be happening to the solution field?

Part IV (10 marks)

Comment on the limitations of your solver. Do you believe that your implementation is correct and stable?

Go back to the discretised equations you developed in Part II and determine how the behaviour of the solver can be inferred from the scheme we developed. How could we address this issue by redesigning our scheme?

Part V (10 marks)

As we observed in Part III and IV, a naive discretisation of a system of equations can lead to unphysical results. Specifically, for the incompressible Navier–Stokes equations the numerical schemes we implemented decouple adjacent grid points and result in instability. For this reason, the projection method is nearly always implemented on a staggered grid, where the velocity components are defined on the grid cell faces and the pressure in the grid cell centres (Figure 2). Explain why using a staggered grid makes sense, and speculate on what kind of complications may arise from using the staggered approach when implementing a numerical solver.

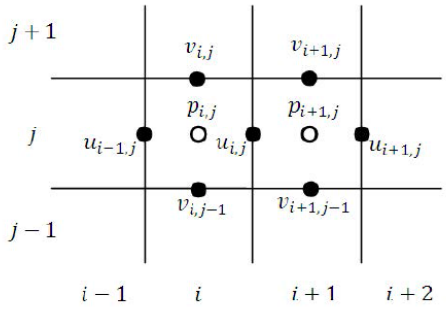

Figure 2: A staggered grid with pressures defined in the grid cell centres and velocities at the centres of the grid cell faces.

One tricky part in staggered grids is the implementation of boundary conditions. Let's focus on the bottom boundary. What boundary conditions would you implement on this boundary for velocities and pressure? How would you implement these boundary conditions? Repeat this question for the left boundary, and then the top boundary. Carefully explain how you will go about the boundary condition implementation.

Part VI (30 marks)

Implement the method that you outlined for a staggered grid, using the forward Euler discretisation in time. For a spatial grid spacing of $h = 1/40$ in the x and y-direction, time step of $\Delta t = h^2/2U$ and driving velocity, $U = 10$, run the code for four Reynolds numbers,

$\mathrm{Re} = [10, 100, 1000, 10000]$. Plot the velocity and pressure fields at $t = 10$ for each of these cases and comment on the behaviour of the system at different Reynold's numbers. Are these results physical or are there numerical artefacts?

Refine your spatial discretisation and create well-designed plots to illustrate your findings, and some nice looking flow plots of your solution that show beautiful eddies. One example of a beautiful and informative plot is overlaying velocity streamlines on top of the pressure field.

Part VII (10 marks)

What are the constraints on the time step size as function of the inputs (viscosity, lid speed, spatial grid step sizes)? Explain.

Perform convergence studies of your numerical solver for two of the three cases, both in time and space (but careful with that pressure singularity!) and analyse the accuracy of your scheme. What are the observed rates of convergence? Do these results give you confidence that your implementation is correct and stable?

We mentioned earlier that a pressure singularity will arise in the upper right corner. Explain why. How would this complicate grid convergence studies (hint: think about how the pressure solution may change as you refine the grid)?

Extra Credit (up to 76 marks)

Thus far, we have used a relatively straightforward discretisation. Take your code to the next level by implementing one of the following modifications:

1. Use backward (implicit) Euler as time-stepping method instead of forward Euler. Careful: the equations are nonlinear.

2. Use an upwind discretisation in space for the convective terms ($u\partial u/\partial x, v\partial u/\partial y$, etc.). Careful: the direction of the velocities changes throughout the domain.

3. Use a higher order explicit discretisation in time. Careful: you need to think a bit about how the various substeps in the projection method could be implemented.

Please save the working code you have *before* you start putting in any changes as we'd like you to compare the results of the new code with your old one for at least two of the four regimes discussed in the previous section. Comment on any difference you see.