COMPUTATIONAL PHYSICS

Lid-Driven Cavity Flow

Authors:
Adrian P.J. Sidhu

Supervisor:
Utku Gürel

University of Groningen
Faculty of Science & Engineering
November 2022

Preface

It should be noted that in this problem all the parameters (e.g. length, velocity) are dimensionless, except time, which has been assigned the dimension of seconds. In particular, the time discretisation Δt is given in seconds, and the full in-simulation time is calculated by the total number of time steps multiplied by the time discretisation. This was done to give a better feel of how the system evolves - in reality, the time units are equally as arbitrary as the other parameter units.

The term *project notebook* refers to the Jupyter notebook where the code has been written.

Contents

Chapter 1

Introduction

The aim of this project is to apply the finite differences method to a well-known problem in fluid dynamics: lid-driven cavity flow. Specifically, the vorticity form of the Navier-Stokes equations will be solved. This approach is often used to test simulation methods for incompressible viscous flows. The following chapter closely follows the structure of the introduction in the *Lid-Driven Cavity Flow Using Finite Differences* manual [1], but expands on the content in a few parts. Furthermore, throughout this report, Chapters 5 & 25 in Computational Physics by Landau et al.[2] are also made use of.

1.1 Lid-Driven Cavity Flow

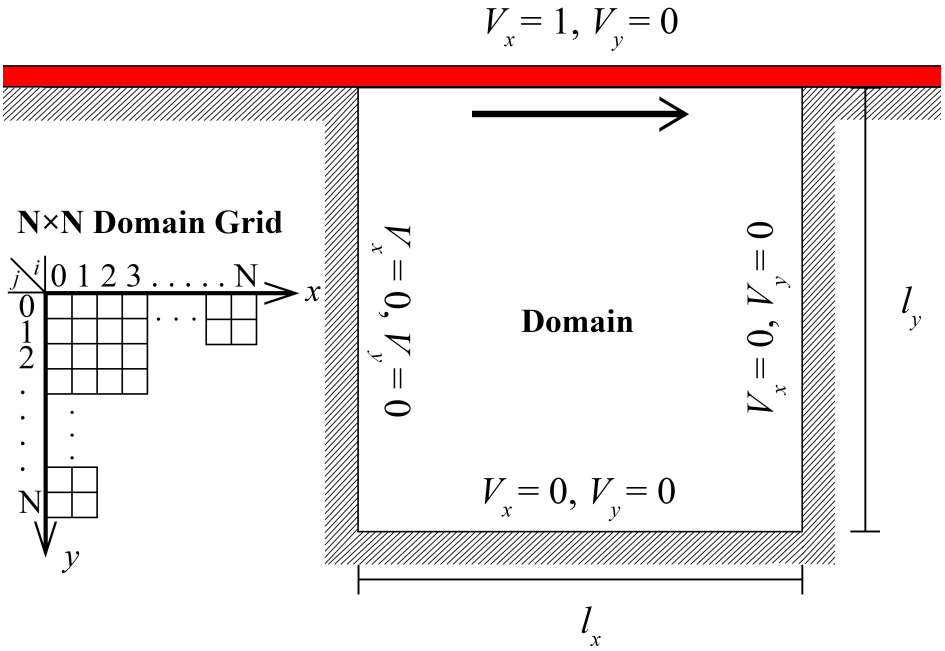


Figure 1.1: The computational domain in question. Here the sides l_x and l_y are both equal to 1. The red rectangle at the top of the domain represents the lid, which is driven in the x-direction with a velocity of $V_x = 1$.

The lid-driven cavity flow problem entails an incompressible flow in a two-dimensional rectangular domain. The domain can be seen in Fig. 1.1. The domain has sides $l_x = l_y = 1$. The top wall (the lid) of the domain is moved with a constant velocity $V_x = V_{\text{wall}} = 1$. It has no other velocity components, and hence $V_y = 0$. Due to the no-slip condition of fluid dynamics [3], this is also the fluid velocity in the top boundary of the domain. In Fig. 1.1, this is indicated by the arrow at the top of the domain. The velocities (both in the x- and y-directions) for all the other walls are zero.

1.1.1 Vorticity Form of the Navier-Stokes Equations

As mentioned previously, to tackle the lid-driven cavity flow problem, the vorticity form of the Navier-Stokes equation must be solved. The non-dimensionalized vorticity form of the Navier-Stokes equations can be written as

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -w \tag{1.1}$$

$$\frac{\partial w}{\partial t} = -\frac{\partial u}{\partial y}\frac{\partial w}{\partial x} + \frac{\partial u}{\partial x}\frac{\partial w}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right) \tag{1.2}$$

where u is the stream function and w is the vorticity. Here Re is the Reynolds number

$$Re = \frac{\rho v L}{\mu} \tag{1.3}$$

where ρ is the fluid density, v is the flow speed, L is the characteristic linear dimension (for the domain described above, $L = 1$), and μ is the dynamic viscosity of the fluid. The Reynolds number quantifies the relative effects of inertial forces versus viscous forces in a fluid, and can predict whether the flow will be laminar or turbulent [4]. In the simulation, the Reynolds number will be one of the input parameters.

1.1.2 Velocity and Pressure Fields

The velocity $\vec{V}$ is related to the stream function u by

$$V_x = \frac{\partial u}{\partial y} \qquad V_y = -\frac{\partial u}{\partial x} \tag{1.4}$$

and the vorticity by

$$w = \frac{\partial V_y}{\partial x} - \frac{\partial V_x}{\partial y}. \tag{1.5}$$

The pressure field in the interior (i.e. at non-boundary points) is given by the solution to the following differential equation

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} = 2\left[\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} + \left(\frac{\partial^2 u}{\partial x \partial y}\right)^2\right]. \tag{1.6}$$

1.1.3 Boundary Conditions

Velocity Boundary Conditions

The pressure field at the boundaries is subject to a set of boundary conditions. The velocity boundary conditions are a result of the no-slip condition and the no-penetration condition. The no-penetration condition, simply put, states that the normal component of the fluid velocity is zero at a boundary, such that $\vec{v} \cdot \hat{n} = 0$ [5]. In the no-penetration condition, the tangential component of the velocity is actually unrestricted, but since the no-slip condition is also imposed, the tangential fluid velocity equals the boundary velocity. These conditions can be written as

$$\begin{aligned}
&x = 0: && V_x = V_y = 0 \\
&x = l_x: && V_x = V_y = 0 \\
&y = l_y: && V_x = V_y = 0 \\
&y = l_y: && V_x = V_{\text{bot}} = \pm 1,\ V_y = 0 \text{ (moving bottom wall)} \\
&y = 0: && V_x = V_{\text{wall}} = 1,\ V_y = 0.
\end{aligned} \tag{1.7}$$

Note that the bottom wall ($y = l_y$) will eventually be set to move with a velocity V_{bot}, which will be set equal or opposite to V_{wall}. Also note, for the purpose of the computational domain, the top wall is set to $y = 0$, while the bottom wall is set to $y = l_y$. This ensures that at $j = 0$, $y = j \cdot h = 0$, as seen in Fig. 1.1 (for futher information on the spatial discretisation, see Section 2.1.1).

Stream Function & Vorticity Boundary Conditions

Eq. (1.7) only provides the boundary conditions of the velocity field - what is of interest to the algorithmic form of the simulation is the stream function u and vorticity w. Using Eq. (1.4) in combination with (1.7) gives

$$\frac{\partial u}{\partial y} = V_{\text{wall}} = V_x = 1 \qquad \frac{\partial u}{\partial x} = 0 \tag{1.8}$$

for the top boundary, while for the other three boundaries (in contact with stationary walls)

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial y} = 0. \tag{1.9}$$

The first boundary condition above states that u can be a function of y. For the moving top wall however, y is constant (and $V_y = 0$ when $y = 0$). Furthermore, the other boundary conditions imply that u is independent of x and y on all other (static wall) boundaries. This means that the stream function u *must be constant along the entire boundary.* Taking one step further, since it is only the *relative differences* that matter, the exact value of u is irrelevant. Thus, for simplicity, u can be set to zero along the entire boundary.

To find the boundary conditions for the vorticity field, Eq. (1.1) is used. The stream function is constant on the boundaries. For the top and bottom boundaries, u does not change with respect to x, and thus Eq. (1.1) reduces to

$$\frac{\partial^2 u}{\partial y^2} = -w_{\text{wall}} \tag{1.10}$$

and similarly, for the left and right boundaries (u does not change with respect to y) Eq. (1.1) reduces to

$$\frac{\partial^2 u}{\partial x^2} = -w_{\text{wall}}. \tag{1.11}$$

Pressure Field Boundary Conditions

The pressure field in the interior of the domain is calculated using (1.6). The boundary conditions for the pressure field can be derived using the Navier-Stokes equations (pressure-velocity form, not vorticity form). The boundary conditions end up being (taken from the manual [1]) for the top and bottom boundaries

$$\frac{\partial P}{\partial x} = -\frac{1}{Re}\frac{\partial w}{\partial y} \tag{1.12}$$

and the left and right boundaries

$$\frac{\partial P}{\partial y} = \frac{1}{Re}\frac{\partial w}{\partial x}. \tag{1.13}$$

1.2 Computational Approach

A Gauss-Seidel method paired with successive relaxation will be used for the code. Array slicing will be used for spatial sweeping and relaxation, while a for-loop will be used for the time iteration. The Gauss-Seidel method implies that each grid cell uses the updated values of the grid cells around it *during* the spatial sweep. A Gauss-Seidel method is used as it is easier to implement array slicing than in the Jacobi method. Successive relaxation is used to be able to analyse the residuals and determine whether the solution converges or not.

Chapter 2

Program Structure

The program consists of several different components and initialisation steps. These are outlined beginning on the next page. The structure of this chapter is as follows:

1 Discretisation of Partial Differential Equations;

2 Pseudocode and Explanation of Navier-Stokes Solver;

3 Errors & Stability of Algorithms

2.1 Discretisation of Partial Differential Equations

2.1.1 Discretisation of Navier-Stokes Equations

Discretising Equation (1.1)

First, Eq. (1.1) must be discretised using the central difference approximation. The central difference method can be applied by stepping forward a step, stepping backwards a step, and then taking this difference to form an approximation to the derivative. In Eq. (1.1), the partial derivatives are of the second order. Therefore, the approach is slightly more complex than simply taking the difference between a step forward and a step back. Starting by taking a Taylor expansion of the stream function u a step Δx forward and a step Δx back (ignoring for now the time dependence in u)

$$u(x + \Delta x,\, y) = u(x,\, y) + \Delta x \frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2}\frac{\partial^2 u}{\partial x^2} + \mathcal{O}((\Delta x)^3)$$

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

By only focusing on the order $(\Delta x)^2$ terms and below (since Δx is small, and since only the $(\Delta x)^2$ term is of interest, it is valid to neglect the higher order terms), and adding the two equations, results in

$$u(x + \Delta x,\, y) + u(x - \Delta x,\, y) = 2u(x,\, y) + (\Delta x)^2 \frac{\partial^2 u}{\partial x^2}$$

and rearranging this gives

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x + \Delta x,\, y) + u(x - \Delta x,\, y) - 2u(x,\, y)}{(\Delta x)^2}. \tag{2.1}$$

An identical approach for y yields the same result

$$\frac{\partial^2 u}{\partial y^2} = \frac{u(x,\, y + \Delta y) + u(x,\, y - \Delta y) - 2u(x,\, y)}{(\Delta y)^2}. \tag{2.2}$$

Plugging these back into (1.1) gives

$$-w = \frac{u(x + \Delta x,\, y) + u(x - \Delta x,\, y) - 2u(x,\, y)}{(\Delta x)^2} + \frac{u(x,\, y + \Delta y) + u(x,\, y - \Delta y) - 2u(x,\, y)}{(\Delta y)^2}. \tag{2.3}$$

Eq. (2.3) is a mathematical discretisation of Eq. (1.1). What is needed then, is a translation of Eq. (2.3) into a computationally compatible statement. To do this, the domain described in the introduction (see Fig. 1.1) is divided into an $N \times N$ grid of cells, with each cell having a centre described by $x_i = ih$ and $y_j = jh$. Here, h is the grid cell length. Thus, the total length of a wall is $l_x = l_y = Nh$. Furthermore, the problem is time-dependent, and this time dependence also has to be discretised. Using Δt for the time discretisation/ step, the time at t_n is given by $t_n = n\Delta t$. Thus the total time t is given by $t = N_t \Delta t$ where N_t is the total number of time steps. For (2.3), Δx and Δy can both be set to h. For brevity of notation, superscript will indicate the time step, and

subscript will indicate the spatial steps in x and y respectively.[1] For example, $w^n_{i,\,j}$ is the vorticity at time $t_n = n\Delta t$ and at $x = x_i \equiv ih$ and $y = y_j \equiv jh$. Applying this to (2.3) gives

$$-w^n_{i,\,j} = \frac{u^n_{i+1,\,j} + u^n_{i-1,\,j} + u^n_{i,\,j+1} + u^n_{i,\,j-1} - 4u^n_{i,j}}{h^2}.$$

Note in the subscript that $i+1$ translates to $x_{i+1} = x_i + h = (i+1)h$. Since it is the stream function u that needs to be updated in this equation, the above can be rearranged to give

$$u^n_{i,\,j} = \frac{1}{4}\big(h^2 w^n_{i,\,j} + u^n_{i+1,\,j} + u^n_{i-1,\,j} + u^n_{i,\,j+1} + u^n_{i,\,j-1}\big).$$

Discretising Equation (1.2)

Now, Eq. (1.2) must be discretized. This will be done using a central difference approximation for the spatial components, but a forward difference approximation for the time components. Starting on the left-hand side (LHS), the time derivative of w can be approximated using the forward difference approximation (ignoring the spatial dependency for the time being)

$$w(t + \Delta t) = w(t) + \Delta t \frac{\partial w}{\partial t} + \mathcal{O}\big((\Delta t)^2\big).$$

Ignoring the higher order terms, and rearranging for the partial derivative, gives

$$\frac{\partial w}{\partial t} = \frac{w(t + \Delta t) - w(t)}{\Delta t}.$$

In the subscript-superscript notation developed above, this can be written as

$$\frac{\partial w}{\partial t} = \frac{w^{n+1}_{i,\,j} - w^n_{i,\,j}}{\Delta t}. \tag{2.4}$$

For the right-hand side (RHS), things are slightly more complicated. To derive the first derivative of u with respect to x, start with the Taylor expansion of each spatial difference in u

$$u(x + \Delta x,\, y) = u(x,\, y) + \Delta x \frac{\partial u}{\partial x} + \mathcal{O}((\Delta x)^2)$$

$$u(x - \Delta x,\, y) = u(x,\, y) - \Delta x \frac{\partial u}{\partial x} + \mathcal{O}((\Delta x)^2).$$

Then, by subtracting the second expression from the first

$$u(x + \Delta x,\, y) - u(x - \Delta x,\, y) = 2\Delta x \frac{\partial u}{\partial x}$$

and finally

$$\frac{\partial u}{\partial x} = \frac{u(x + \Delta x,\, y) - u(x - \Delta x,\, y)}{2\Delta x}.$$

In terms of the subscript-superscript notation

$$\frac{\partial u}{\partial x} = \frac{u^n_{i+1,\,j} - u^n_{i-1,\,j}}{2h}. \tag{2.5}$$

[1] To clarify $i,\, j = 0, \ldots, N$ and $n = 0, \ldots, N_t$.

This expression holds for both x and y and both u and w. Then, going back to (a part of) the RHS of (1.2)

$$-\frac{\partial u}{\partial y}\frac{\partial w}{\partial x}+\frac{\partial u}{\partial x}\frac{\partial w}{\partial y}$$

$$=\frac{\left(u^n_{i+1,\,j}-u^n_{i-1,\,j}\right)\left(w^n_{i,\,j+1}-w^n_{i,\,j-1}\right)-\left(u^n_{i,\,j+1}-u^n_{i,\,j-1}\right)\left(w^n_{i+1,\,j}-w^n_{i-1,\,j}\right)}{4h^2}. \tag{2.6}$$

The second part of the RHS can be calculated using the same expression for the second derivative in the central difference approximation derived prior

$$\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}=\frac{w^n_{i+1,\,j}+w^n_{i-1,\,j}+w^n_{i,\,j+1}+w^n_{i,\,j-1}-4w^n_{i,j}}{h^2}.$$

Altogether, Eq. (1.2) can be written in the following discretised form

$$\frac{w^{n+1}_{i,\,j}-w^n_{i,\,j}}{\Delta t}=\frac{1}{Re}\left(\frac{w^n_{i+1,\,j}+w^n_{i-1,\,j}+w^n_{i,\,j+1}+w^n_{i,\,j-1}-4w^n_{i,j}}{h^2}\right)+$$

$$\frac{\left(u^n_{i+1,\,j}-u^n_{i-1,\,j}\right)\left(w^n_{i,\,j+1}-w^n_{i,\,j-1}\right)-\left(u^n_{i,\,j+1}-u^n_{i,\,j-1}\right)\left(w^n_{i+1,\,j}-w^n_{i-1,\,j}\right)}{4h^2}.$$

To summarise, the discretised, vorticity forms of the Navier-Stokes equations are

$$u^n_{i,\,j} = \frac{1}{4}\big(h^2 w^n_{i,\,j} + u^n_{i+1,\,j} + u^n_{i-1,\,j} + u^n_{i,\,j+1} + u^n_{i,\,j-1}\big) \tag{2.7}$$

and having rearranged for $w^{n+1}_{i,\,j}$

$$\begin{aligned} w^{n+1}_{i,\,j} =& w^n_{i,\,j} + \Delta t\Bigg[\bigg(\frac{w^n_{i+1,\,j} + w^n_{i-1,\,j} + w^n_{i,\,j+1} + w^n_{i,\,j-1} - 4w^n_{i,j}}{h^2 Re}\bigg) + \\ & \frac{\big(u^n_{i+1,\,j} - u^n_{i-1,\,j}\big)\big(w^n_{i,\,j+1} - w^n_{i,\,j-1}\big) - \big(u^n_{i,\,j+1} - u^n_{i,\,j-1}\big)\big(w^n_{i+1,\,j} - w^n_{i-1,\,j}\big)}{4h^2}\Bigg] \end{aligned} \tag{2.8}$$

2.1.2 Discretisation of the Velocities

This is quite straightforward. Starting with Eq. (1.4), and using a central difference approximation, the velocities are given by

$$\begin{aligned} V_x \equiv V^n_{x:\,i,\,j} &= \frac{u^n_{i,\,j+1} - u^n_{i,\,j-1}}{2h} \\ V_y \equiv V^n_{y:\,i,\,j} &= -\frac{u^n_{i+1,\,j} - u^n_{i-1,\,j}}{2h} \end{aligned} \tag{2.9}$$

2.1.3 Discretisation of the Pressure Field

It can be seen in the above section (Section 2.1.1) how the first and second derivative of a function can be discretised using a central difference approximation, and how the first derivative of a function can be discretised using a forward difference approximation. The pressure field equation, Eq. (1.6), consists of only second order spatial derivatives. Using the fact that the discretised second derivative of a function $f(x,\,y,\,t) \equiv f^n_{i,\,j}$ (using the notation explained in Section 2.1.1) in the central forward approximation is

$$\frac{\partial^2 f}{\partial x^2} = \frac{f(x + \Delta x,\,y,\,t) + f(x - \Delta x,\,y,\,t) - 2 \cdot f(x,\,y,\,t)}{(\Delta x)^2} = \frac{f^n_{i+1,\,j} + f^n_{i-1,\,j} - 2f^n_{i,\,j}}{h^2} \tag{2.10}$$

where the step forward $h \equiv \Delta x$ and $x + \Delta x = ih + h = h(i+1)$ and $t = n\Delta t$, but since the above is a spatial derivative, everything is evaluated at the same time step n. Similarly, the first derivative is

$$\frac{\partial f}{\partial x} = \frac{f(x + \Delta x,\,y,\,t) - f(x - \Delta x,\,y,\,t)}{2\Delta x} = \frac{f^n_{i+1,\,j} - f^n_{i-1,\,j}}{2h}. \tag{2.11}$$

In this project, the spatial derivatives are discretised with the central difference approximation. The time derivatives however, are discretised with the forward difference approximation. For the function $f(x,\,y,\,t)$ its first time derivative, discretised, is

$$\frac{\partial f}{\partial t} = \frac{f(x,\,y,\,t + \Delta t) - f(x,\,y,\,t)}{\Delta t} = \frac{f^{n+1}_{i,\,j} - f^n_{i,\,j}}{\Delta t}. \tag{2.12}$$

Using these equations, the pressure field equation can be easily discretised. Starting with the LHS of Eq. (1.6)

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} = \frac{P^n_{i+1,j} + P^n_{i-1,j} + P^n_{i,j+1} + P^n_{i,j-1} - 4P^n_{i,j}}{h^2}$$

and then the first part of the RHS

$$\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} = \left(\frac{u^n_{i+1,j} + u^n_{i-1,j} - 2u^n_{i,j}}{h^2}\right) \cdot \left(\frac{u^n_{i,j+1} + u^n_{i,j-1} - 2u^n_{i,j}}{h^2}\right)$$

and finally, the second part of the RHS (neglecting the square for now)

$$\begin{aligned}
\frac{\partial^2 u}{\partial x \partial y} &= \frac{\partial}{\partial x}\left(\frac{u^n_{i,j+1} - u^n_{i,j-1}}{2h}\right) \\
&= \frac{\partial}{\partial x}\left(\frac{u^n_{i,j+1}}{2h}\right) - \frac{\partial}{\partial x}\left(\frac{u^n_{i,j-1}}{2h}\right) \\
&= \frac{1}{2h}\left(\frac{u^n_{i+1,j+1} - u^n_{i-1,j+1}}{2h}\right) - \frac{1}{2h}\left(\frac{u^n_{i+1,j-1} - u^n_{i-1,j-1}}{2h}\right) \\
&= \frac{1}{4h^2}\Big(u^n_{i+1,j+1} + u^n_{i-1,j-1} - u^n_{i-1,j+1} - u^n_{i+1,j-1}\Big).
\end{aligned}$$

Squaring this gives

$$\left(\frac{\partial^2 u}{\partial x \partial y}\right)^2 = \frac{1}{16h^4}\Big(u^n_{i+1,j+1} + u^n_{i-1,j-1} - u^n_{i-1,j+1} - u^n_{i+1,j-1}\Big)^2.$$

Altogether then, Eq. (1.6)

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} = 2\left[\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} + \left(\frac{\partial^2 u}{\partial x \partial y}\right)^2\right]$$

can be discretised as

$$\begin{aligned}
P^n_{i,j} = -\frac{h^2}{2}\Bigg[&\left(\frac{u^n_{i+1,j} + u^n_{i-1,j} - 2u^n_{i,j}}{h^2}\right) \cdot \left(\frac{u^n_{i,j+1} + u^n_{i,j-1} - 2u^n_{i,j}}{h^2}\right) \\
&+ \frac{1}{16h^4}\Big(u^n_{i+1,j+1} + u^n_{i-1,j-1} - u^n_{i-1,j+1} - u^n_{i+1,j-1}\Big)^2\Bigg] \\
&+ \frac{1}{4}\Big(P^n_{i+1,j} + P^n_{i-1,j} + P^n_{i,j+1} + P^n_{i,j-1}\Big).
\end{aligned}$$

Expanding the first bracket gives then a final expression for $P^n_{i,j}$

$$\boxed{\begin{aligned}
P^n_{i,j} = &\frac{1}{4}\Big(P^n_{i+1,j} + P^n_{i-1,j} + P^n_{i,j+1} + P^n_{i,j-1}\Big) \\
&- \frac{1}{2h^2}\big(u^n_{i+1,j} + u^n_{i-1,j} - 2u^n_{i,j}\big) \cdot \big(u^n_{i,j+1} + u^n_{i,j-1} - 2u^n_{i,j}\big) \\
&- \frac{1}{32h^2}\Big(u^n_{i+1,j+1} + u^n_{i-1,j-1} - u^n_{i-1,j+1} - u^n_{i+1,j-1}\Big)^2.
\end{aligned}} \tag{2.13}$$

Seeing that it can be tedious to implement so many terms arising from the second order derivatives in u, it is useful to replace the stream function derivatives with $V_x = \frac{\partial u}{\partial y}$ and $V_y = -\frac{\partial u}{\partial x}$ (see Eq. (1.4). Doing so results in a similar form to Eq. (2.13)

$$\begin{aligned} P^n_{i,j} = \frac{1}{4}\Big(P^n_{i+1,j} + P^n_{i-1,j} + P^n_{i,j+1} + P^n_{i,j-1}\Big) \\ + \frac{1}{8}\Big[\big(V^n_{y:\,i+1,j} - V^n_{y:\,i-1,j}\big)\big(V^n_{x:\,i,j+1} - V^n_{x:\,i,j-1}\big) \\ - \big(V^n_{x:\,i+1,j} - V^n_{x:\,i-1,j}\big)^2\Big]. \end{aligned} \tag{2.14}$$

2.1.4 Discretisation & Implementation of the Boundary Conditions

Stream Function & Velocity Boundary Conditions

The boundary conditions for the fluid velocity are given in Eq. (1.7). The discretised versions of these are

$$\begin{array}{ll} \text{Top moving wall:} & V_{x:\,i,\,j=0} = V_{\text{wall}},\ V_{y:\,i,\,j=0} = 0 \\ \text{Bottom wall:} & V_{x:\,i,\,j=N} = V_{y:\,i,\,j=N} = 0 \\ \text{Bottom moving wall:} & V_{x:\,i,\,j=N} = V_{\text{bot}},\ V_{y:\,i,\,j=N} = 0 \\ \text{Left wall:} & V_{x:\,i=0,\,j} = V_{y:\,i=0,\,j} = 0 \\ \text{Right wall:} & V_{x:\,i=N,\,j} = V_{y:\,i=N,\,j} = 0. \end{array} \tag{2.15}$$

As mentioned in Section 1.1.3, the value of u can be set to 0 along the entire boundary. This implies that (recall that $i,\, j = 0, \ldots, N$)

$$\begin{array}{ll} \text{Top moving wall:} & u^n_{i,\,j=0} = 0 \\ \text{Bottom (moving) wall:} & u^n_{i,\,j=N} = 0 \\ \text{Left wall:} & u^n_{i=0,\,j} = 0 \\ \text{Right wall:} & u^n_{i=N,\,j} = 0. \end{array} \tag{2.16}$$

Vorticity Boundary Conditions

In general, the programme discretises space using the central difference approximation. However, at the boundaries, this is a bit trickier to implement (in the system used here, quite impossible actually - if $i = 0$ is a boundary, there is no $i - 1$). An easier, and more useful approach, is by using a Taylor expansion to acquire the forward difference approximation. Considering first the moving top wall, and Taylor expanding the stream function for the first row below the boundary (the boundary is at $(i,\, j = 0) \to (x,\, y = 0)$ for the top wall)

$$u(x,\, y + \Delta y,\, t) = u(x,\, y,\, t) + \Delta y \frac{\partial u}{\partial y} + \frac{(\Delta y)^2}{2}\frac{\partial^2 u}{\partial y^2} + \mathcal{O}\big((\Delta y)^3\big)$$

and using $\Delta y = h$ and $y = 0$

$$u(x,\, h,\, t) = u(x,\, 0,\, t) + h\frac{\partial}{\partial y}\big(u(x,\, 0,\, t)\big) + \frac{h^2}{2}\frac{\partial^2}{\partial y^2}\big(u(x,\, 0,\, t)\big) + \mathcal{O}\big(h^3\big).$$

Since $y = hj$ and $h \neq 0$, when $y = 0$ then $j = 0$, and when $y = h$, then $j = 1$. Rewriting this in subscript-superscript notation

$$u^n_{i,\,j=1} = u^n_{i,\,j=0} + h\frac{\partial u^n_{i,\,j=0}}{\partial y} + \frac{h^2}{2}\frac{\partial^2 u^n_{i,\,j=0}}{\partial y^2} + \mathcal{O}\big(h^3\big).$$

Eq. (1.10) states that ${}^{\partial^2 u}/_{\partial y^2} = -w_{\text{wall}}$, and (1.8) states that ${}^{\partial u}/_{\partial y} = V_{\text{wall}}$. Using these it is possible to rewrite the above Taylor expansion

$$u^n_{i,\,j=1} = u^n_{i,\,j=0} + hV_{\text{wall}} - \frac{h^2}{2}w^n_{i,\,j=0} + \mathcal{O}\big(h^3\big) \tag{2.17}$$

which gives for the vorticity

$$w^n_{i,\,j=0} = \frac{2}{h^2}\big(u^n_{i,\,j=0} - u^n_{i,\,j=1}\big) + \frac{2}{h}V_{\text{wall}} + \mathcal{O}(h). \tag{2.18}$$

Now the same must be done for all other boundaries. Starting with the bottom wall now (such that $(i,\, j = N)$) and Taylor expanding the first row off the bottom ($j = N - 1$)

$$u^n_{i,\,j=N-1} = u^n_{i,\,j=N} - h\frac{\partial u^n_{i,\,j=N}}{\partial y} + \frac{h^2}{2}\frac{\partial^2 u^n_{i,\,j=N}}{\partial y^2} + \mathcal{O}\big(h^3\big).$$

Replacing Eq. (1.9) (${}^{\partial u}/_{\partial y} = 0$) and Eq. (1.10) (${}^{\partial^2 u}/_{\partial y^2} = -w_{\text{wall}}$) for the derivatives gives

$$u^n_{i,\,j=N-1} = u^n_{i,\,j=N} - hV_{\text{bottom}} - \frac{h^2}{2}w^n_{i,\,j=N} + \mathcal{O}\big(h^3\big) \tag{2.19}$$

and the vorticity becomes

$$w^n_{i,\,j=N} = \frac{2}{h^2}\big(u^n_{i,\,j=N} - u^n_{i,\,j=N-1}\big) - \frac{2}{h}V_{\text{bottom}} + \mathcal{O}(h). \tag{2.20}$$

Similarly, the stream function just off the boundary and the vorticity for the left boundary is

$$u^n_{i=1,\,j} = u^n_{i=0,\,j} - \frac{h^2}{2}w^n_{i=0,\,j} + \mathcal{O}\big(h^3\big) \tag{2.21}$$

$$w^n_{i=0,\,j} = \frac{2}{h^2}\big(u^n_{i=0,\,j} - u^n_{i=1,\,j}\big) + \mathcal{O}(h) \tag{2.22}$$

and the right boundary

$$u^n_{i=N-1,\,j} = u^n_{i=N,\,j} - \frac{h^2}{2}w^n_{i=N,\,j} + \mathcal{O}\big(h^3\big) \tag{2.23}$$

$$w^n_{i=N,\,j} = \frac{2}{h^2}\big(u^n_{i=N,\,j} - u^n_{i=N-1,\,j}\big) + \mathcal{O}(h). \tag{2.24}$$

These boundary conditions can be further simplified by applying the boundary conditions of Eq. (2.16) - setting all stream functions to zero on the boundary gives the final boundary conditions for the vorticity

$$\begin{aligned}
&\text{Top moving wall:} && w^n_{i,\,j=0} = \frac{2}{h^2}\left(u^n_{i,\,j=0} - u^n_{i,\,j=1}\right) + \frac{2}{h}V_{\text{wall}} \\
&\text{Bottom wall:} && w^n_{i,\,j=N} = \frac{2}{h^2}\left(u^n_{i,\,j=N} - u^n_{i,\,j=N-1}\right) \\
&\text{Bottom moving wall:} && w^n_{i,\,j=N} = \frac{2}{h^2}\left(u^n_{i,\,j=N} - u^n_{i,\,j=N-1}\right) - \frac{2}{h}V_{\text{bottom}} \\
&\text{Left wall:} && w^n_{i=0,\,j} = \frac{2}{h^2}\left(u^n_{i=0,\,j} - u^n_{i=1,\,j}\right) \\
&\text{Right wall:} && w^n_{i=N,\,j} = \frac{2}{h^2}\left(u^n_{i=N,\,j} - u^n_{i=N-1,\,j}\right).
\end{aligned} \tag{2.25}$$

Pressure Boundary Conditions

Now the final boundary conditions must be discretized - these are for the pressure. For the top and bottom wall, the pressure along the y-direction is constant. The boundary condition to discretise then is Eq. (1.12)

$$\frac{\partial P}{\partial x} = -\frac{1}{Re}\frac{\partial w}{\partial y}.$$

Starting with a Taylor expansion[2]

$$P(x+h,\,y,\,t) = P(x,\,y,\,t) + h\frac{\partial}{\partial x}\big(P(x,\,y,\,t)\big) + \mathcal{O}(h^2)$$

and replacing the first order derivative of P with the boundary condition Eq. (1.12), along with replacing $y = 0$ (as this is at the boundary of the top wall) gives

$$P(x+h,\,0,\,t) = P(x,\,0,\,t) - \frac{h}{Re}\frac{\partial}{\partial y}\big(w(x,\,0,\,t)\big) + \mathcal{O}(h^2).$$

Taylor expanding $w(x,\,y+h,\,t)$ to discretise the first derivative, and setting $y = 0$ (top wall boundary condition)

$$w(x,\,h,\,t) = w(x,\,0,\,t) + h\frac{\partial}{\partial y}\big(w(x,\,0,\,t)\big) + \mathcal{O}(h^2).$$

Rearranging for the derivative, neglecting higher order terms, and using the subscript-superscript notation

$$\frac{\partial w^n_{i,\,j=0}}{\partial y} = \frac{w^n_{i,\,j=1} - w^n_{i,\,j=0}}{h}$$

and replacing back into the pressure equation gives the boundary equation for the top wall

$$P^n_{i+1,\,j=0} = P^n_{i,\,j=0} - \frac{1}{Re}\left(w^n_{i,\,j=1} - w^n_{i,\,j=0}\right).$$

[2]Here the $(x,\,y,\,t)$ notation has been used in favour of the subscript-superscript notation to keep track of the different x and y contributions. The final answers are given in subscript-superscript notation.

The bottom wall has a similar result. Starting with the Taylor expansion in subscript-superscript notation (neglecting higher order terms) and replacing the first order derivative with the boundary condition

$$P^n_{i+1,\,j=N} = P^n_{i,\,j=N} - \frac{h}{Re}\frac{\partial w^n_{i,\,j=N}}{\partial y}$$

and replacing the derivative in w with a backward difference approximation

$$w(x,\, l_y - h,\, t) = w(x,\, l_y,\, t) - h\frac{\partial}{\partial y}\big(w(x,\, l_y,\, t)\big) + \mathcal{O}(h^2).$$

$$\frac{\partial w^n_{i,\,j=N}}{\partial y} = \frac{w^n_{i,\,j=N} - w^n_{i,\,j=N-1}}{h}.$$

Plugging this back into the equation for the pressure gives the boundary equation for the bottom wall

$$P^n_{i+1,\,j=N} = P^n_{i,\,j=N} - \frac{1}{Re}\big(w^n_{i,\,j=N} - w^n_{i,\,j=N-1}\big).$$

The left boundary equation is given by (using Eq. (1.13) for the boundary condition)

$$P^n_{i=0,\,j+1} = P^n_{i=0,\,j} + \frac{1}{Re}\big(w^n_{i=1,\,j} - w^n_{i=0,\,j}\big)$$

and the right boundary equation is given by

$$P^n_{i=N,\,j+1} = P^n_{i=N,\,j} + \frac{1}{Re}\big(w^n_{i=N,\,j} - w^n_{i=N-1,\,j}\big).$$

Altogether then the pressure boundary equations are

$$\begin{array}{ll}
\text{Top moving wall:} & P^n_{i+1,\,j=0} = P^n_{i,\,j=0} - \dfrac{1}{Re}\big(w^n_{i,\,j=1} - w^n_{i,\,j=0}\big) \\
\text{Bottom (moving) wall:} & P^n_{i+1,\,j=N} = P^n_{i,\,j=N} - \dfrac{1}{Re}\big(w^n_{i,\,j=N} - w^n_{i,\,j=N-1}\big) \\
\text{Left wall:} & P^n_{i=0,\,j+1} = P^n_{i=0,\,j} + \dfrac{1}{Re}\big(w^n_{i=1,\,j} - w^n_{i=0,\,j}\big) \\
\text{Right wall:} & P^n_{i=N,\,j+1} = P^n_{i=N,\,j} + \dfrac{1}{Re}\big(w^n_{i=N,\,j} - w^n_{i=N-1,\,j}\big).
\end{array} \tag{2.26}$$

2.1.5 Summary of Discretised Equations and Boundary Conditions

Description of Domain

The domain is a square of length $l_x = l_y = l = 1$ divided into an $N \times N$ grid using Cartesian indexing. For the indices $i,\ j$, the index i specifies the column of the grid (x-axis) and the index j specifies the row of the grid (y-axis). Since the problem is also time-dependent, the total time is divided into N_t time discretisations Δt. The total time is thus given by $T = N_t \cdot \Delta t$. A subscript-superscript notation is used for brevity. For a function $f(x,\, y,\, t) \equiv f(ih,\, jh,\, n\Delta t)$, its subscript-superscript equivalent is given by $f^n_{i,\,j}$. The `numpy` library is used throughout the Navier-Stokes solver script (project notebook). An important note is that the `numpy` library in Python (and Python itself) uses matrix indexing. Thus, the indices given by $i,\ j$ in the domain are indexed as $[j,\ i]$ in Python (j being the row, i.e. y, and i being the column, i.e. x). The discretised equations and boundary conditions are given below.

Stream Function Equation

$$u^n_{i,\,j} = \frac{1}{4}\big(h^2 w^n_{i,\,j} + u^n_{i+1,\,j} + u^n_{i-1,\,j} + u^n_{i,\,j+1} + u^n_{i,\,j-1}\big).$$

Stream Function Boundary Conditions

Top moving wall:	$u^n_{i,\,j=0} = 0$
Bottom (moving) wall:	$u^n_{i,\,j=N} = 0$
Left wall:	$u^n_{i=0,\,j} = 0$
Right wall:	$u^n_{i=N,\,j} = 0.$

Vorticity

$$\begin{aligned} w^{n+1}_{i,\,j} =& w^n_{i,\,j} + \Delta t \Bigg[\left(\frac{w^n_{i+1,\,j} + w^n_{i-1,\,j} + w^n_{i,\,j+1} + w^n_{i,\,j-1} - 4w^n_{i,j}}{h^2 Re} \right) + \\ & \frac{\big(u^n_{i+1,\,j} - u^n_{i-1,\,j}\big)\big(w^n_{i,\,j+1} - w^n_{i,\,j-1}\big) - \big(u^n_{i,\,j+1} - u^n_{i,\,j-1}\big)\big(w^n_{i+1,\,j} - w^n_{i-1,\,j}\big)}{4h^2} \Bigg]. \end{aligned}$$

Vorticity Boundary Conditions

Top moving wall:	$w^n_{i,\,j=0} = \frac{2}{h^2}\big(u^n_{i,\,j=0} - u^n_{i,\,j=1}\big) + \frac{2}{h} V_{\text{wall}}$
Bottom wall:	$w^n_{i,\,j=N} = \frac{2}{h^2}\big(u^n_{i,\,j=N} - u^n_{i,\,j=N-1}\big)$
Bottom moving wall:	$w^n_{i,\,j=N} = \frac{2}{h^2}\big(u^n_{i,\,j=N} - u^n_{i,\,j=N-1}\big) - \frac{2}{h} V_{\text{bottom}}$
Left wall:	$w^n_{i=0,\,j} = \frac{2}{h^2}\big(u^n_{i=0,\,j} - u^n_{i=1,\,j}\big)$
Right wall:	$w^n_{i=N,\,j} = \frac{2}{h^2}\big(u^n_{i=N,\,j} - u^n_{i=N-1,\,j}\big).$

Velocities

$$V_x \equiv V^n_{x:\,i,\,j} = \frac{u^n_{i,\,j+1} - u^n_{i,\,j-1}}{2h}$$
$$V_y \equiv V^n_{y:\,i,\,j} = -\frac{u^n_{i+1,\,j} - u^n_{i-1,\,j}}{2h}.$$

Velocity Boundary Conditions

Top moving wall:	$V_{x:\,i,\,j=0} = V_{\text{wall}},\ V_{y:\,i,\,j=0} = 0$
Bottom wall:	$V_{x:\,i,\,j=N} = V_{y:\,i,\,j=N} = 0$
Bottom moving wall:	$V_{x:\,i,\,j=N} = V_{\text{bot}},\ V_{y:\,i,\,j=N} = 0$
Left wall:	$V_{x:\,i=0,\,j} = V_{y:\,i=0,\,j} = 0$
Right wall:	$V_{x:\,i=N,\,j} = V_{y:\,i=N,\,j} = 0.$

Pressure

$$\begin{aligned} P^n_{i,\,j} = \frac{1}{4}\Big(P^n_{i+1,\,j} + P^n_{i-1,\,j} + P^n_{i,\,j+1} + P^n_{i,\,j-1}\Big) \\ + \frac{1}{8}\Big[\big(V^n_{y:\,i+1,\,j} - V^n_{y:\,i-1,\,j}\big)\big(V^n_{x:\,i,\,j+1} - V^n_{x:\,i,\,j-1}\big) \\ - \big(V^n_{x:\,i+1,\,j} - V^n_{x:\,i-1,\,j}\big)^2\Big]. \end{aligned}$$

Pressure Boundary Conditions

Top moving wall:	$P^n_{i+1,\,j=0} = P^n_{i,\,j=0} - \frac{1}{Re}\big(w^n_{i,\,j=1} - w^n_{i,\,j=0}\big)$
Bottom (moving) wall:	$P^n_{i+1,\,j=N} = P^n_{i,\,j=N} - \frac{1}{Re}\big(w^n_{i,\,j=N} - w^n_{i,\,j=N-1}\big)$
Left wall:	$P^n_{i=0,\,j+1} = P^n_{i=0,\,j} + \frac{1}{Re}\big(w^n_{i=1,\,j} - w^n_{i=0,\,j}\big)$
Right wall:	$P^n_{i=N,\,j+1} = P^n_{i=N,\,j} + \frac{1}{Re}\big(w^n_{i=N,\,j} - w^n_{i=N-1,\,j}\big).$

2.2 Pseudocode and Explanation of Navier-Stokes Solver

2.2.1 Pseudocode

Below pseudocode of the Navier-Stokes solver seen in the project notebook is given. The pseudocode is written to ensure wide readability and generality with all coding languages, not just Python. There is a colour scheme in the pseudocode:

- BLUE: Blue, fully capitalised terms represent standard functions and operations included in many coding languages. Examples include DEFINE FUNCTION, COPY, IMPORT, RETURN, INITIALIZE;
- Lilac: Lilac terms represent user-defined functions;
- Red: Red terms represent user-defined operations. Examples include Set boundaries, Calculate interior, Define term;
- # Brown: Brown terms preceded by a hash # represent comments (i.e. not running parts of the script);
- Yellow: Yellow terms represent print statements;
- Green 0, 1, 2,...: Green terms represent numbers;
- Pink: Pink terms represent logical values (True/ False).

```
# A hash '#' indicates a comment (i.e. not a running part of the script)
# Code for the Navier-Stokes Solver

# Notation
# Re - Reynolds number
# u - stream function
# w - vorticity
# Vx - x-direction velocity
# Vy - y-direction velocity
# p - pressure
# h - spatial discretisation
# dt - time discretisation
# N - number of grid cells along one axis (NxN domain)
# Nt - total number of time steps
# l - domain length (set to 1)
# Vx_history - size Nt array (time history) of Vx at domain...
# centre (N/2, N/2)
# Vtop - velocity of top wall
# Vbot - velocity of bottom wall
# Vbot - velocity of bottom wall
# bottomVel - logical value. If False Vbot effectively 0,...
```

```
# if True Vbot=Vbot
# tolerance - the point at which the domain centre...
# acceleration can be considered steady-state (set to 1E-10)
# epsilon - the error tolerance of the residuals
# maxIters - maximum number of iterations performed during
# residual calculations if epsilon is not reached

IMPORT necessary libraries

# Build a residual calculator (see the section on
# errors for information on residuals)
DEFINE FUNCTION Residuals(f_new, f_old, omega=1)
    # In the successive over-relaxation, an omega factor is included.
    # This is initially set to 1
    # To accelerate convergence, this amplification factor omega
    # can be increased or decreased
    Calculate residual using residual definition
    # See section on errors for residual definition
    RETURN residual

# Build a stream function solver/ relaxer
DEFINE FUNCTION StreamFunctionSolver(u, w, h, epsilon, maxIters)
    INITIALIZE iteration number as 0
    # Begin successive (over-)relaxation
    WHILE conditions hold True
        COPY u as u_new
        # If a variable is not first copied in the function,
        # then recursions can occur, which results in both wrong
        # outputs and loops that crash due to very large numbers
        Set boundaries of u to 0
        Calculate interior of u using u_new, w, and h
        in discretised form of Navier-Stokes equation
        Calculate residuals using Residuals function
        ADD 1 to iteration
        END WHILE if residual less than epsilon or
        iteration more than maxIters
    RETURN u, residual
END FUNCTION


# Then build a vorticity update function
DEFINE FUNCTION VorticityUpdate(u, w, h, dt, Re, Vtop, Vbot, bottomVel)
    COPY w as w_new
    Set boundaries of w_new using discretised
```

```
    boundary conditions with u, h, Vtop, and Vbot
    Update interior of w_new using u, w, h, dt, and Re
    in discretised form of Navier-Stokes equation
    RETURN w_new
END FUNCTION

# Build a velocity update function
DEFINE FUNCTION VelocityUpdate(u, h, Vx, Vy, Vtop, Vbot, bottomVel)
    COPY Vx, Vy as Vxn, Vyn respectively
    Set boundaries of Vxn at top wall to Vtop, Vyn to  0
    Set boundaries of Vxn, Vyn at left and right walls to 0
    IF bottomVel is True
        Set boundaries of Vxn at bottom wall to Vbot, Vyn to 0
    ELSE
        Set boundaries of Vxn, Vyn at bottom wall to 0
    END IF
    Update interior of Vxn, Vyn using u and h in
    discretised form of velocity-stream function equations
    RETURN Vxn and Vyn
END FUNCTION

# Build a velocity probe function
DEFINE FUNCTION VelocityProbe(Vx_history, Nt, dt, tolerance)
    INITIALIZE Ax as size Nt array of zeros
    Update Ax using discretised time derivative of Vx_history
    Find index T0index where Ax reaches below tolerance
    IF T0index not found
        print The tolerance is too low or high
    END IF
    MULTIPLY T0index with dt to get time T0time at which tolerance
    is reached
    RETURN Ax, T0index, and T0time
END FUNCTION

# Build a pressure update function
DEFINE FUNCTION PressureUpdate(p, w, h, Vx, Vy)
    COPY p as p_new
    Set boundaries of p_new using discretised
    boundary conditions with p, w, and Re
    Update interior of p_new using p, Vx, Vy, and h
    RETURN p_new
END FUNCTION

########################################
```

```
110
111 # Now that all functions have been defined,
112 # an initialisation function should be made
113
114 DEFINE FUNCTION Initialisation(Re, N, l, Nt, dt)
115     DIVIDE l by N to acquire grid spacing h
116     MULTIPLY Nt by dt to acquire total time T
117     Define stability as dt / (Re*h*h)
118     IF stability is under 0.25
119         print Simulation is stable!
120     ELSE
121         print Simulation is unstable. Change parameters.
122     END IF
123     INITIALIZE u, w, Vx, Vy, and p as size NxN arrays of zeros
124     INITIALIZE Vx_history and u_residuals as size Nt array of zeros
125     RETURN u, w, Vx, Vy, p, Vx_history, u_residuals, h
126 END FUNCTION
127
128 # Finally, the Navier-Stokes solver
129 DEFINE FUNCTION NavierStokesSolver(u, w, Vx, Vy, Vx_history,
130 p, u_residuals, h, N, Nt, epsilon, maxIters)
131     Define domain_centre as integer of N/2
132     FOR time steps in total number of time steps Nt
133         Update stream function u and u_residuals  with
134         StreamFunctionSolver(u, w, h, epsilon, maxIters)
135         for time step t
136
137         Update velocities Vx, Vy
138         with VelocityUpdate(u, h, Vx, Vy, Vtop, Vbot, bottomVel)
139         for time step t
140
141         Update Vx_history by taking Vx at
142         (domain_centre, domain_centre)=(N/2, N/2) for time step t
143
144         COPY w as w1 for time step t+1
145
146         Update vorticity w1 with
147         VorticityUpdate(u, w, h, dt, Re, Vtop, Vbot, bottomVel)
148         for time step t+1
149
150         Set w to w1 (w=w1) to repeat process for next time step,
151         until time step Nt
152     END FOR
153     RETURN u, w, Vx, Vy, Vx_history, p, u_residuals
154 END FUNCTION
```

```
155
156 ########################################
157
158 # Now for an example
159 Set Re to 10
160
161 Set N to 200
162 Set l to 1
163
164 Set Vtop to 1
165 Set bottomVel to False
166 Set Vbot to 0
167
168 Set Nt to 100000
169 Set dt to 0.00005
170
171 Set epsilon to 0.00005
172 Set maxIters to 300
173
174 INITIALISE with Initialisation(Re, N, l, Nt, dt) to
175 acquire u, w, Vx, Vy, p, Vx_history, u_residual arrays and h
176
177 Solve the Navier-Stokes equation with
178 NavierStokesSolver(u, w, Vx, Vy, Vx_history, p, u_residuals, h, N, Nt)
179
180 PLOT u, w, p, and (Vx, Vy) as functions of space (x, y)
```

2.2.2 Approach to Assignments and Explanation of Navier-Stokes Solver

The general approach to the Navier-Stokes solver can be seen in the pseudocode above (Section 2.2.1). There are a few important points that need to be clarified however, as they are not mentioned in the pseudocode.

Iteration and Slicing

First, is the method of iteration. There are three variables to iterate each function over - two spatial (the x- and y- directions) variables and one time variable. For example, for the stream function u, the function `StreamFunctionSolver` (seen in the pseudocode) iterates over the spatial variables, and then in the `NavierStokesSolver`, the `StreamFunctionSolver` is called at each time step (hence the time iteration happens within the `NavierStokesSolver`). Since what is of interest for the stream function, vorticity, pressure, and velocities, is their spatial dependence at the *end* of the time range $T = N_t \cdot \Delta t$, they are initialised as $N \times N$ arrays - i.e. spatial arrays. Then, within the `NavierStokesSolver`, these $N \times N$ arrays are updated at each time step.

The question to ask then is, what is the best method of iteration? The spatial update of the $N \times N$ array requires individual access to each row and column - i.e. the value at `[i, j]` of function A (or at time step $n+1$) requires the value at `[i, j]` or others (for example, `[i+1, j-1]`) of function B (or at time step n). One method of accessing individual rows and columns is to use a nested loop (a for-loop within another for-loop, each with distinct loop iterators). However, nested loops are notoriously inefficient. Another method, that is slightly more efficient than a nested loop, is to use one loop iterating over all permutations of the `[i, j]` values. In Python, `itertools.product(x, y)` can be used to create 2D tuples of each index in `x` (x-axis) and each index in `y` (y-axis), and within the loop these indices can be called, similar to the nested loop. Despite being slightly faster than a nested for loop, this method is still quite inefficient. Both methods do not update the entire array at once, but iterate through it $N \times N$ times. For small N, this is fine, but for larger N this becomes more and more computationally expensive.

The ideal method to use, and what is utilised in the Navier-Stokes solver, is array slicing. The `numpy` library is used for all arrays (`numpy.array`) in the solver, and has a very useful method of accessing individual rows and columns *simultaneously*, instead of iteratively. Consider the array A and the definition of B

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

$$B[i,\ j] = A[i+1,\ j] + A[i-1,\ j].$$

Instead of iterating over each i, and then each j, B can be formed by 'overlapping' the $i+1$ matrix and the $i-1$ matrix, as seen below

$$A[i+1,\ j] + A[i-1,\ j] = \begin{pmatrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} = \begin{pmatrix} 5 & 6 & 7 & 8 \\ 10 & 12 & 14 & 16 \\ 18 & 20 & 22 & 24 \\ 9 & 10 & 11 & 12 \end{pmatrix} = B.$$

For example, using zero-based indexing (as in Python), and taking the random indices $[2, 1]$, $B[2, 1] = 20$ in the above matrix and

$$B[2, 1] = A[3, 1] + A[1, 1] = 14 + 6 = 20$$

as expected. Thus, `numpy` array slicing was used to iterate over all the spatial variables, and was tested to be up to ~ 40 times faster (at least) than iterating through `itertools.product()` tuples.[3]

There is a caveat with this method however, and that is the inaccessibility of boundaries when *both* indices are modified within the equation. The only example of where this happens is in the boundary conditions for the pressure, seen in Eq. (2.26). In these boundary conditions, both the indices i and j are modified (i changes in the vorticity w and j changes in the pressure P). This is a problem because in slicing, the array shape can change (whereby in looping, only floats are involved, not arrays), and an array of shape x cannot be broadcast (see `numpy` documentation) with an array of shape y unless $x = y$ or one of them is equal to 1. Thus, if both indices are sliced/ modified in an array, it might be too short to broadcast with another array in the equation. Thankfully, there is a solution for this - padding the array (using `numpy.pad`) with zeros. Adding a border of zeros along the boundary of the entire array ensures that the arrays remain the same shape after slicing, and also does not add anything to the actual equation, as calling any border value just results in the addition of 0. Looking at a new matrix C

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad C = \begin{pmatrix} 1 & 2 & 5 \\ 2 & 3 & 3 \\ 7 & 8 & 6 \end{pmatrix}$$

and one of its boundary conditions

$$C[0,\ j+1] = C[0,\ j] - A[1,\ j].$$

Looking at this in terms of slicing (whereby the second set of indices give the `numpy` slicing equivalent)

$$C[0,\ j+1] = C[0,\ 1:] = \begin{pmatrix} 2 & 5 \end{pmatrix}^{4}$$
$$C[0,\ j] = C[0,:] = \begin{pmatrix} 1 & 2 & 5 \end{pmatrix}$$
$$A[1,\ j] = A[1,:] = \begin{pmatrix} 4 & 5 & 6 \end{pmatrix}$$

it can be seen that the first array has a shape incompatible with the other two. Adding a boundary around both A and C then

$$A' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad C' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 5 & 0 \\ 0 & 2 & 3 & 3 & 0 \\ 0 & 7 & 8 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

[3] This time test was done by testing a few different, simple equations using the two methods, and arrays with high dimensions ($N \sim 1000$).

[4] This is technically incorrect, as by adding 1 to the slice [0:end] results in [1:end+1], *not* [1:], and end+1 does not exist in the array.

and changing the slicing of $[i,\ j] = [\,0:,\ 0:]$ to $[i,\ j] = [1:-1,\ 1:-1]$ (only accessing within the border) gives

$$C[0,\ j+1] \to C'[1,\ j+1] = C'[1,\ 2:] = \begin{pmatrix} 2 & 5 & 0 \end{pmatrix}$$
$$C[0,\ j] \to C'[1,\ j] = C'[1,\ 1:-1] = \begin{pmatrix} 1 & 2 & 5 \end{pmatrix}$$
$$A[1,\ j] \to A'[2,\ j] = A'[2,\ 1:-1] = \begin{pmatrix} 4 & 5 & 6 \end{pmatrix}.$$

It can be seen then that by adding the border, the first array now has a shape that can be broadcast with the other two without changing the numerical value of the equation, as only a zero is added. Once all necessary calculations are done, the padding is removed from the arrays.

Choice of Variables $(N, N_t, \ldots)$ and Initialisation

As seen in the pseudocode, the variables needed to be set for initialisation are given below. From these below values, all other values and functions follow via either the initialisation function, the Navier-Stokes solver, or both. The description of each variable and reasoning behind their respective numerical values are also given below.

- `Re`: the Reynolds number. This changes between 10, 100, and 200 based on the given assignments;
- `N`: the number of grid cells in one axis. This is kept at 200, which results in a `h` of 0.005. This is quite a precise value (see Section 2.3 for more on errors) while ensuring that the von Neumann stability condition can be fulfilled with reasonable values of `dt`;
- `l`: length of the square domain. Given as 1 in the project description;
- $\texttt{V}_{\texttt{top}}$: velocity of the top wall. Given as 1 in the project description;
- `bottomVel`: logical value to indicate whether bottom wall velocity should be considered. `bottomVel` acts as a control for the bottom wall velocity - even if the bottom wall velocity has been set to a non-zero value, if `bottomVel` is still set to `False` then $\texttt{V}_{\texttt{bot}}$ is effectively 0. This is to prevent mistakenly forgetting to set $\texttt{V}_{\texttt{bot}}$ back to 0;
- $\texttt{V}_{\texttt{bot}}$: velocity of the bottom wall. Given as ± 1 in the project description;
- `dt`: the time discretisation. This is set to 0.00005 throughout the entire notebook. The reason for this is twofold. First, when the Reynolds number is 10, to satisfy the von Neumann stability condition with a `h` of 0.005, `dt` has to be around 0.00005.[5] Second, due to the way the steady-state conditions are determined (see Section 2.2.2), it is best to keep `dt` constant. This comes at the price of longer computational times, as to achieve steady-state, larger values of $\texttt{N}_\texttt{t}$ are required if `dt` is kept constant;
- $\texttt{N}_\texttt{t}$: the total number of time steps. This varies to ensure the condition set for steady-state is met.

[5] The actual maximum value `dt` can be is 0.000063, but 0.00005 is chosen as it is more rounded and hence divides more easily.

Determination of Steady-State

Within the project description, it states that a probe is to be placed in the centre of the domain to measure the time-history of the horizontal velocity. This is straightforward - a size N_t array, `Vx_history` in the pseudocode and the project notebook, is initialised, and at each step in the time loop the value of V_x at the domain centre saved. To determine whether the system reaches a steady-state, the change in the horizontal velocity over time (i.e. horizontal acceleration) can be analysed.

To calculate the acceleration, a forward difference approximation is used, whereby the horizontal velocity is differentiated using

$$A_x = \frac{V_{x:\text{hist}}^{n+1} - V_{x:\text{hist}}^{n}}{\Delta t}. \tag{2.27}$$

Ideally, in a steady-state condition, the domain centre acceleration should go to zero. However, as this is a computational process with discretised derivatives, errors arise which prevent the acceleration from reaching zero. Thus, a tolerance is set which indicates when the system can be considered steady-state. If the acceleration falls below this tolerance at time step n, then the probe determines that steady-state conditions are reached at time step n. Throughout the project notebook, the tolerance is set to 1×10^{-6} which is as close to zero the tolerance can go while ensuring that N_t remains reasonable. It also corresponds to velocity changes on the order of $10^{-6} \cdot \Delta t = 10^{-6} \cdot (5 \cdot 10^{-5}) \sim 10^{-10}$. Thus, if the acceleration falls below 1×10^{-6} at time step n, the probe returns this time step.

Since errors arise from discretisation, to ensure consistency when comparing both different Reynolds numbers and different bottom wall velocities, it is best to keep the time discretisation `dt` constant. Otherwise, due to the presence of `dt` in Eq. (2.27) and other discretised equations in the Navier-Stokes solver, different `dt` values will result in the same tolerance being reached at different times for some fixed Reynolds number and fixed V_{bot}. Therefore, any conclusions made by comparing steady-state reaching times[6] if `dt` is changed between Reynolds numbers, would be slightly inaccurate. Note that this necessity of keeping `dt` fixed is only really necessary for this method (the tolerance method) of steady-state determination. The only caveat of this method is that by necessitating `dt` remain constant, N_t needs to be increased to compensate for longer steady-state reaching times.

Note on Updating Runtime Status

The Navier-Stokes solver can take time to compute so it is therefore useful to update the user on its current status. Implementing this is quite straightforward, and the update code used for the Navier-Stokes solver in the project notebook is given below. Note that the `time` module is required.

```
1 def NS_solver(..., Nt, outputNo):
2     # outputNo is the number of updates to be printed set by the user
3     ...
4     # Define the output frequency (output per n time steps)
```

[6] The time it takes for the system to reach steady-state conditions.

```
    outputFreq = Nt/outputNo

    # Set a starting time right before the for loop
    start = time.time()

    # Loop over time in the NS solver
    for tt in range(Nt):
        ...
        ...
        # Print first update (tt=0) and print every
        # time tt is a multiple of outputFreq
        # (results in outputNo print statements)
        if tt == 0 or tt % outputFreq == 0:
            # Time now for 'this' loop (constantly updated)
            t_now = time.time()
            # Calculate time left using difference between
            # the start and now divided by the number of steps
            # passed (this gives time per loop) then multiply
            # by number of steps left
            # (tt+1 to avoid tt=0 singularity)
            time_left = ((t_now-start)/(tt+1)) * (Nt-tt)
            # Update time left with the local time as well
            # for clarity
            # (e.g. what time did the simulation start, finish etc.)
            t_now_24 = time.strftime("%H:%M:%S", time.localtime())

            # Print everything
            print('Local time: {0}\n'.format(t_now_24) + \
                  'Time left: {0} s'.format(round(time_left,2)) + \
                  '\nIteration: {0}/{1}\n'.format(tt,Nt))

    # Print time elapsed for entire simulation
    end = time.time()
    print('\n\nTime taken: {0} s'.format(round(end-start,2)))
```

Note on Indexing

As mentioned earlier in the chapter, the domain given in the project description uses Cartesian indexing, whereas `numpy` arrays use matrix indexing. Thus, the indices given by i, j in the domain are indexed as $[j, i]$ in Python (j being the row, i.e. the y-position, and i being the column, i.e. the x-position).

Note on Results Analysis

The Results & Discussion (Chapter 3) is structured by increasing Reynolds number (and subsequently, different bottom wall velocities). As a reminder, the Reynolds number is given by Eq. (1.3)

$$Re = \frac{\rho v L}{\mu} = \frac{v}{\nu}$$

where ρ is the fluid density, v is the flow speed, L is the characteristic linear dimension (for the project domain, $L = 1$), μ is the dynamic viscosity of the fluid, and ν is the kinematic viscosity. The last inequality uses the fact that $L = 1$ for the domain. The dynamic viscosity is a measure of a fluid's resistance to deformation (stress and strain). The higher the dynamic viscosity, the harder it is to deform the fluid. For example, honey is more viscous than water, and is evidently harder to deform (compare pouring honey versus water out of a beaker). The Reynolds number, as mentioned before, is a measure of the effect of inertial forces versus viscous forces in a fluid. The inertial forces are characterised by vL and the viscous forces are characterised by $\rho/\mu = 1/\nu$. The exact values will not be explored in this project, but the effect of the Reynolds number on the stream function, vorticity, velocity, and pressure will be examined.

It is important to note that the Reynolds number is inherently linked to the type of flow a fluid has. If the Reynolds number is low, then the viscous forces dominate the flow, and the flow is consequently laminar. Higher viscosities result in laminar flow as the fluid's response to deformation is low and its resistance to inertia is high; a lower deformation response means less fluid deformation, and hence more constant motion. At higher Reynolds numbers, turbulent flow occurs. This is because the inertial forces dominate the viscous forces, and fluid response to deformation is high/ inertial resistance is low. Turbulent flow can then result in chaotic processes, such as eddies. An analaysis of the flow type will be carried out for each respective case in the results.

2.3 Errors & Stability of Algorithms

2.3.1 Residual Calculations: Successive (Over-)Relaxation

For any function $f(x,\ y,\ t)$ that is calculated via iterations, and where at each iteration $f(x,\ y,\ t)$ is relaxed, a method exists to determine whether the simulation converges or not. Suppose that f, theoretically, equals to X, such that $f(x,\ y,\ t) = X$. When sweeping through the spatial domain to update the values of f, due to computational and iterative errors (see next sections) a residue r remains between f and its theoretical value, $r = X - f$. Now, in this project, this ideal theoretical value X is not known, but it can be approached by sweeping the spatial grid successively, until the residue is reduced to a tolerance ϵ set by the user. In terms of the subscript-superscript notation

$$f_{i,j}^{n:\text{new}} = f_{i,j}^{n:\text{old}} + \omega r_{i,j}^{n}$$

$$r_{i,j}^{n} = \frac{1}{\omega}\left(f_{i,j}^{n:\text{new}} - f_{i,j}^{n:\text{old}}\right) \tag{2.28}$$

where ω is an amplifying factor. Note that this all happens within the same time step. The successive relaxation of the function in the spatial domain occurs within one time step, at every time step. The higher the number of relaxations, the lower the residue becomes, and hence the more precise the simulation becomes. To increase the speed of convergence, ω can be set to something other than 1 ($\omega < 1$ is under-relaxation, while $\omega > 1$ is over-relaxation). Ideal values of ω are $1 \leq \omega \leq 2$, and ω is set to 1 throughout the project notebook. The tolerance ϵ is set to 10^{-5} throughout the project notebook, which indicates a well-converging simulation.

The only function that the residual calculations are applied to is the stream function. The reason being that the two base functions (i.e. functions that every other function depends on) are the stream function and the vorticity. The vorticity however, is not relaxed in the spatial domain, but instead updated at each time step. This means that the vorticity cannot be successively relaxed. Thus only the stream function is successively relaxed. Technically, while all spatially relaxing functions (velocities, pressure) should be successively relaxed, to ensure reasonable processing times, this was avoided. Furthermore, since the stream function appears in all other functions, the successive relaxation of the stream function permeates through to all these functions. Due to this, the stream function residuals are then a good representation of the convergence of the whole simulation.

2.3.2 Determining Error Magnitudes of Order

The Navier-Stokes equations are partial differential equations. As computational processes work via discrete, numerical methods, the partial differential equations must be discretised, as seen in the former sections of this chapter (see Section 2.1). There are many numerical methods used to solve partial differential equations - the method used in this project is the finite difference method, and in particular, both the central and forward difference methods (which are types of finite difference methods). These methods inevitably result in errors which should be considered during implementation and analysis.

The spatial interiors are all updated via a central difference method, whereas the time is updated with a forward difference method. The spatial boundaries are updated via a backward or a forward

difference method, depending on the type of boundary. The reason a central difference method is not used for the boundaries is because for the starting boundary, there is no `i-1` cell, whereas for the final boundary, there is no `i+1` cell, so central differences cannot be used.

Looking first at the forward difference method used in the time discretisation, and expanding some function $f(x,\ y,\ t)$ in t around Δt (x and y have been omitted for brevity)

$$f(t+\Delta t) = f(t) + \Delta t \frac{\partial f(t)}{\partial t} + \mathcal{O}\big((\Delta t)^2\big)$$

$$\frac{\partial f(t)}{\partial t} = \frac{f(t+\Delta t) - f(t)}{\Delta t} + \mathcal{O}(\Delta t).$$

It is easy to see that using this method to discretise the first-order derivative of f in t results in an error of magnitude Δt. Now, whether or not this is a significant error depends on the choice of Δt and how this compares to the magnitude of $f(t)$ and its derivative. Throughout the project notebook, a $\Delta t \equiv$`dt` of 0.00005 is chosen (an order of magnitude of 10^{-4} or 10^{-5}, depending on convention). This then is the corresponding error for the vorticity and the acceleration, which are the only functions which contain time discretisation. This value of `dt` also ensures that the von Neumann stability condition is fulfilled for small Reynolds numbers - see next subsection for more on this.

The spatial discretisation `h` used in the project notebook is given by `h` $= 1/$`N` $= 1/200 = 0.005$. For the boundaries, this results in an associated error of 0.005 (the boundaries also utilise forward/ backward difference methods). The only boundaries which utilise this discretisation are the vorticity and pressure boundaries. The spatial interiors use a central difference method, which results in (y and t omitted for brevity)

$$f(x+h) = f(x) + h\frac{\partial f(x)}{\partial x} + \frac{h^2}{2}\frac{\partial^2 f(x)}{\partial x^2} + \mathcal{O}(h^3)$$

$$f(x-h) = f(x) - h\frac{\partial f(x)}{\partial x} + \frac{h^2}{2}\frac{\partial^2 f(x)}{\partial x^2} + \mathcal{O}(h^3)$$

$$\frac{\partial f(x)}{\partial x} = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2).$$

The error of the central difference method is the square of the discretisation - clearly a more precise method than the forward/ backward difference which results solely in an error equal to the discretisation. Again, like the boundaries, 0.005 (`h`) is used for the discretisation, resulting in an error on the order of $\texttt{h}^2 = 2.5 \cdot 10^{-5}$. A summary of these errors can be seen in Table 2.1.
It is expected that the largest inconsistencies will arise from the calculation of the spatial boundaries, seeing as the error associated with the boundaries (`h`=0.005) is several orders of magnitude larger than the errors associated with the time and spatial interior. Due to the chosen value of `h`, the spatial boundaries are only precise up to the second decimal place; at the third decimal (order 10^{-3}) the error starts to become significant, and after the third, the error dominates. Furthermore, while the error can be improved by reducing `h` this is not practical, as to ensure the von Neumann stability criterion is satisfied `dt` would have to be decreased, which would then mean that for longer times (i.e. to reach a steady-state) N_t would have to be increased to unreasonable values (resulting

Table 2.1: Table of errors arising from different finite difference methods used in discretising the Navier-Stokes equations.

Part of Simulation	**Differentiation Variable**	**Finite Difference Method**	**Associated Error**	**Numerical Value**
Time	t	Forward	$\Delta t \equiv$ `dt`	0.00005
Spatial boundary	x	Forward & Backward	`h`	0.005
Spatial interior	x	Central	$\texttt{h}^2$	0.000025

in processing times of several hours or days). How significant a role `h` being 0.005 plays in the simulation can only be understood after seeing the magnitude of the boundaries. If on average the boundaries (of a general function f) are on the order of 10^{-2} or above, then this does not pose too much of a problem, as the third decimal hardly contributes to the value. However, if the boundaries (on average) have an order lower than 10^{-2}, a proper analysis must be conducted to understand the effects of this relatively high error. A similar analysis should be conducted with `dt` dependent functions (vorticity and acceleration) and the spatial interiors, to ensure that the errors are small relative to the values of their associated functions.

Close attention should be paid particularly to the vorticity, as it has errors arising from all three of these methods, unlike all other functions which only contain one or two sources of error.[7] The vorticity is the only function which is updated in time with an explicit time derivative, and subsequently has its borders and interiors updated with the respective difference method.

The three sources of error outlined in Table 2.1 are the primary error sources. Since they arise from discretisation, these errors are called *truncation errors*.[8] The other major errors present in computational and numerical methods are the *round-off error* and *representation error*. Round-off errors arise due to the fact that computers have a limited storage ability - in other words, computers cannot store an infinite string of digits. While harmless for one or two calculations, these rounding errors can accumulate and become significant. Python has a precision of 16 digits, and has to approximate floating point numbers to these 16 digits. Furthermore, as it cannot represent all floats exactly due to the way Python interprets floats (using binary/ base 2 fractions), this also adds to the imprecision - this is the representation error. For example, running the following code

```
A=0
N = 50000000
for i in range(N):
    A += 0.555
```

results in an `A` of 27749999.987904016, which is 0.01209... off from the actual value of 27750000. This is the error arising from the use of binary fractions in representing floats, the representation

[7] The acceleration function has one source of error arising from time discretisation, and the stream function and velocities have one source of error arising from the spatial interior discretisation. The pressure has two sources, one arising from the boundary calculations and one from the interior calculations.

[8] In general, a truncation error is an error that arises due to approximating a physical and/ or mathematical process.

error, as 0.555 is not a dyadic rational. If the round-off error is also included, it is evident that these errors can be substantial. For example,

```
B=0
N = 50000000
a = 1/3 # Here 1/3 will be rounded off to the 16 digit precision
for i in range(N):
    B += a
for i in range(N):
    B -= a
```

results in a `B` of $4.964\ldots \cdot 10^{-10}$, when in reality it should be zero. While the round-off error here is seemingly not as significant as the representation error, it can still accumulate and cause inaccuracies, especially when paired with the representation error. These two must also be considered when analysing the final results.

2.3.3 Von Neumann Stability Analysis

To ensure that a simulation utilising a finite difference method is stable, a von Neumann stability analysis must be carried out. The von Neumann stability analysis is done first through a decomposition of the errors via a Fourier series. Thankfully, the von Neumann stability criterion has been given in the project description, and therefore no analysis via Fourier series needs to be conducted. The von Neumann stability criterion for the lid-driven cavity flow problem is

$$\frac{\Delta t}{Re \cdot h^2} \leq \frac{1}{4}. \tag{2.29}$$

This condition is implemented in the `Initialisation` function, whereby if the condition in Eq. (2.29) is not satisfied, the user will be informed that the simulation conditions are unstable. Furthermore, the code provides recommended values of Re, Δt, and N that the user can use to satisfy the stability criterion. The reason N is given as a recommended value as opposed to h is because N defines h in the project notebook. The recommended values are calculated using the following inequalities

$$\begin{aligned} Re &\geq \frac{4\Delta t}{h^2} \\ \Delta t &\leq \frac{Re \cdot h^2}{4} \\ N &\leq \sqrt{\frac{Re \cdot l^2}{4\Delta t}} \longrightarrow h = \frac{l}{N} \,:\, h \geq \sqrt{\frac{4\Delta t}{Re}}. \end{aligned}$$

Here, l is the square domain length, as defined previously in the report.

Chapter 3

Results & Discussion

In this chapter, the results of the simulation are presented, analysed and discussed. The chapter is organised by increasing Reynolds number, and then by bottom wall velocity. The sections are:

1 Reynolds Number: 10;

2 Reynolds Number: 100;

3 Reynolds Number: 200;

wherein each section the subsections are given by:

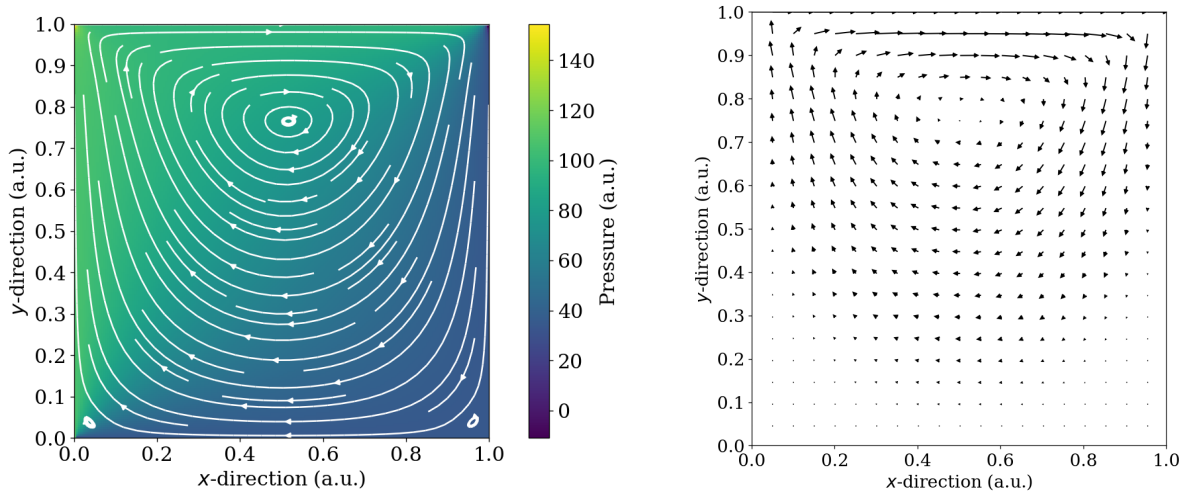
1 $V_{\text{bot}} = 0$;

2 $V_{\text{bot}} = 1$;

3 $V_{\text{bot}} = -1$.

3.1 Reynolds Number: 10

3.1.1 Bottom Wall Velocity $V_{\mathbf{bot}} = 0$



(a) Velocity stream & pressure plot.

(b) Velocity quiver plot.

Figure 3.1: Velocity and pressure plots for Reynolds number 10 and $V_{\text{bot}} = 0$ at steady-state (in-simulation time of 5 seconds).

A Reynolds number of 10 implies a fluid in which the viscous forces dominate (high kinematic viscosity) the inertial forces. An example of this would be honey in a small cavity. Fig. 3.1 shows the associated velocity and pressure plots for the system in a steady-state (in-simulation time of 5 seconds). As expected, the flow (as seen from the velocity stream and quiver plots) is mostly laminar. The flow vortex is almost centered about the x-axis (a Reynolds number approaching 0 would have an increasingly smaller vortex centered exactly in the centre of the x-axis), and there is no major chaotic behaviour. The vortex itself can be seen as turbulent and chaotic behaviour, but its high symmetry indicates a significant laminar contribution. There are very small eddies which can be seen in the bottom two corners of Fig. 3.1a, which arise from the fact that the Reynolds number is not below 1 - a value of 10 implies that the inertial forces still play an important role in the flow of the fluid.

It can also be seen that there is a flow symmetry about $x = 0.5$ (or close to 0.5), which arises from the laminar flow. This makes sense, as due to the absence of major turbulence and chaos, the flow of particles in the fluid are following smooth layered paths which are mirrored along an axis perpendicular to the driving direction. In general, the motion of the fluid is uniform throughout the cavity - despite the creation of a vortex, the fluid particles still follow neat paths around it. In the quiver plot (Fig. 3.1b) the magnitude of the velocities can be seen. As expected, the highest velocities occur at the boundary with the driving lid (no-slip condition implies that the fluid at the boundary moves with the same velocity as the boundary itself), and decrease with decreasing y to near-zero values. The velocities in the vortex centre are also near-zero, as expected.

Interestingly, the quiver plot shows that at the upper corners (corners where the driving lid contacts the cavity), the flow moves in opposite directions. On the left, at the beginning of the driving lid contact, the flow pushes fluid up into the corner, causing a pressure increase which can be seen in Fig. 3.1. While on the right, at the end of the driving lid contact, the flow pushes fluid away from the corner, causing a pressure decrease. This is not a feature of the Reynolds number, but of the dynamics of the system. Since the driving lid causes the fluid in direct contact with it to travel at the same velocity, this fluid interacts with the right wall and subsequently reflects back and away from it. Furthermore, the vortex created in the centre causes a negative pressure, which pulls fluid away from the right wall and towards the left. Then, at the left wall, this fluid is pushed up into the corner due to the vortex current. The high pressure in the left corner is not experienced in the rest of the top boundary, as the driving lid dissipates the flow and pressure along it length.

In general, the velocities as seen from the stream and quiver plots are as expected. One driving lid at the top, driving a low Reynolds number fluid, causes a vortex to form symmetrically about the centre of the driving lid axis, with the vortex centre closer to the lid. The velocities at the bottom (low y-values) of the cavity are all near-zero, which is also as expected. Unfortunately, the pressure is not as expected. After consulting a range of literature, and analysing the velocity-pressure form of the Navier-Stokes equations, a pressure gradient equally as symmetric as the flow velocity was expected. Besides a high pressure point in the upper-left corner, and a low pressure point in the upper right corner, the pressure in the left wall should be more or less constant in y, and decrease in layers with increasing x. There should also be a clear symmetry about the symmetry axis of the vortex. In Fig. 3.1 this is clearly not the case, as the gradient is diagonal to the driving wall rather than perpendicular.

There might be a few reasons for this, but the most likely culprits are incorrect implementation of the boundary conditions and/ or lack of successive relaxation in the pressure. In order to troubleshoot the pressure, the boundary conditions were removed. The result was as expected for a constant boundary pressure - the centre of the vortex was an area of extreme low pressure, with the pressure increasing radially from the vortex until reaching a maximum at the boundaries. The boundary conditions should help to dissipate the pressure appropriately around the cavity, and the interiors subsequently calculated from the boundary conditions. Getting expected/ accurate results in the absence of boundary conditions implies that the boundary conditions could be the source of the inaccuracy. A successive relaxation was also applied to the pressure function (and soon after removed, as it slowed the processing times), which seemed to improve the diagonal gradient, causing it to tilt more towards the vortex symmetry axis. While not as big of an effect as removing the boundary conditions, implementing successive relaxation did seem to help slightly. To improve on this, it is recommended to implement stricter successive relaxation requirements on the pressure, and/ or try a new method in applying the boundary conditions.

The flow velocities are defined via the stream function; the spatial derivative of the stream function in $x(y)$ corresponds to the negative(positive) of velocity $V_{y(x)}$. Thus, it only makes sense that the stream function has a structure which closely resembles that of the flow velocities, as seen in Fig. 3.2. There are some differences however. Firstly, the vortex centre where the magnitude of the velocities go to zero, is also the area with the highest stream function values. Despite it being a maximum area for the stream function, it is also quite constant in spatial changes, which reflects

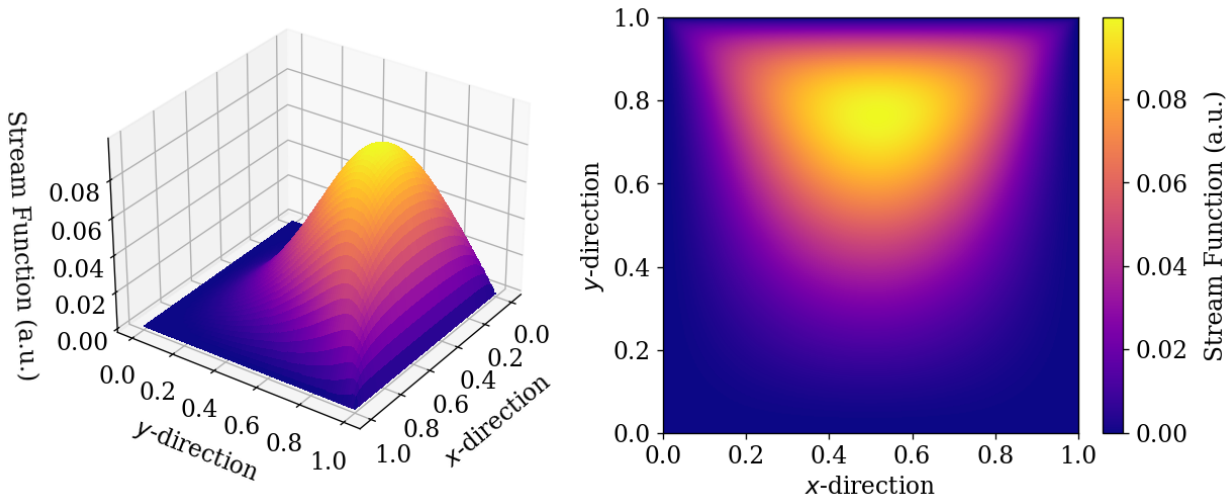


Figure 3.2: 3D and heatmap plots of stream function for Reynolds number 10 and $V_{\text{bot}} = 0$.

in the velocities being low at the vortex centre. The highest velocities are those at the driving lid, and those around the vortex. This can also be seen in Fig. 3.2. Particularly, the velocities in and around the driving lid are highest as the gradient of the stream function is highest here. At and around $x = 0.5$, and from $y = 1.0$ to $y = 0.8$, the stream function increases from 0.00 to over 0.09. However, in the same x region, and from $y = 0.8$ to $y = 0.2$, the stream function goes back to 0.00 (from over 0.09), which is a smaller gradient. This is also reflected in the velocities at the bottom edge of the vortex, whose magnitudes are smaller than those at the driving lid.

On a final note regarding the stream function, another major difference as compared to the structure of the velocities is the boundary conditions. The fluid boundaries of the velocity are determined by the no-slip condition - the velocity of the fluid at the boundaries is equal to the boundary velocity (cavity walls). This results in all the velocities being zero at the boundary, except that of the driving lid. The stream function however, as mentioned in the problem overview (Section 1.1), is only interested in the differences of the velocities. The only non-zero derivative of the stream function u is $\partial u/\partial y = V_{\text{top}}$. The top wall is constant in y though, meaning that u has to be constant in y as well. Since there is no x- or y-dependence in u for the other boundaries, u must be constant throughout the entire boundary. As seen in Fig. 3.2, it is set to 0 along the entire boundary, because for the flow velocities it is only the gradient of u that matters, not the actual values. This setting of $u = 0$ along the boundaries is actually still a consequence of the no-slip conditions, since the velocities at all the other boundaries being 0 implies that u cannot be a function of x nor y. Without the implementation of these boundary conditions (u_{boundary} =constant), the physics of the simulation would be wrong.

Fig. 3.3 shows a 3D and contour plot of the vorticitity. The vorticity is a quantification of the (local) spinning or rotational motion of a continuous fluid medium at some point, and as such is given by the curl of the flow velocity ($\vec{V} = (V_x,\ V_y)$). The vorticity, like the stream function and velocity, is symmetric about the centre of the x-axis, as expected for a low Reynolds number/

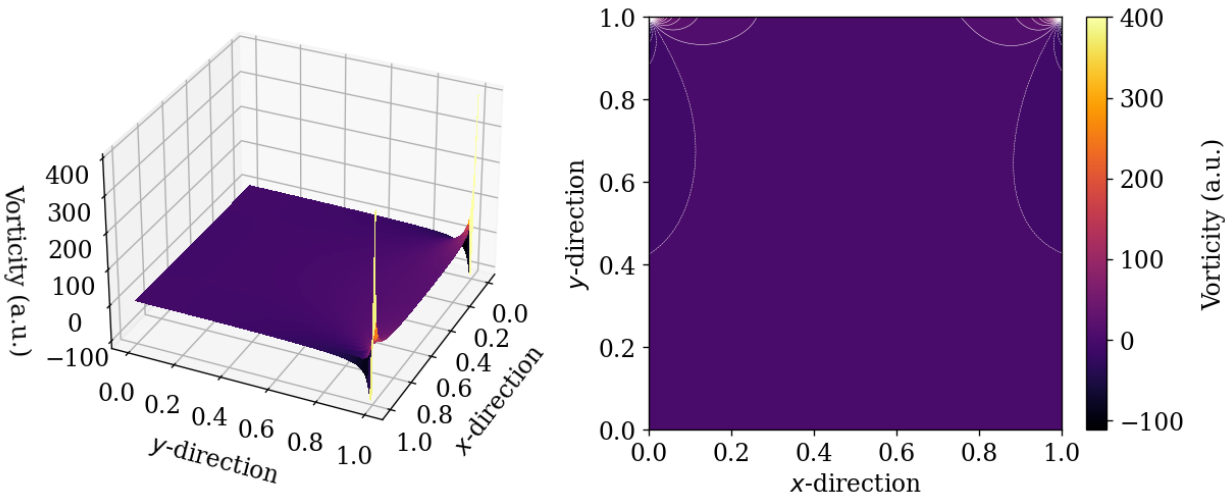


Figure 3.3: 3D and contour plots of vorticity for Reynolds number 10 and $V_{\text{bot}} = 0$.

laminar flow. Also as expected, the highest points of rotation are at the corners of the driving lid, where there are significant changes in the flow direction. The vorticity is relatively homogeneous in the rest of the cavity. Despite the changes in flow direction (for example, at the vortex), the vorticity remains low. Looking at the theoretical form of the vorticity given by Eq. (1.5)

$$w = \frac{\partial V_y}{\partial x} - \frac{\partial V_x}{\partial y}$$

it is clear to see that the vorticity is dependent on changes in both V_x and V_y. At the corners, there is a point of extremely high vorticity followed by extremely low vorticity (seen quite easily in the 3D plot). This implies that first, V_y increases as x increases and there is either a small V_x increase, or that V_x decreases as y decreases. Looking at Fig. 3.1b, it is clear that the dominant contribution first comes from the changes in V_y, as V_y is practically 0 at the lid at $x = 0.8$, but quickly increases to become the only velocity contribution at $x = 0.9$. The V_x contribution decreases, until the right boundary, where V_x flips direction; from $y = 1.0$ to $y = 0.9$ at the upper right corner, there is a large change in V_x, where not only does the magnitude change, but the direction too. This leads to the low vorticity point seen in Fig. 3.3. The responses of the flow velocity to the stream function and vorticity are all as expected. This implies that the discretised equations and iteration approach are correct for the stream function, vorticity, and velocities, at least for a Reynolds number of 10 and no bottom wall velocity. The pressure formulation needs revision, as while there is some semblance of physical reality (e.g. pressure building in the upper corners), the pressure gradient is unphysical.

An important checkpoint of a simulation is to check whether it can reach a steady-state.[1] Furthermore, the Reynolds number plays an important role in the attainment of steady-state conditions. To analyse the point at which the system reaches a steady-state, a probe was placed in the domain centre (for $N = 200$, this was (100, 100)) which recorded the time history of the horizontal

[1] Along with ensuring that residuals are minimized, reaching a steady-state is an indication of simulation convergence.

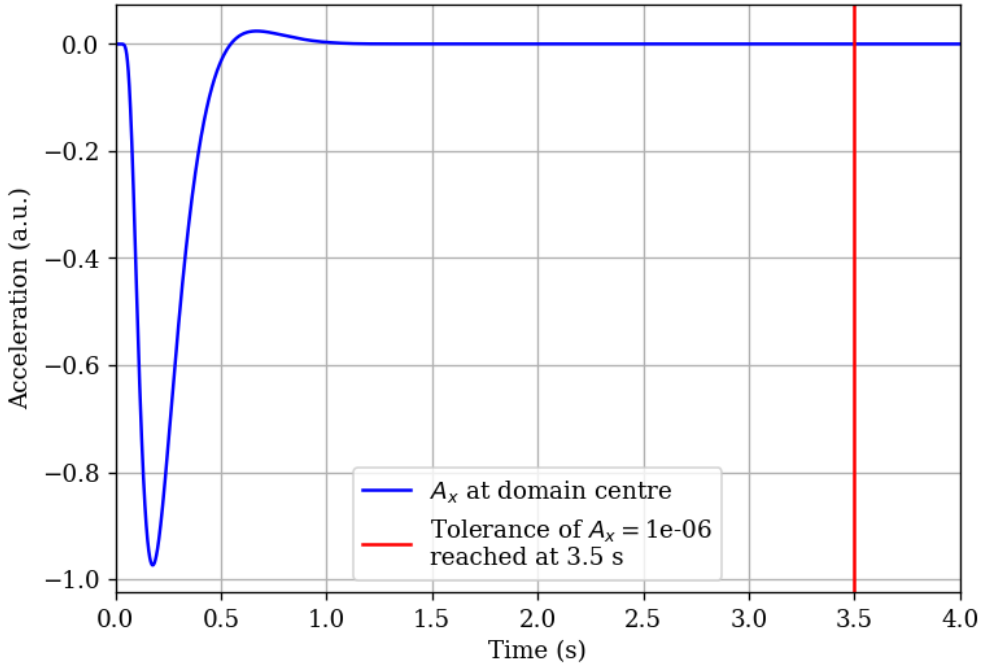


Figure 3.4: Horizontal acceleration A_x (change of V_x with respect to time) at domain centre (100, 100) showing time when tolerance is reached for Reynolds number 10 and $V_{\text{bot}} = 0$.

velocity V_x. Then, a function calculated the time changes in V_x i.e. the horizontal acceleration A_x. Fig. 3.4 shows this acceleration at the domain centre (100, 100), and the point at which the user defined steady-state tolerance was reached. A tolerance of 10^{-6} was set to be the point at which the system could be considered to be in a steady-state. This corresponds to changes in V_x on the order of $(10^{-6}) \cdot \Delta t = (10^{-6}) \cdot (5 \cdot 10^{-5}) = 5 \cdot 10^{-11} \approx 10^{-10}$. The mean of V_x at the domain centre was around -0.2, which is $\sim 10^9$ times larger than the changes, so for all practical purposes $10^{-10} \approx 0$. This choice of 10^{-6} was relatively arbitrary. Since it was unknown what the magnitude of the velocities would be, a tolerance needed to be set that was small enough to ensure a good approximation to steady-state conditions while also ensuring that the computational times were reasonable (smaller tolerances means longer steady-state reaching times). After running the simulation a few times, and seeing the velocity order of magnitudes, changes on the order of 10^{-10} were deemed appropriate to be considered steady-state.

In Fig. 3.4, the tolerance is reached at 3.5 seconds. Thus, based on the criteria described above, the system reaches a steady-state at 3.5 seconds. This is relatively low and as expected for a low Reynolds number. The more laminar the flow is, the more structured it is, resulting in higher uniformity. This high uniformity allows the system to quickly reach steady-state - since there is no unexpected chaos (e.g. eddies), the changes in the flow are allowed to stabilise quickly. Not much can be said about this steady-state reaching time of 3.5 seconds until all cases are considered (different Reynolds numbers, different bottom wall velocities).

Beyond the steady-state reaching time, there is something very interesting about Fig. 3.4, the be-

haviour of the acceleration. The acceleration behaves as an under-damped (about 1/2 of the critical damping) harmonic oscillator. This implies that the horizontal velocity accelerates rapidly in the first 0.2 seconds, before starting to decrease in acceleration (it does *not* decelerate) in the next 0.3 seconds, indicating a stabilising velocity. Then, at 0.5 seconds, the velocity decelerates (indicated by the bump in the acceleration at 0.5 seconds) and stabilises, with the acceleration going to 0. The physical reasoning for this is as follows. At the start, the driving lid causes a vortex to form, in which the probe is placed at the bottom of the vortex. The velocity increases rapidly at first as the vortex continues to increase in intensity, before slowly stabilising as the flow reaches an equilibrium. Due to reactionary forces originating from the boundaries, which reflect the flow back, the velocity slightly decelerates. Soon after, all the forces in the fluid come to an equilibrium, and the velocity stabilises. This behaviour is seen for all Reynolds numbers and all bottom wall velocities, but the magnitudes of damping differ.

Error Analysis for Reynolds number 10 and $V_{\mathbf{bot}} = 0$

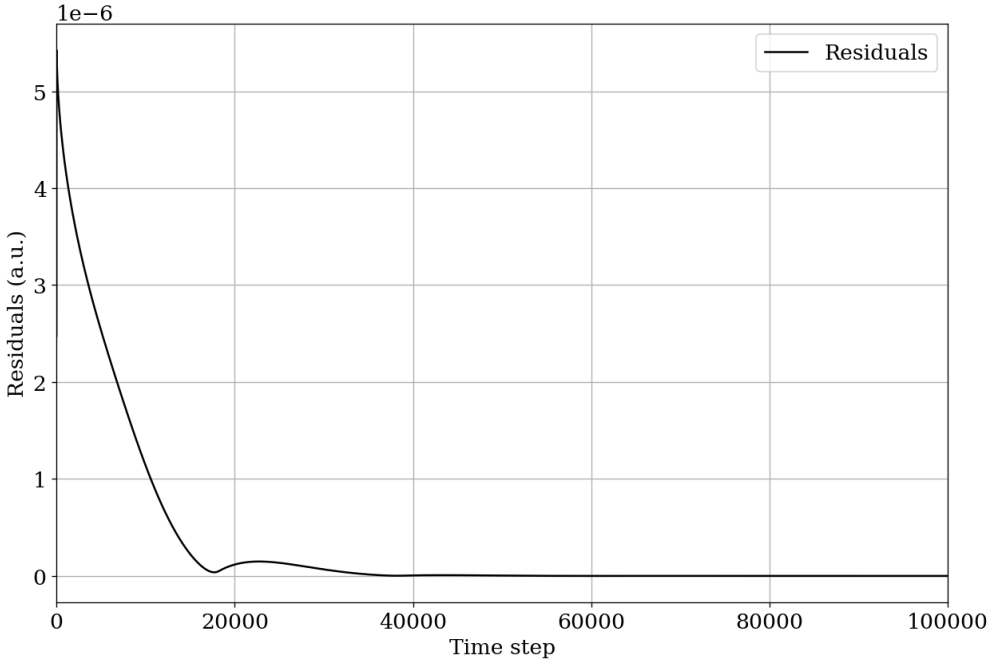


Figure 3.5: Residuals of stream function over all time steps.

Fig. 3.5 shows the stream function residuals from successive relaxation (ω was set to 1 throughout the project notebook as there was not enough time to determine an ideal ω). It can be clearly seen that the residuals start off very small, and decrease rapidly with increasing time step. In general, residuals on the order of 10^{-6} are considered to be tightly converged [6]. The residuals seen in Fig. 3.5 start at $5 \cdot 10^{-6}$ (which already implies tight convergence) and then decrease to below 10^{-12}. This, admittedly, is suspiciously low, but no faults could be found in the script. There is a doubt though, in the calculation of the residuals. To determine whether $r = f^{\text{new}} - f^{\text{old}}$ (see Eq. (2.28)) was below the tolerance ϵ (distinct from the tolerance mentioned in Fig. 3.4), the mean of r was taken (r is an array). This could have resulted in larger residuals being averaged out by significant numbers of smaller residuals. A method that might prove to be more precise is by using normalisation instead of averaging, whereby the norm[2] of r is taken and compared to the tolerance ϵ. This is a much stricter requirement than taking the mean, and was initially applied. Due to this strictness, many times the code resorted to calculating until the user defined max iteration number was reached. This resulted in very long calculation times, and while it increased the residuals and the precision requirement, was abandoned in favour of the mean. Another method that would prove more precise than the mean, and less computationally exhaustive as the norm, would be by only stopping the successive relaxation when every element in r is below the tolerance. This could be done easily and efficiently by using the `.all()` function in Python (e.g. `r.all()<epsilon`) but was unfortunately not implemented.

[2]Norm of a vector/ array $\vec{v} = (v_1, v_2, \ldots, v_n)$ is given by $v = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$.

The truncation errors given in Table 2.1 should be compared to the mean values of the different functions, to get an idea of how precise (and maybe even accurate) each function is. For the pressure, the sources of error arise from the boundary condition discretisation (error `h`=0.005) and the spatial interior discretisation (error $\texttt{h}^2$=0.000025). The boundaries of u are set to 0, so no truncation error arises here. Similarly, V_x and V_y have no truncation errors arising from boundary discretisation, as their boundaries are set. However u, V_x, and V_y all suffer from spatial interior truncation errors. The vorticity suffers from time truncation errors, and both spatial boundary and interior truncation errors. Table 3.1 shows the relevant truncation errors.

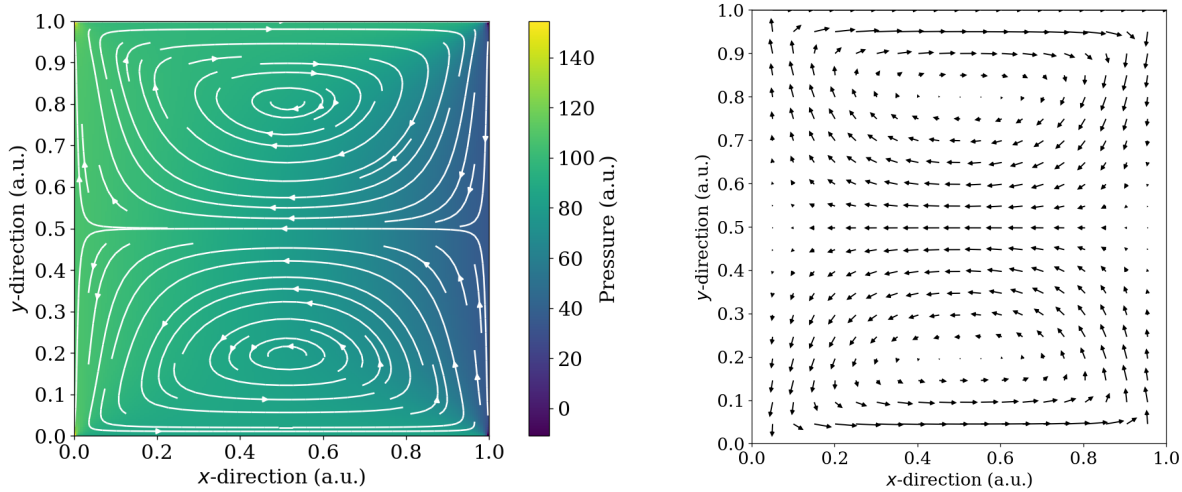
Table 3.1: Errors, error sources, and mean values of different functions for Reynolds number 10 and $V_{\text{bot}} = 0$.

Function	**Error Source**	**Mean of Function at Error Source (a.u.)**	**Limiting Errors**	**Limiting Error Values**
Stream Function u	Interior	0.02921	$\texttt{h}^2$	0.000025
Horizontal Velocity V_x	Interior	0.12666	$\texttt{h}^2$	0.000025
Vertical Velocity V_y	Interior	0.09909	$\texttt{h}^2$	0.000025
Vorticity w	Interior	1.88714	`t`	0.00005
Vorticity w	Boundary	10.420	`h`	0.005
Pressure p	Interior	68.50007	$\texttt{h}^2$	0.000025
Pressure p	Boundary	68.576	`h`	0.005

It can be seen that the errors of concern are the boundaries, as these are the largest errors. Thankfully, both w and p at the boundaries are on the order of 10^1 (p can be considered 10^2, depending on convention), while the error is only $5 \cdot 10^{-3}$, which implies that both w and p are precise up to four significant figures (including decimals). Furthermore, for w, despite errors arising from the time discretisation and spatial interior descretisation as well, it can be seen that both the interior of w (order of magnitude 10^0) and boundaries of w (order of magnitude 10^1) have values that are several orders of magnitude higher than the corresponding errors (10^{-5} for both interior and time truncation errors). Thus the values of the interior of w are precise up to five significant figures. The interiors of the velocities V_x and V_y are precise up to four significant figures, while the interior of u is precise up to three significant figures (comparing the means with the limiting error values).

Since the residuals all tightly converge (10^{-12} is a brilliant, if suspiciously good, residual value), and the means of all the functions are several orders of magnitude higher than their respective truncation errors, it can be said that for a Reynolds number of 10 and a $V_{\text{bot}} = 0$, the simulation is accurate and precise, with the exception of the accuracy of the pressure (explained in Fig. 3.1.

3.1.2 Bottom Wall Velocity $V_{\text{bot}} = 1$



(a) Velocity stream & pressure plot.

(b) Velocity quiver plot.

Figure 3.6: Velocity and pressure plots for Reynolds number 10 and $V_{\text{bot}} = 1$ at steady-state (in-simulation time of 5 seconds).

The results seen in Fig. 3.6 mirror those seen in Fig. 3.1, with the exception that now, two vortices have been created due to the presence of a moving bottom wall with a velocity equal to that of the top wall. Due to the low Reynolds number, the central x-axis symmetry remains (the vortices are both centered around $x = 0.5$), with another symmetry arising in the centre of the y-axis. However, this latter symmetry is not due to the Reynolds number, but rather the fluid dynamics of the system. Since the driving lid velocities are of equal magnitude (and direction), the vortices created are of equal strength, and hence the border between the two vortices lies exactly in the middle of the cavity. If one of the walls had a higher velocity, one of the vortices would overpower the other, and this $y = 0.5$ axis symmetry would be broken.

A difference from the previous case is that at the border between the two vortices, the velocity is higher. That is because there are velocity contributions from both vortices, which add up to give velocity magnitudes on par with those bordering the lids. Furthermore, the fluid velocities in general are higher around the lower borders (side of vortex facing away from the driving wall) of the vortices, due to the contribution from the other vortex. While the two vortices indicate chaotic behaviour, the vortices being centered around $x = 0.5$ and the flow having the central x-axis symmetry indicate that the viscous forces are still very significant. Furthermore, the eddies present in the bottom corners of Fig. 3.1 have disappeared, due to the moving bottom wall. The rest of the velocity profile of each vortex matches that of the $V_{\text{bot}} = 0$ case, with the same analysis thus applying.

The pressure profile of Fig 3.6a matches the upper-half of Fig. 3.1, wherein the pressure profile is reflected around the perpendicular axis at $y = 0.5$. This results in a high pressure along the entire left wall ($x = 0$), and pressure build ups in the upper and lower left corners, due to the flow velocity pushing fluid up into these corners. A pressure decrease is experienced in the middle ($y = 0.5$) of

the right wall ($x = 1$), wherein the flow pulls fluid away from the wall and towards the left. While the gradient of this pressure profile is wrong (similar to the $V_{\text{bot}} = 0$ case), the high pressures in the left corners, and the pressure decrease in the middle of the right wall (1.0, 0.5), are both physically correct. Again, the erroneous gradient is most likely due to the boundary conditions.

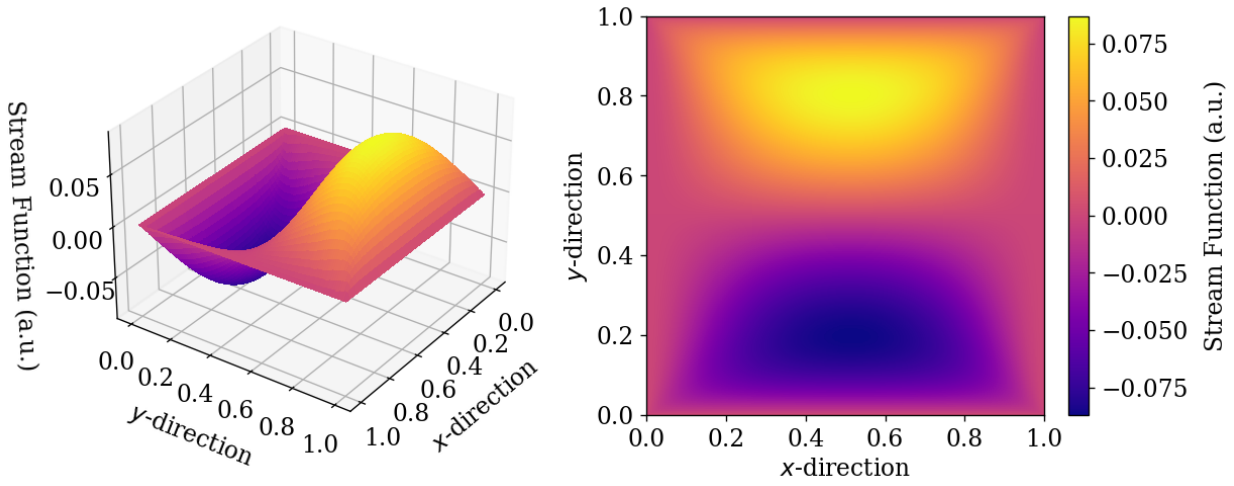


Figure 3.7: 3D and heatmap plots of stream function for Reynolds number 10 and $V_{\text{bot}} = 1$.

The stream function seen in Fig. 3.7 is similar to that of the stream function in Fig. 3.2, with the same symmetry about $x = 0.5$. The difference being however, is that due to the presence of the moving bottom wall, a depression in the stream function is seen bordering the bottom wall. This area of negative stream function has the exact same magnitude as the stream function bordering the top wall (as the velocities of the walls are the same), but is negative due to the flow direction. While the horizontal velocities V_x in the upper and bottom halfs of the cavity are the same, the vertical velocities V_y are equal and opposite (see Fig. 3.6b). This is a result of the flow being equal and opposite in the y-direction, and is reflected in the stream function by the negative gradient seen in Fig. 3.7. A positive stream function in place of the stream function depression would result in an unstable system, and the two stream function peaks would eventually join into one. Physically, this means that two vortices are created, rotating in the same direction (for $V_{\text{bot}} = 1$ they rotate in opposite directions, see Fig. 3.6a). This is unstable, and eventually they form one large vortex. This is the case when $V_{\text{bot}} = -1$.

In Fig. 3.8 the vorticity plots can be seen. The vorticity is symmetric about $x = 0.5$ and $y = 0.5$, and just as in the case of $V_{\text{bot}} = 0$ has large peaks followed by dips in the corners. The difference being that here the peaks and dips are seen in all corners, since the top and bottom walls both move. The explanation of these peaks and dips are the same as given in the $V_{\text{bot}} = 0$ subsection. Also similar to the $V_{\text{bot}} = 0$ case is the homogeneity of the vorticity in the centre (in particular, its zero value). This is also explained by the same reasoning as given in that subsection. One difference is the magnitude of the negative vorticity. In the $V_{\text{bot}} = 0$ case, the most negative vorticity was around -100. Here however, the magnitude of the vorticity minimum matches that of the magnitude of the vorticity maximum (both around 400). This is due to the same phenomena seen

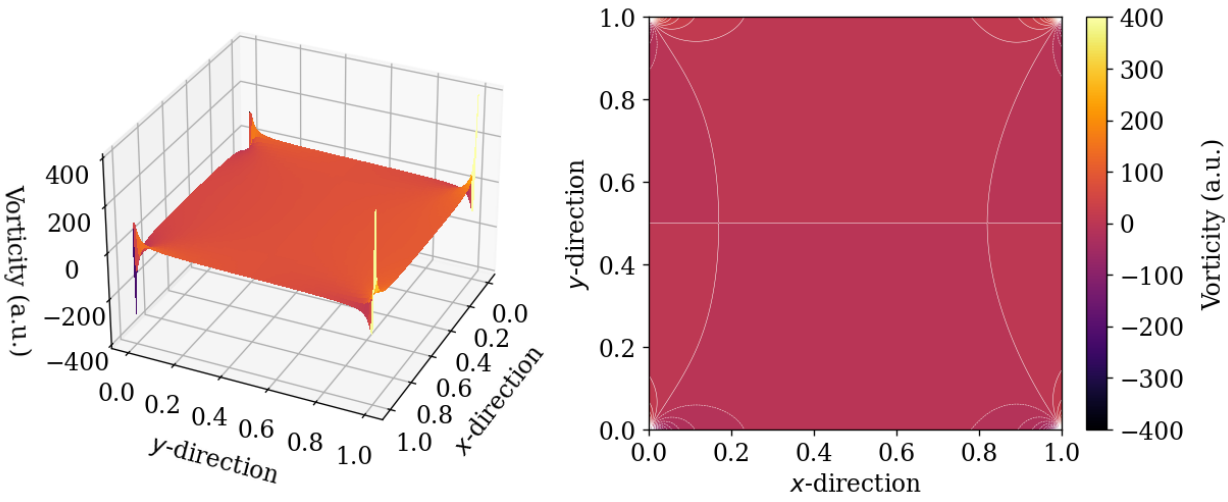


Figure 3.8: 3D and contour plots of vorticity for Reynolds number 10 and $V_{\text{bot}} = 1$.

in the stream function. Due to the opposite and equal values of V_y for the two vortices, the bottom corners experience the same vorticity effects, but flipped in sign. First, the vorticity decreases substantially to around -400, and then increases to about 100, the opposite of what happens at the top wall (increase to 400 and decrease to -100). The reasoning behind this decrease(increase) and subsequent increase(decrease) for the bottom(top) wall is the same as that of the increase and subsequent decrease for the top wall in the $V_{\text{bot}} = 0$ case.

The acceleration and steady-state reaching time can be seen in Fig. 3.9, with the same damped oscillatory behaviour seen for $V_{\text{bot}} = 0$. The difference being that the steady-state reaching time is now considerably lower (over 1 second) than that of the $V_{\text{bot}} = 0$ case. This is due to the fact that the two vortices are of equal strength, and meet exactly at $y = 0.5$. The probe is placed at $(x,\ y) = (0.5,\ 0.5)$, meaning that it feels the effect of these two vortices meeting, and hence stabilises much quicker due to the enhanced velocity contributions from both vortices. In general, having two, oppositely-rotating vortices perfect centered within the cavity will result in a quicker stabilisation, as there are velocity contributions from the top *and* bottom sides. Whether $V_{\text{bot}} = 1$ or $V_{\text{bot}} = -1$ results in a smaller steady-state reaching time depends on the Reynolds number of the fluid.

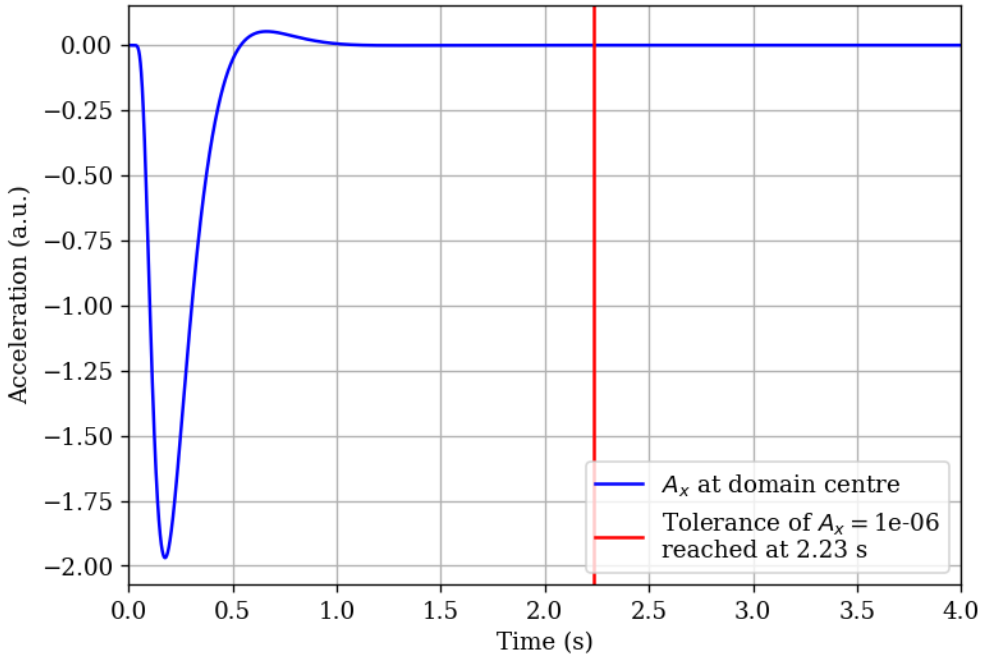


Figure 3.9: Horizontal acceleration A_x (change of V_x with respect to time) at domain centre (100, 100) showing time when tolerance is reached for Reynolds number 10 and $V_{\text{bot}} = 1$.

Error Analysis for Reynolds number 10 and $V_{\text{bot}} = 1$

Fig. 3.10 shows the residuals for $V_{\text{bot}} = 1$. Just as the $V_{\text{bot}} = 0$ case, the residuals already start tightly converged, and quickly reduce, implying that the simulation converges very well. Again, these residuals are suspiciously small, and the same reasoning discussed in the $V_{\text{bot}} = 0$ case as to why, also applies here. Table 3.2 shows the errors, error sources, and means of the different functions. Again, like the $V_{\text{bot}} = 0$ case, the orders of magnitude of the (means of the) function values are several orders higher than those of the limiting error values. The least precise mean value is that of the stream function, which has a precision of three significant figures based on its limiting error. Every other (mean) function value has a precision of four significant figures or higher. Despite the high precision in the pressure, the pressure values themselves are not accurate. In conclusion, the simulation for this case converges, and all the calculated functions have at least three significant figures of precision based on truncation errors.

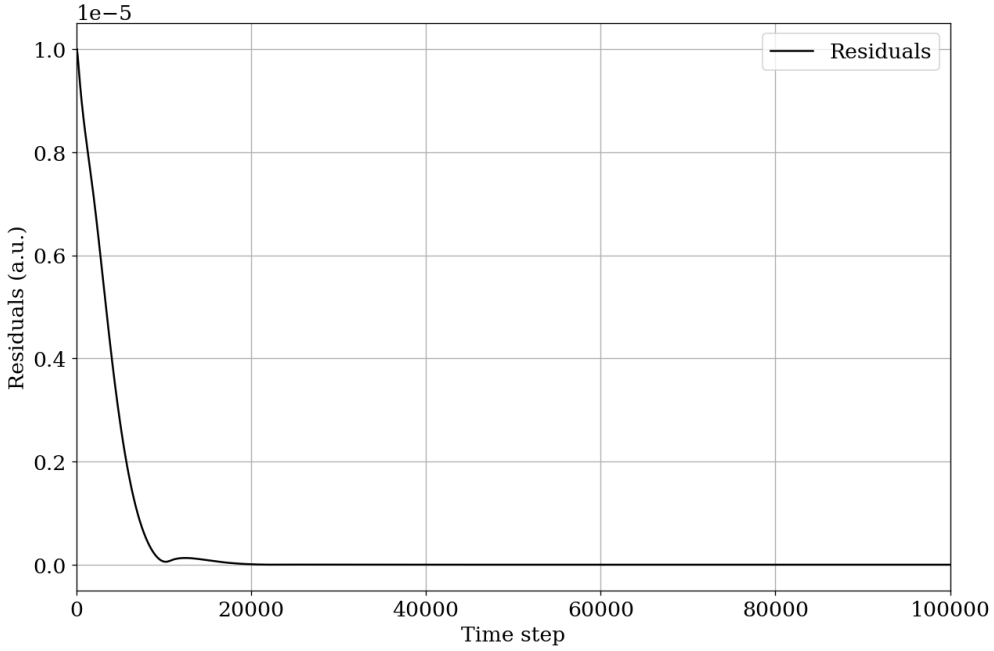
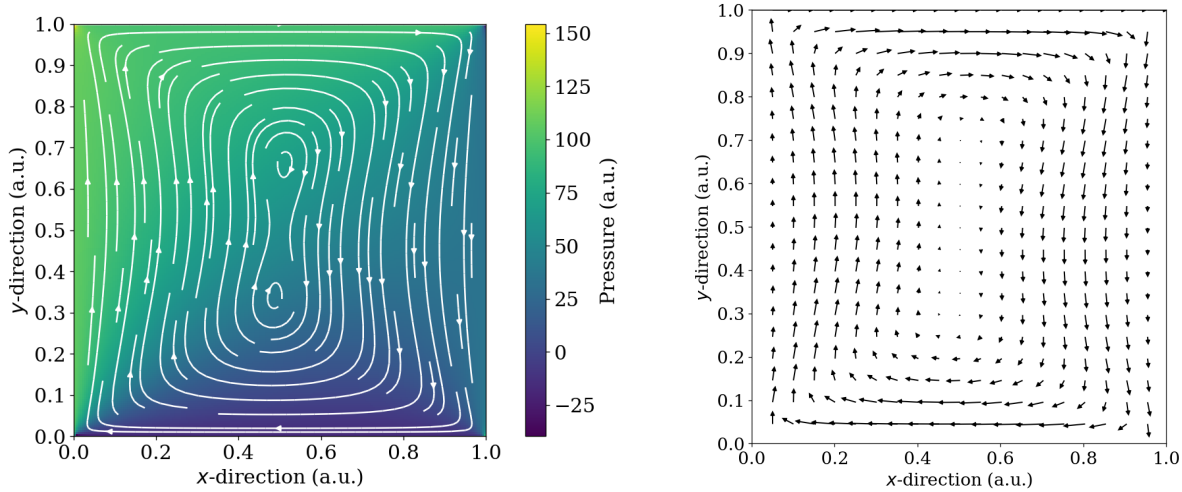


Figure 3.10: Residuals of stream function over all time steps for Reynolds number 10 and $V_{\text{bot}} = 1$.

Table 3.2: Errors, error sources, and mean values of different functions for Reynolds number 10 and $V_{\text{bot}} = 1$.

Function	**Error Source**	**Mean of Function at Error Source (a.u.)**	**Limiting Errors**	**Limiting Error Values**
Stream Function u	Interior	0.03646	$\mathtt{h}^2$	0.000025
Horizontal Velocity V_x	Interior	0.23225	$\mathtt{h}^2$	0.000025
Vertical Velocity V_y	Interior	0.11339	$\mathtt{h}^2$	0.000025
Vorticity w	Interior	3.42006	$\mathtt{t}$	0.00005
Vorticity w	Boundary	19.319	$\mathtt{h}$	0.005
Pressure p	Interior	81.42768	$\mathtt{h}^2$	0.000025
Pressure p	Boundary	81.460	$\mathtt{h}$	0.005

3.1.3 Bottom Wall Velocity $V_{\text{bot}} = -1$



(a) Velocity stream & pressure plot.

(b) Velocity quiver plot.

Figure 3.11: Velocity and pressure plots for Reynolds number 10 and $V_{\text{bot}} = -1$ at steady-state (in-simulation time of 5 seconds).

Fig. 3.11 shows the plots of the flow velocities and pressure. Here, the top and bottom walls have the same speed, but opposite velocities. This results in two vortices being created, both of which rotate in the same direction. After some time, their respective currents merge into each other, forming a large loop around the entire cavity. The original vortices can be seen in the stream plot (Fig. 3.11a) but as seen in the quiver plot (Fig. 3.11b), it is as if the fluid follows a square loop around the cavity. The low Reynolds number again implies laminar flow - mainly viscous, with some inertial, contributions. This can be seen by the symmetry about the $x = 0.5$ axis, which has been identifiable in all three cases of a Reynolds number of 10. The magnitudes of the velocity mirror those of the previous cases; the velocities bordering the driving walls are larger, and decrease radially toward the vortex centres. Since the driving walls have equal speeds, the loop is also somewhat symmetric about $y = 0.5$, as seen in the $V_{\text{bot}} = 1$ case.

The pressure is mainly focused in the upper left corner, with an increase as well in the lower right corner. These pressure build-ups are again due to fluid being pushed into the respective corners. What is interesting to note however, is that while the gradient is symmetric about $x = 0.5$ in contours, it is not symmetric in magnitude. The right wall has a lower pressure than the left, which is not physical, as they experience equal and opposite flow velocities. Furthermore, the upper left corner has a much higher pressure concentration than the lower right corner, even though both corners undergo the same contact process with the driving wall. This again signifies that there is a problem with the pressure calculations, and in particular, the boundary conditions are suspect.

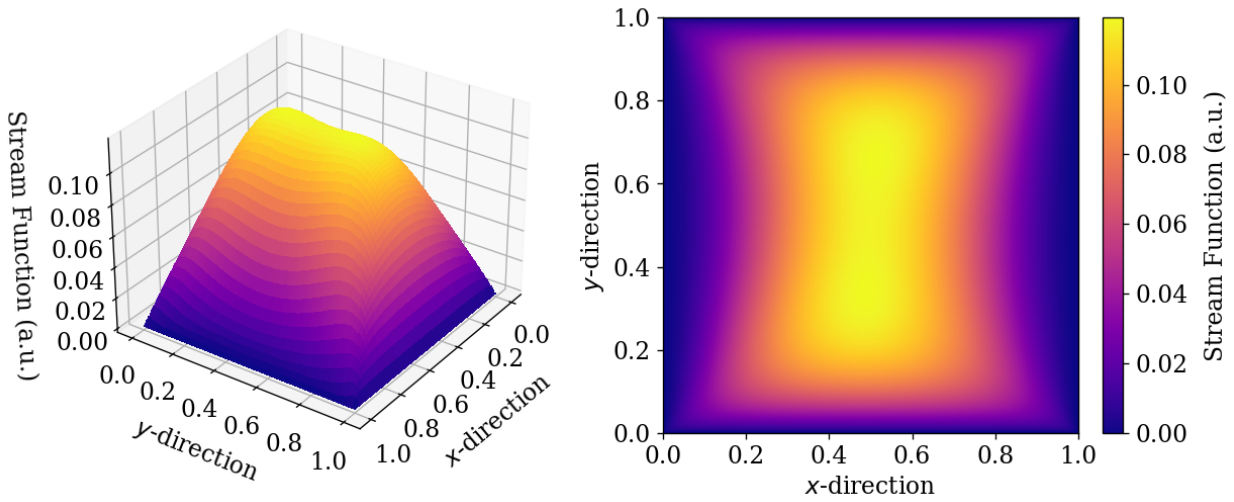


Figure 3.12: 3D and heatmap plots of stream function for Reynolds number 10 and $V_{\text{bot}} = -1$.

Fig. 3.12 shows the stream function for this case. As discussed in the $V_{\text{bot}} = 1$ case, when two vortices are produced which rotate in the same direction, this is an unstable situation. After some time, the vortices eventually merge to form a fluid loop around the cavity, driven by two central vortices. This is reflected in the stream function. Unlike the $V_{\text{bot}} = 1$ case, there are no negative stream function values, as the vortices both travel/ rotate in the same direction. This same direction rotation is also seen in the vorticity.

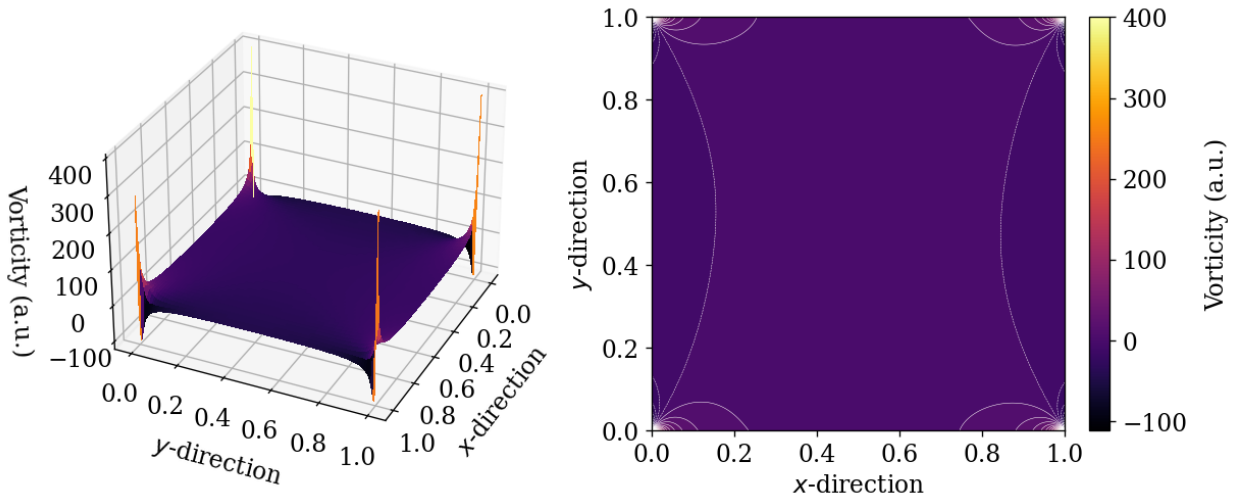


Figure 3.13: 3D and contour plots of vorticity for Reynolds number 10 and $V_{\text{bot}} = -1$.

The vorticity plots are shown in Fig. 3.13. The four corner peaks are again seen, but unlike the $V_{\text{bot}} = 1$ case, the negative vorticity values are not of the same magnitude as the positive values. Instead, the range of vorticities reflects that of the $V_{\text{bot}} = 0$ case, which makes sense as both vortices

are rotating in the same direction. The physical mechanisms that explain the peaks and troughs of the stream function and vorticity are the same as those explained in the previous cases.

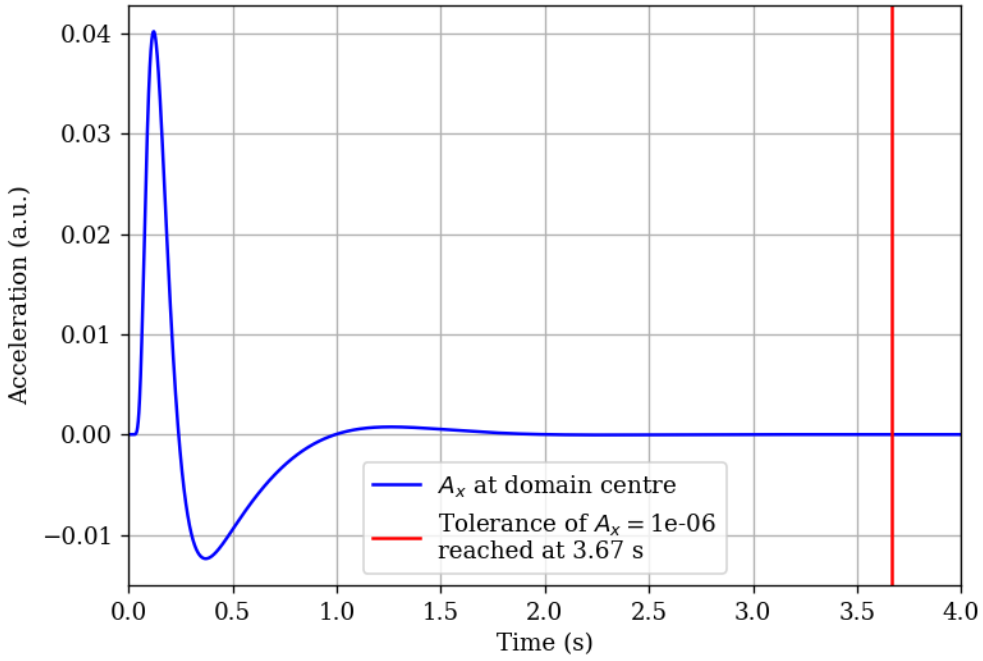


Figure 3.14: Horizontal acceleration A_x (change of V_x with respect to time) at domain centre (100, 100) showing time when tolerance is reached for Reynolds number 10 and $V_{\text{bot}} = -1$.

The acceleration and steady-state reaching time can be seen in Fig. 3.14, with the same damped oscillatory behaviour seen in the previous cases - however, the damping here is less than with the other two. Here the steady-state reaching time of 3.67 seconds is more than both the previous cases. It is only slightly more than the $V_{\text{bot}} = 0$ case, and over a second more than the $V_{\text{bot}} = 1$ case. What is interesting, is that the $V_{\text{bot}} = -1$ steady-state reaching time actually decreases as the Reynolds number increases (for Reynolds numbers of 100 and 200, $V_{\text{bot}} = -1$ results in the quickest stabilising times). An explanation for this is given at the end of this chapter.

Error Analysis for Reynolds number 10 and $V_{\mathbf{bot}} = -1$

Fig. 3.15 shows the residuals for $V_{\text{bot}} = -1$. Unlike the previous two cases, the residuals start well converged (10^{-5}) but not tightly converged (10^{-6}), but they do reduce rapidly to reach tightly converged conditions. A reason for these slightly higher residuals has not been found. It is posited that they might arise from rounding errors or representation errors when sweeping the grid, and that these errors were higher than in the previous two cases, due to either some hidden cause in the script, or a random float calculation. Either way, there was no time to identify the cause, and seeing as the residuals nonetheless indicate a converging simulation, the cause for their increase was ignored.

Table 3.3 shows the errors, error sources, and means of the different functions. Again, like the previous cases, the orders of magnitude of the (means of the) function values are several orders higher than those of the limiting error values. The least precise mean value is that of the stream function again, which has a precision of three significant figures based on its limiting error. Every other (mean) function value has a precision of four significant figures or higher. Despite the high precision in the pressure, the pressure values themselves are not accurate. In conclusion, the simulation for this case converges, and all the calculated functions have at least three significant figures of precision based on truncation errors.

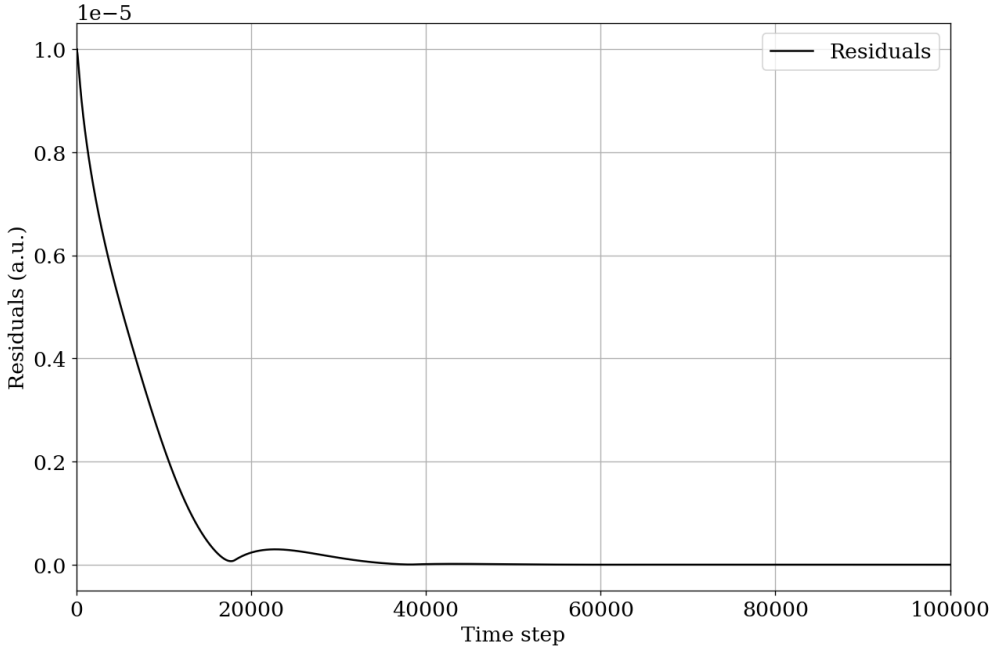


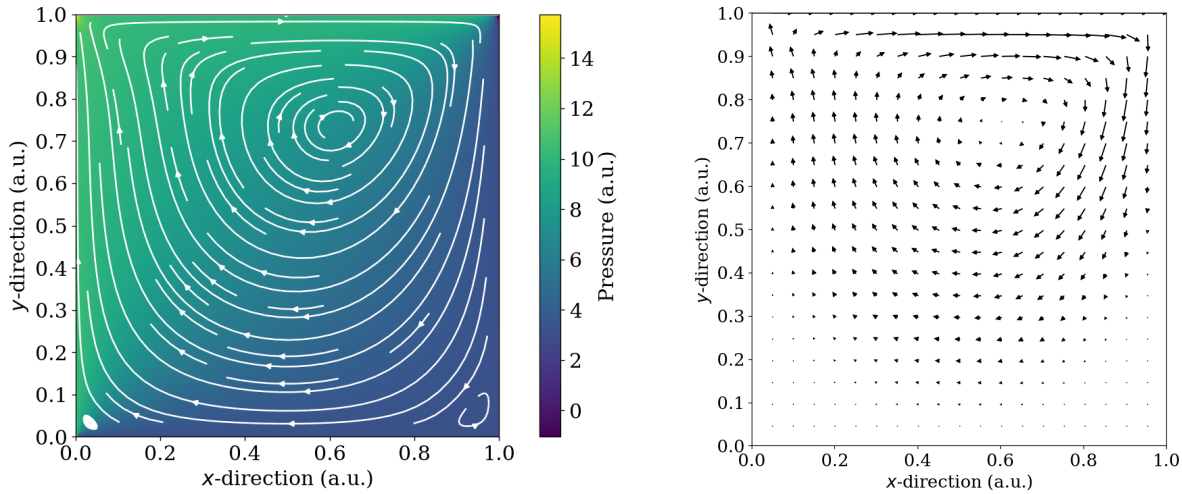
Figure 3.15: Residuals of stream function over all time steps for Reynolds number 10 and $V_{\text{bot}} = -1$..

Table 3.3: Errors, error sources, and mean values of different functions for Reynolds number 10 and $V_{\text{bot}} = -1$.

Function	**Error Source**	**Mean of Function at Error Source (a.u.)**	**Limiting Errors**	**Limiting Error Values**
Stream Function u	Interior	0.05845	$\mathtt{h}^2$	0.000025
Horizontal Velocity V_x	Interior	0.15727	$\mathtt{h}^2$	0.000025
Vertical Velocity V_y	Interior	0.19820	$\mathtt{h}^2$	0.000025
Vorticity w	Interior	3.50725	`t`	0.00005
Vorticity w	Boundary	19.527	`h`	0.005
Pressure p	Interior	56.70883	$\mathtt{h}^2$	0.000025
Pressure p	Boundary	63.928	`h`	0.005

3.2 Reynolds Number: 100

3.2.1 Bottom Wall Velocity $V_{\mathbf{bot}} = 0$



(a) Velocity stream & pressure plot.

(b) Velocity quiver plot.

Figure 3.16: Velocity and pressure plots for Reynolds number 100 and $V_{\text{bot}} = 0$ at steady-state (in-simulation time of 25 seconds).

Fig. 3.16 shows the velocity and pressure plots for the case of $V_{\text{bot}} = 0$ for a Reynolds number Re of 100. The (fluid) dynamics of the system remain the same as the $Re = 10$ case, with the only difference being now a situation where the inertial forces play a more significant role. Immediately, the effect of the inertial forces can be seen in Fig. 3.16, with the breaking of the $x = 0.5$ symmetry. In the $Re = 10$ case, the vortex and flow loop around the cavity was centered and symmetrical at the $x = 0.5$ axis. Now the vortex and flow is no longer centered, nor symmetrical. This is to be expected with a higher Reynolds number, as it implies that the inertial forces being provided by the driving lid are dominating the viscous forces in the fluid.

The reason the vortex moves right of centre is because the inertial forces at the top right are stronger than the viscous forces which resist the flow propagation around the vortex. Fig. 3.16b shows this clearly - the velocities on the right of the vortex are much greater than the velocities on the left. The fluid is driven at the top boundary, before hitting the right wall. This fluid is then reflected off of said wall, and constantly aided in its motion by the inertia of the fluid behind it. This motion is then propagated via inertial forces around the vortex. However, as the viscosity is lower, the inertial forces are more easily dissipated, which results in a non-uniform distribution of forces around the vortex. In the $Re = 10$ case, the viscous forces (which are forces that resist deformation - a resistance force) resist the inertial forces to the extent that the inertial forces are evenly transferred throughout the whole fluid, and the viscous and inertial forces reach an equilibrium that results in a flow symmetry about $x = 0.5$. Since the resistance is lower here due to the lower viscosity, more inertia is able to be transferred to the upper right part of the cavity, and hence less to the rest of the cavity (as said, the inertia dissipates more easily).

Thus the reason the vortex is shifted to the right is because to get an even amount of force on both sides (a vortex centre has zero force and zero velocity), there must be a greater amount of fluid on the left, as the fluid particles' (read: fluid grid cells) on the left have smaller forces and flow velocities. An example of this can be seen quite easily. Say that the average force a particle has on the right of the vortex is 10 (arbitrary units), and on the left is 2. For 5 particles on the right, the left needs 25 particles to even the forces out (both resulting in forces of 50), such that the net force in the vortex centre is 0. It is important to note as well that the vortex centre is also deeper. This is because these stronger inertial forces on the right wall push the vortex centre further down.

Note that the eddy in the bottom right corner has gotten larger as compared to the $Re = 10$ case. This is due to the fact that since the inertial forces are less evenly transferred across the cavity, and there is less resistance in the fluid, the fluid response starts becoming more chaotic. This chaos then results in internal currents forming in the fluid, and since the fluid resistance is lower, these internal currents and eddies do not die out over time, and are still present in the steady-state system. The pressure resembles that of $Re = 10$, $V_{\text{bot}} = 1$ case, with the exception that now the pressures are a lot lower. This makes sense, as a less viscous fluid can more easily dissipate inertial force, and hence more easily dissipate pressure within the system. When viscous forces dominate, the high resistance implies that pressure can build up more easily in areas of high flow concentration. Besides the upper corners however, the pressure gradient is wrong.

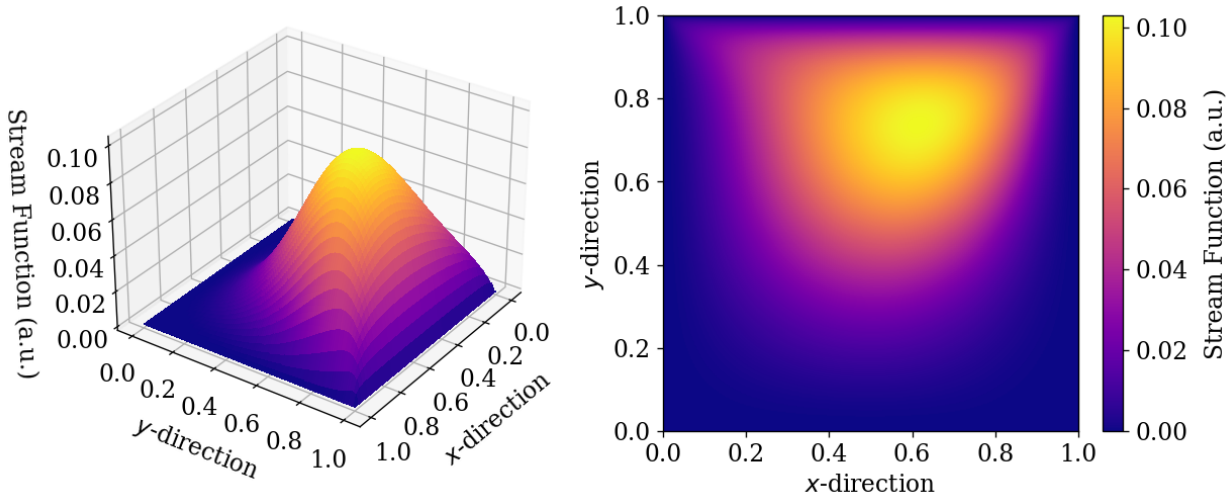


Figure 3.17: 3D and heatmap plots of stream function for Reynolds number 100 and $V_{\text{bot}} = 0$.

The stream function plots can be seen in Fig. 3.17. Just as in the case of $Re = 10$, the stream function reflects the flow velocity, as it should. One difference to the $Re = 10$ case (beyond the shift of the maximum stream function values, which is explained by the same mechanism as the vortex shift) is the higher stream function values. The $Re = 10$, $V_{\text{bot}} = 0$ case had a maximum stream function value just below 0.1, whereas here, the maximum is just above 0.1. This is again due to the higher inertial forces being transferred to the fluid as a consequence of lower resistance/

viscous forces. All other physical mechanism explanations are the same as in the previous cases.

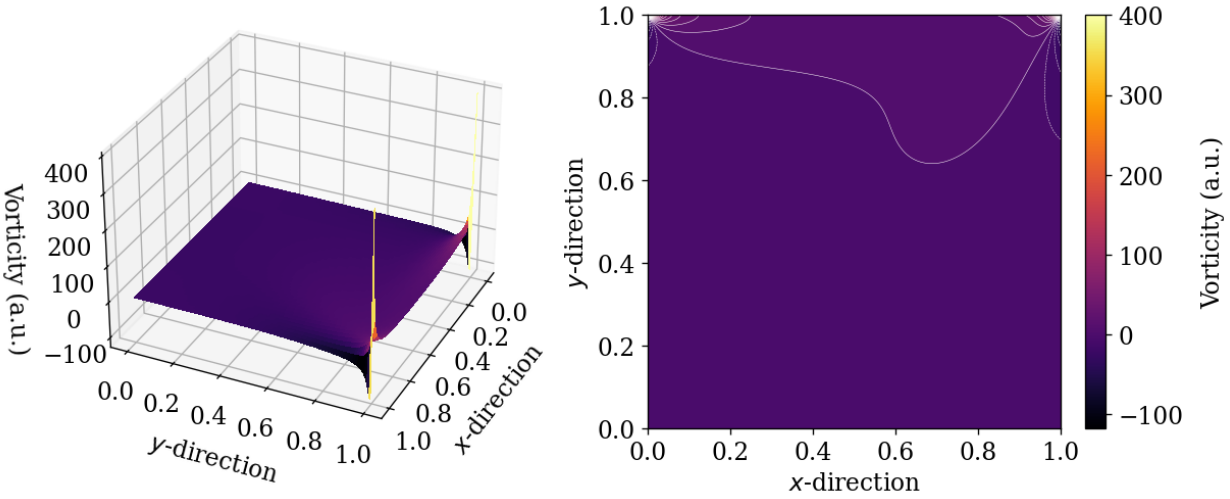


Figure 3.18: 3D and contour plots of vorticity for Reynolds number 100 and $V_{\text{bot}} = 0$.

The vorticity plots can be seen in Fig. 3.18. The corner peaks and troughs are still present, and their magnitudes are equal to those in the $Re = 10$ cases. The reason for this is that the vorticity peak/ trough magnitudes are determined by the magnitudes of the velocities, which at the top boundaries are determined by the lid velocity. Since the lid velocity has not changed, neither have the vorticity peak/ trough magnitudes. What has changed, as expected, is the $x = 0.5$ symmetry. It is clear that the vorticity in the vortex area is slightly higher (see the contour plot on the right), which makes sense, as both the magnitude and direction of V_x and V_y are constantly changing. In particular, this higher vorticity indicates a substantial change in V_x as y decreases, which can be seen in Fig. 3.16b. For example, at $x = 0.8$, as y decreases from $y = 1.0$ to $y = 0.7$, V_x flips orientation almost completely. While at $x = 0.2$, from $y = 0.7$ upwards, V_x is more or less constant until $y = 0.9$, which is also reflected in the vorticity by an increase at $(x,\ y) = (0.2,\ 0.9)$. This severe change in V_x on the right side of the vortex is again attributed to the turbulent flow caused by the high Reynolds number.

The acceleration and steady-state reaching time is shown in Fig. 3.19. The acceleration here is critically damped. Due to the turbulent flow, the system takes longer to stabilise. The lower viscous forces, resulting in lower intertial resistance, means that the response of the fluid is not as strong. This weaker response means that no oscillations occur in the acceleration, as the fluid does not reflect as intensely off of the walls, and the system slowly reaches steady-state by accelerating. This acceleration first increases, like in the $Re = 10$ case, but slowly decreases, meaning that the velocity first accelerates rapidly before starting to accelerate slowly, and finally not accelerating at all.

The steady-state reaching time of 16.9 seconds, which is almost six-fold the $Re = 10$ case, is as expected. A higher Reynolds number means that the flow is more turbulent, and inertial forces dominate. Thus, there is more chaos in the system, and due to this chaos it takes a lot longer for

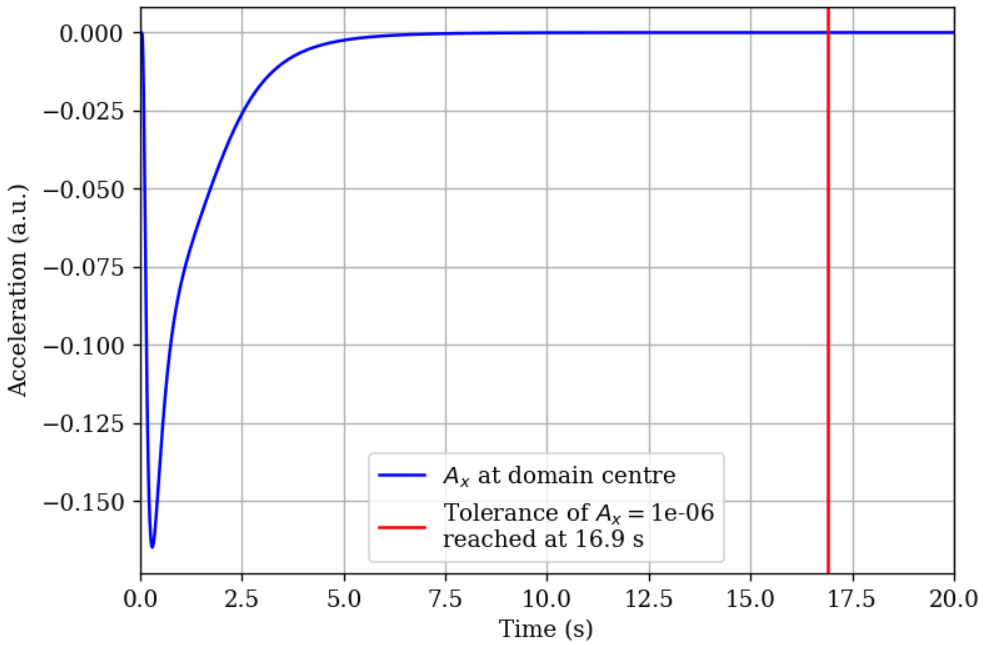


Figure 3.19: Horizontal acceleration A_x (change of V_x with respect to time) at domain centre (100, 100) showing time when tolerance is reached for Reynolds number 100 and $V_{\text{bot}} = 0$.

the system to stabilise and reach a point where it does not change in time.

Error Analysis for Reynolds Number 100 and $V_{\text{bot}} = 0$

Fig. 3.20 shows the residuals for $V_{\text{bot}} = 0$ and $Re = 100$. Here, the residuals start even smaller than in any of the previous cases, implying that the simulation is tightly converged. Again, the residuals are suspiciously low, and the reasoning for this discussed prior still applies.
Table 3.4 shows the errors, error sources, and means of the different functions. Again, like the previous cases, the orders of magnitude of the (means of the) function values are several orders higher than those of the limiting error values. The least precise mean value is that of the stream function again, which has a precision of three significant figures based on its limiting error. Every other (mean) function value has a precision of four significant figures or higher. Despite the high precision in the pressure, the pressure values themselves are not accurate. In conclusion, the simulation for this case converges, and all the calculated functions have at least three significant figures of precision based on truncation errors.

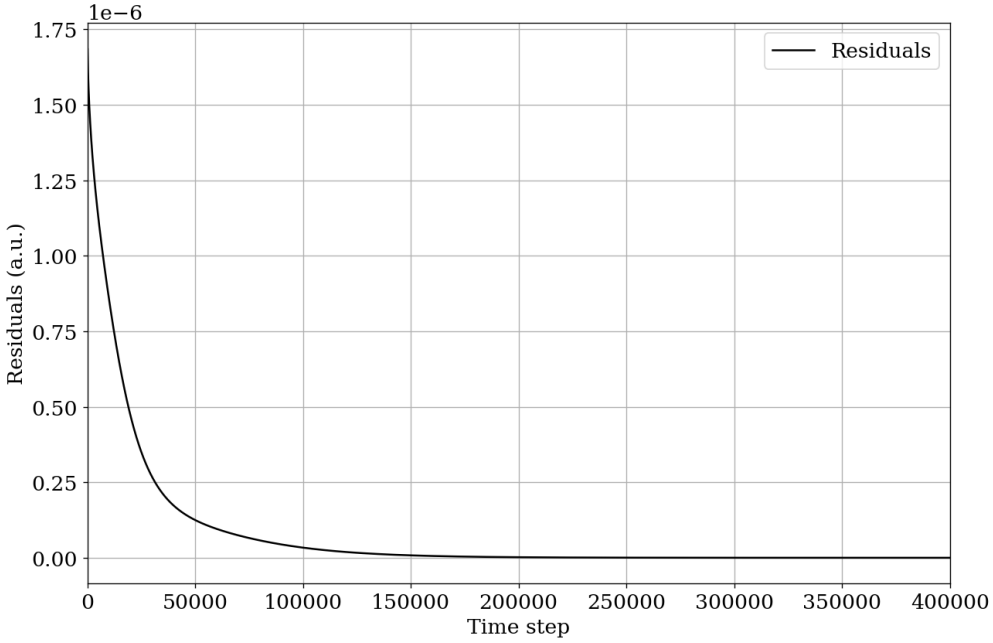
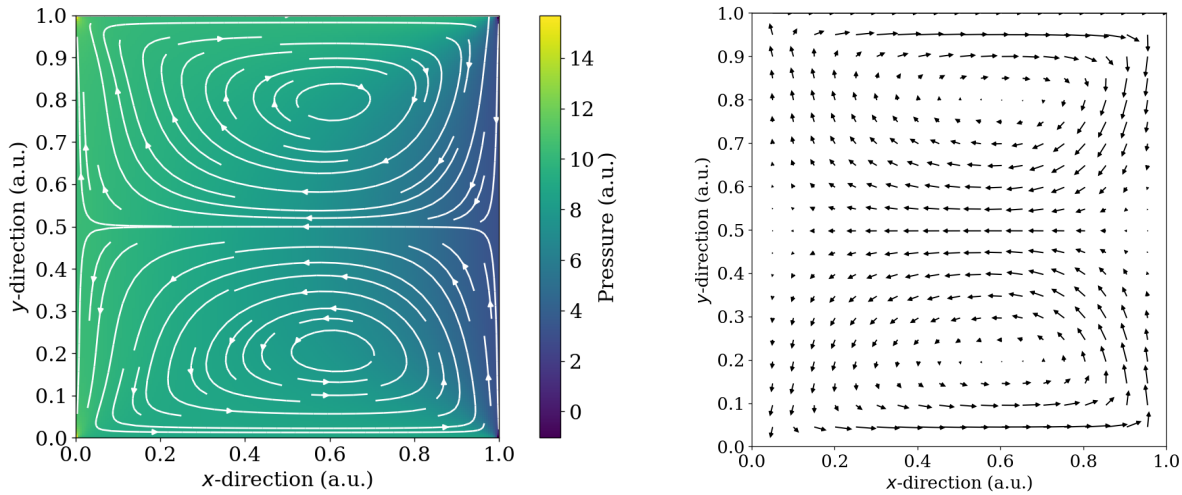


Figure 3.20: Residuals of stream function over all time steps for Reynolds number 100 and $V_{\text{bot}} = 0$.

Table 3.4: Errors, error sources, and mean values of different functions for Reynolds number 100 and $V_{\text{bot}} = 0$.

Function	**Error Source**	**Mean of Function at Error Source (a.u.)**	**Limiting Errors**	**Limiting Error Values**
Stream Function u	Interior	0.03048	$\mathtt{h}^2$	0.000025
Horizontal Velocity V_x	Interior	0.1249	$\mathtt{h}^2$	0.000025
Vertical Velocity V_y	Interior	0.10625	$\mathtt{h}^2$	0.000025
Vorticity w	Interior	1.97698	$\mathtt{t}$	0.00005
Vorticity w	Boundary	10.982	$\mathtt{h}$	0.005
Pressure p	Interior	6.57094	$\mathtt{h}^2$	0.000025
Pressure p	Boundary	6.648	$\mathtt{h}$	0.005

3.2.2 Bottom Wall Velocity $V_{\text{bot}} = 1$



(a) Velocity stream & pressure plot.

(b) Velocity quiver plot.

Figure 3.21: Velocity and pressure plots for Reynolds number 100 and $V_{\text{bot}} = 1$ at steady-state (in-simulation time of 25 seconds).

Fig. 3.21 shows the velocity and pressure plots. Again, like the $Re = 10$, $V_{\text{bot}} = 1$ case, two distinct, oppositely-rotating vortices are formed. This time however, due to the higher Reynolds number (flow is more turbulent as inertial forces dominate), the vortices are shifted to the right, as explained in the previous case of $Re = 100$, $V_{\text{bot}} = 0$. The pressure mechanism is the same as in the $Re = 10$, $V_{\text{bot}} = 1$ case, with the exception of lower pressures discussed in the previous case. Furthermore, as explained before, the symmetry about $y = 0.5$ is due to the dynamics of the system (in particular, equal moving wall velocities) rather than the Reynolds number. Raising or lowering the Reynolds number will not affect this symmetry.

Fig. 3.22 shows the stream function for the $Re = 100$, $V_{\text{bot}} = 1$ case. As expected, the stream function reflects the velocity profile. The negative stream function values (the depression in the 3D graph) is explained by the same mechanisms explained in the $Re = 10$, $V_{\text{bot}} = 1$ case, while the shifted stream function maximum/ minimum (read: vortices) is explained in the previous case of $Re = 100$, $V_{\text{bot}} = 0$.

The vorticity plots are seen in Fig. 3.23. The corner peaks & troughs at the corners can be seen, and behave in the same way as the $Re = 10$, $V_{\text{bot}} = 1$ case (with opposite vorticity signs at the corner). The $x = 0.5$ axis symmetry is lost, but the $y = 0.5$ symmetry remains (as this symmetry is dependent on the wall velocities). The vorticity at the right wall increases as it goes towards the centre, while on the left wall it remains more or less constant throughout. This asymmetry is again due to the turbulent flow arising from a higher Reynolds number, and its physical mechanism has been explained in the previous cases.

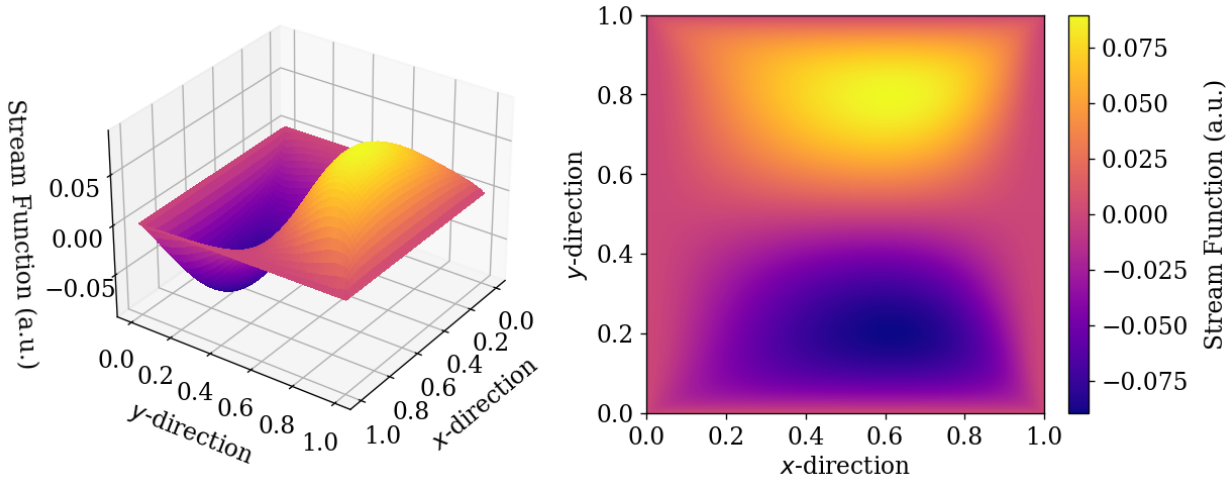


Figure 3.22: 3D and heatmap plots of stream function for Reynolds number 100 and $V_{\text{bot}} = 1$.

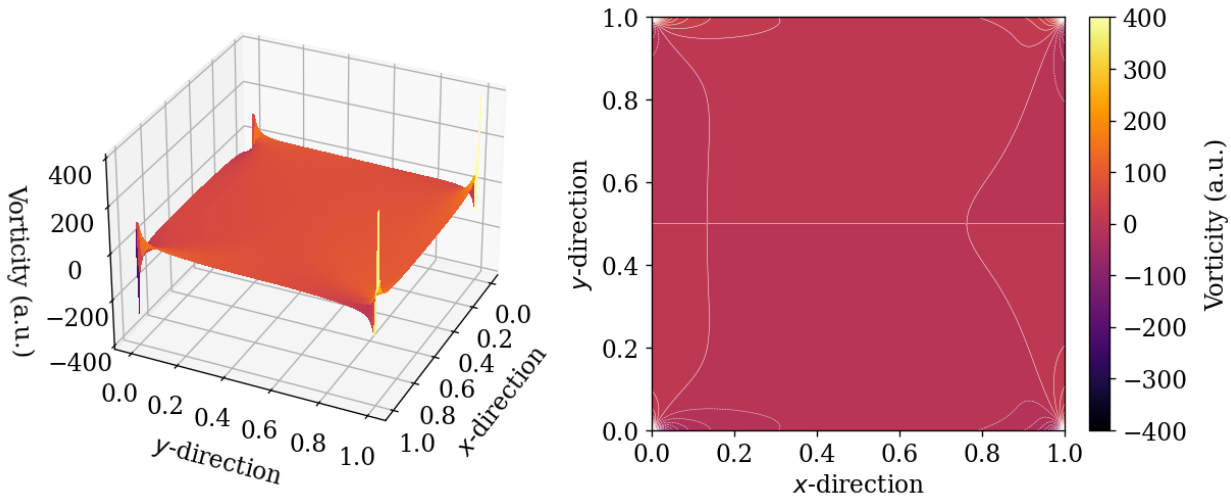


Figure 3.23: 3D and contour plots of vorticity for Reynolds number 100 and $V_{\text{bot}} = 1$.

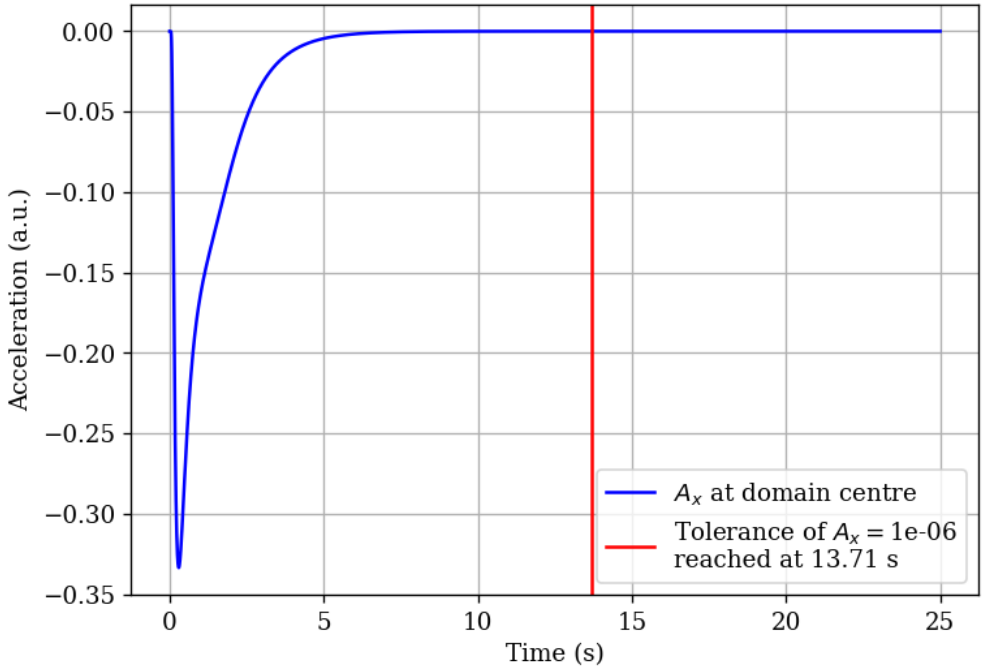


Figure 3.24: Horizontal acceleration A_x (change of V_x with respect to time) at domain centre (100, 100) showing time when tolerance is reached for Reynolds number 100 and $V_{\text{bot}} = 1$.

The acceleration is again critically damped, but the steady-state reaching time of 13.71 seconds is a few seconds lower than the previous case. This makes sense, as explained before, the vortices meeting in the centre enhance the velocity contributions, which results in the centre horizontal velocity stabilising quicker.

Error Analysis for Reynolds Number 100 and $V_{\mathbf{bot}} = 1$

Fig. 3.25 shows the residuals for $V_{\text{bot}} = 1$ and $Re = 100$. The residuals imply that the simulation is tightly converging.

Table 3.5 shows the errors, error sources, and means of the different functions. The conclusions of the previous cases also apply here - the simulation converges, and the least precise function has a precision of up to three significant figures, with all others having four or more.

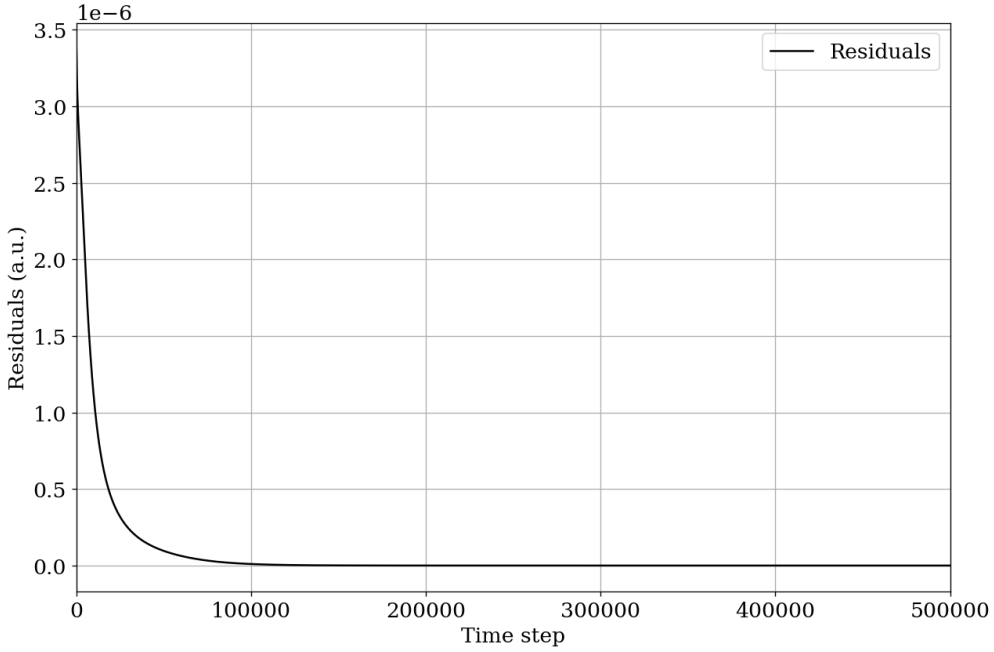
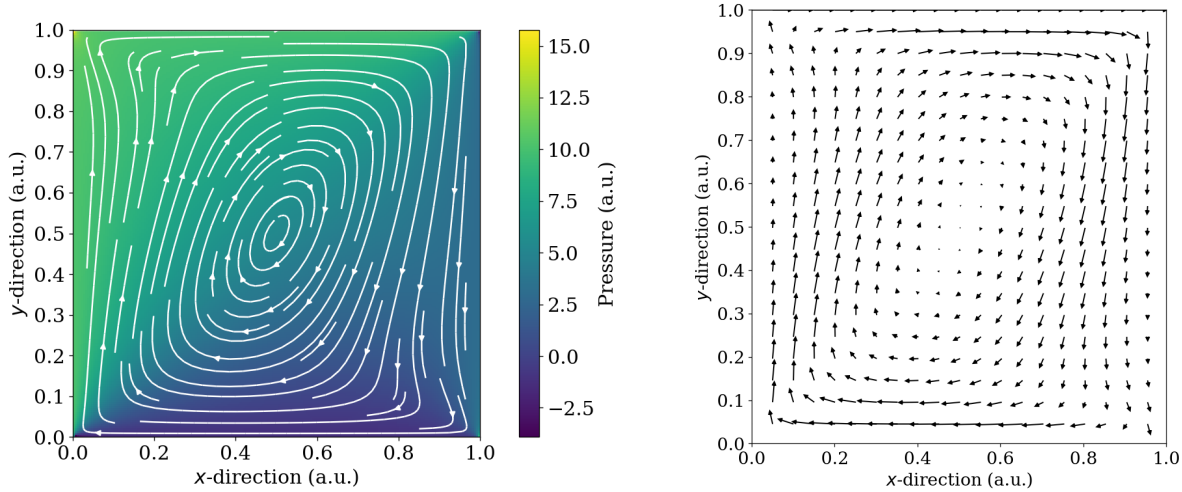


Figure 3.25: Residuals of stream function over all time steps for Reynolds number 100 and $V_{\text{bot}} = 1$.

Table 3.5: Errors, error sources, and mean values of different functions for Reynolds number 100 and $V_{\text{bot}} = 1$.

Function	**Error Source**	**Mean of Function at Error Source (a.u.)**	**Limiting Errors**	**Limiting Error Values**
Stream Function u	Interior	0.03678	$\mathtt{h}^2$	0.000025
Horizontal Velocity V_x	Interior	0.23037	$\mathtt{h}^2$	0.000025
Vertical Velocity V_y	Interior	0.11753	$\mathtt{h}^2$	0.000025
Vorticity w	Interior	3.49239	$\mathtt{t}$	0.00005
Vorticity w	Boundary	20.058	$\mathtt{h}$	0.005
Pressure p	Interior	7.96514	$\mathtt{h}^2$	0.000025
Pressure p	Boundary	8.108	$\mathtt{h}$	0.005

3.2.3 Bottom Wall Velocity $V_{\text{bot}} = -1$



(a) Velocity stream & pressure plot.

(b) Velocity quiver plot.

Figure 3.26: Velocity and pressure plots for Reynolds number 100 and $V_{\text{bot}} = -1$ at steady-state (in-simulation time of 25 seconds).

Fig. 3.26 shows the pressure and velocity plots. Similar to the $Re = 10$, $V_{\text{bot}} = -1$ case, the two rotating vortices of equal rotation direction merge to form a big loop. A major difference here however, is that there is only one vortex in the loop, as opposed to two in the $Re = 10$, $V_{\text{bot}} = -1$ case. The dominant viscous forces in the $Re = 10$ case keep the individual vortices from collapsing into a large vortex. Here however, the inertial forces overpower these viscous forces, and the two vortices merge to not only form a loop around the cavity, but in fact a vortex which flows around the entire cavity. Furthermore, the vortices themselves would be shifted - the upper vortex to the right, and the lower one to the left. This creates a very energetic, unstable system. Thus, to minimise the energy, these vortices collapse to form one. The pressure contributions have the same physical mechanisms as explained in previous cases.

The stream function plots can be seen in Fig. 3.27. The stream function reflects the velocity profile and has a higher maximum than the $Re = 10$ case, as explained before. The vorticity plots can be seen in Fig. 3.28. The vorticity has an increase around the regions of (x, y)=(0.8, 0.8) and (0.2, 0.2). The profile of these increases resemble that of the $V_{\text{bot}} = 0$ case for $Re = 100$ (see Fig. 3.18). This makes sense, as these areas of vorticity increase are actually the areas where the vortex centres would be, if they had not merged to create one large vortex. The reasoning behind these increases then are the same as explained in the $Re = 100$, $V_{\text{bot}} = 0$ case, but just mirrored for an additional moving wall at the bottom (which moves with an opposite velocity).

The acceleration and steady-state reaching time of 12.65 seconds is shown in Fig. 3.29. Just like the acceleration for the $Re = 10$, $V_{\text{bot}} = -1$ case, the acceleration here is also underdamped, actually by a greater degree. The physical mechanism behind this is the same as in the $Re = 10$, $V_{\text{bot}} = -1$ case. Interestingly, the steady-state reaching time is *less* than both the $V_{\text{bot}} = 0$ and $V_{\text{bot}} = 1$

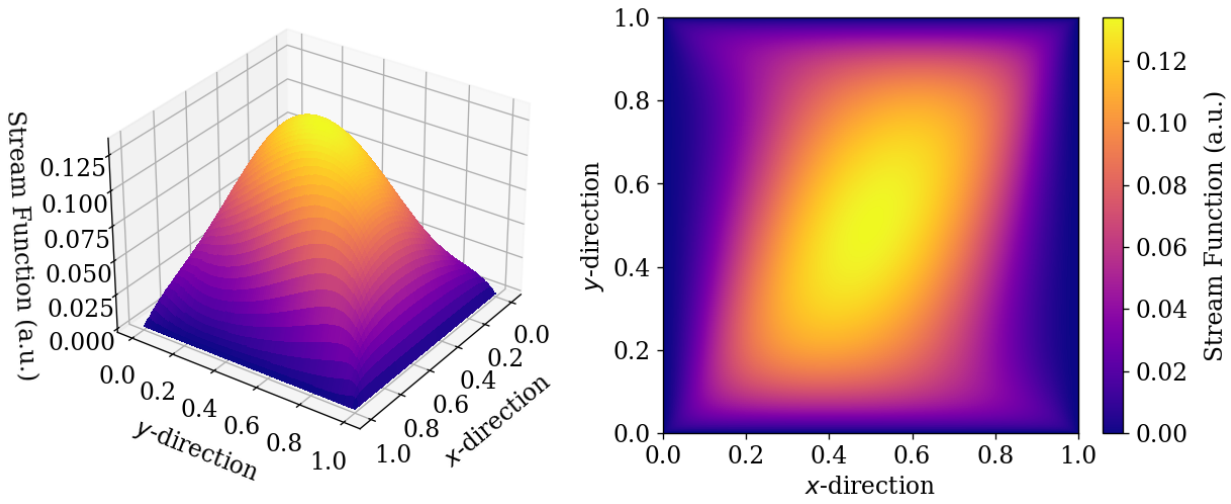


Figure 3.27: 3D and heatmap plots of stream function for Reynolds number 100 and $V_{\text{bot}} = -1$.

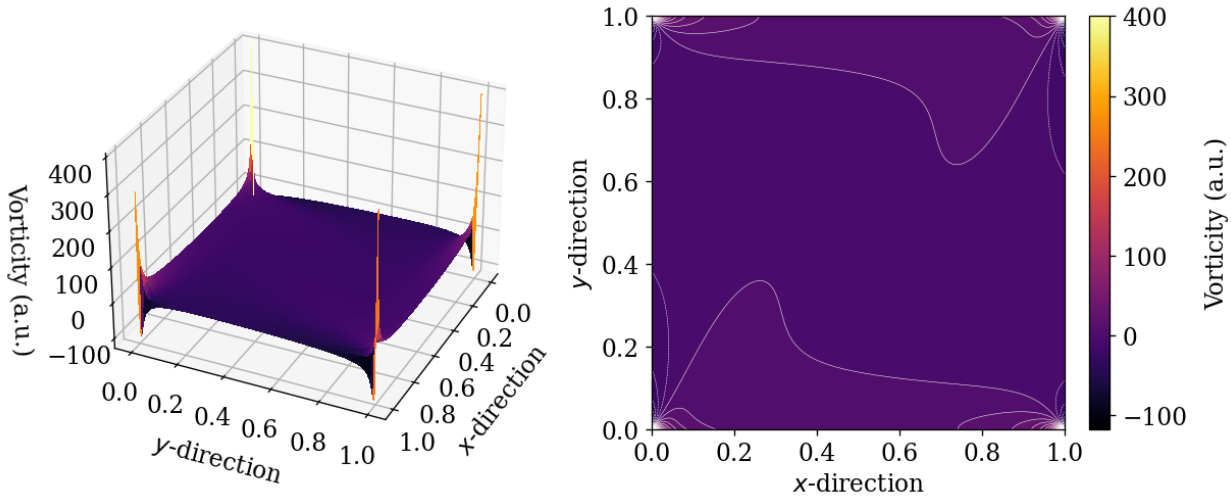


Figure 3.28: 3D and contour plots of vorticity for Reynolds number 100 and $V_{\text{bot}} = -1$.

cases, whereas for $Re = 10$, it was *more* than the two cases. This can maybe be explained by the collapsing vortices. Since the two vortices combine to form one, as opposed to the $Re = 10$ case wherein the two vortices remained thanks to the strong viscous forces, the system was allowed to stabilise a lot faster, as there was only one large flow around the entire cavity. Furthermore, since there were equal contributions to the central vortex from the top and bottom walls, this also enhanced the stability of said vortex.

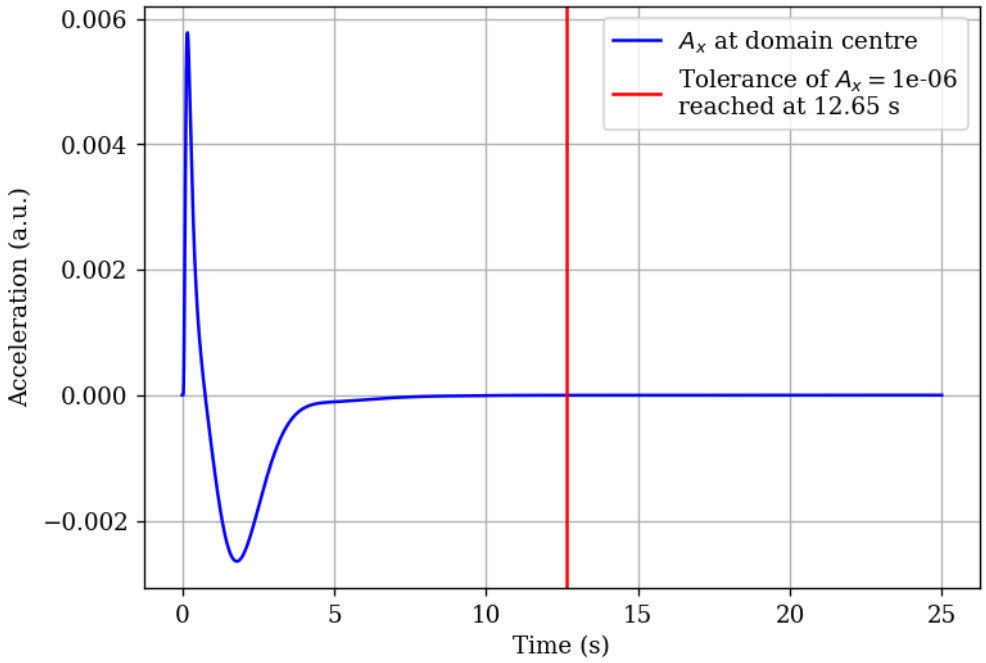


Figure 3.29: Horizontal acceleration A_x (change of V_x with respect to time) at domain centre (100, 100) showing time when tolerance is reached for Reynolds number 100 and $V_{\text{bot}} = -1$.

Error Analysis for Reynolds Number 100 and $V_{\mathbf{bot}} = -1$

Fig. 3.30 shows the residuals for $V_{\text{bot}} = -1$ and $Re = 100$. The residuals imply that the simulation is tightly converging.

Table 3.6 shows the errors, error sources, and means of the different functions. The conclusions of the previous cases also apply here - the simulation converges, and the least precise function has a precision of up to three significant figures, with all others having four or more.

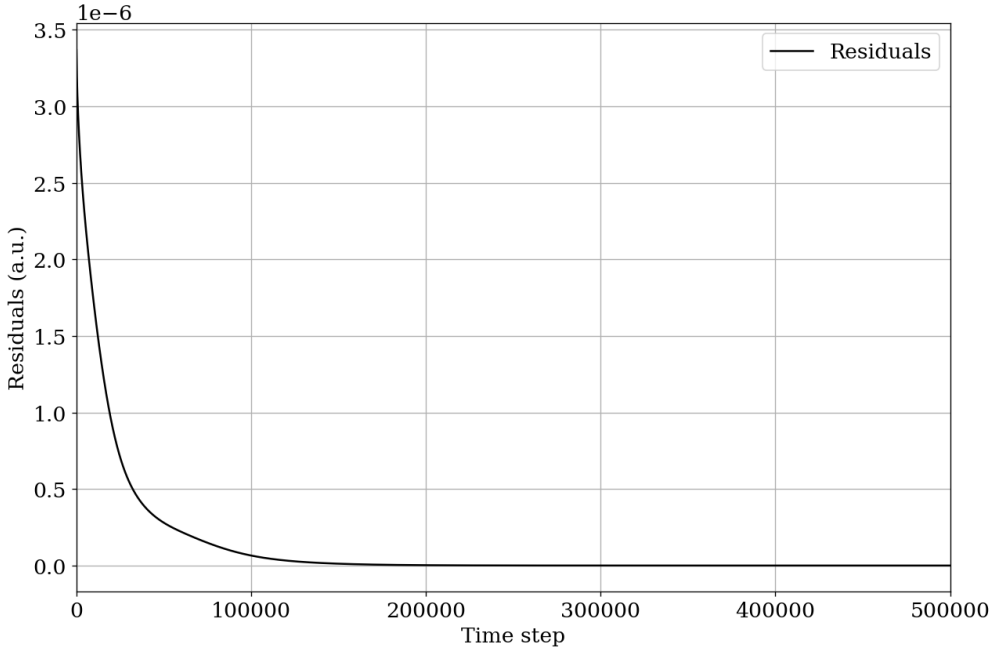


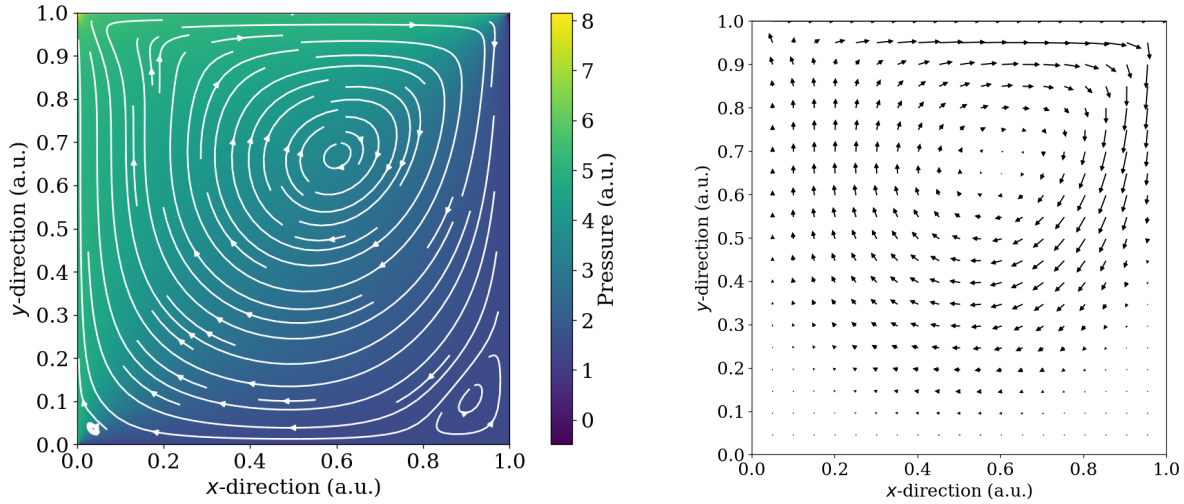
Figure 3.30: Residuals of stream function over all time steps for Reynolds number 100 and $V_{\text{bot}} = -1$.

Table 3.6: Errors, error sources, and mean values of different functions for Reynolds number 100 and $V_{\text{bot}} = -1$.

Function	**Error Source**	**Mean of Function at Error Source (a.u.)**	**Limiting Errors**	**Limiting Error Values**
Stream Function u	Interior	0.0619	$\mathtt{h}^2$	0.000025
Horizontal Velocity V_x	Interior	0.17857	$\mathtt{h}^2$	0.000025
Vertical Velocity V_y	Interior	0.20614	$\mathtt{h}^2$	0.000025
Vorticity w	Interior	3.68645	$\mathtt{t}$	0.00005
Vorticity w	Boundary	21.051	$\mathtt{h}$	0.005
Pressure p	Interior	5.49729	$\mathtt{h}^2$	0.000025
Pressure p	Boundary	6.214	$\mathtt{h}$	0.005

3.3 Reynolds Number: 200

3.3.1 Bottom Wall Velocity $V_{\mathbf{bot}} = 0$



(a) Velocity stream & pressure plot.

(b) Velocity quiver plot.

Figure 3.31: Velocity and pressure plots for Reynolds number 200 and $V_{\text{bot}} = 0$ at steady-state (in-simulation time of 45 seconds).

Fig. 3.31 shows the velocity and pressure plots for $Re = 200$, $V_{\text{bot}} = 0$. These plots are very similar to the case of $Re = 100$, $V_{\text{bot}} = 0$, but with the turbulent flow effects amplified. Firstly, the vortex centre has shifted slightly further to the right, and even deeper into the cavity, with the same physical mechanisms as explained in the aforementioned case. The eddy in the bottom right corner has also gotten bigger, as the inertial forces in this case are even stronger. The pressure gradient has not changed much, except in magnitude, where the pressures are now even lower (again, as explained by the same physical mechanisms explained in previous cases).

The stream function plots can be seen in Fig. 3.32. The stream function reflects the velocity profile, as expected. There is an even larger shift towards the upper right corner, as a result of the increased turbulent behaviour. All the physical mechanisms at play here are the same as those for $Re = 100$, $V_{\text{bot}} = 0$, but amplified. The vorticity function can be seen in Fig. 3.33. While the peak/ trough magnitudes are the same (due to the constant moving wall velocity) as the previous cases, there is now even more chaos in the upper right corner, and in particular, the upper right wall. This increased region of vorticity is due to changes in V_x as y decreases.

The acceleration and steady-state reaching time can be seen in Fig. 3.34. The acceleration here is just under critical damping, but also exhibits intermediate oscillatory behaviour during the damping - this is due to chaos arising from the turbulent flow. The critical damping feature is explained as in the previous cases. The steady-state reaching time of 34.18 seconds is to be expected - the more turbulent the flow, the less predictable it is, and hence the longer it takes to stabilise.

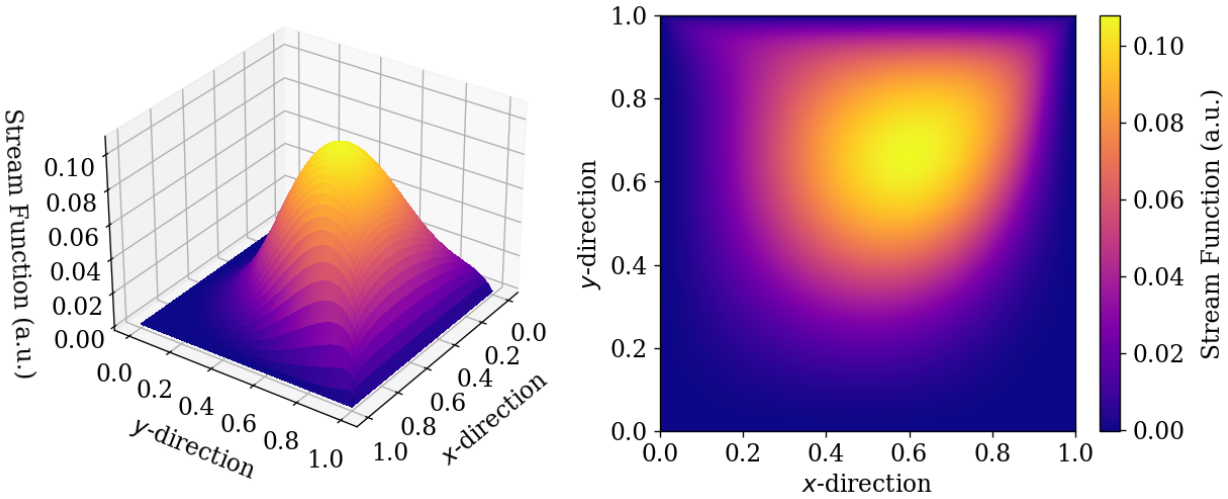


Figure 3.32: 3D and heatmap plots of stream function for Reynolds number 200 and $V_{\text{bot}} = 0$.

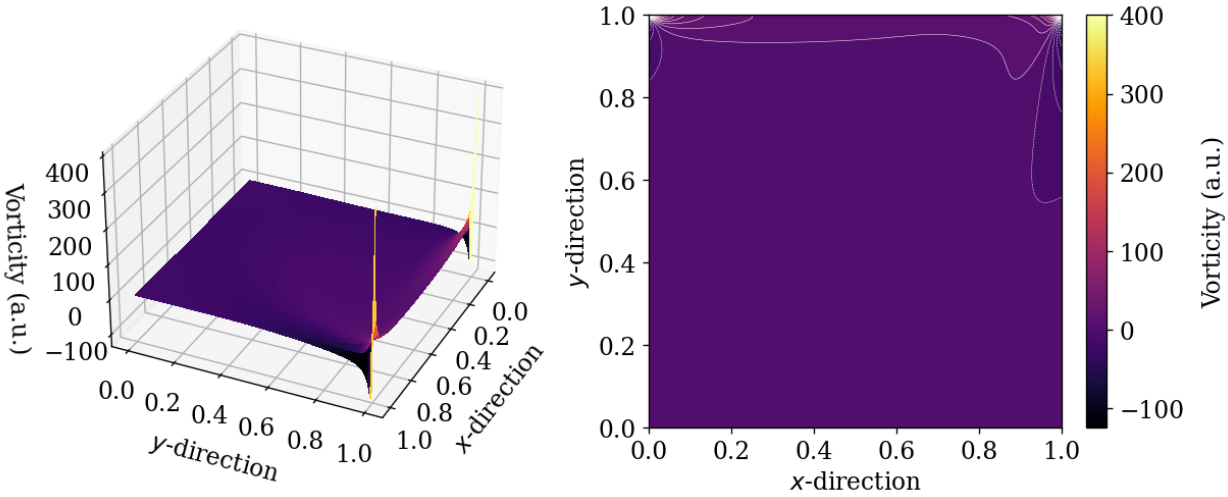


Figure 3.33: 3D and contour plots of vorticity for Reynolds number 200 and $V_{\text{bot}} = 0$.

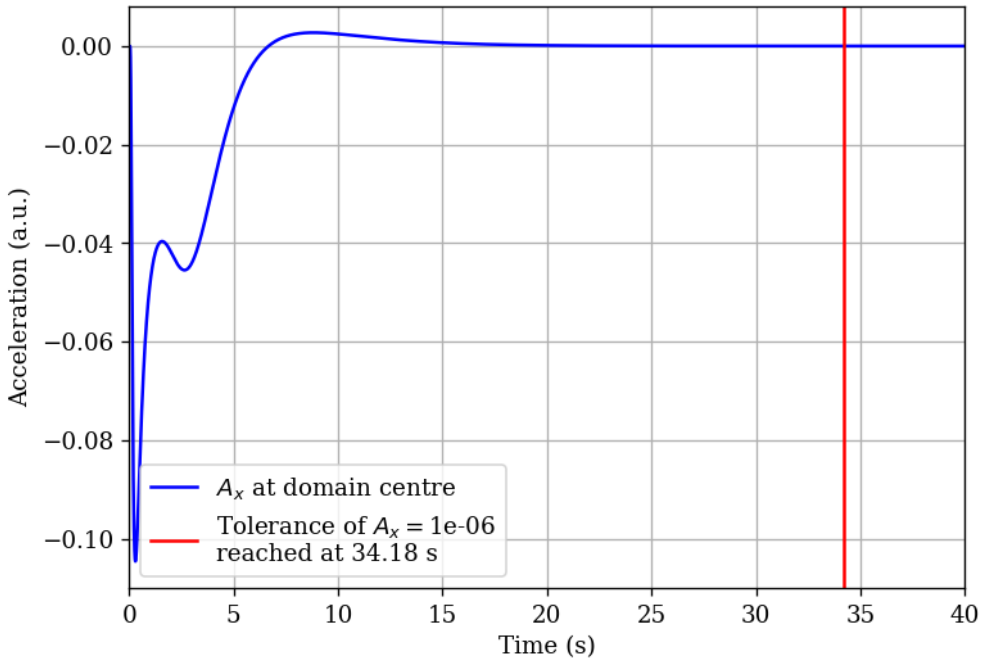


Figure 3.34: Horizontal acceleration A_x (change of V_x with respect to time) at domain centre (100, 100) showing time when tolerance is reached for Reynolds number 200 and $V_{\text{bot}} = 0$.

Error Analysis for Reynolds Number 200 and $V_{\mathbf{bot}} = 0$

Fig. 3.35 shows the residuals for $V_{\text{bot}} = 0$ and $Re = 200$. The residuals imply that the simulation is tightly converging.

Table 3.7 shows the errors, error sources, and means of the different functions. The conclusions of the previous cases also apply here - the simulation converges, and the least precise function has a precision of up to three significant figures, with all others having four or more.

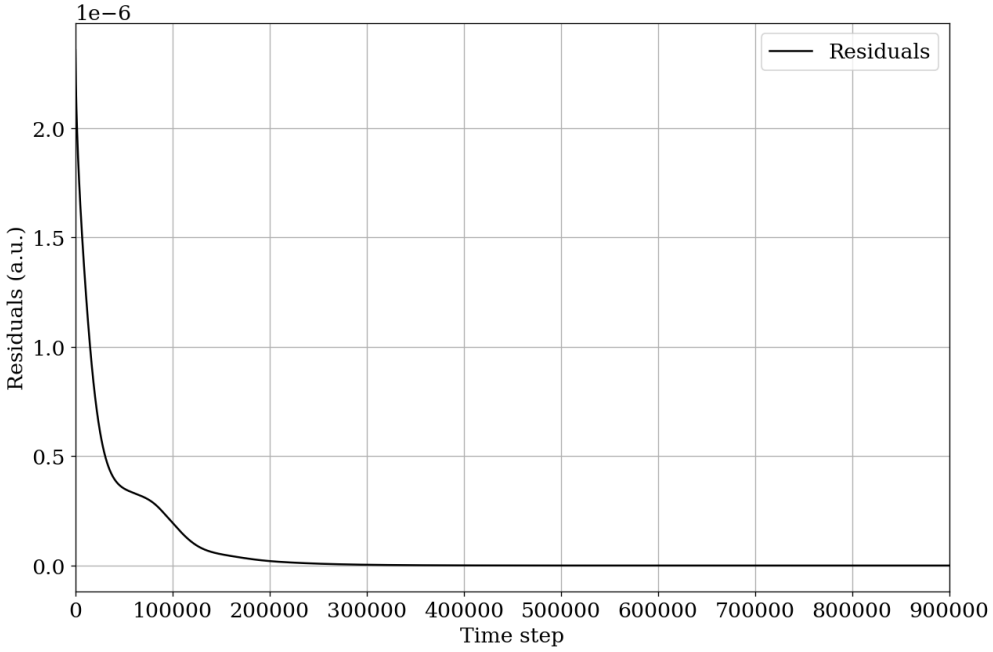
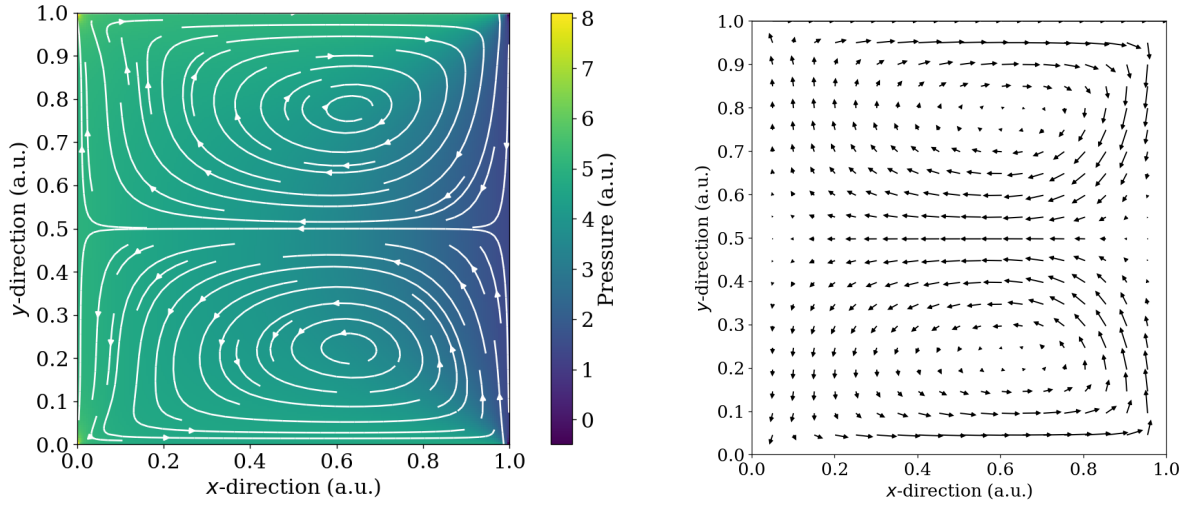


Figure 3.35: Residuals of stream function over all time steps for Reynolds number 200 and $V_{\text{bot}} = 0$.

Table 3.7: Errors, error sources, and mean values of different functions for Reynolds number 200 and $V_{\text{bot}} = 0$.

Function	**Error Source**	**Mean of Function at Error Source (a.u.)**	**Limiting Errors**	**Limiting Error Values**
Stream Function u	Interior	0.03435	$\mathtt{h}^2$	0.000025
Horizontal Velocity V_x	Interior	0.12976	$\mathtt{h}^2$	0.000025
Vertical Velocity V_y	Interior	0.12057	$\mathtt{h}^2$	0.000025
Vorticity w	Interior	2.16146	$\mathtt{t}$	0.00005
Vorticity w	Boundary	11.932	$\mathtt{h}$	0.005
Pressure p	Interior	3.19196	$\mathtt{h}^2$	0.000025
Pressure p	Boundary	3.277	$\mathtt{h}$	0.005

3.3.2 Bottom Wall Velocity $V_{\text{bot}} = 1$



(a) Velocity stream & pressure plot.

(b) Velocity quiver plot.

Figure 3.36: Velocity and pressure plots for Reynolds number 200 and $V_{\text{bot}} = 1$ at steady-state (in-simulation time of 45 seconds).

Fig. 3.36 shows the velocity and pressure plots. As expected, the vortices are shifted to the right, but there is still a $y = 0.5$ symmetry. The velocity profile closely resembles that of the $Re = 100$, $V_{\text{bot}} = 1$ case, with slightly more turbulence resulting in deeper and more shifted vortices. The pressure magnitudes are again lower than their $Re = 100$ counterpart, as explained in the previous cases.

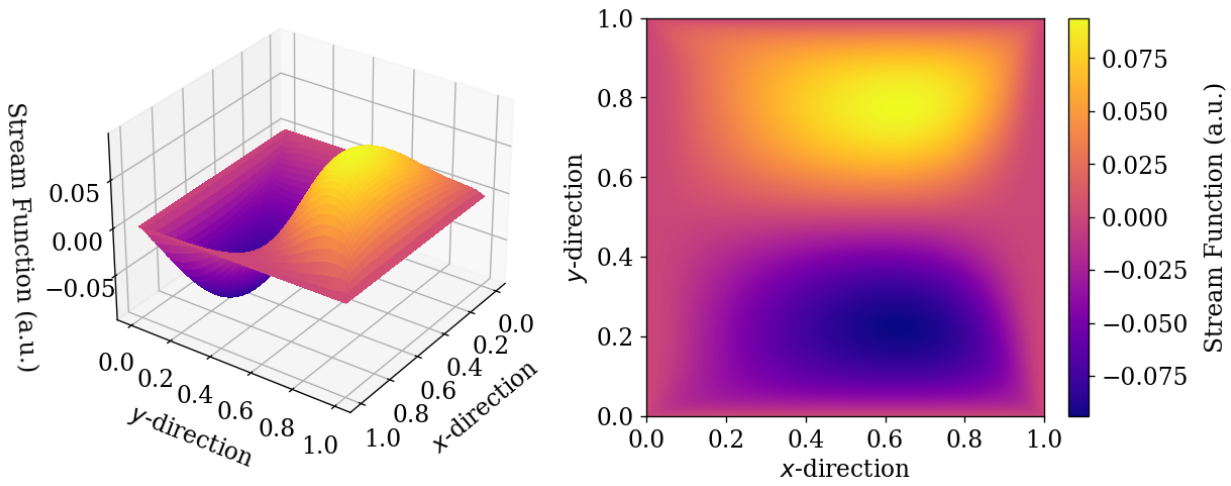


Figure 3.37: 3D and heatmap plots of stream function for Reynolds number 200 and $V_{\text{bot}} = 1$.

Fig. 3.37 shows the stream function plots. All the physical mechanisms here are the same as explained in the $V_{\text{bot}} = 1$ case for $Re = 10$ and $Re = 100$. Of course, the stream function more closely resembles that of the $Re = 100$ case, as both $Re = 100$ and $Re = 200$ imply much more turbulent flow than the $Re = 10$ case. The stream function reflects the velocity profile, and as such, is more shifted than the $Re = 100$ case due to the higher turbulence. The vorticity profile also closely resembles that of the $Re = 100$, $V_{\text{bot}} = 1$ case, but with more exaggerated features due to the higher turbulence. The physical mechanisms are the same as those explained in said case.

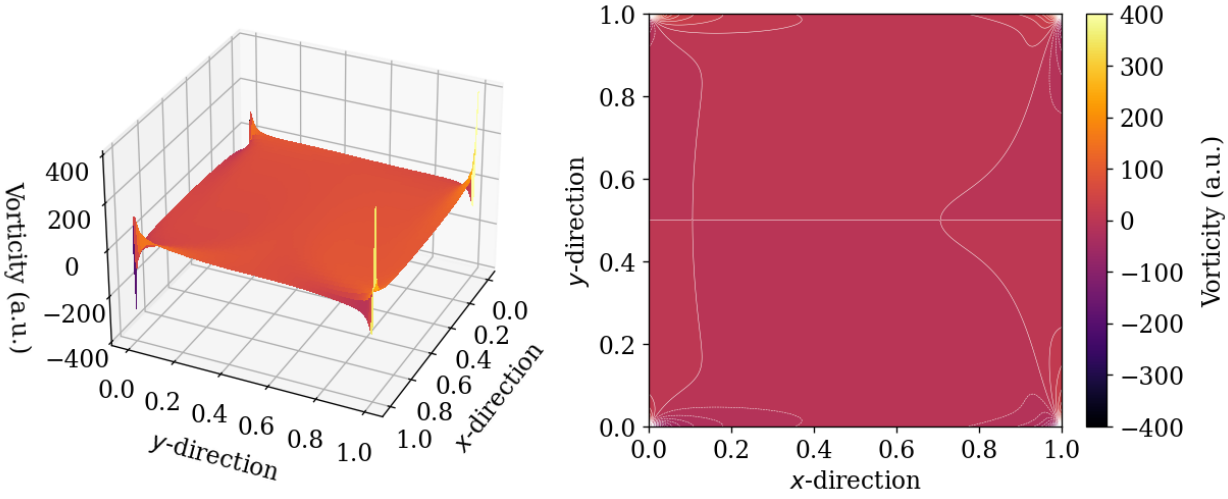


Figure 3.38: 3D and contour plots of vorticity for Reynolds number 200 and $V_{\text{bot}} = 1$.

The acceleration and steady-state reaching time is seen in 3.39. The acceleration is critically damped (see case $Re = 10$, $V_{\text{bot}} = 1$ for a physical explanation), and during the damping there are intermediate oscillations, indicative of chaotic behaviour in the system. The steady-state reaching time is as expected for the two-vortex system, being much lower than the $V_{\text{bot}} = 0$ case, due to the enhanced velocity contributions from the two vortices.

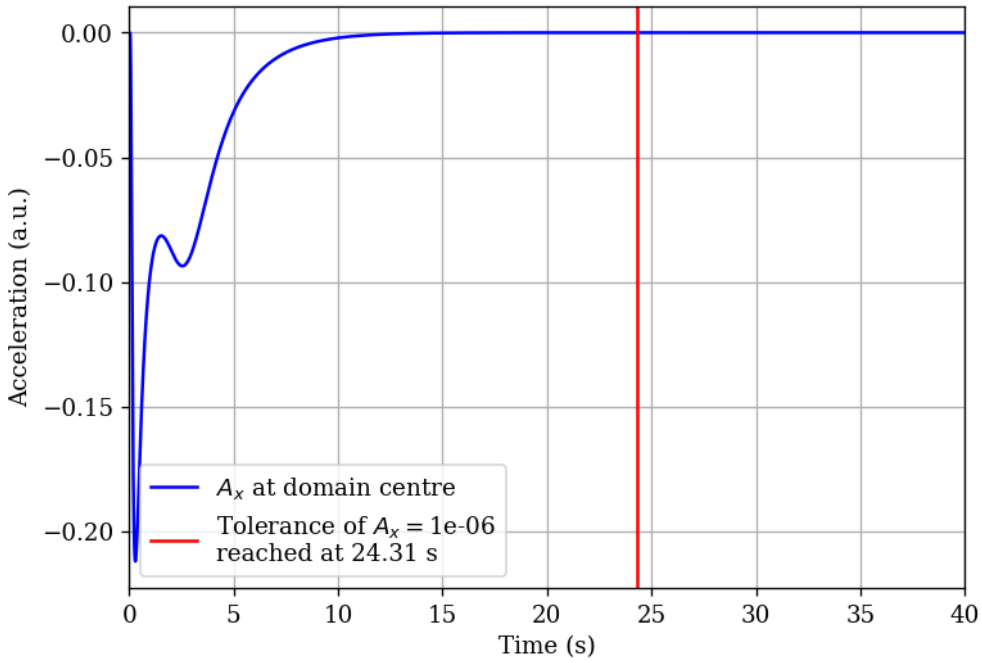


Figure 3.39: Horizontal acceleration A_x (change of V_x with respect to time) at domain centre (100, 100) showing time when tolerance is reached for Reynolds number 200 and $V_{\text{bot}} = 1$.

Error Analysis for Reynolds Number 200 and $V_{\mathbf{bot}} = 1$

Fig. 3.40 shows the residuals for $V_{\text{bot}} = 1$ and $Re = 200$. The residuals imply that the simulation is tightly converging.

Table 3.8 shows the errors, error sources, and means of the different functions. The conclusions of the previous cases also apply here - the simulation converges, and the least precise function has a precision of up to three significant figures, with all others having four or more.

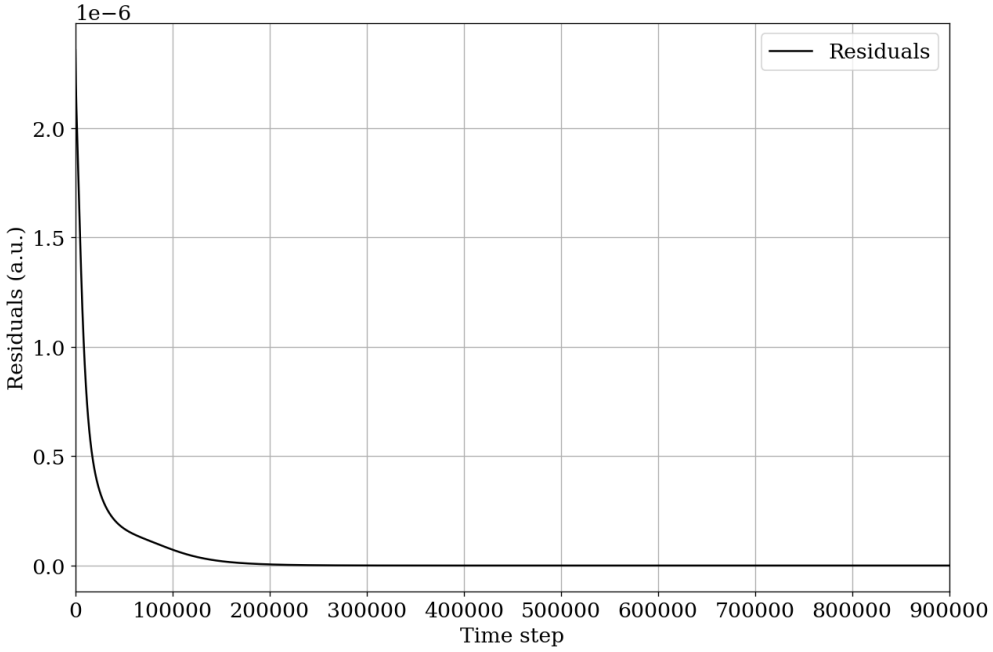
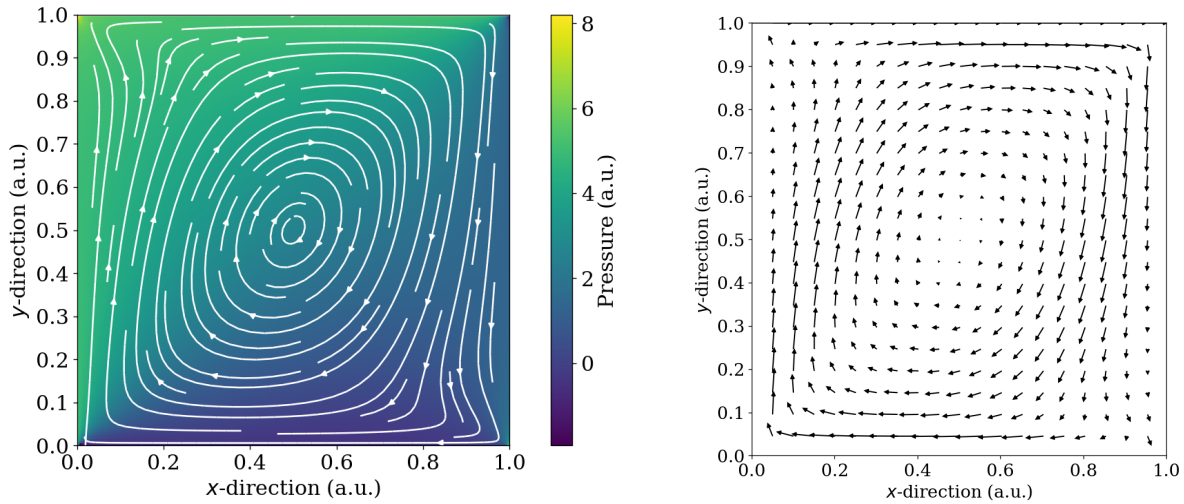


Figure 3.40: Residuals of stream function over all time steps for Reynolds number 200 and $V_{\text{bot}} = 1$.

Table 3.8: Errors, error sources, and mean values of different functions for Reynolds number 200 and $V_{\text{bot}} = 1$.

Function	**Error Source**	**Mean of Function at Error Source (a.u.)**	**Limiting Errors**	**Limiting Error Values**
Stream Function u	Interior	0.03863	$\mathtt{h}^2$	0.000025
Horizontal Velocity V_x	Interior	0.2351	$\mathtt{h}^2$	0.000025
Vertical Velocity V_y	Interior	0.1251	$\mathtt{h}^2$	0.000025
Vorticity w	Interior	3.6537	$\mathtt{t}$	0.00005
Vorticity w	Boundary	21.355	$\mathtt{h}$	0.005
Pressure p	Interior	3.92748	$\mathtt{h}^2$	0.000025
Pressure p	Boundary	4.095	$\mathtt{h}$	0.005

3.3.3 Bottom Wall Velocity $V_{\text{bot}} = -1$



(a) Velocity stream & pressure plot.

(b) Velocity quiver plot.

Figure 3.41: Velocity and pressure plots for Reynolds number 200 and $V_{\text{bot}} = -1$ at steady-state (in-simulation time of 45 seconds).

Fig. 3.41 shows the pressure and velocity plots. The same flow as the $Re = 100$ case occurs, only more exaggerated, with the exception of the upper left and lower right corners. Unlike the $Re = 100$ case, it can be seen that the flow bends more violently at these corners. This is the beginning of the formation of eddies. Increasing the Reynolds numbers, will result in such a formation.

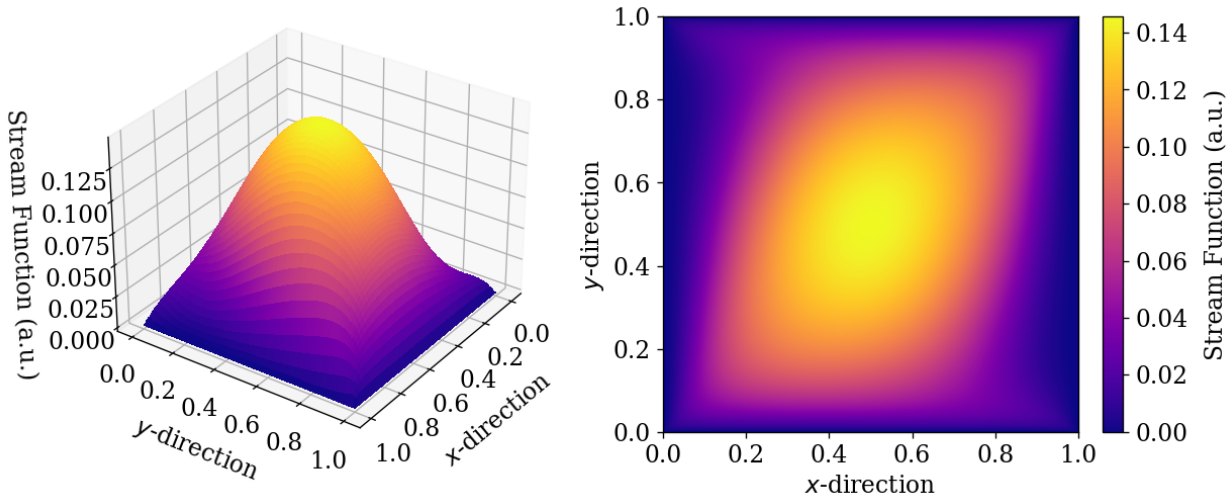


Figure 3.42: 3D and heatmap plots of stream function for Reynolds number 200 and $V_{\text{bot}} = -1$.

The stream function plots can be seen in Fig. 3.42. The stream function reflects the velocity profile,

which is as expected. Beyond a greater shift to the upper right and lower left corners, there is not much of a difference between this and the $Re = 100$ case (see Fig. 3.27). The physical mechanisms are also the same as explained in said case. The vorticity can be seen in Fig. 3.43. The vorticity is as expected, with a mirroring of the $Re = 200$, $V_{\text{bot}} = 0$ vorticity at the bottom wall, as explained in the $Re = 100$, $V_{\text{bot}} = -1$ case.

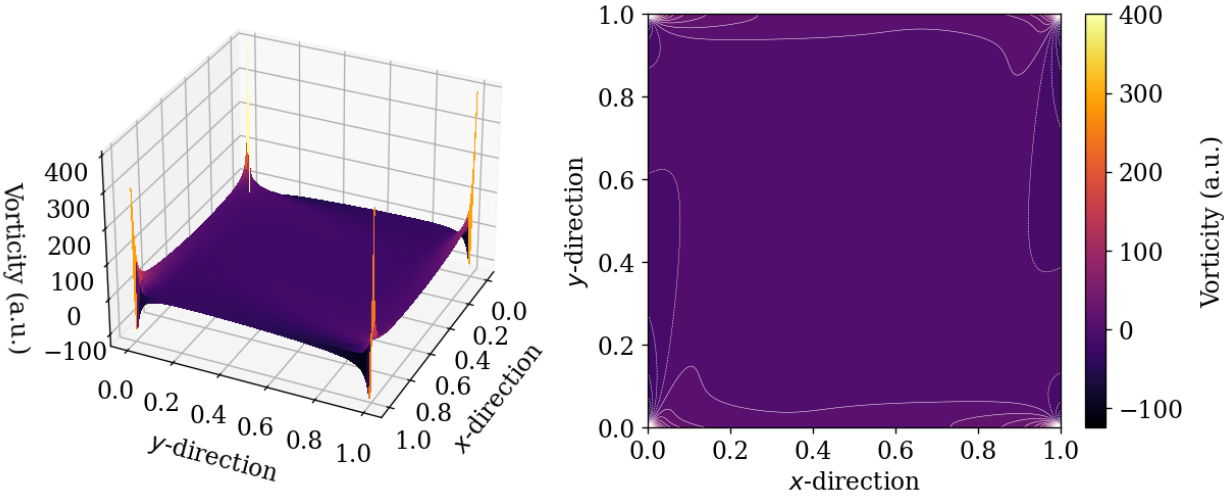


Figure 3.43: 3D and contour plots of vorticity for Reynolds number 200 and $V_{\text{bot}} = -1$.

The acceleration and steady-state reaching time of 22.26 seconds is shown in Fig. 3.44. Similar to the $Re = 100$ case, $V_{\text{bot}} = -1$ results in the shortest steady-state reaching time. Furthermore, this acceleration is similarly underdamped, with intermediate chaotic behaviour (for example, at around 7 seconds).

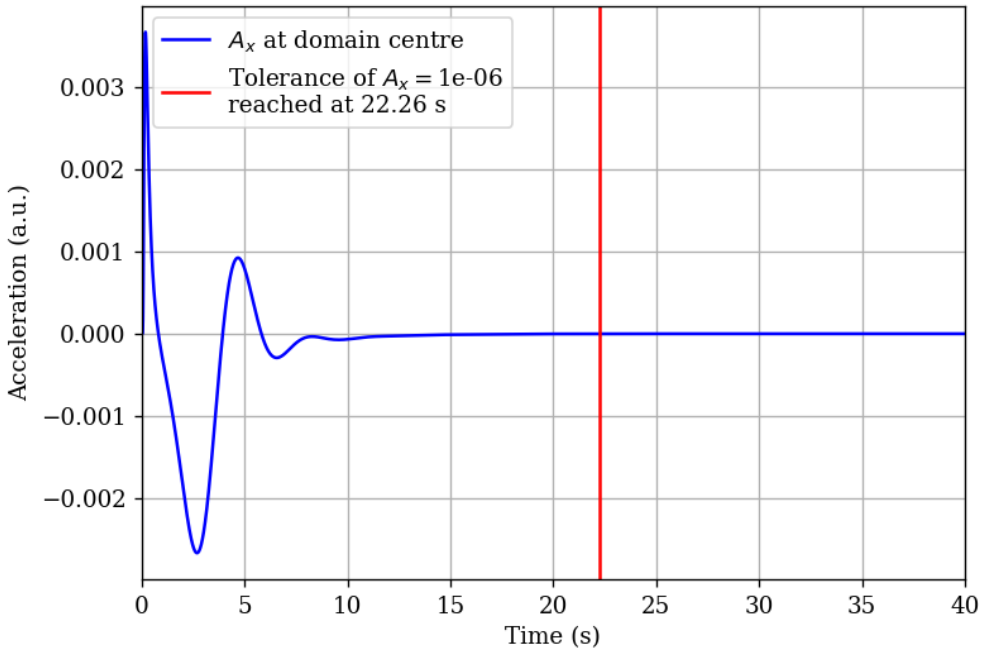


Figure 3.44: Horizontal acceleration A_x (change of V_x with respect to time) at domain centre (100, 100) showing time when tolerance is reached for Reynolds number 200 and $V_{\text{bot}} = -1$.

Error Analysis for Reynolds Number 200 and $V_{\mathbf{bot}} = -1$

Fig. 3.45 shows the residuals for $V_{\text{bot}} = -1$ and $Re = 200$. The residuals imply that the simulation is tightly converging.

Table 3.9 shows the errors, error sources, and means of the different functions. The conclusions of the previous cases also apply here - the simulation converges, and the least precise function has a precision of up to three significant figures, with all others having four or more.

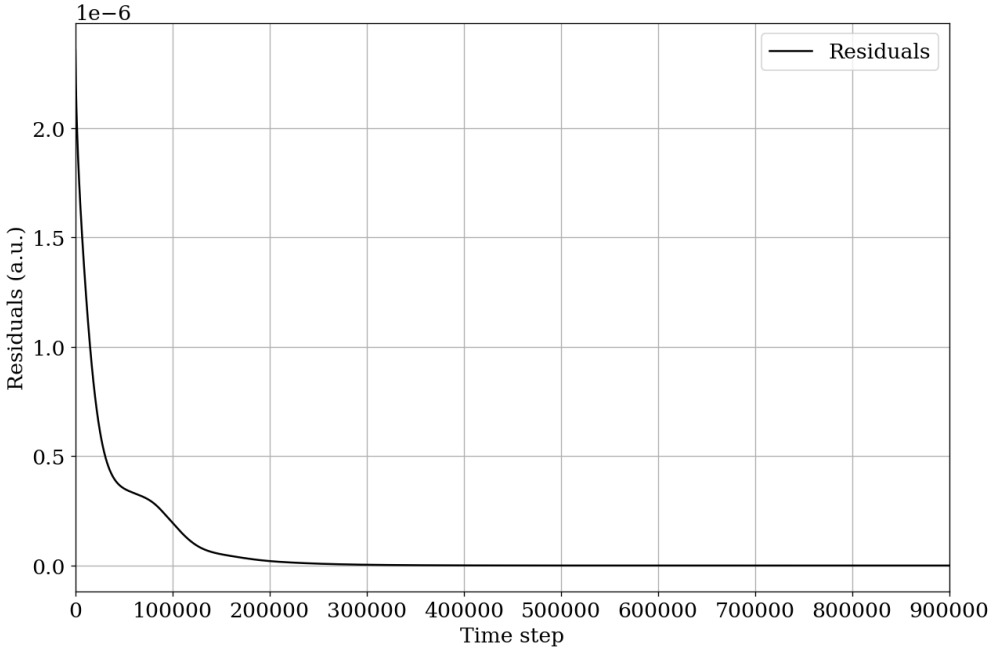


Figure 3.45: Residuals of stream function over all time steps for Reynolds number 200 and $V_{\text{bot}} = -1$.

Table 3.9: Errors, error sources, and mean values of different functions for Reynolds number 200 and $V_{\text{bot}} = -1$.

Function	**Error Source**	**Mean of Function at Error Source (a.u.)**	**Limiting Errors**	**Limiting Error Values**
Stream Function u	Interior	0.06524	$\mathtt{h}^2$	0.000025
Horizontal Velocity V_x	Interior	0.19155	$\mathtt{h}^2$	0.000025
Vertical Velocity V_y	Interior	0.21247	$\mathtt{h}^2$	0.000025
Vorticity w	Interior	3.84515	`t`	0.00005
Vorticity w	Boundary	22.955	`h`	0.005
Pressure p	Interior	2.70602	$\mathtt{h}^2$	0.000025
Pressure p	Boundary	3.049	`h`	0.005

3.4 General Discussion

3.4.1 Changing Reynolds Numbers

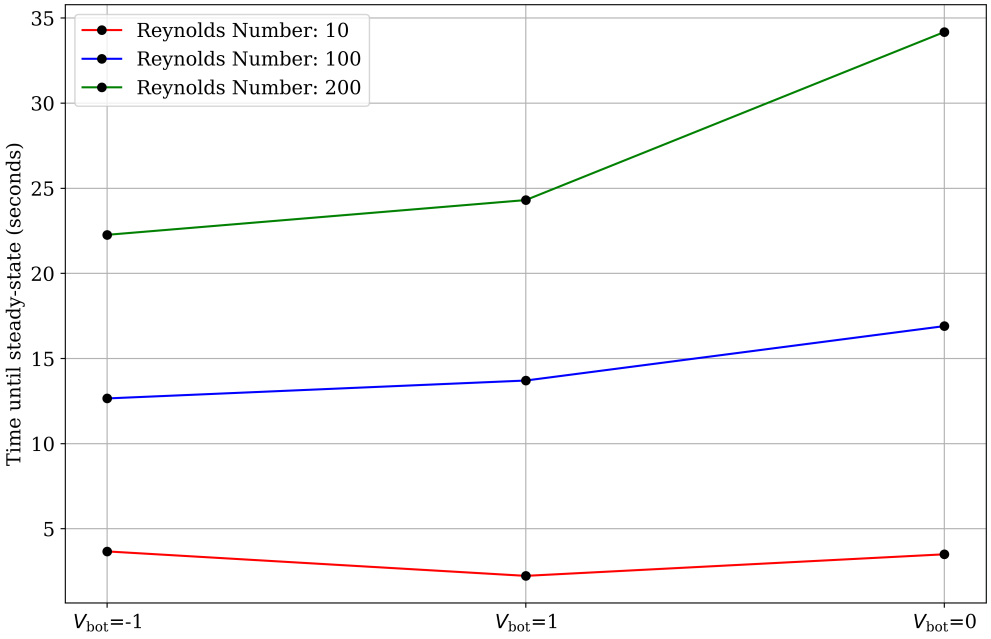


Figure 3.46: Reynolds numbers as compared to the steady-state reaching time for different bottom wall velocities (V_{bot} vs. steady-state reaching time).

The Figures 3.46 and 3.47 show how the Reynolds numbers compare to the steady-state reaching times. It is abundantly clear that as the Reynolds number is increased, so does the steady-state reaching time. This is explained quite easily. When the Reynolds number is low, the viscous (resistant) forces of the fluid dominate, meaning that the inertial forces are not as easily spread and dissipated throughout fluid. In this way, the individual particles/ fluid grid cells reach an equilibrium between the inertial forces and viscous forces a lot faster (even if the inertial force from the driving lid is strong, the viscous forces balance it out very quickly within the fluid), and as such the fluid reaches an equilibrium faster as well. This quick-reaching equilibrium results in a short steady-state reaching time.

A higher Reynolds number means that the inertial forces are the main actors in the fluid motion. This means that the inertial forces are easily spread, but also easily dissipated (if all the inertial force is spread throughout the top right of the lid, the rest of the cavity will not experience as strong as a force). This results in one portion of the fluid cavity having stronger (inertial) forces applied to it, while another portion having much weaker forces applied to it, consequently resulting in a flow velocity heterogeneity as well. This imbalance means it takes the system as a whole longer to stabilise, and thus longer to reach steady-state reaching times.

It is also seen that for higher Reynolds number fluids, the steady-state reaching times increase with

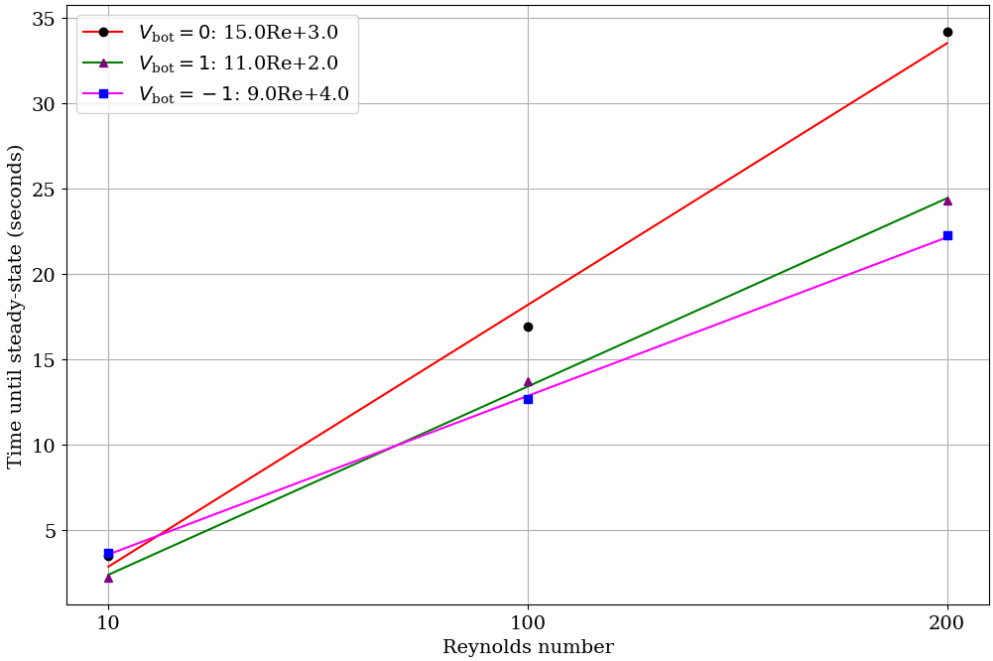


Figure 3.47: Reynolds numbers as compared to the steady-state reaching time for different bottom wall velocities (Re vs. steady-state reaching time).

$V_{\text{bot}} = -1$, $V_{\text{bot}} = 1$, and $V_{\text{bot}} = 0$. While for $Re = 10$, the lowest steady-state reaching time occurs at $V_{\text{bot}} = 1$, increasing then with $V_{\text{bot}} = 0$, and $V_{\text{bot}} = -1$. The reason for this discrepancy in $V_{\text{bot}} = -1$ is due to the presence or absence of two vortices within the fluid. For $Re = 10$, there were two vortices in the fluid, which did not collapse into one due to the resistant viscous forces. However, for $Re = 100$ and $Re = 200$, these vortices did collapse into one, resulting in one large vortex. A large vortex, being driven by two opposite walls with opposite and equal velocities, will reach equilibrium quickly as compared to other wall velocity configurations, as the centre of the vortex is exactly at the centre of the cavity. It might also be the case that since the velocity probe is also at the centre, the information gathered by the probe is biased, but this was not investigated further on account of time.

In Fig. 3.47, each data set is fit with a linear plot. The relation between Re and the steady-state reaching time is most likely nonlinear, but a linear fit was used to get an idea of the extent by which the time changed by increasing Re. This can be seen in the corresponding legend.

3.4.2 Applicability & Feasability

As a whole, the applicability of this approach to solving the vorticity form of the Navier-Stokes equation can be said to be very promising. Besides the relatively large inaccuracies of the pressure calculations (see next section), the fluid response of the computational system was very similar to what was expected of the actual physical response. This means that the methods used here can

definitely be applied to other, similar fluid dynamics problems.[3] In particular, the stream function, vorticity, and velocities were all accurate (and precise) in terms of what was theoretically expected of the physical response. If this approach is to be used for other simulations, care must be taken in adjusting the domain properly and ensuring that the initialisation is correct for the system parameters. The pressure calculations, and in particular the boundary conditions for the pressure, should definitely be reevaluated. A successive (over-)relaxation scheme should also be considered for the pressure, which might improve both the precision and accuracy of the simulation, while reducing simulation runtime.

In terms of feasibility, it can be said that this approach is reasonable. The major check for feasibility (beyond the actual accuracy and precision) is the runtime of a script. The grid number and total number of time steps are the two main contributors to runtime, as these are what need to be swept over. For a grid size of 200×200, and an N_t of around 10^6, the simulation took around 1.25 hours to run. For the number of parameters being swept over/ iterated ($200 \cdot 200 \cdot 10^6 = 4 \cdot 10^{10}$) this is very efficient. Thankfully, the array slicing as described in Chapter 2 helped to drastically reduce computation time (and in fact removed the need for spatial iteration). Thus, it can be said that this is quite a feasible script to run relatively small- to medium-sized simulations with reasonable runtimes.

3.4.3 Results Quality: Accuracy, Precision, and Improvements

In general both the accuracy and the precision of the simulation were good. The lowest precision function was consistently the stream function, with a precision of three significant figures (e.g. $u = 0.01234 \pm 0.00003$ - the fourth significant figure has an error), with all other functions having a precision of four significant figures or more. Unfortunately the pressure was generally inaccurate. There might be a few reasons for this, but the most likely is incorrect implementation of boundary conditions.

The precision analysis was based on truncation errors solely. On account of the lack of time, representation and round-off errors were not considered in the error analysis. Furthermore, there was difficulty implementing analysis tools for these errors in the code, so they were ignored. The final results being as expected mean that these errors could not have been significant then, except for maybe the pressure.

Residual analysis was done by examining residuals from a successive relaxation process. The residuals of all nine cases indicated that the simulation converged very well. However, they were all suspiciously low. A reason for this might be the way the residuals were calculated. Only the mean of the residuals was considered (at each time step), rather than each residual individually or the norm of the residuals, which are both stricter requirements. Despite this, the mean should still be an applicable form of checking the residual tolerance, and it might simply be the case that the code is very efficient at sweeping through the spatial grid to achieve the necessary tolerance. The convergence of the pressure was never checked, which might have shed light on both the very low u residuals, and the pressure inaccuracies.

[3] Note: The response of each function should still be checked to ensure that they are physically reasonable.

One final, quite important, note is on the y-dimension of the domain. As seen in Fig. 1.1, $y = 0$ is at the lid, while $y = 1.0$ is at the bottom of the cavity. When plotting however, this was erroneously flipped. The functions also all had to be flipped then (which can be seen in the plotting functions in the project notebook) when plotted, as the equations were discretised according to the domain described in Fig. 1.1. This was not a huge problem, as the physics remained the same, but if this project were to be repeated care should be taken to keep the cavity axes in plotting consistent with the initial domain axes. In this way, the functions would not need to be flipped when plotting the functions in a cavity.

Chapter 4

Conclusion

The purpose of this project was to discretise and solve the vorticity form of the Navier-Stokes equation applied to a lid-driven cavity flow, and subsequently simulate the flow of an incompressible fluid in this cavity. Along with simulating a fluid in a lid-driven cavity flow, the effects of changing the Reynolds number and bottom wall velocity were also investigated. Each equation needed for the simulation was discretised using the finite difference method; the central difference approximation was used for spatial interiors, the forward difference approximation was used for time discretisations, and a combination of forward and backward difference approximations were used for the spatial boundaries. These equations were then implemented using a Gauss-Seidel method paired with successive relaxation in a Python code. Array slicing was used to update the spatial grid, while a for-loop was used to iterate through the time steps.

There were many results that were obtained from these simulations and subsequent analyses. First and foremost, it was found that by increasing the Reynolds number, the time it took for the system to reach a steady-state also increased. This is not surprising, as a higher Reynolds number implies turbulent flow, which means that the system has to take longer to stabilise on account of the chaos. Furthermore, it was also found that by changing the bottom wall velocity, this steady-state reaching time also changed. In general, by adding a driving bottom wall, the steady-state reaching time decreased, with the exception of $Re = 10$ and $V_{\text{bot}} = -1$, wherein it increased. This increase is attributed to the fact that two vortices are created (in the $Re = 10$, $V_{\text{bot}} = -1$ case) whose flows are rotating in the same direction, which results in opposite flow directions at their meeting, and hence longer stabilising times. Additionally, it was found that by increasing the Reynolds number, the irregularities of the system increased. Symmetries originally present at low Reynolds numbers were broken, and chaotic behaviour such as eddies occurred.

The accuracy and precision of the simulation were, in general, good. The simulation results showed substantial agreement with the theoretically predicted response, and behaved as how a physical system would be expected to behave. This was true for all functions except the pressure. The pressure, except for a handful of points in the cavity, was mostly inaccurate for all cases. The most likely culprit is the boundary conditions - if this simulation were to be used, the boundary conditions of the pressure must be reevaluated. All the functions had mean values with a precision of at least three significant figures or more, as compared with their respective limiting truncation error, which indicates reasonable precision. Residual analysis concluded that all the simulations

converged very well (residuals well below 10^{-6}). The residuals were suspiciously low at points, but the reason behind this was never really determined.

To improve, over-relaxation might help to improve the speed of the simulation. In that case, the over-relaxation could then also be applied to the pressure without sacrificing the runtime too much, and the grid spacing could also be decreased while ensuring reasonable runtimes (remembering that a decrease in grid spacing means that the time discretisation must also decrease, to ensure stability). The Gauss-Seidel method could also be replaced by the Jacobi method, but it is unsure what the effects of that would be. Regarding the steady-state determination, a caveat with the method used was that only one probe was utilised, and it was placed in the centre. This could have led to biases, as in some cases the centre stabilised quicker than, for example, grid point (80, 30). To improve on this, a range of different probe positions should have been used. Finally, as mentioned, the pressure function and its boundary conditions should be reevaluated to identify the source of the pressure inaccuracies.

All in all, the simulation proved to work quite well, and is a promising simulation method for small- to medium-sized simulations of fluid dynamics systems with reasonable runtimes.

Bibliography

[1] P. Onck, "Lid-Driven Cavity Flow," University of Groningen.

[2] R. H. Landau, M. J. Páez, and C. C. Bordeianu, *Computational Physics: Problem Solving with Python*, 3rd ed. WILEY-VCH Verlag GmbH & Co.KGaA, 2015, pp. 86-87 & 575-590.

[3] B. E. Rapp, "No-Slip Boundary Condition," [Online]. Available: https://www.sciencedirect.com/topics/engineering/no-slip-boundary-condition. Accessed: 19th October 2022.

[4] B. Rehm, A. Haghshenas, J. Schubert, J. Hughes, and A. S. Paknejad, *Managed Pressure Drilling*. Gulf Professional Publishing, 2008, pp. 59–60.

[5] "Theory for the Wall Boundary Condition," [Online]. Available: https://doc.comsol.com/5.5/doc/com.comsol.help.cfd/cfd_ug_fluidflow_single.06.065.html. Accessed: 21 October 2022.

[6] M. Kuron, "3 Criteria for Assessing CFD Convergence," [Online]. Available: https://www.engineering.com/story/3-criteria-for-assessing-cfd-convergence. Accessed: 1st November 2022.