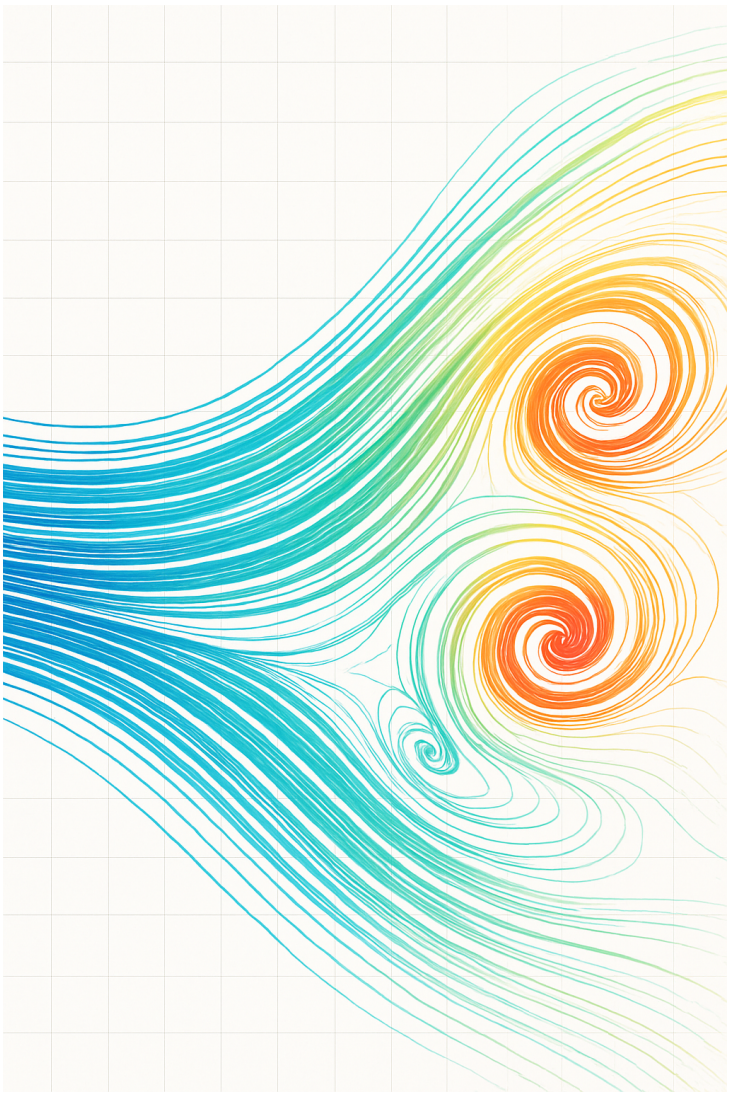
Simulation of Hydrodynamics and the Navier-Stokes equations using Python

Des De Borger

May 21, 2025



Contents

1 Introduction **2**

2 Methodology **2**
2.1 The Navier-Stokes equations . . . 2
2.2 Discretization scheme . . . 3

3 Code implementation **4**
3.1 Initialisation methods . . . 6
3.2 Solving methods . . . 6
3.3 Plotting methods . . . 7
3.4 Hidden methods . . . 10
3.5 NoSlipObject . . . 11
3.6 External funtions . . . 11

4 Exercise 1: Driven cavity **11**
4.1 Effect of the grid size/time step . . . 13
4.1.1 Convergence criterium . . . 13
4.1.2 Convergence speed . . . 14
4.1.3 Numerical instabilities . . . 18
4.2 Effect of the fluid parameters/Reynolds number . . . 18
4.2.1 Same Re, different parameters . . . 19
4.2.2 Increasing Re . . . 21
4.2.3 Decreasing Re . . . 24

5 Exercise 2: Laminar flow **25**
5.1 Increasing F . . . 27
5.2 Changing Re by changing the viscosity . . . 28

6 Exercise 3: Laminar flow over an obstacle **29**
6.1 Increasing F . . . 30
6.2 Changing μ . . . 31
6.3 Changing the obstacle height . . . 36
6.4 Speed profiles . . . 37

7 Exercise 4: Laminar flow over trough **40**
7.1 Increasing F . . . 40
7.2 Changing μ . . . 41
7.3 Speed profiles . . . 42

8 Conclusion **44**

9 Extras **44**

Abstract

The Navier-Stokes equations are partial differential equations that describe the flow of viscous fluids. In theory, these equations give the solutions for the pressure and velocity fields, given appropriate boundary conditions; however, analytical solutions rarely exist. Thus, we have to turn to numeric methods to approximate the solutions to the problem. In this report, we will discuss how the 2D Navier-Stokes equations can be solved using iteration-based solvers in Python.

1 Introduction

Our goal is to solve the Navier-Stokes equations for different systems, and to study the effects of the different fluid-parameters on the solution of the system. To do this, we first define the type of fluid under consideration—in our case, a Newtonian (constant viscosity), incompressible (constant density) fluid—and derive the corresponding Navier-Stokes equations (see chapter The Navier-Stokes equations). Next, we will discretize these equations by approximating the differential operators by finite-difference approximations (see chapter Discretization scheme).

The next step is to implement the found algorithm in Python, using object-oriented programming (see chapter Code Implementation). Our goal is to create a robust yet user-friendly package capable of simulating a wide range of customizable fluid systems.

Finally, we apply our code to simulate four systems—driven cavity, laminar flow, laminar flow over an obstacle, and laminar flow over a trough—analyzing the influence of different parameters in each case.

2 Methodology

In this section, we will briefly explain the specifics of the hydrodynamic problem we are trying to solve (based on [VRDL]), and the discretization scheme we are going to use to solve it. Furthermore, we will give an overview of the different classes, methods that are used to solve this problem.

2.1 The Navier-Stokes equations

We will begin by considering the *conservation of momentum* in a fluid. If we consider a fluid running through a volume element, the conservation of momentum can be expressed as follows:

$$\rho\left(\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V}\cdot\nabla\mathbf{V}\right) = \rho\mathbf{f} + \nabla\cdot\mathbf{\Pi_{ij}} \tag{1}$$

In this expression, ρ is the density of the fluid, $\mathbf{V}$ is the speed of the fluid, $\mathbf{f}$ is the force enacted on the fluid, and $\mathbf{\Pi_{ij}}$ is the *deviatoric stress tensor*, a tensor of order 2 that describes stress within the system.

The terms on the left describe the change of momentum per unit of volume. On the right, the term $\rho\mathbf{f}$ describes external forces acting on the fluid. The term $\nabla\cdot\mathbf{\Pi_{ij}}$ be rewritten as the sum of a pressure term and a viscous term; $\nabla\cdot\mathbf{\Pi_{ij}} = -\nabla p + \mu\nabla^2\mathbf{V}$, where μ is the dynamic viscosity. Substituting this, we get the following expression for the conservation of momentum:

$$\rho\left(\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V}\cdot\nabla\mathbf{V}\right) = \rho\mathbf{f} - \nabla p + \mu\nabla^2\mathbf{V} \tag{2}$$

For now, we will assume that there are no external forces acting on the fluid. We will also define $\nu = \frac{\mu}{\rho}$ as the kinematic viscosity. This gives us:

$$\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V}\cdot\nabla\mathbf{V} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{V} \tag{3}$$

Now, we will start discretization this expression. We can approximate the time derivative by its forward finite difference approximation: $\frac{\partial \mathbf{V}}{\partial t} \approx \frac{\mathbf{V}^{n+1}-\mathbf{V}^n}{\Delta t}$, where Δt is our chosen time step. If we also substitute all other expressions for the velocity by $\mathbf{V}^n$, we get:

$$\mathbf{V}^{n+1} = \frac{\rho}{\Delta t}\mathbf{V}^n - \rho\mathbf{V}^n \cdot \nabla\mathbf{V}^n - \nabla p + \mu\nabla^2\mathbf{V}^n \tag{4}$$

From this equation, we can solve for the velocity field $\mathbf{V}$.

Now, we just have to find an expression to solve for the pressure p. T0 do this, we will first consider *conservation of mass*. The equation expressing this continuity equation is:

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{V}) = \frac{\partial\rho}{\partial t} + \rho\nabla\cdot\mathbf{V} + \mathbf{V}\nabla\cdot\rho = 0 \tag{5}$$

As it turns out, our system becomes a lot simpler if we consider the fluid to be *incompressible*, meaning that the density ρ is constant and doesn't change. In this case, equation 5 simply becomes $\nabla\cdot\mathbf{V} = 0$. We can now easily get an expression for the pressure by taking the divergence of equation 4 and setting $\nabla\cdot\mathbf{V}^{n+1} = 0$. We then get:

$$\nabla^2 p = \frac{\rho}{\Delta t}\nabla\cdot\mathbf{V}^n - \rho\nabla\cdot(\mathbf{V}^n\cdot\nabla\mathbf{V}^n) + \mu\nabla^2(\nabla\cdot\mathbf{V}^n) \tag{6}$$

This equation can be solved for p, and so we can solve the system.

The Reynolds number is defined as:

$$Re = \frac{LU\rho}{\mu} \tag{7}$$

Here, the variable L is the characteristic length of the system, and U is the characteristic speed of the system. In our simulations, we will always set L equal to the smallest dimension of the system. U is generally chosen as the average velocity of the field, but sometimes we already have a good estimate for the characteristic speed (see section (DrivenCavity)). The Reynolds number is a dimensionless number which predicts the behavior of the fluid. A low Reynolds number corresponds with a viscous fluid and very laminar flow, while a high Reynolds number corresponds with a non-viscous fluid and very turbulent flow with vortices.

2.2 Discretization scheme

Our goal is now to discretize equations 4 and 6. We will work in two dimensions, meaning that $\mathbf{V}$ has two components, which we will call u (horizontal) and v (vertical).

Furthermore, we will neglect second-order or higher derivatives in the pressure equation.In this case, the term $\mu\nabla^2(\nabla\cdot\mathbf{V}^n)$ is neglected, and $\nabla\cdot(\mathbf{V}^n\cdot\nabla\mathbf{V}^n)$ becomes $\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + 2\left(\frac{\partial u}{\partial y}\right)\left(\frac{\partial v}{\partial x}\right)$.

Now, we will consider an arbitrary scalar quantity f, and we will show how different derivatives can be approximated by finite difference expressions.

1. First order time derivative, forward time approximation: $\frac{\partial f}{\partial t} \approx \frac{f^{n+1}-f^n}{\Delta t}$.
2. First order spatial derivative, forward space approximation: $\frac{\partial f}{\partial x} \approx \frac{f_{i+1}-f_i}{\Delta x}$.
3. First order spatial derivative, center space approximation: $\frac{\partial f}{\partial x} \approx \frac{f_{i+1}-f_{i-1}}{2\Delta x}$.
4. Second order spatial derivative, center space approximation: $\frac{\partial^2 f}{\partial x^2} \approx \frac{-2f_i+f_{i+1}+f_{i-1}}{\Delta x^2}$.
5. Using the second order spatial derivative, we can discretize the laplacian: $\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2} \approx \frac{-2f_{i,j}+f_{i+1,j}+f_{i-1,j}}{\Delta x^2} + \frac{-2f_{i,j}+f_{i,j+1}+f_{i,j-1}}{\Delta y^2}$

Now, we can substitute these approximations in equations 4 and 6. We will also choose[1] $\Delta x = \Delta y = h$ By then solving for $p_{i,j}$ and $\mathbf{V_{i,j}^{n+1}} = \begin{pmatrix} u_{i,j}^{n+1} \\ v_{i,j}^{n+1} \end{pmatrix}$ get the following discretization schemes:

[1] In our code, we keep $\Delta x \neq \Delta y$. We only choose this equality for readability.

$$
\begin{aligned}
p_{i,j}^{n+1} = & \frac{(p_{i+1,j}^n + p_{i-1,j}^n)\Delta y^2 + (p_{i,j+1}^n + p_{i,j-1}^n)\Delta x^2}{2(\Delta x^2 + \Delta y^2)} \\
& - \frac{\rho \Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)} \cdot \left[\frac{1}{\Delta t} \left(\frac{u_{i+1,j}^n - u_{i-1,j}^n}{2\Delta x} + \frac{v_{i,j+1}^n - v_{i,j-1}^n}{2\Delta y} \right) \right. \\
& - \left(\frac{u_{i,j}^n (u_{i,j}^n - u_{i-1,j}^n)}{\Delta x} + \frac{u_{i+1,j}^n u_{i+1,j}^n - u_{i,j}^n u_{i,j}^n}{\Delta x} + \frac{v_{i,j}^n (u_{i,j+1}^n - u_{i,j-1}^n)}{2\Delta y} \right) \\
& \left. + \frac{1}{\Delta y} \left(v_{i,j}^n \cdot \frac{v_{i,j+1}^n - v_{i,j-1}^n}{2} \right) \right]
\end{aligned} \tag{8}
$$

$$
u_{i,j}^{n+1} = u_{i,j}^n - \frac{\Delta t}{2\rho \Delta x}(p_{i+1,j}^n - p_{i-1,j}^n) + \mu \left[\frac{\Delta t}{\Delta x^2}(u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + \frac{\Delta t}{\Delta y^2}(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n) \right] \tag{9}
$$

$$
v_{i,j}^{n+1} = v_{i,j}^n - \frac{\Delta t}{2\rho \Delta y}(p_{i,j+1}^n - p_{i,j-1}^n) + \mu \left[\frac{\Delta t}{\Delta x^2}(v_{i+1,j}^n - 2v_{i,j}^n + v_{i-1,j}^n) + \frac{\Delta t}{\Delta y^2}(v_{i,j+1}^n - 2v_{i,j}^n + v_{i,j-1}^n) \right] \tag{10}
$$

By analyzing these schemes further, it can be proven that stability is guaranteed if [VRDL]:

$$
\Delta t \leq \frac{h^2}{2\mu + hU} \tag{11}
$$

Here, U is the typical speed of the system.

3 Code implementation

Now that we have discussed the methodology for solving the Navier-Stokes equations, we will explain in more detail how we have structured our code. The package we have built is called `NavierStokes`, and it consists of 2 classes (`NewtonianFluid` and `NoSlipObstacle`), and 5 external functions which are mainly used to typecheck inputted values. Figure 1 gives a general overview of the structure of the code.

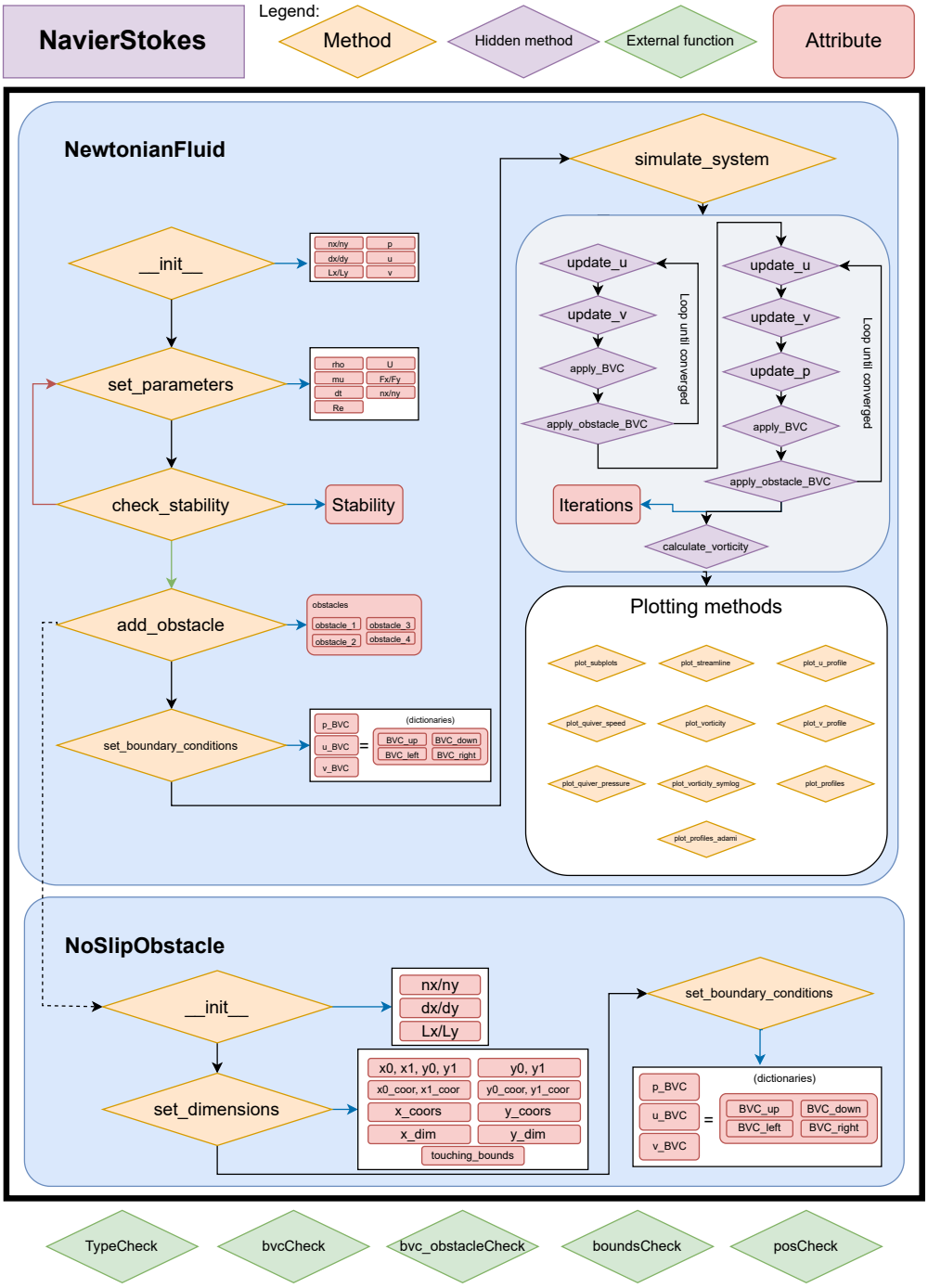


Figure 1: Flowchart for the NavierStokes package.

We will now explain all implemented methods in more detail.

3.1 Initialisation methods

__init__()

The method `__init__()` initializes the system, and has inputs $L_x, L_y, \Delta t, \Delta x, \Delta y$. Furthermore, the parameters μ, ρ, U can be initialized, but they have default values and don't need to be inputted. Furthermore, this method calculates the grid size, created the p, v, u fields, and already calculates the Reynods number (for the case that we already have an estimate for U).

set_parameters()

In this method, we set values for the parameters $\Delta t, \Delta x, \Delta y, \mu, \rho, U, F_x, F_y$. Furthermore, the Reynolds number is recalculated, and since the grid size can change, the hidden method `_update_grid()` is called for each obstacle (see later).

set_boundary_conditions_[p/u/v]()

These are actually three separate methods, one for each field (u, p, v). These methods initialize or change dictionaries, which contain the boundary conditions for the system. These conditions can be Dirichlet, Neumann or Periodic, depending on the system itself.

add_obstacle()

This method simply creates and initializes a `NoSlipObstacle` object and appends it to a list within the `NewtonianFluid` object.

check_stability()

This method checks if equation 11 holds true. Note that this method can only be used after the system has converged to an equilibrium (in which case we know the average of U), or if we already have a guess for U.

update_dt()

This method uses the stability criterium (Equation 4.1.1) to change the chosen time step accordingly. Δt is set to a fraction of $\frac{h^2}{2\mu + hU}$ ($\frac{1}{10}$ by default).

3.2 Solving methods

simulate_system()

This function contains the main loop which solves the Navier-Stokes equations for the system. The basic idea is the following:

1. Apply equations 8, 9, and 10 using the hidden `_update_p`, `_update_p`, and `_update_p` methods (see Hidden methods).
2. Apply the boundary conditions for the bounds of the system using the hidden `_apply_all_boundary_conditions()` method (see Hidden methods).
3. Apply the boundary conditions for the obstacles using the hidden `_apply_all_obstacle_boundary_conditions()` method (see Hidden methods).
4. Check if convergence is reached, or if the iteration cap is exceeded.

Both the convergence tolerance and the iteration cap have default values (10^{-4} and 10000 respectively), but can be changed by the user. The convergence is checked by taking the maximum of the variables $p_{diff}, u_{diff}, v_{diff}$ (which are calculated as $\left|\frac{f_{new} - f_{old}}{f_{new}}\right|$), and comparing it to the set tolerance.

This function is allowed to run, regardless of the stability criterium. If divergence occurs, all fields are set to zero.

In actuality, this function works in two phases. In the first phase, we try to find a solution for the u and v fields (by applying equations 9 and 10), while keeping the pressure at $p = 0$. In the second phase, we add equation 8 to the loop. Finally, we also calculate the vorticity of the system using the hidden `_calculate_vorticity()` method (see Hidden methods).

update_reynolds_number()

This method recalculates the Reynolds number of the system, substituting U with the average velocity of the field.

3.3 Plotting methods

To show to solution of the system, we have implemented a dozen of plotting methods. Below, we will show an example for every method, for the driven cavity (see Driven Cavity)

plot_quiver_speed()

This method makes a quiver plot, with the norm of the velocity as the background (Figure 2).

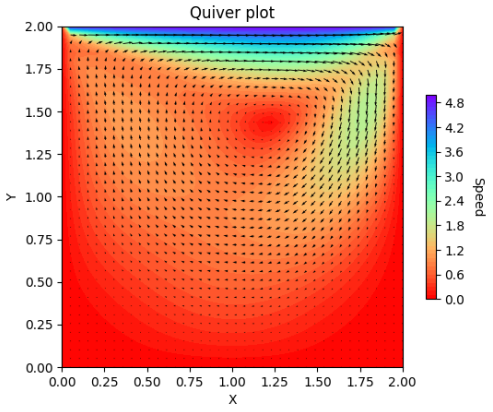


Figure 2: Plot generated by plot_quiver_speed().

plot_quiver_pressure()

This method makes a quiver plot, with the pressure as the background (Figure 3).

plot_streamline()

This method makes a streamline plot, with the vorticity as the background. The user can choose the thickness of the line via input. The default value for the line thickness is proportional to the norm of the velocity. (Figure 4).

plot_vorticity()

This method makes a plot of the vorticity of the system. (Figure 5).

plot_vorticity_symlog()

This method makes a plot of the symmetric logarithm of the - vorticity of the system. This quantity is calculated via $\text{sgn}(\zeta)\log(|\zeta| + 1)$. This plot makes the vorticity of the entire system more readable, whereas the `plot_vorticity` shows the location of the vortices clearly. (Figure 6).

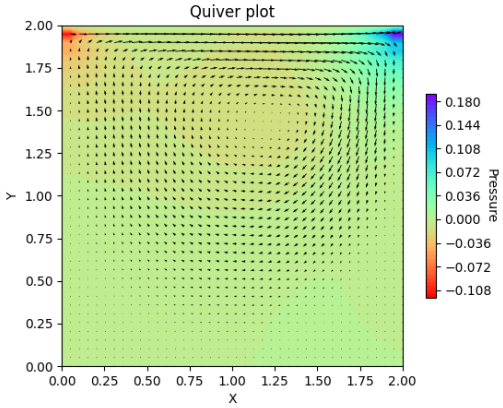


Figure 3: Plot generated by plot_quiver_pressure().

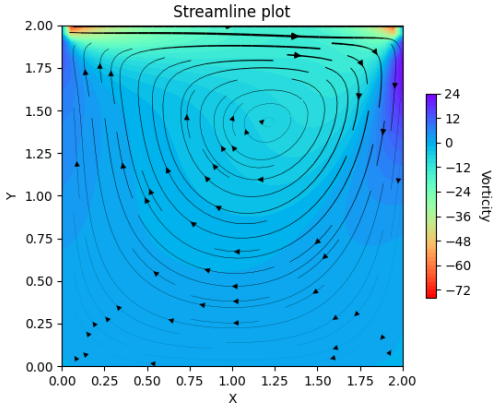


Figure 4: Plot generated by plot_streamline().

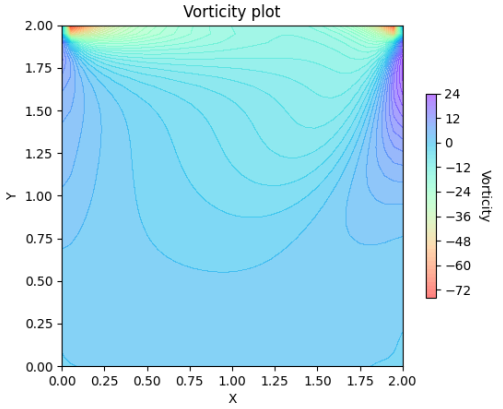


Figure 5: Plot generated by plot_vorticity().

plot_subplots()

This method makes a 2×2 subplot of the `plot_quiver_speed()`, `plot_quiver_pressure()`, `plot_streamline()`[2], and `plot_vorticity_symlog()` methods.(Figure 11).

[2]With the line thickness set to 1.

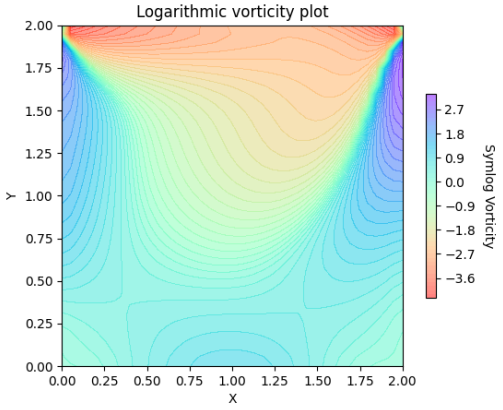


Figure 6: Plot generated by plot_vorticity_symlog().

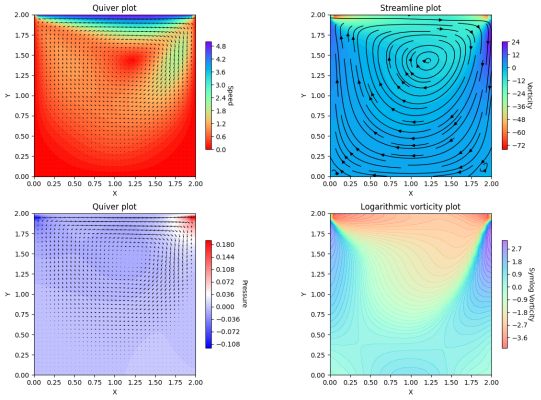


Figure 7: Plot generated by plot_subplots().

plot_profiles()

This method makes plots of the horizontal and vertical speed profiles of the system. The user can choose the desired cross section, and the default values for these parameters are $\frac{L_x}{2}$ and $\frac{L_y}{2}$. There are actually three methods, `plot_u_profile`, `plot_v_profile`, and `plot_profiles` (which is simply the first two next to each other). (Figure 12).

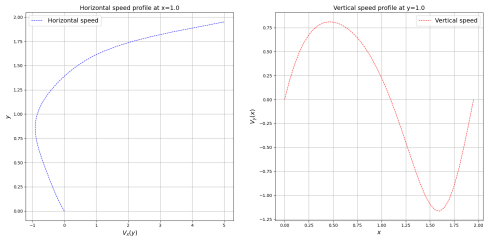


Figure 8: Plot generated by plot_profiles().

plot_profiles_adami()

This method makes plots of the horizontal and vertical speed profiles of the system, in the style of [AHA13]. (Figure 12). Again, the user can choose which cross sections to plot, with the default values

being set to $\frac{L_x}{2}$ and $\frac{L_y}{2}$. In the remainder of this report, we have elected to use the `plot_profiles` method instead of the `plot_profiles_adami` method, since it is more intuitive to understand.

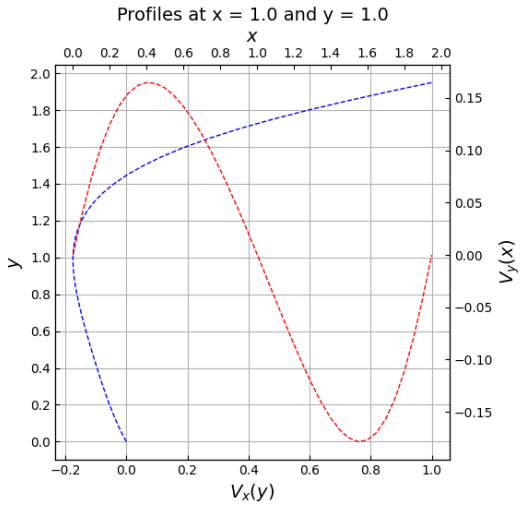


Figure 9: Horizontal and vertical speed profiles of a driven cavity, in the style of [AHA13].

3.4 Hidden methods

To make the code more readable, we have implemented a few *hidden* methods, which are not accessible to the user.

update[p/u/v]()

This method applies the discretization (equations 8, 9, 10) scheme to one of the fields.

_calculate_vorticity()

This method calculates the vorticity of the system. For grid points that do not lie on an edge, central difference approximations are used for both derivatives. If the grid point lies on an edge (or corner), a forward or backward difference approximation is used.

_apply_boundary_conditions()

This method applies inputted boundary conditions to an inputted matrix.

_apply_all_boundary_conditions()

This method uses the `_apply_boundary_conditions()` method to apply the boundary conditions (stored in the `BVC` dictionaries) to the p, u, v fields.

_apply_obstacle_boundary_conditions()

This method applies inputted boundary conditions (stored in a `NoSlipObstacle` object) to an inputted matrix.

_apply_all_obstacle_boundary_conditions()

This method uses the `_apply_obstacle_boundary_conditions()` method to apply the boundary conditions (stored in the `BVC` dictionaries, which are stored in `NoSlipObstacle` objects) to the p, u, v fields.

3.5 NoSlipObject

These objects are made via the `add_obstacle()` method. However, their attributes have to be changed. For this, we use the following methods:

set_dimensions()

This method takes the four corners of the object as an input, and converts them to their nearest coordinates in the grid. Then, it stores the relevant x- and y-coordinates to use later for `numpy`-slicing. Furthermore, the methods checks if the object is touching the bounds of the system, which might be a problem when implementing Neumann boundary conditions.

set_boundary_conditions_[p/u/v]()

Just like the method with the same name in the `NewtonianFluid` class, this method takes boundary conditions as inputs and stores them in dictionaries. This method can be used to make obstacles that are not non-slip, like moving belts.

_update_grid()

This method is called when the grid size of the system changes, and will recalculate where the corners of the obstacles are on the grid.

3.6 External funtions

Finally, we have also implemented a few external functions, which are used to typecheck the inputs

typeCheck()

Checks the type of a list of objects.

posCheck

Checks if a float is greater than 0.

boundsCheck

Checks if a float is between to given values.

boundsCheck

Checks if a float is between to given values.

bvcCheck()

Checks if a dictionary of boundary conditions is allowed (Ńeumann; Ṕeriodic; or a numeric value).

bvc_obstacleCheck()

Checks if a dictionary of boundary conditions for obstacles is allowed (Ńeumann; or a numeric value).

4 Exercise 1: Driven cavity

In the first exercise, we study the problem of *lid-driven driven cavity flow*. The system we are considering essentially consists of a box containing a liquid, which is being driven by a moving belt at the top of the box. We consider all walls (and the belt) to be no-slip, which gives us the following boundary conditions shown in figure 10. We will start from a motionless liquid ($p = 0, u = 0, v = 0$) at $t = 0$.

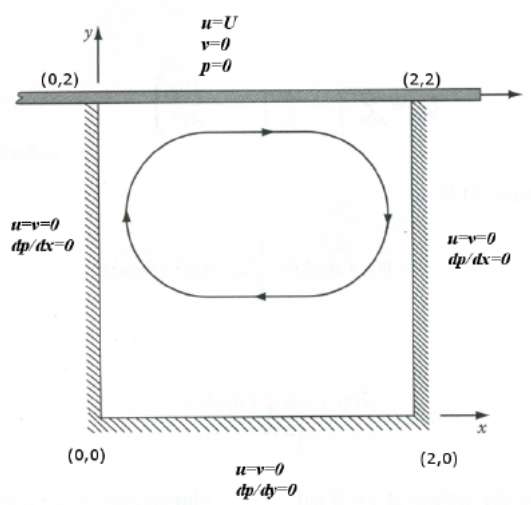


Figure 10: Schematic illustration of the boundary conditions of the driven cavity. [VRDL]

For the initial simulation, I have chosen a spatial spacing size of $\Delta x = \Delta y = 0.05$, which gives us a grid of 40×40 points. Furthermore, I will choose a time step of $\Delta t = 10^{-3}$. In order to make the simulation stable, the parameters have to be chosen in a way so that equation 11 holds. Furthermore, we want to begin with a low Reynolds number, given by 7. In this equation, there is a dependence on the variable U, the *typical* size of the velocity field. Since we do not really have a good way of determining this before simulating the system, we will set U equal to the speed of the moving belt.

I have chosen for the parameters $U = 5, \rho = 0.01, \mu = 0.1$, which guarantees stability and gives $Re = 1$. Running this simulation (with a tolerance of 10^{-4}), we get a stable solution after 5903 iterations. This takes $1.54s$, which is relatively quick. The solution is shown in plots 11 and 12.

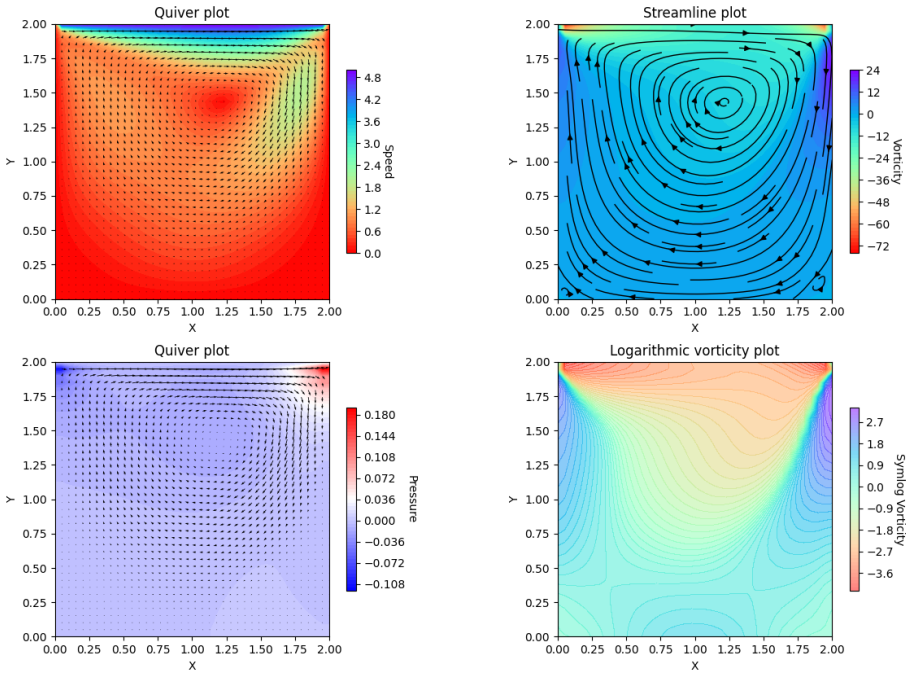


Figure 11: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 1$.

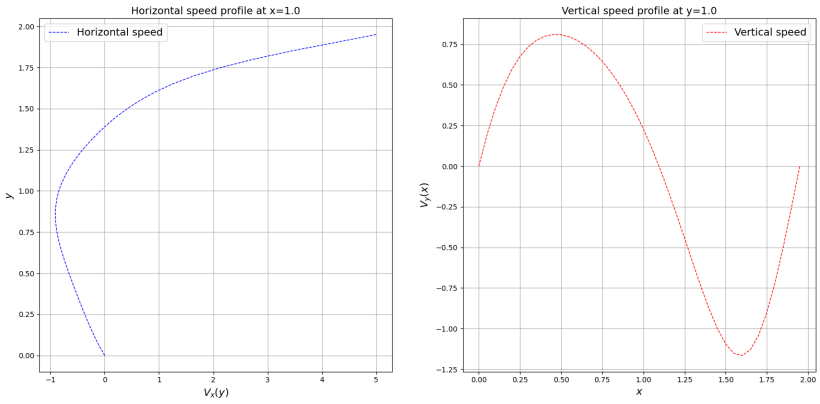


Figure 12: Speed profiles for a driven cavity with $Re = 1$.

In these figures, we see the expected behavior for lid-driven cavity flow. The profile plots (Figure 12) look very similar to the plots found in [AHA13]. Moreover, we can see small vortices in the bottom corners, even with relatively low Reynolds numbers. It should, however, be noted that these vortices are really only visible on the streamline plots, and actually only consist of 3-4 grid points. This means that they are not actually smooth circles, even if they are displayed as such in the streamline plot.

4.1 Effect of the grid size/time step

Now that we can find stable solutions for the driven cavity, we will analyze the effect of the grid size and time step on the simulation. In the entirety of this section, we will continue working with the parameters $U = 5, \rho = 0.01, \mu = 0.1$, which gives $Re = 1$.

4.1.1 Convergence criterium

First, we will keep the time step constant while changing the grid size. First, we will look for the smallest possible distance between the grid points, with which a stable solution is still found. We can rewrite the equation 7 as a quadratic equation in h:

$$h^2 - hU\Delta t - 2\mu\Delta t \geq 0 \tag{12}$$

Solving this equation for $h > 0$ yields:

$$h > \frac{U\Delta t + \sqrt{(U\Delta t)^2 + 8\mu\Delta t}}{2} \approx 0.0169 \tag{13}$$

In practice however, we see that equation 11 is not strict enough. The minimum value for h for which convergence is reached, is approximately equal to 0.030, which corresponds to a grid size of 66×66. For this value, a stable solution is reached after 5056 iterations, with a computation time of $2.68s$. This system is shown on Figures 13 and 14:

We can see that the behavior is almost identical to the first simulation with $h = 0.05$. For any value of h smaller than 0.030, the discretization scheme is not stable, and divergence occurs.

Next, we will keep $h = 0.05$ constant, and analyze the effect of the time step. This time, there is not a minimal, but a maximal value for Δt for which stability is expected. Using equation 11, the maximum time step should be ≈ 0.00556. In practice however, we find the maximum time step to be around 0.00275. In this case, convergence is reached after 2529 iterations, which takes $0.693s$. The solution is shown on Figures 15 and 16:

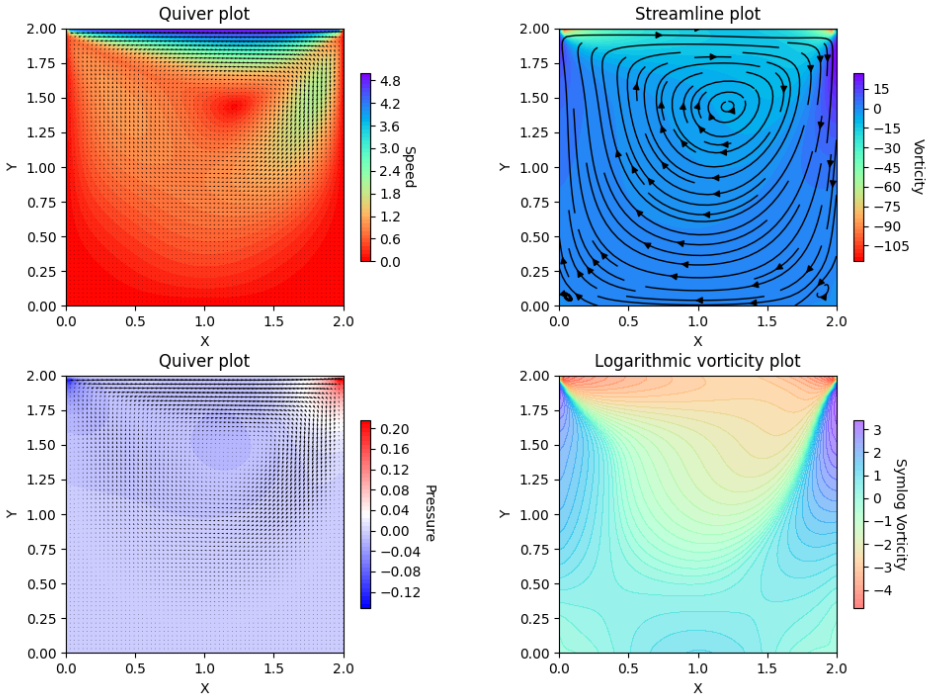


Figure 13: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 1$ and $h = 0.029$.

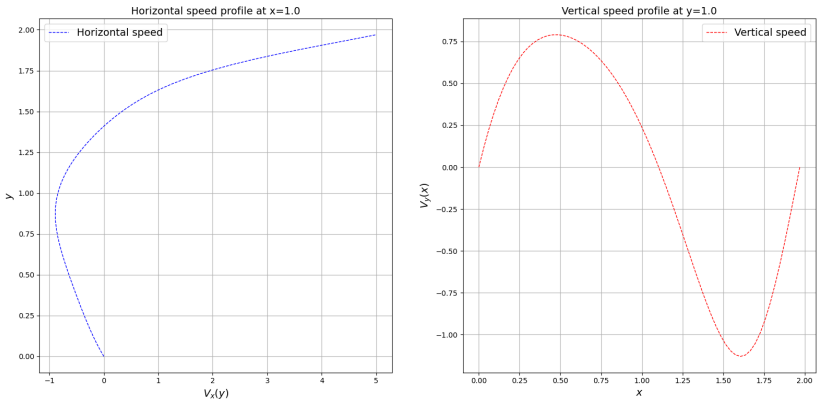


Figure 14: Speed profiles for a driven cavity with $Re = 1$ and $h = 0.029$.

From this analysis, we can conclude that equation 11 does not actually hold true in reality. This is most likely due to the vague definition of the variable U; it is equal to the *typical* size of the velocity field, which can't be defined rigorously. Equation 11 can thus only be used to estimate the order of magnitude of the maximum Δt and minimum h. The stability criterium is **not strict enough**: if the parameters do not satisfy the criterium, divergence always occurs, but if the parameters do satisfy the criterium, stability *may or may not* be reached.

4.1.2 Convergence speed

Next, we will study the relationship between the spatial step h and the amount of iterations. In order to do this, we will choose $\Delta t = 10^{-3}$ constant, and choose spatial steps between $h = 0.03$ and $h = 0.5$,

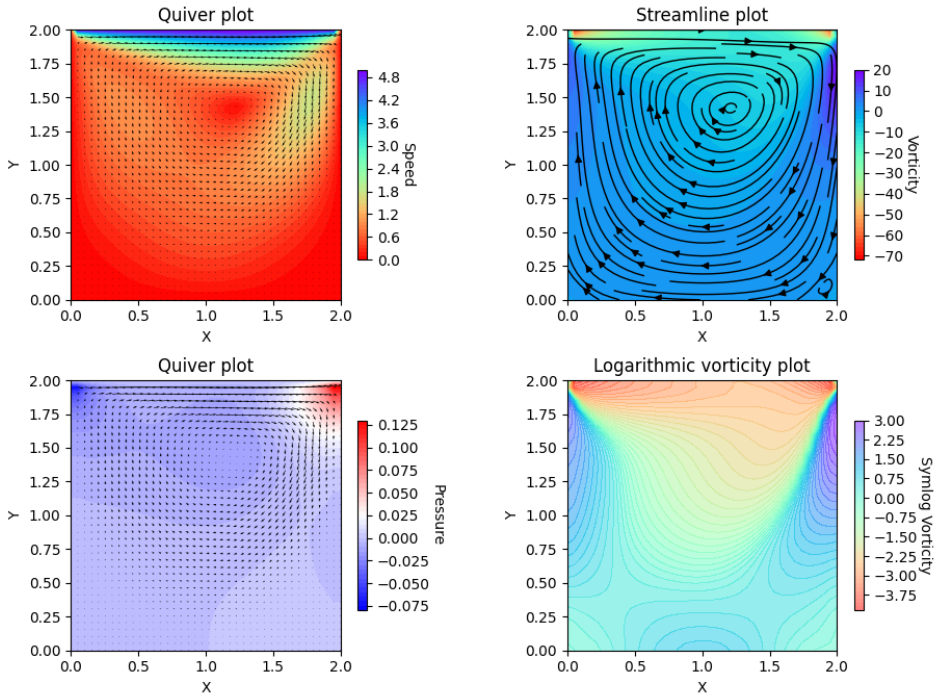


Figure 15: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 1$ and $\Delta t = 0.0032$.

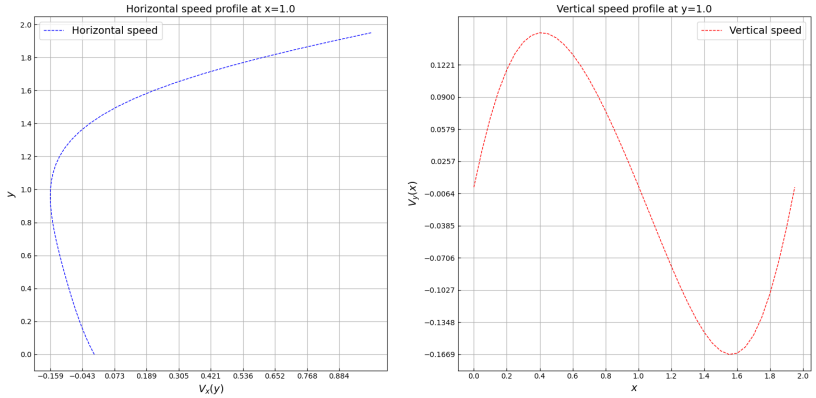


Figure 16: Speed profiles for a driven cavity with $Re = 1$ and $\Delta t = 0.0032$.

(which respectively correspond to grid sizes of 66×66 and 4×4), and see how long it takes before convergence is reached. The results are shown in Table 1.

If we plot these data points on a graph and add a function fit, we get Figure 17.

As we can see, there is not a distinct linear relation between the grid size and the number of iterations. As a matter of fact, the number of iterations doesn't increase strictly with increasing grid size, sometimes it goes down! When adding function fits using Excel, we find that the best model is given by a logarithmic relationship. To analyze this further, we calculate the correlation coefficient of this dataset. We find $r = 0.625$, which shows us that the grid size, and number of iterations are indeed positively correlated, as expected. For demonstration purposes, the solution for the system with a grid size of 10×10 is shown on Figures 18 and 19.

Next, we will study the relationship between the time step Δt and the amount of iterations. In

Grid size	h	Iterations	Computation time (s)
66	0.03	6845	2.834
50	0.04	7075	2.163
40	0.05	5903	1.45
33	0.06	5273	1.196
28	0.07	5094	1.071
25	0.08	4632	0.931
22	0.09	5094	0.931
20	0.1	4702	0.846
16	0.12	4892	0.836
13	0.15	6055	0.967
10	0.2	1467	0.267
4	0.5	1571	0.273

Table 1: Convergence speed for the driven cavity for different grid sizes.

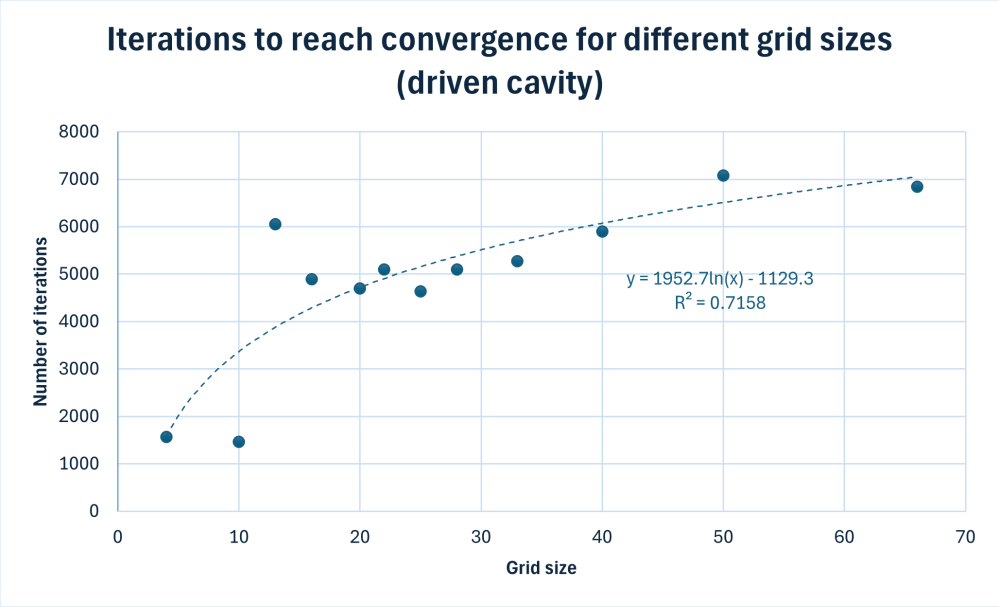


Figure 17: Graph of the number of iterations to reach convergence in function of the grid size.

order to do this, we will keep $h = 0.05$ constant, and choose time steps between $\Delta t = 2.5 \cdot 10^{-3}$ and $\Delta t = 1 \cdot 10^{-5}$, and see how long it takes before convergence is reached. The results are shown in Table 2. In this table, the last column (Simulated time) is equal to the product of the time step and the number of iterations.

By plotting this data on a graph and adding some function fits, we get Figure 20.

As we can see, the number of iterations shoots up steeply for very small time steps, which is expected behavior. The data is best modeled by a logarithmic function fit. However, this function fit becomes negative for some time step, which is not realistic. A better function fit is a power fit, where we find that the data is best modeled by the following relation: $\#iterations \propto (\Delta t)^{-0.623}$. Furthermore, we can see that the simulated time is not always equal. This is interesting, because ideally, the physical 'time' it takes to reach a stable state should be independent of the chosen time step. However, the discretization scheme is not perfect; we are approximating derivatives by finite-differences, for which the discretization error depends on the chosen time/spatial step. For large time steps, the discretization error is larger, and we need more iterations [3]. Thus, the simulated time is longer. For demonstration purposes, the solution for the system with a timestep of $\Delta t = 1 \cdot 10^{-5}$ is shown on

[3] *More iterations* means relative to the chosen time step: the number of iterations is smaller, but the simulated time is longer.

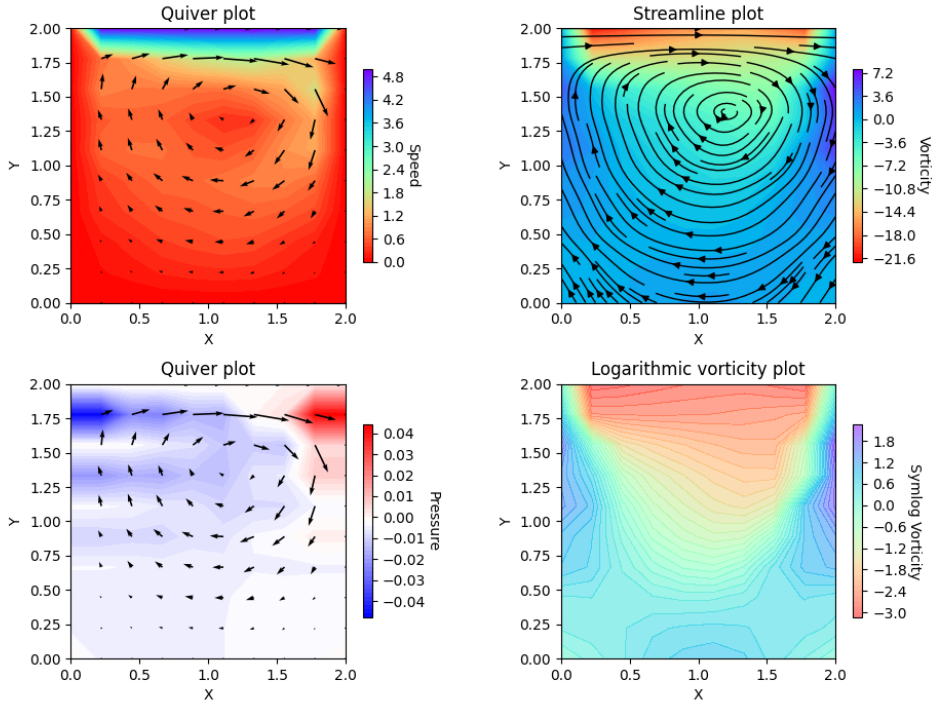


Figure 18: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 1$ and a grid size of 10×10.

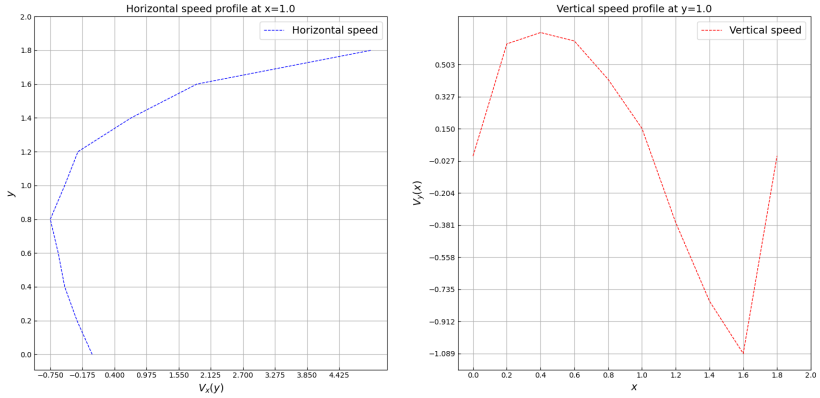


Figure 19: .Speed profiles for a driven cavity with $Re = 1$ and and a grid size of 10×10

Figures 21 and 22.

When comparing to Figures 11 and 12, we see some differences. Interestingly, the vortices in the corners seem to have disappeared. A possible explanation is that when Δt is smaller, the system comes to an equilibrium more gradually, and because of this there is less chaotic behavior. A deeper analysis of these corner vortices will be made further into this report.

Δt	Iterations	Computation time (s)	Simulated time
0.00001	85509	20.362	0.85509
0.000025	84286	20.078	2.10715
0.00005	53494	12.695	2.6747
0.000075	44113	10.51	3.308475
0.0001	32662	7.85	3.2662
0.00025	19950	4.829	4.9875
0.0005	9229	2.28	4.6145
0.00075	6200	1.613	4.65
0.001	5903	1.506	5.903
0.0025	2454	0.679	6.135

Table 2: Convergence speed for the driven cavity for different time steps.

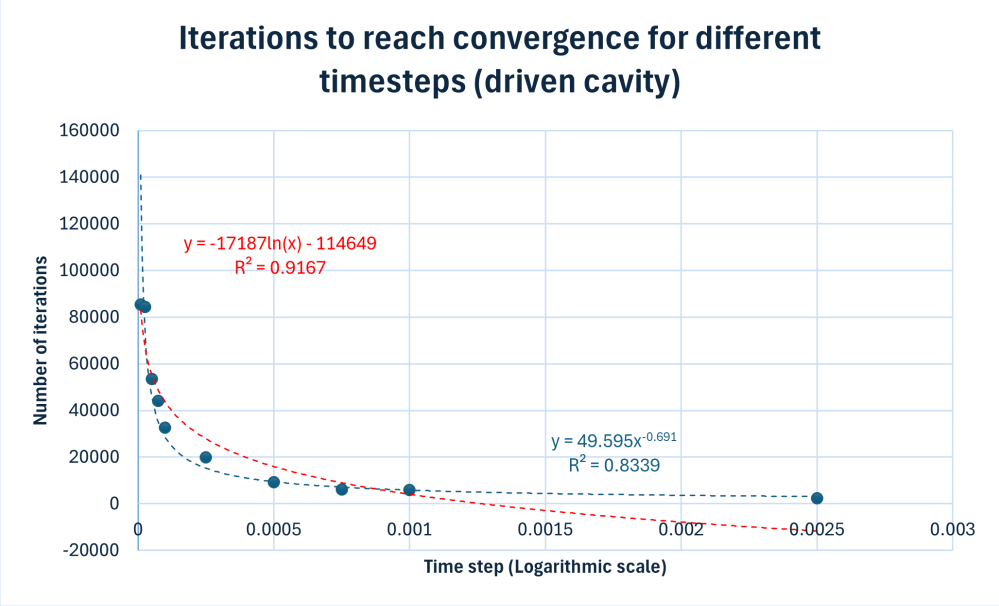


Figure 20: Graph of the number of iterations to reach convergence in function of the time step with two function fits.

4.1.3 Numerical instabilities

Finally, we will analyze if any numerical instabilities occur if we keep iterating, even after convergence has been reached. To do this, we set the tolerance equal to 10^{-14}, and we run the simulation for $h = 0.05$ and $\Delta t = 10^{-3}$, for a maximum of 10^6 iterations. The solution, of course will 'not converge', since the chosen tolerance is smaller than machine precision. It takes $238.456s$ to execute 10^6 iterations, and the results are shown on Figures 23 and 24.

When comparing to Figures 11 and 24, we see little to no difference, meaning that no numeric instabilities occur.

4.2 Effect of the fluid parameters/Reynolds number

In the previous section, we only used the parameters $U = 5, \rho = 0.01, \mu = 0.1$. Now, we will change these values and analyze the behavior of the system. However, first we must discuss the role of the density ρ of the fluid. When testing different values for the pressure, we noticed that the pressure actually does not have an influence on the velocity of the fluid; it is merely a factor that scales the pressure. It does however, have an influence on the Reynolds number. This could pose a problem, since we can make the Reynolds number arbitrarily large or small (without changing the velocity field) by changing the pressure. Because of this, we will set $\rho = 1$ as constant in the rest of this chapter.

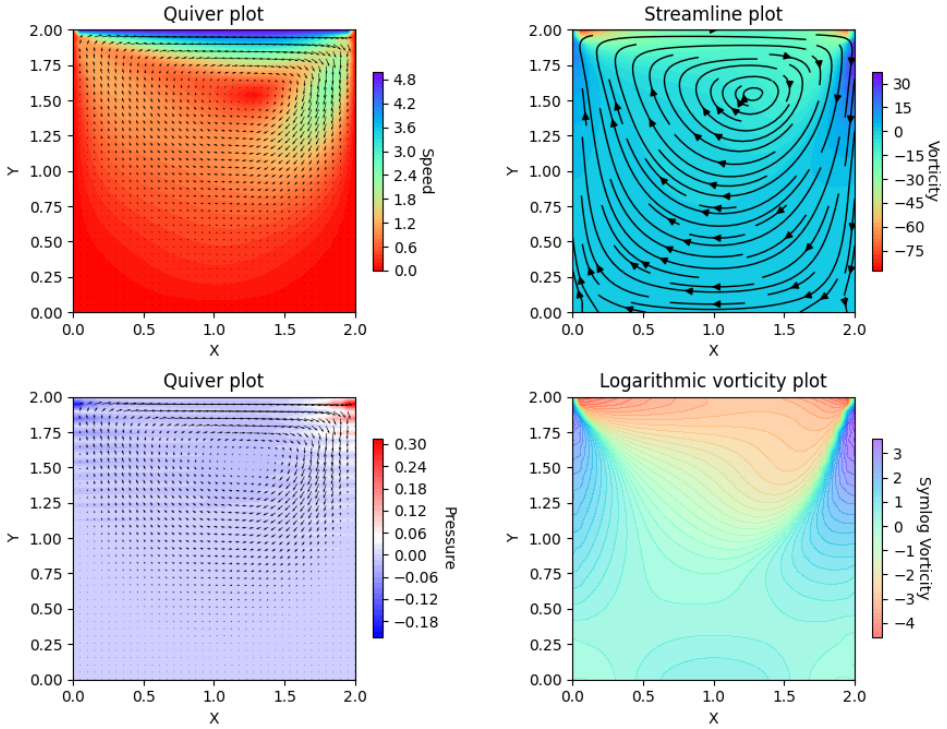


Figure 21: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 1$ and $\Delta t = 1 \cdot 10^{-5}$.

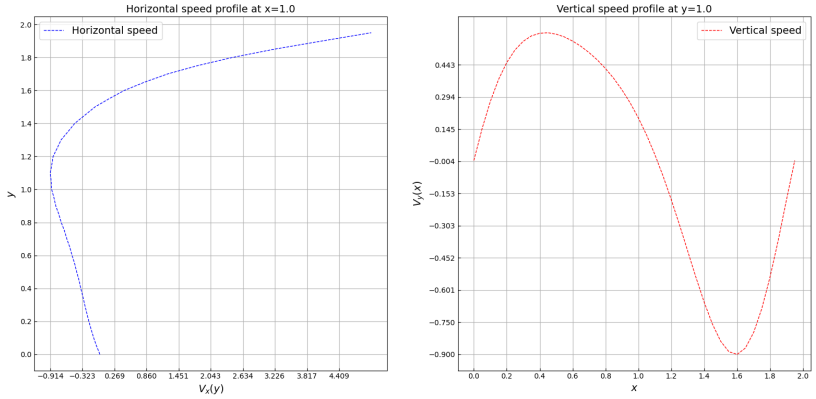


Figure 22: Speed profiles for a driven cavity with $Re = 1$ and $\Delta t = 1 \cdot 10^{-5}$.

Furthermore, we can always make the Reynolds number 0 by setting $U = 0$, which gives the trivial solution of a non-moving fluid. I will therefore set a minimum value of 1 for U.

4.2.1 Same Re, different parameters

First, we will consider two systems with the same Reynolds number, but different parameters. Next, we will try to increase the Reynolds number as much as possible, and see what this does to the system. For the first system, we choose the parameters $U = 5, \mu = 1$, and for the second system, we choose the parameters $U = 1, \mu = 0.2$. Both of these give $Re = 10$. Furthermore, I will use $h = 0.03$ and $\Delta t = 10^{-4}$. The first system taken 4918 iterations and $2.0223s$ to converge, and the second system

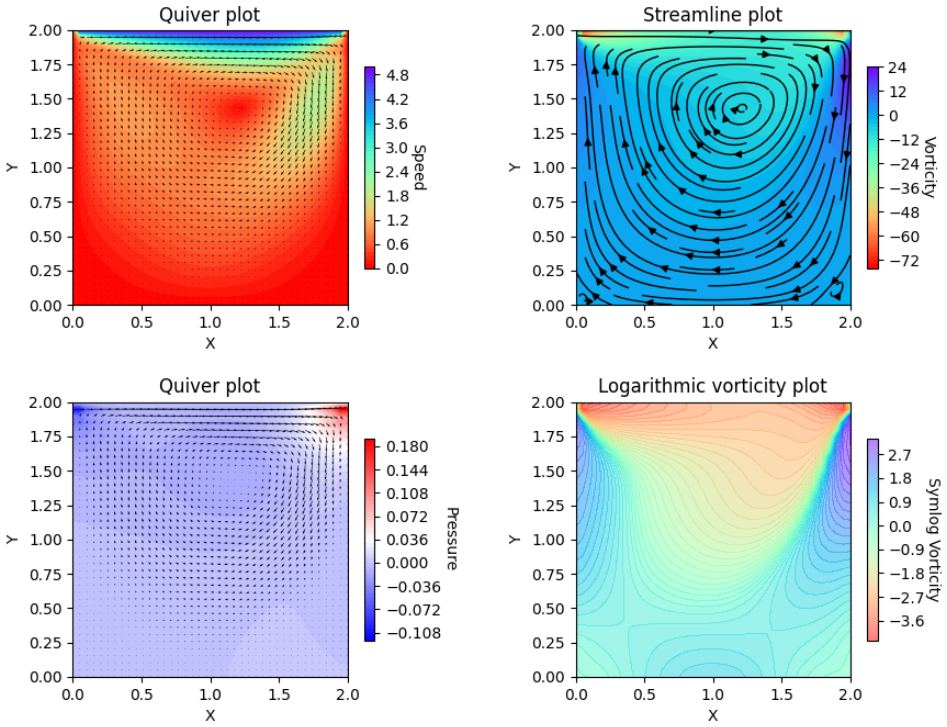


Figure 23: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 1$ after 10^6 iterations.

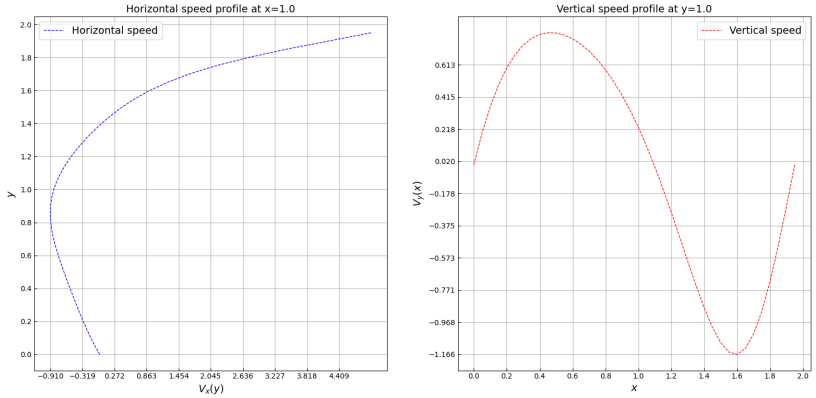


Figure 24: Speed profiles for a driven cavity with $Re = 1$ after 10^6 iterations.

takes 24000 iterations and $9.321s$ to converge. The results are shown on Figures 25- 28.

When comparing these figures, we see that they are very similar. The only notable difference is the pressure scale, which is ≈ 10 larger in the first system. This is because of the greater speed of the moving belt. The belt pushes the fluid into the top right corner, so that pressure builds up here. Conversely, the belt pulls the fluid away from the top left corner, and an under pressure forms. It is thus logical that the faster moving belt creates larger pressures. We can conclude that systems with similar Reynolds numbers will behave similarly, meaning that the Reynolds number is a good indicator for the behavior of the system.

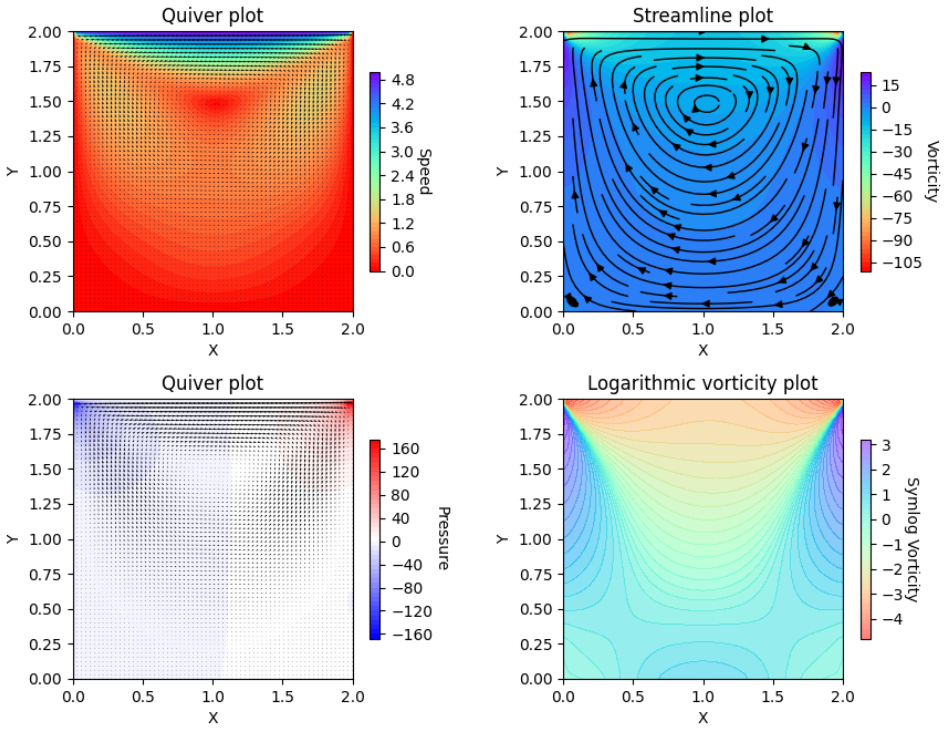


Figure 25: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 10$ and $U = 5, \mu = 1$.

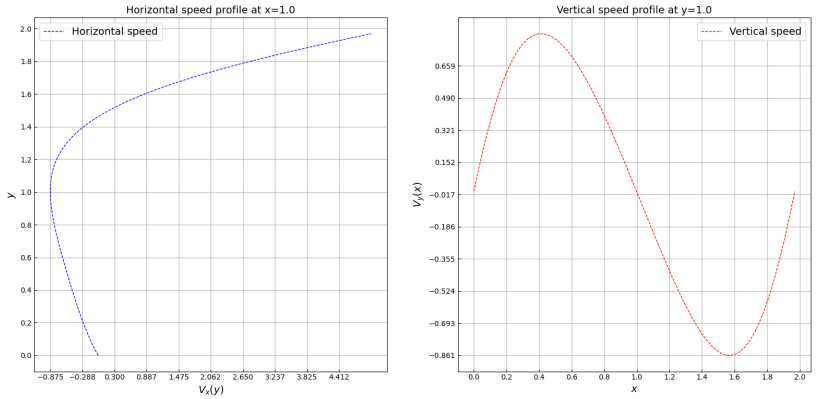


Figure 26: Speed profiles for a driven cavity with $Re = 10$ and $U = 5, \mu = 1$.

4.2.2 Increasing Re

Next, we will drive up the Reynolds number, and see what occurs. We will continue to use $h = 0.03$ and $\Delta t = 10^{-4}$. First, we will try a Reynolds number of 50, with parameters $U = 5, \mu = 0.2$. Convergence is reached after 19264 iterations and $7.334s$. The results are shown on Figures 29 and 30.

This system is very similar to Figures 27 and 28. However, the vortices in the bottom corners are a little larger, and the pressure has decreased by a factor of about 4. Next, we will increase the Reynolds number to 125, with $U = 5, \mu = 0.08$. This system converges after 51192 iterations and $19.338s$. The results are shown on Figures 31 and 32.

In these figures, we can see that the vortex in the bottom right has become even larger, which means that the simulation is working as expected. A higher Reynolds number means a less viscous

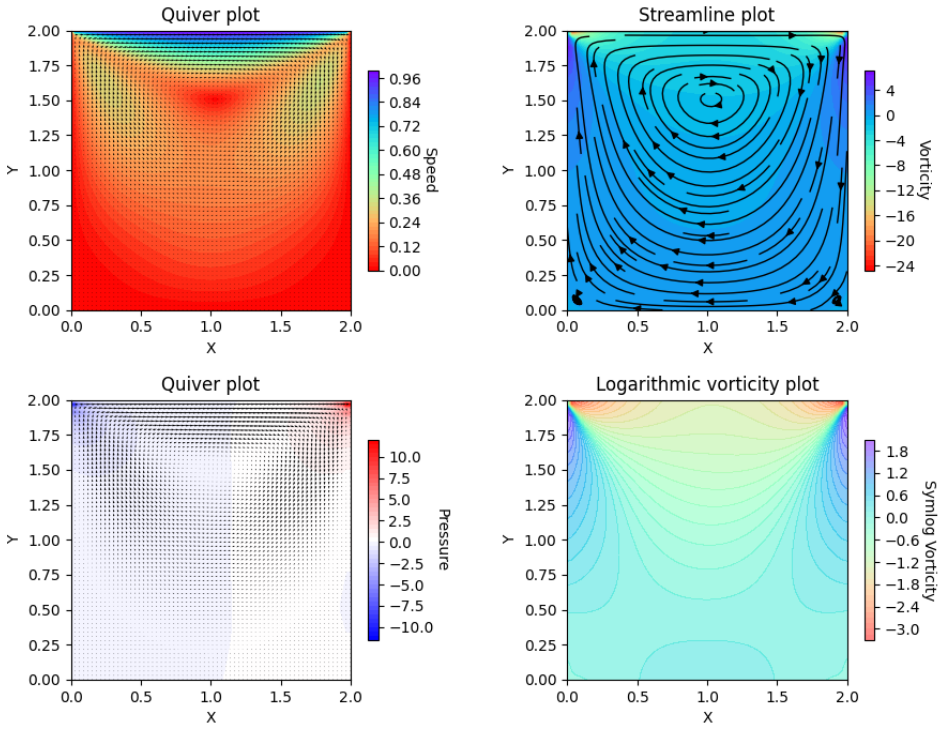


Figure 27: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 10$ and $U = 1, \mu = 0.2$.

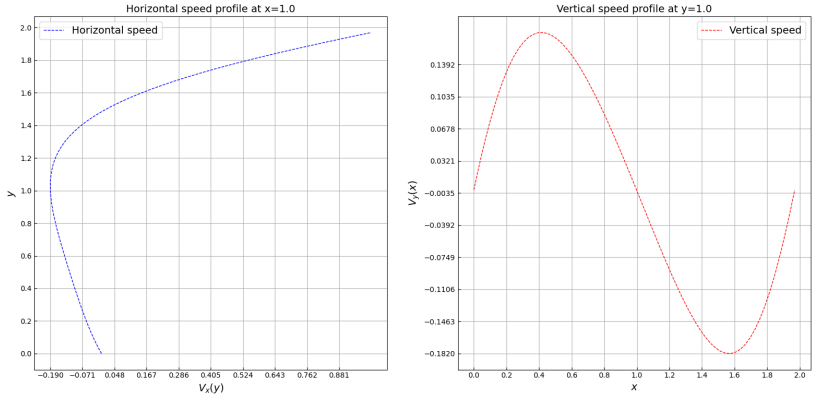


Figure 28: Speed profiles for a driven cavity with $Re = 10$ and $U = 1, \mu = 0.2$.

fluid, and more turbulent flow.

Finally, we will look to increase the Reynolds number as much as possible. Through trial and error, I have determined that the maximum Reynolds number[4] that doesn't cause divergence is equal to 280, with $U = 7, \mu = 0.05$. This system converges after 66047 iterations and $24.916s$. The results are shown on Figures 33 and 34.

The bottom right vortex is now the largest possible without the discretization scheme diverging. Interestingly enough, the bottom left vortex has disappeared. A possible explanation is that the right vortex is bigger, so the laminar flow at the bottom of the cavity now extends fully into the bottom left

[4]For $\rho = 1, h = 0.03, \Delta t = 10^{-4}$.

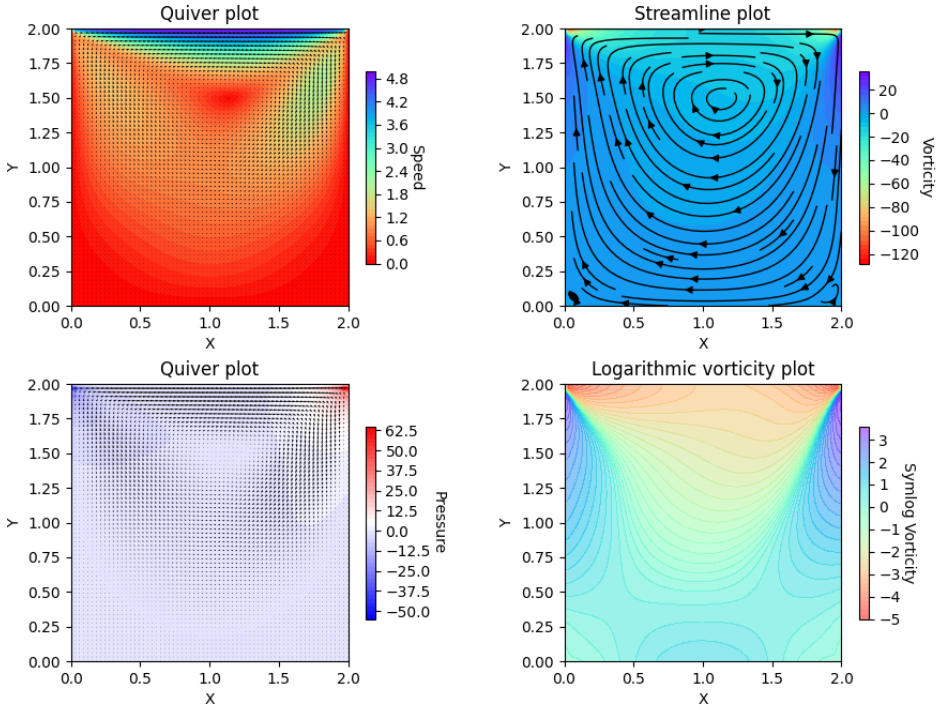


Figure 29: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 50$ and $U = 5, \mu = 0.2$.

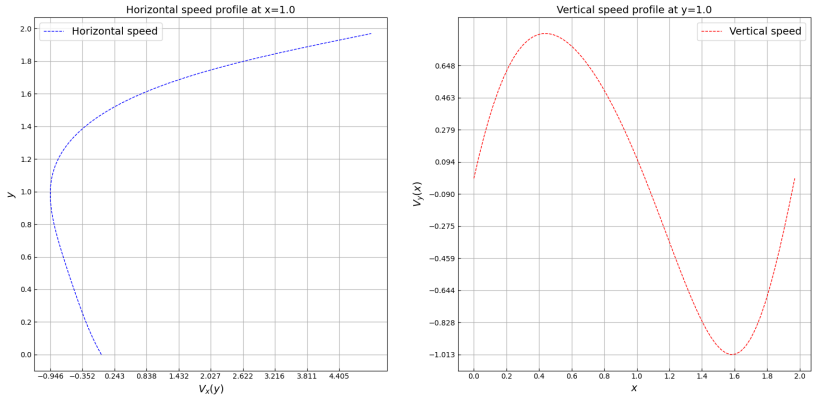


Figure 30: Speed profiles for a driven cavity with $Re = 50$ and $U = 5, \mu = 0.2$.

corner, where the bottom left vortex previously used to be.

Another interesting observation, is that the number of iterations until convergence increases with the Reynolds number. This makes sense: a higher Re means a more turbulent flow, so obviously it will take longer for an equilibrium to form. Furthermore, to get to a Reynolds number of 280, a grid spacing of $h = 0.03$ is needed. When we try to simulate this system with $h = 0.05$, divergence occurs. The grid needs to be refined for the system to be stable.

Interestingly, changing the time step does not allow us to further increase the Reynolds number. This is further indication that the stability criterium (Equations 4.1.1) is not strict enough.

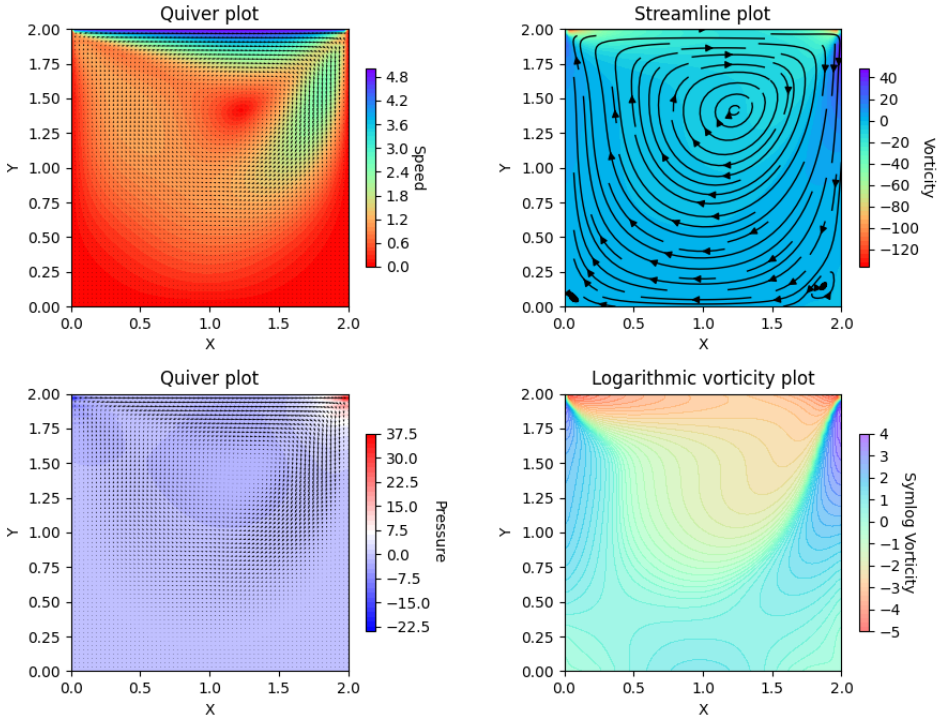


Figure 31: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 125$ and $U = 5, \mu = 0.08$.

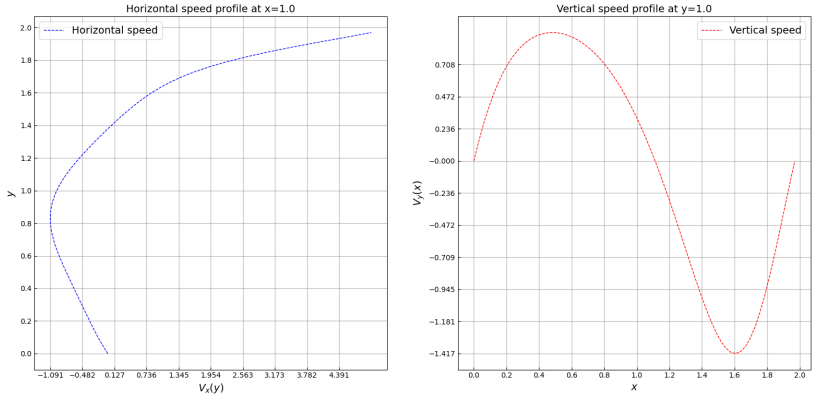


Figure 32: Speed profiles for a driven cavity with $Re = 125$ and $U = 5, \mu = 0.08$.

4.2.3 Decreasing Re

In this section, we will do the opposite as the last section: decrease the Reynolds number as much as possible. The lowest stable Reynolds number[5] is equal to 1.74, with $U = 1, \mu = 1.15$. The system converges after 4565 iterations and $1.86s$. The results are shown on Figures 35 and 36.

Here, we again see very small vortices in the bottom corners. The flow is very laminar and symmetrical, and the speed of the fluid is a lot smaller compared to the systems with high Re. This is obviously because the belt is moving a lot slower. It should also be noted that $Re = 1.74$ is the

[5]For $\rho = 1, h = 0.03, \Delta t = 10^{-4}$. In the first section [(]Convergence criterium], we got Reynolds numbers of 1, but there the density ρ was equal to 0.01.

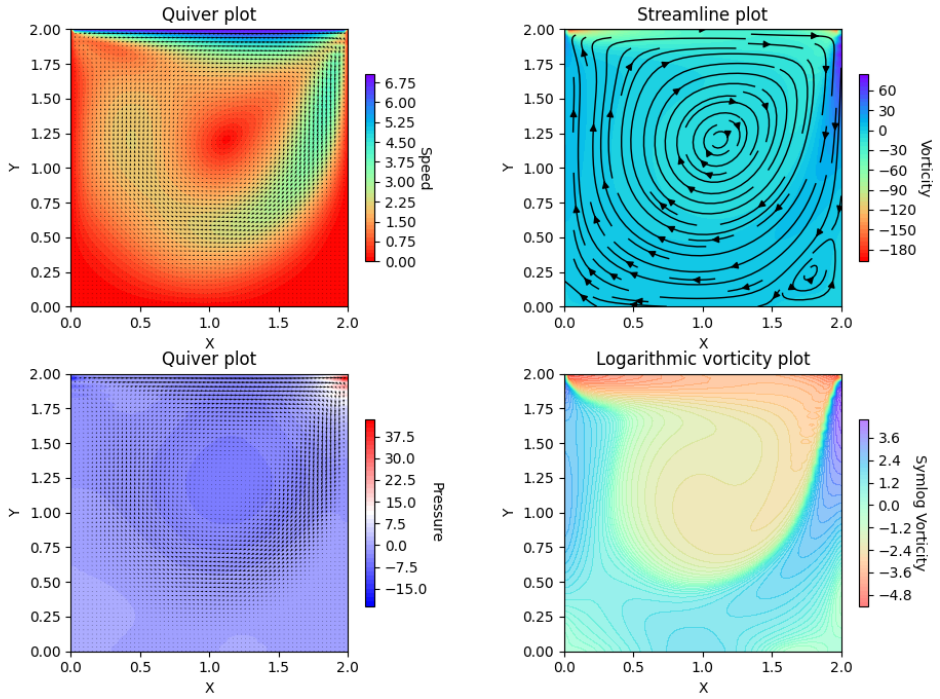


Figure 33: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 280$ and $U = 7, \mu = 0.05$.

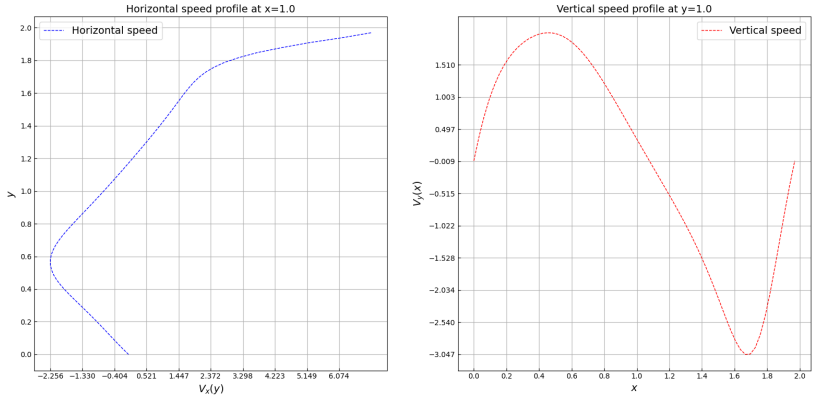


Figure 34: Speed profiles for a driven cavity with $Re = 280$ and $U = 7, \mu = 0.05$.

lowest limit, but only for $\Delta t = 10^{-4}$. By decreasing Δt, we can further decrease Re; for example, for $\Delta t = 5 \cdot 10^{-10}$, we can get $Re = 2 \cdot 10^{-5}$.

5 Exercise 2: Laminar flow

For the second task, we will simulate laminar flow through a pipe. For this exercise, we will assume the the fluid is pushed through the pipe by a constant pressure gradient $F = \frac{\partial p}{\partial x}$. Because of this, the discretization schemes will be changed a little. In the discretization scheme for u (equation 9), we will add an extra term $+F\Delta t$ on the right hand side. The expected speed profile is shown on Figure 37. For this task, have chosen the dimensions $L_x = 2, L_y = 1$.

This system can be solved analytically, the horizontal speed profile through the pipe should be

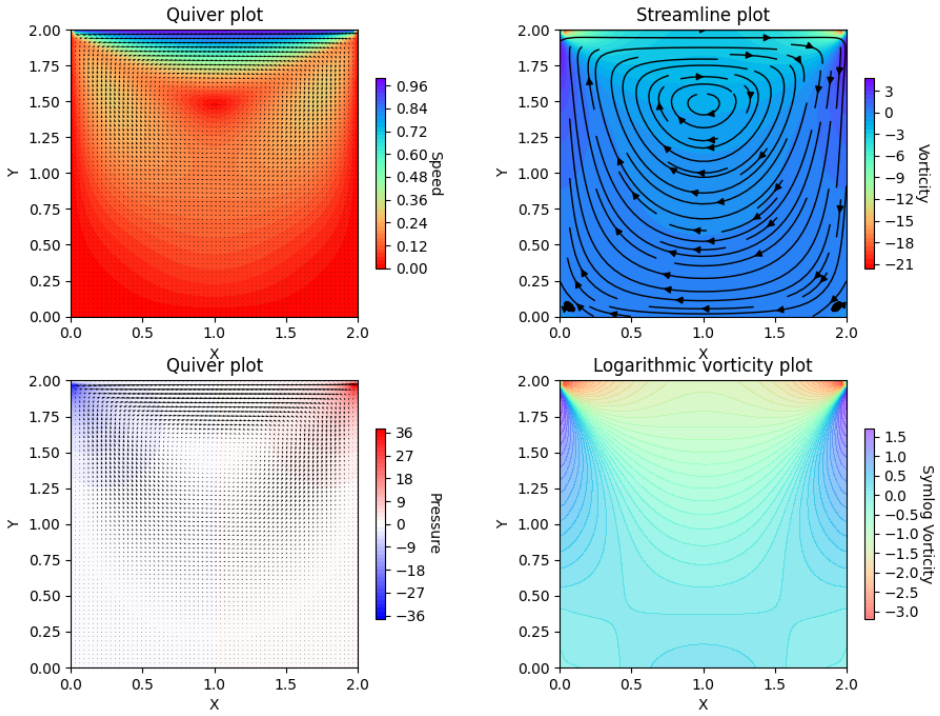


Figure 35: Plots for speed, pressure and vorticity, for a driven cavity with $Re = 1.74$ and $U = 7, \mu = 1.15$.

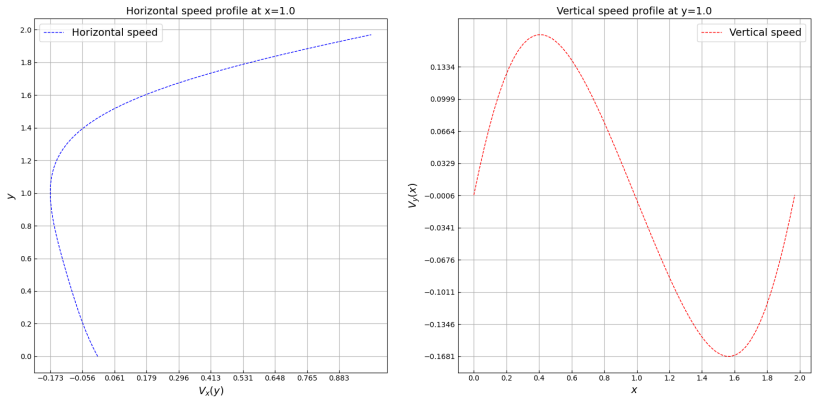


Figure 36: Speed profiles for a driven cavity with $Re = 1.74$ and $U = 1, \mu = 1.15$.

given by:

$$u = U_0 \left(1 - \frac{r^2}{R^2}\right) \tag{14}$$

Checking the convergence behavior of the laminar flow system is a little different than for the driven cavity system, since we don't actually know the maximum speed until after we have simulated the system. We can thus only calculate the Reynolds number after the simulation. We will then substitute U by the average velocity of the field. For our first simulation, we will choose $h = 0.05, \Delta t = 10^{-3}, \mu = 0.5, F = 10^6$. Convergence is reached after 460 iterations and $0.067s$, and the resulting Reynolds number is 2.624. The results are shown on Figures 38 and 39.

[6] We will also keep $\rho = 1$ as constant in this chapter.

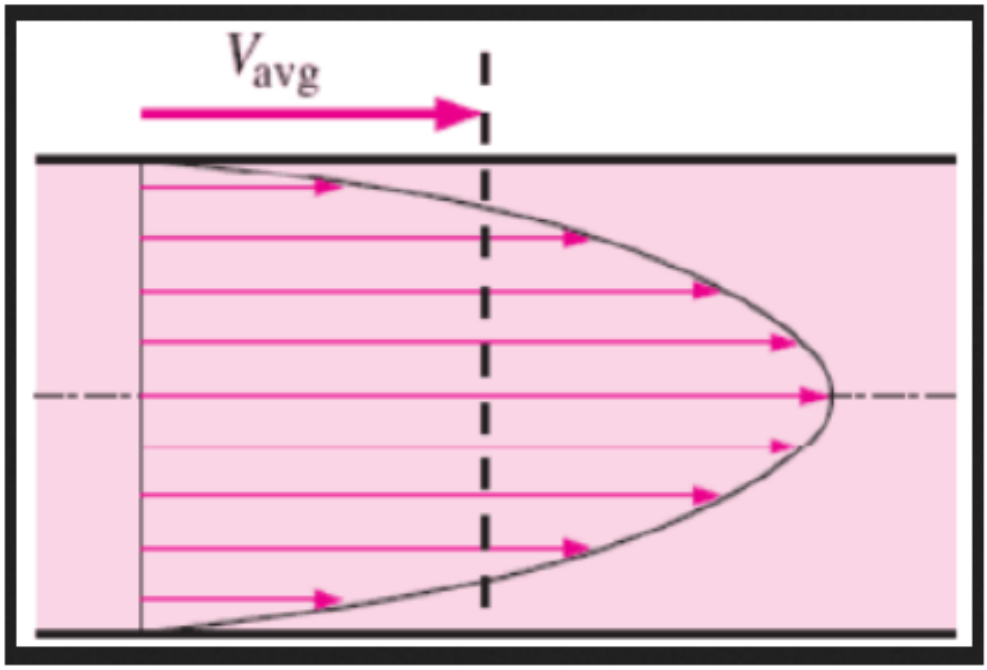


Figure 37: Speed profile for laminar flow through a pipe [VRDL].

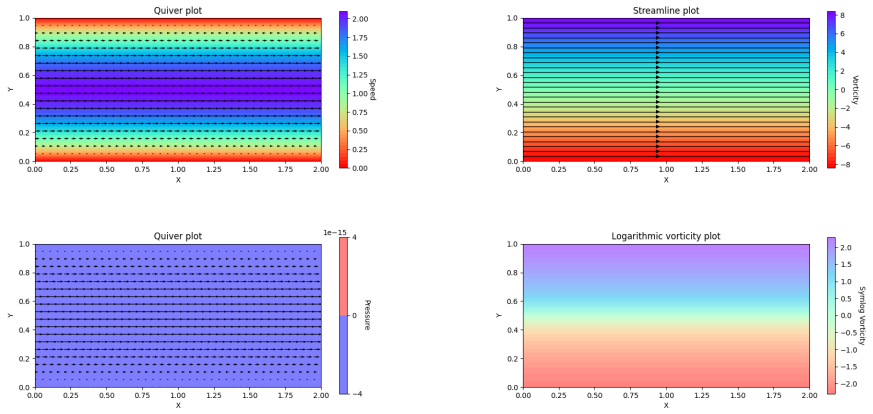


Figure 38: Plots for speed, pressure and vorticity, for laminar flow through a pipe with $\mu = 0.5, F = 10$.

We indeed see the expected quadratic speed profile. Now, we can change the parameters and see how the system changes.

5.1 Increasing F

To begin, we will see what happens if we increase the force through the tube. We will keep working with $h = 0.05, \Delta t = 10^{-3}, \mu = 0.5$. If we choose $F = 100$, convergence is reached after 4.9 iterations and $0.067s$, and the resulting Reynolds number is 2.624. We see that convergence is reached after exactly as many iterations as before, and the Reynolds number the same but 10 time bigger! This could mean that increasing F by a factor (without changing anything else), simply scales the speed by that same factor. The results are shown on Figures 40 and 41.

As we can see, our hypothesis is correct! Scaling F simply scales the speeds of the system. Further analysis reveals that We can scale F up to pretty much whatever we want, the scheme doesn't diverge

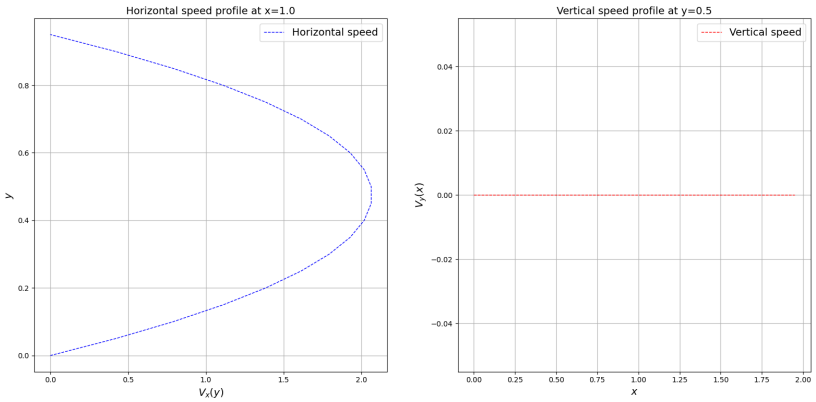


Figure 39: Speed profiles for laminar flow through a pipe with $\mu = 0.5, F = 10$.

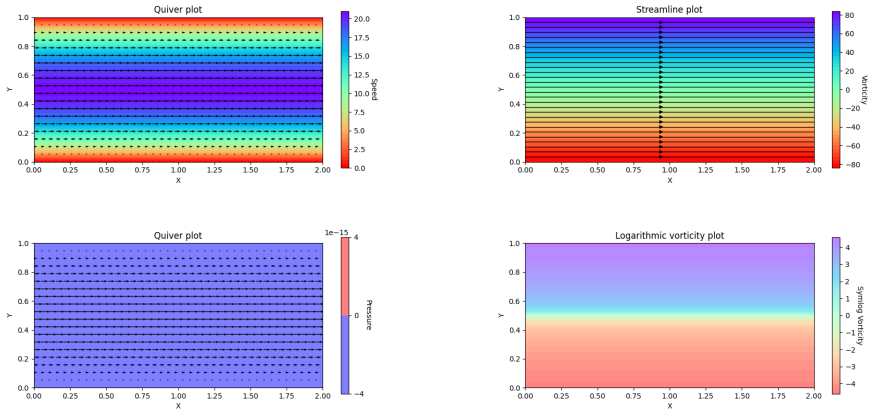


Figure 40: Plots for speed, pressure and vorticity, for laminar flow through a pipe with $\mu = 0.5, F = 100$.

until F is of the order 10^{154}. This also means that we can make the Reynolds number arbitrarily large. because of this, we will fix $F = 10$ in the next section.

5.2 Changing Re by changing the viscosity

Changing the viscosity μ has more interesting effects on the system. We will first try to lower μ to a value of 0.05. The system reaches convergence after 1408 iterations and $0.276s$, and the resulting Reynolds number is equal to 154.74. The results are shown on Figures 42 and 43.

As we can see, lowering the viscosity increases the speed of the system, which is what we would expect. The profile stays quadratic. However, decreasing the viscosity more can break the quadratic profile. If we choose $\mu = 0.005$, the system reaches convergence after 1995 iterations and $0.415s$, and the resulting Reynolds number is equal to 3164.56. The results are shown on Figures 44 and 45.

We see that the profile is lo longer quadratic, and flattens in the middle of the pipe. This is also expected, as a very small viscosity means that the inner layers are held back less by the slower moving outer layers. This allows a thick central layer of fluid to move through the pipe at the same speed.

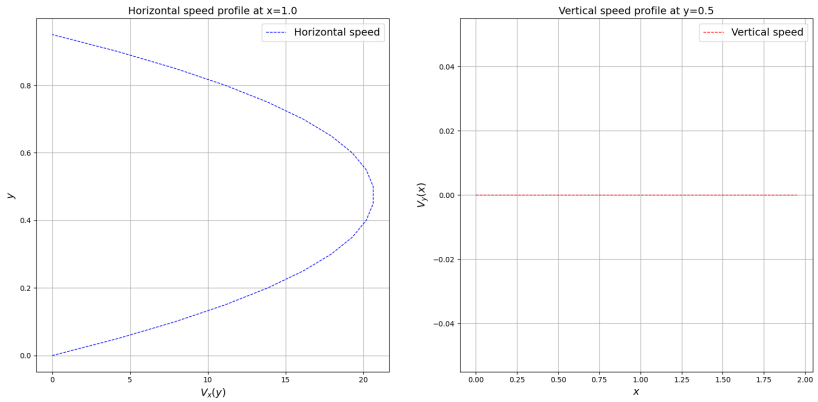


Figure 41: Speed profiles for laminar flow through a pipe with $\mu = 0.5, F = 100$.

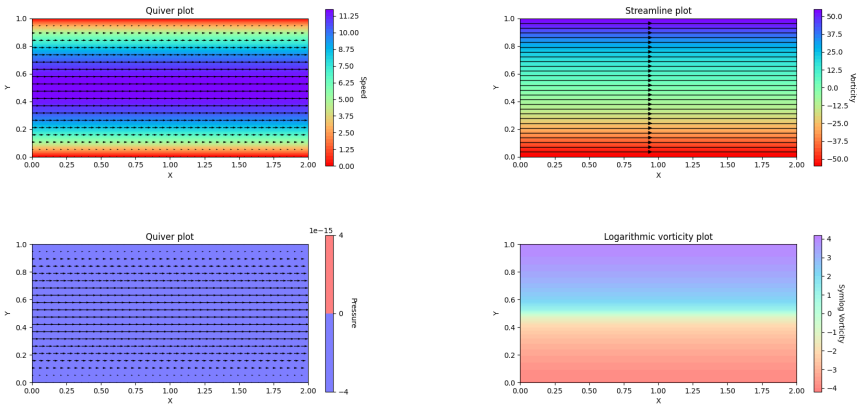


Figure 42: Plots for speed, pressure and vorticity, for laminar flow through a pipe with $\mu = 0.05, F = 10$.

Next, we will increase the viscosity to its maximum possible value. We find that this value is equal to $\mu = 1.28$. Any value greater than this causes divergence. The system reaches convergence after 245 iterations and $0.039s$, and the resulting Reynolds number is equal to 0.42. The results are shown on Figures 46 and 47.

Again, we see a quadratic profile, further solidifying the accuracy of this discretization scheme.

6 Exercise 3: Laminar flow over an obstacle

For the third exercise, the system is very similar to the previous section: we consider laminar flow through a pipe, but with an obstacle. In this system, we should expect a recirculation behind the object (for some choice of parameters), as shown on Figure 48. We will choose to work with a system with parameters $L_x = 3, L_y = 1, h = 0.03, dt = 10^{-4}$. We will now place an obstacle between $x = 0.7$ and $x = 1.3$, $y = 0$ and $y = 0.2$. If we choose $F = 1$ and $\mu = 0.1$, the system converges after 35893 iterations and $14.107s$, and results in $Re = 5.75$. The results are shown on Figure 49.

We see that the system is very similar to laminar flow without an obstacle, with the speed increasing towards the center. We can also observe that the speed increases in the area directly above the obstacle. Since the fluid is incompressible, this makes sense: the flow rate through the pipe has to remain

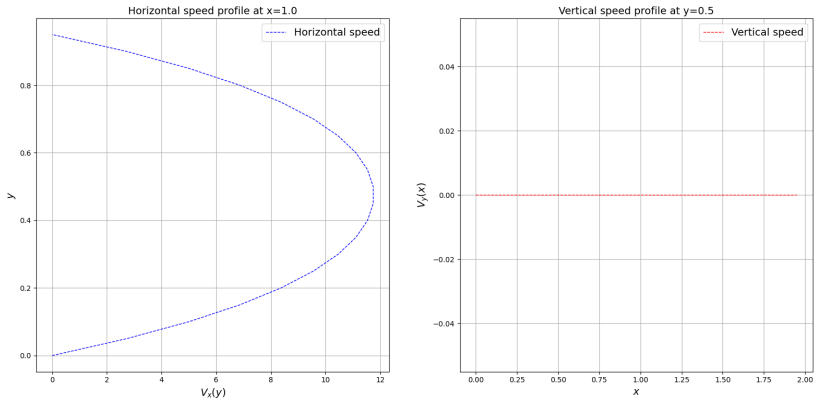


Figure 43: Speed profiles for laminar flow through a pipe with $\mu = 0.05, F = 10$.

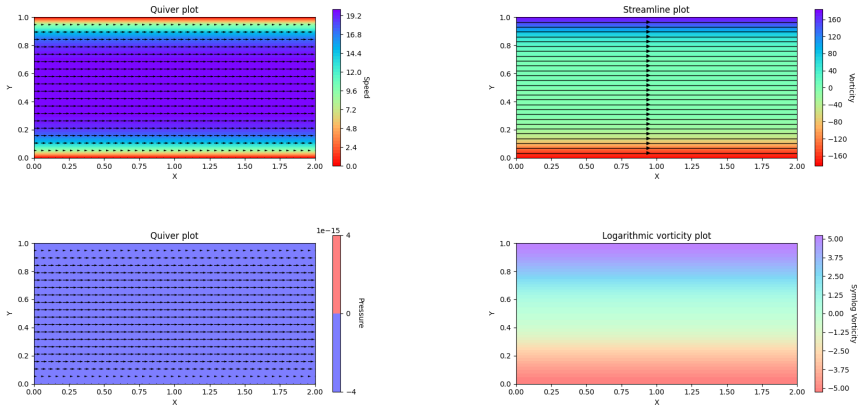


Figure 44: Plots for speed, pressure and vorticity, for laminar flow through a pipe with $\mu = 0.005, F = 10$.

constant at each cross section. Since the object narrows the pipe, the velocity has to increase. As a result of the velocity increase, the pressure has to decrease (Bernoulli's law, $P + \frac{1}{2}\rho v^2 + \rho g h = \text{constant}$). This is also observable in this simulation[7]. This is actually how the wings of an airplane create lift: because the air is flowing faster over the wing than under the wing, the pressure is higher below the wing, which pushes the wing upwards[8].

In the previous . We will now change F and μ to see when recirculation starts to occur.

6.1 Increasing F

We will start by increasing F to a value of 50. This system converges after 37599 iterations and $14.600s$, and results in $Re = 241.23$. The results are shown on Figure 50.

We can see that a recirculation has now appeared behind the obstacle. This makes sense, as the Reynolds number has jumped from 5.75 to 241.23. Furthermore, we see that the speed and vorticity have increased by a factor of ≈ 40, and the pressure by a factor of ≈ 200. This because the fluid is

[7] However, it is more clear in further figures, like Figure 50.
[8] see also on Figure **??**

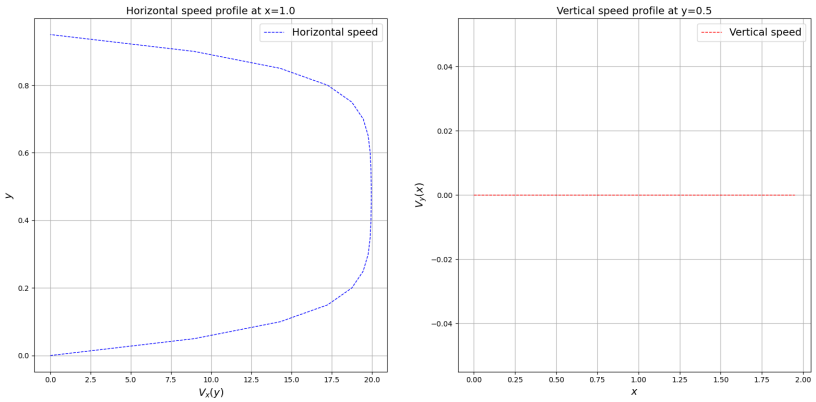


Figure 45: Speed profiles for laminar flow through a pipe with $\mu = 0.005, F = 10$.

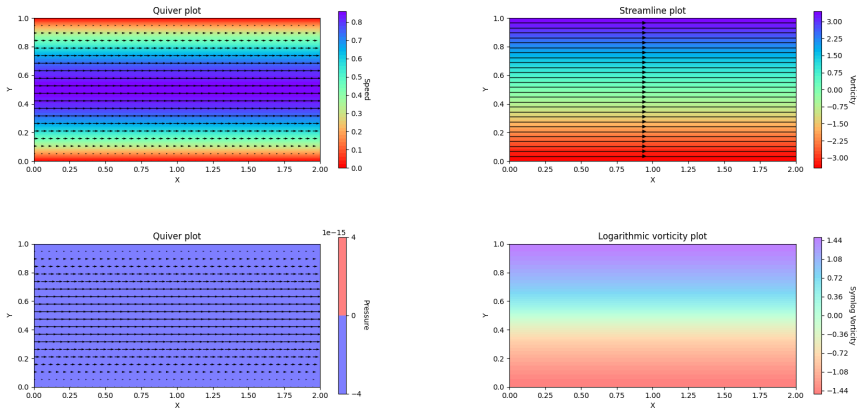


Figure 46: Plots for speed, pressure and vorticity, for laminar flow through a pipe with $\mu = 1.28, F = 10$.

being pushed trough the pipe with a stronger force (which increases the speed and thus the vorticity), causing it to be pushed into the obstacle harder (which increases the pressure).

Next, we will increase the force to a maximum of $F = 205$. Increasing F any further (while keeping other parameters constant), results in divergence. This system converges after 29128 iterations and $11.383s$, and results in $Re = 887.27$. The results are shown on Figure 51.

We can see that the recirculation has grown even larger, and due to the periodic boundary conditions, it even appears again on the left side of the obstacle. Again, we see a sharp increase in speed, vorticity and pressure due to the greater force exerted on the fluid. However, even high a Reynolds number of 887.27, no Eddy currents form. One explanation for this, is that the grid has to be finer for these small vortices to form. However, this would steeply increase number of iterations and the computation time to reach a stable solution.

6.2 Changing μ

Next, we will fix $F = 5$, and study the effect of the viscosity. For this F, the maximum μ which doesn't cause convergence is $\mu = 1.15$. This system converges after 3722 iterations and $1.391s$, and results in

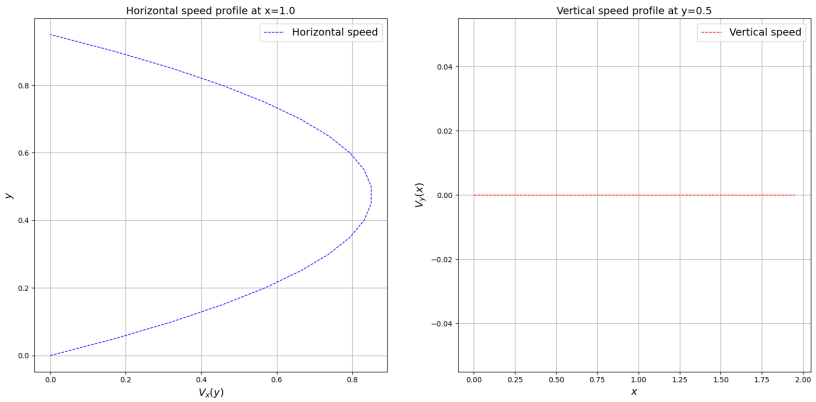


Figure 47: Speed profiles for laminar flow through a pipe with $\mu = 1.28, F = 10$.

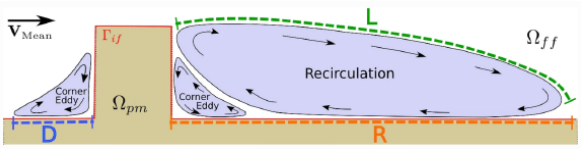


Figure 48: Recirculation behind an obstacle in a pipe with laminar flow.

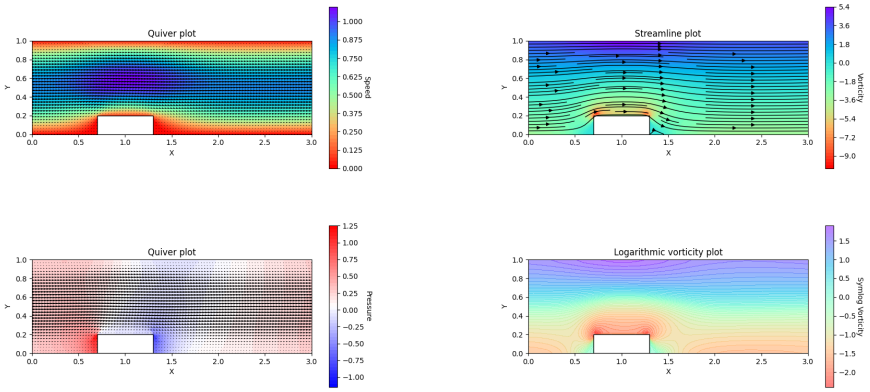


Figure 49: Plots for speed, pressure and vorticity, for laminar flow through a pipe with an obstacle, and $\mu = 0.1, F = 1$.

$Re = 0.22$. The results are shown on Figure 52.

We again see no circulation, and the speed, vorticity, and pressure are all relatively low, as is usual for Reynolds numbers of this order. We also note the low amount of iterations needed to reach stability; it seems that lower Reynolds numbers result in faster convergence.

Now, we will decrease μ. First, we choose $\mu = 0.05$. This system converges after 61237 iterations and $24.171s$, and results in $Re = 103.44$. The results are shown on Figure 54.

We again see a recirculation, just like when we increased the speed. However, the speed, vorticity and pressure do not shoot up as much as before, when we increased F. We now decrease the viscosity further, $\mu = 0.02$. This system converges after 61237 iterations and $24.171s$, and results in $Re = 103.44$.

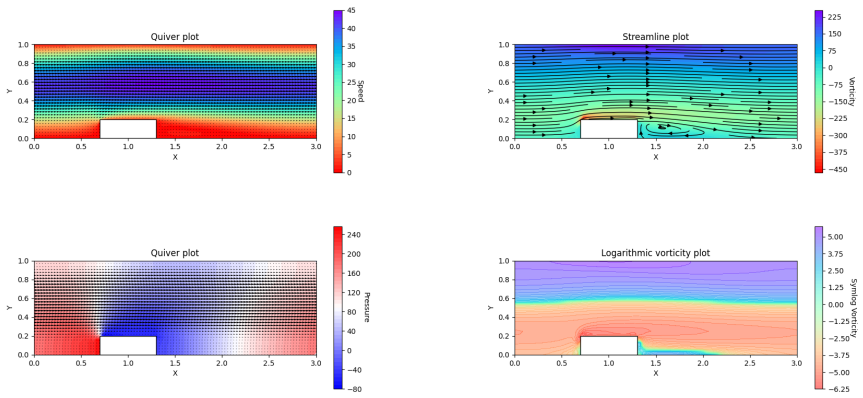


Figure 50: Plots for speed, pressure and vorticity, for laminar flow through a pipe with an obstacle, and $\mu = 0.1, F = 50$.

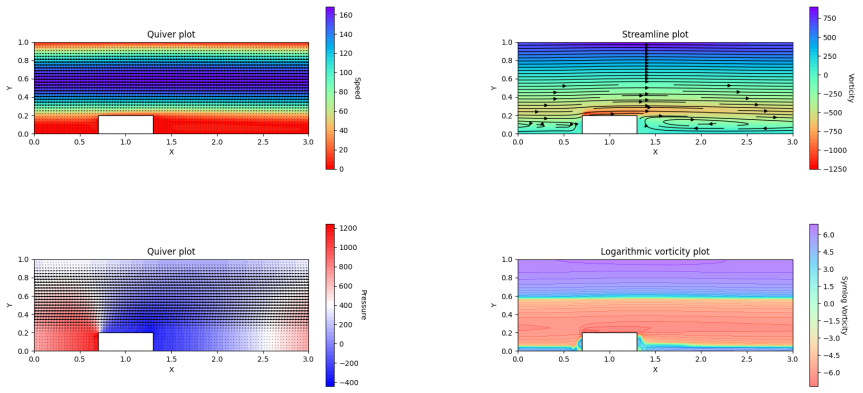


Figure 51: Plots for speed, pressure and vorticity, for laminar flow through a pipe with an obstacle, and $\mu = 0.1, F = 205$.

The results are shown on Figure 54.

And again, we see an even bigger recirculation. We can see once more that vortex circulation is heavily linked to the Reynolds number, meaning that vortices can form both by increasing the pressure gradient F, and by decreasing the viscosity μ of the fluid.

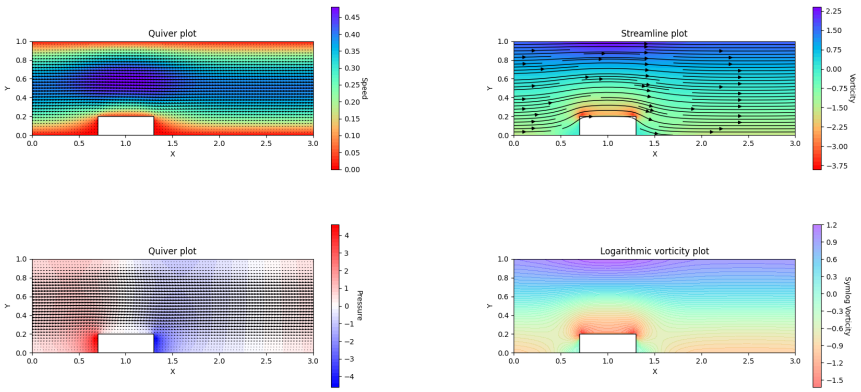


Figure 52: Plots for speed, pressure and vorticity, for laminar flow through a pipe with an obstacle, and $\mu = 1.15, F = 5$.

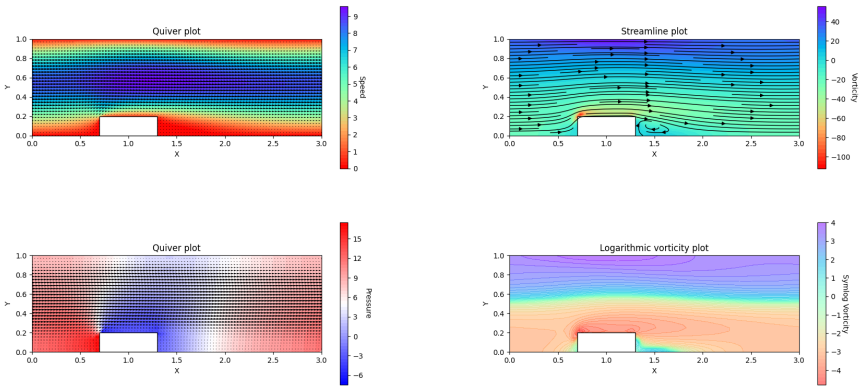


Figure 53: Plots for speed, pressure and vorticity, for laminar flow through a pipe with an obstacle, and $\mu = 0.05, F = 5$.

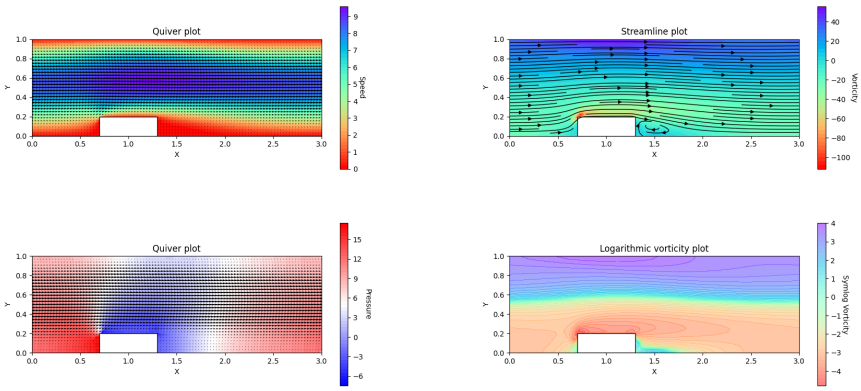


Figure 54: Plots for speed, pressure and vorticity, for laminar flow through a pipe with an obstacle, and $\mu = 0.02, F = 5$.

6.3 Changing the obstacle height

In this next section, we will briefly discuss what happens when we change the object's size. We will use $F = 20, \mu = 0.1$. We will do three simulations, with object heights of 0.4, 0.6, and 0.8. The first system converges after 21557 iterations and $8.503s$, and results in $Re = 152.47$. The results are shown on Figure 55. The second converges after 18859 iterations and $7.438s$, and results in $Re = 70.75$. The results are shown on Figure 56. The third converges after 13081 iterations and $5.1421s$, and results in $Re = 12.54$. The results are shown on Figure 57.

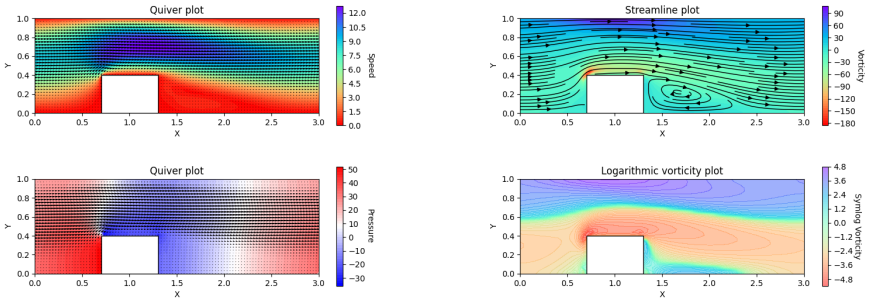


Figure 55: Plots for speed, pressure and vorticity, for laminar flow through a pipe with an obstacle with height 0.4.

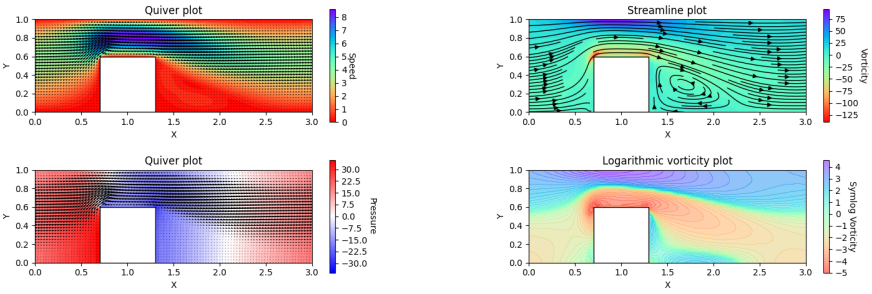


Figure 56: Plots for speed, pressure and vorticity, for laminar flow through a pipe with an obstacle with height 0.6.

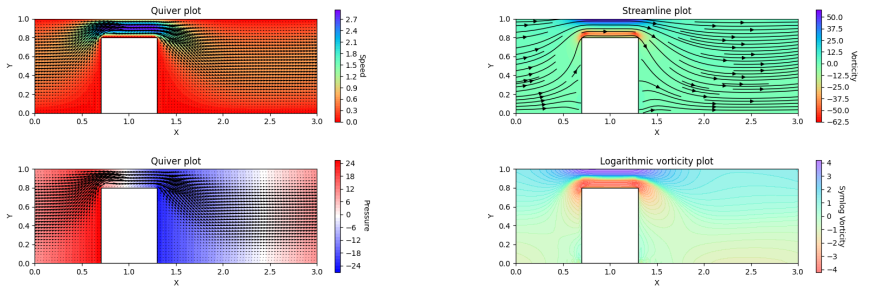


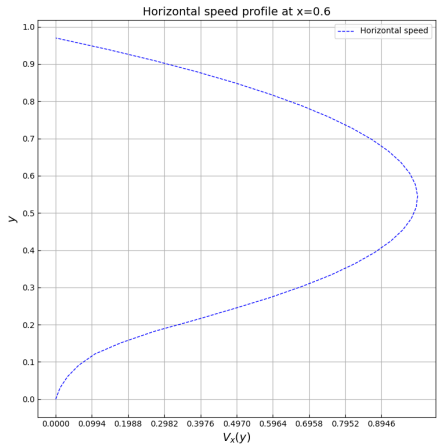
Figure 57: Plots for speed, pressure and vorticity, for laminar flow through a pipe with an obstacle with height 0.8.

On these plots, we see that the average speed and pressure decreases as the obstacle covers more of the cross section of the pipe. Furthermore, we can see that the recirculation grows for heights of 0.4

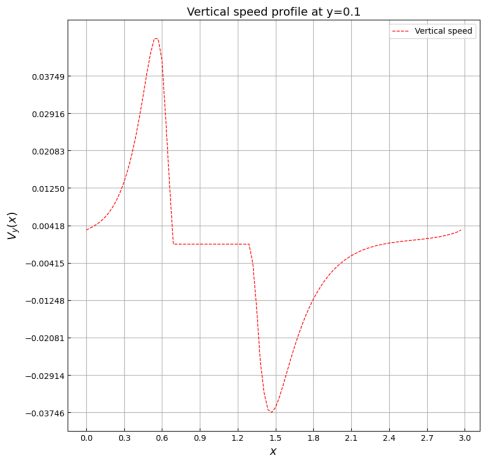
and 0.6, but interestingly, it seems to have dissapeared for a height of 0.8. This is likely because the speed is too slow to cause a recirculation.

6.4 Speed profiles

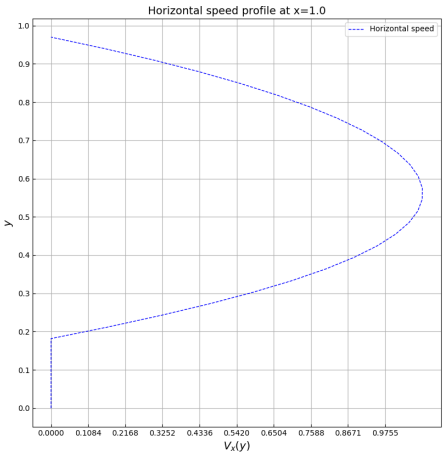
As a final analysis of an object in a pipe, we will take a look at some speed profiles for the two systems shown on Figures 49 and 41. For both systems, we will show vertical speed profiles for $y = 0.1, y = 0.3, y = 0.7$ and horizontal speed profiles for $x = 0.6, <= 1.0, x = 1.4$. These plots are shown on Figures 58 and 58



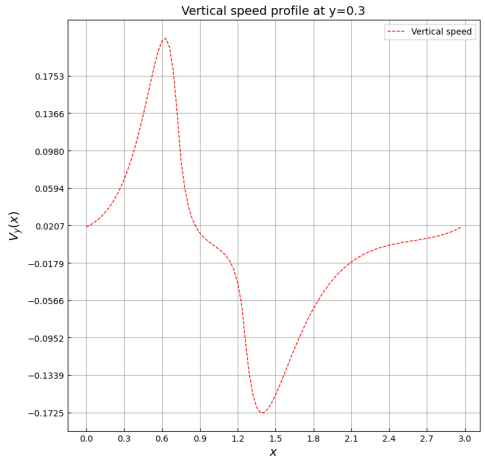
(a) Horizontal speed profile at $x = 0.6$.



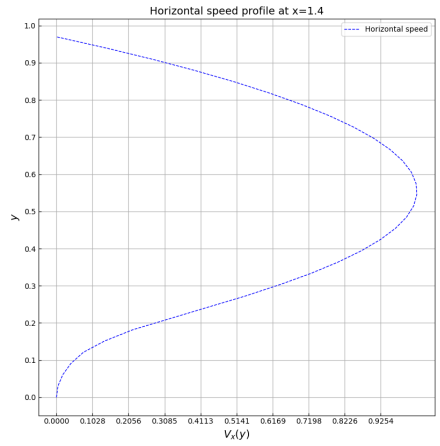
(b) Vertical speed profile at $y = 0.1$



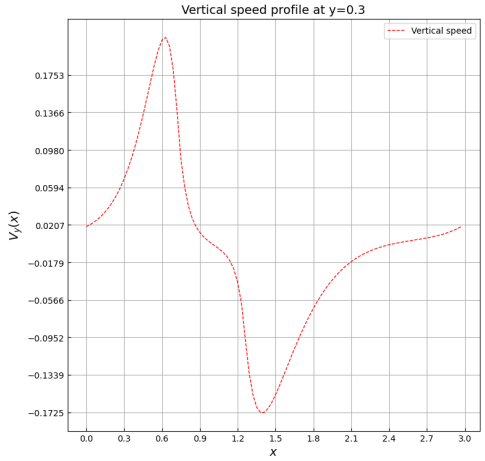
(c) Horizontal speed profile at $x = 1.0$



(d) Vertical speed profile at $y = 0.3$

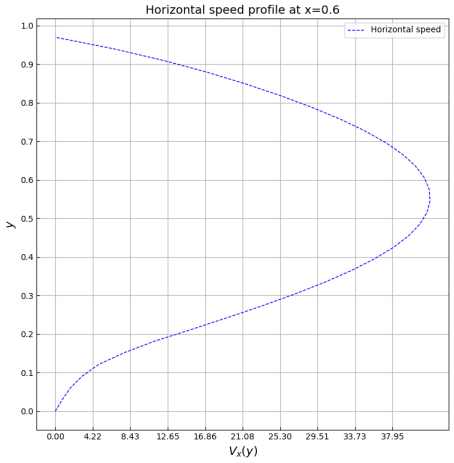


(e) Horizontal speed profile at $x = 1.4$

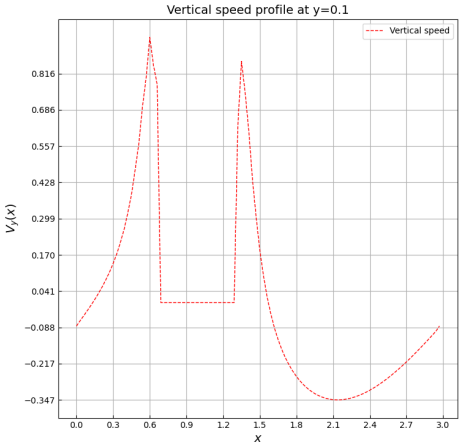


(f) Vertical speed profile at $y = 0.7$

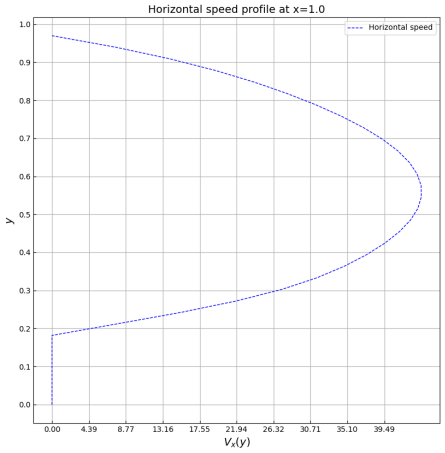
Figure 58: Speed profile plots for $F = 1, \mu = 0.1$.



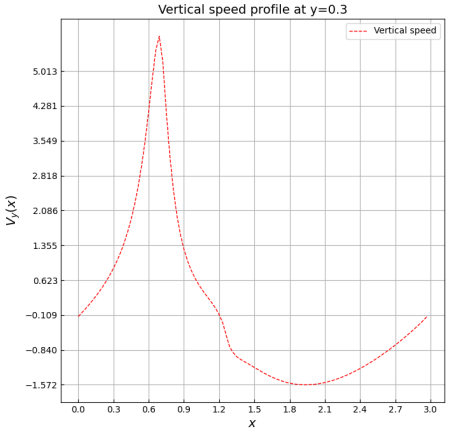
(a) Horizontal speed profile at $x = 0.6$



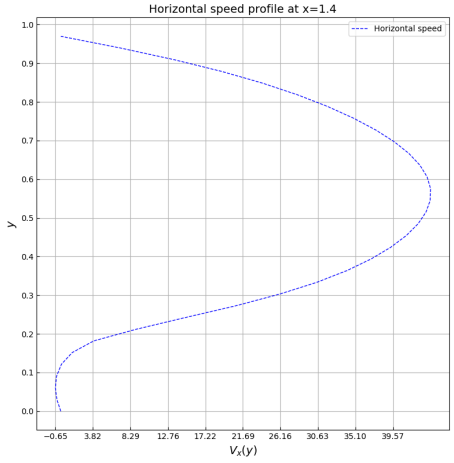
(b) Vertical speed profile at $y = 0.1$



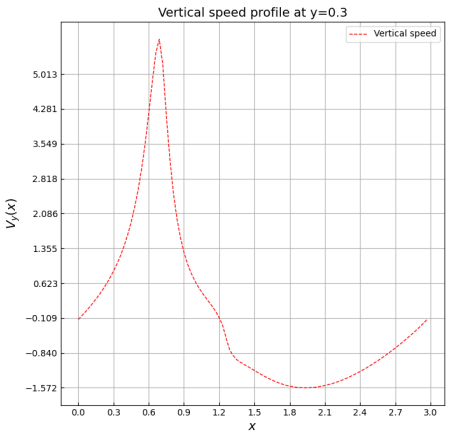
(c) Horizontal speed profile at $x = 1.0$



(d) Vertical speed profile at $y = 0.3$



(e) Horizontal speed profile at $x = 1.4$



(f) Vertical speed profile at $y = 0.7$

Figure 59: Speed profile plots for $F = 50, \mu = 0.1$.

Comparing the subplots for both systems, we can note a few key similarities and differences:

1. The scale is very different for the two systems; the speeds are much higher for system 2 (with $F = 50$), than for system 1 (with $F = 1$). This makes sense, as a higher pressure gradient results in more force and a faster moving liquid.

2. For both systems, we can see that the speed increases when the pipe is made narrower by the obstacle.

3. We can clearly see the presence of a recirculation behind the object in system 2. At the bottom Figure 59e, we can see that the horizontal speed briefly becomes negative, and then increases again. This does not happen on Figure 58e, where no circulation occurs. Similarly, we can see that the vertical speed directly after the obstacle is positive for system 2 (Figure 59d), while it is negative for system 1 (Figure 58d).

7 Exercise 4: Laminar flow over trough

For the final exercise, we will place not one, but two objects in the pipe, creating a trough. We will then again simulate laminar flow through the pipe, and see how the liquid behaves inside the through. The dimensions of the system are $L_x = 2, L_y = 1$, and the trough will have length 2 and depth 0.2. For all following simulations, we will use $h = 0.02, \Delta t = 10^{-4}, \rho = 1$. For the first simulation, we will choose the parameters $F = 0.1, \mu = 0.2$.

This system converges after 19944 iterations and $9.419s$, and results in $Re = 0.11$. The results are shown on Figure 60.

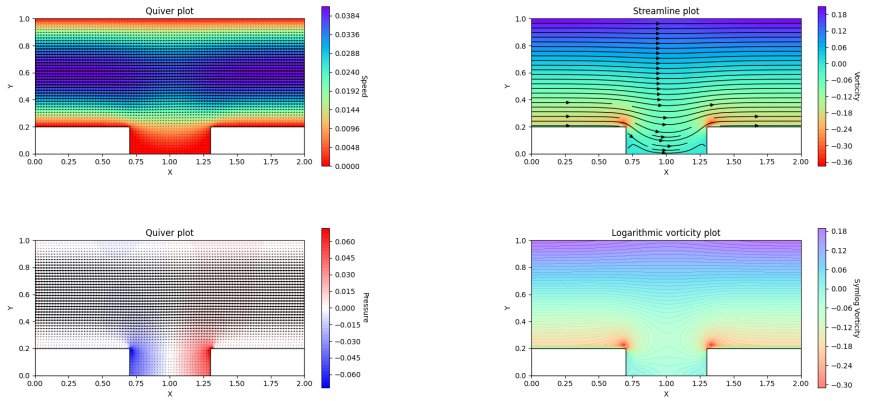


Figure 60: Plots for speed, pressure and vorticity, for laminar flow through a pipe with a trough, and $\mu = 0.2, F = 0.1$.

We can clearly see that for very small Reynolds numbers, no circulation appears in the trough. We will now play around with the parameters F and μ to see how this changes the system.

7.1 Increasing F

First, we are going to increase the pressure gradient F. First, we will increase F to a value of 50. This system converges after 20690 iterations and $10.092s$, and results in $Re = 51.60$. The results are shown on Figure 61.

On these figures, we can see some recirculation forming behind the first obstacle, similar to the obstacle in the pipe. However, we don't yet see full recirculation in the trough. For this, we will increase the pressure gradient to a maximal value of 150. This system converges after 19339 iterations and $9.425s$, and results in $Re = 152.99$. The results are shown on Figure 62.

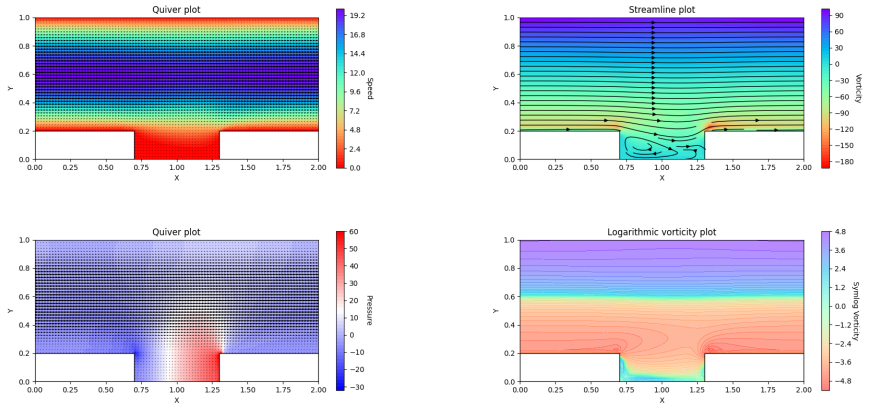


Figure 61: Plots for speed, pressure and vorticity, for laminar flow through a pipe with a trough, and $\mu = 0.2, F = 50$.

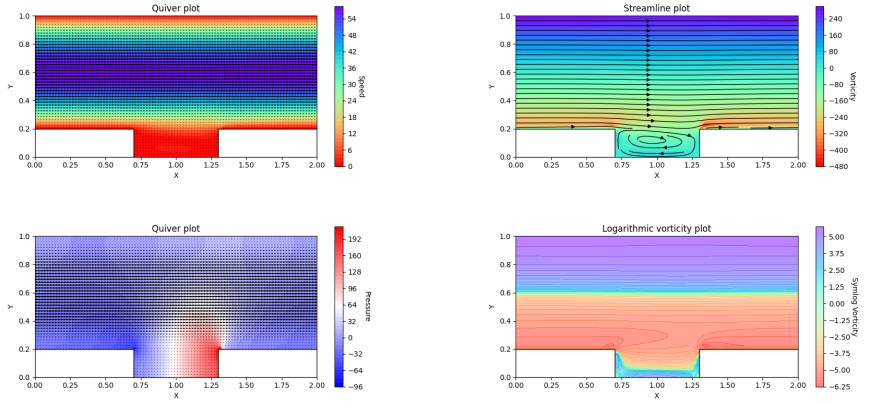


Figure 62: Plots for speed, pressure and vorticity, for laminar flow through a pipe with a trough, and $\mu = 0.2, F = 150$.

Now, we do actually see full recirculation inside of the trough. This recirculation will be further analyzed in section [Speed profiles].

7.2 Changing μ

Now, we will study the effect of the viscosity μ. We will start by decreasing to $\mu = 0.12$, with $F = 10$. This system converges after 29925 iterations and $14.336s$, and results in $Re = 28.78$. The results are shown on Figure 63.

Again, we see a partial recirculation in the trough. To get full recirculation, we will further decrease to a minimal value of $\mu = 0.05$. This system converges after 62598 iterations and $30.266s$, and results in $Re = 162.29$. The results are shown on Figure 64.

We can see that full recirculation occurs. Again, we see a clear correlation between the Reynolds number and the formation of circulation.

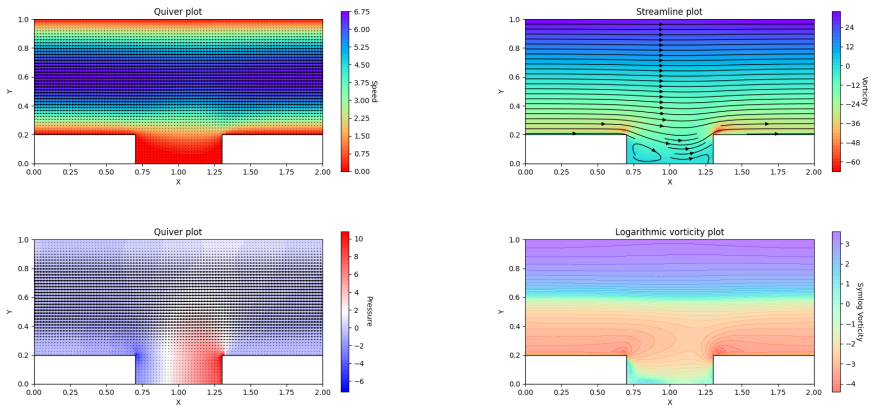


Figure 63: Plots for speed, pressure and vorticity, for laminar flow through a pipe with a trough, and $\mu = 0.12, F = 10$.

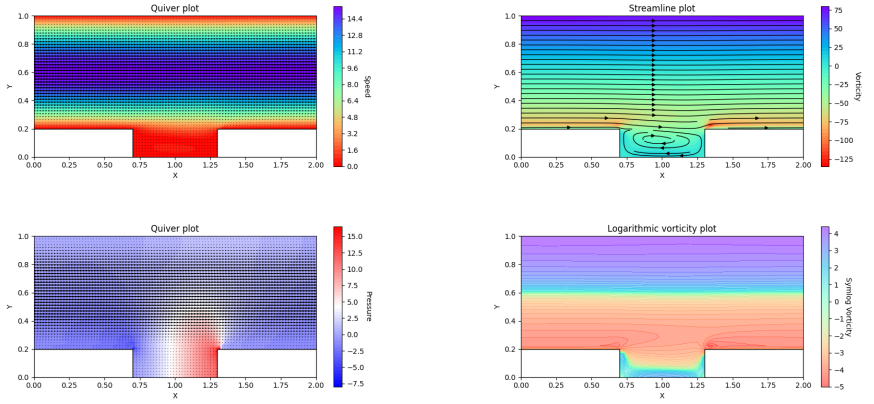
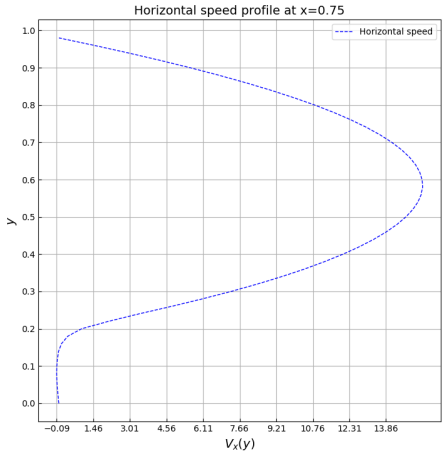


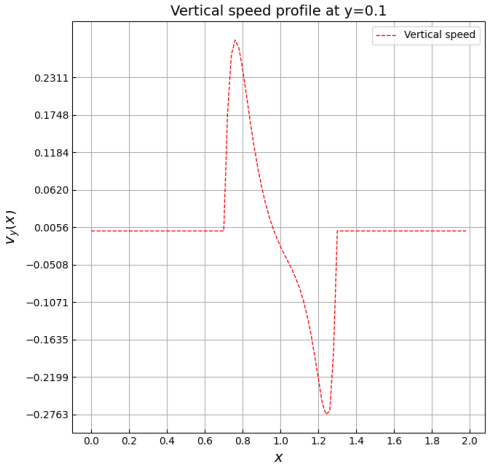
Figure 64: Plots for speed, pressure and vorticity, for laminar flow through a pipe with a trough, and $\mu = 0.05, F = 10$.

7.3 Speed profiles

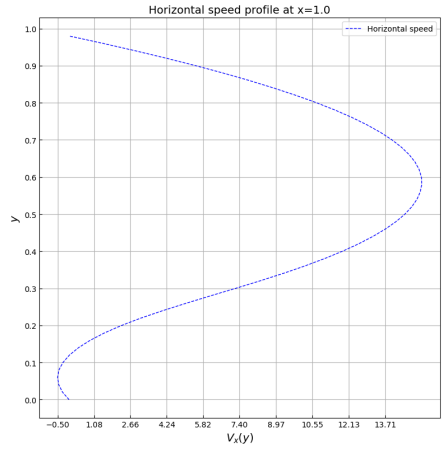
Just like with the obstacle in the pipe, we will look at three horizontal speed profiles, and three vertical speed profiles. We have selected the system shown on Figure 64, and have chosen the profiles $x = 0.1x = 0.25x = 0.5$, and $y = 0.75y = 1.0y = 1.25$. These plots are shown on Figure 65.



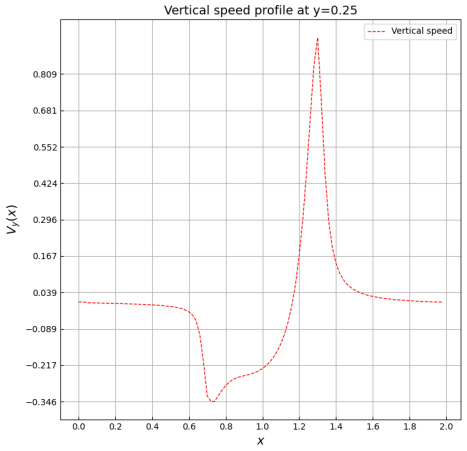
(a) Horizontal speed profile at $x = 0.6$.



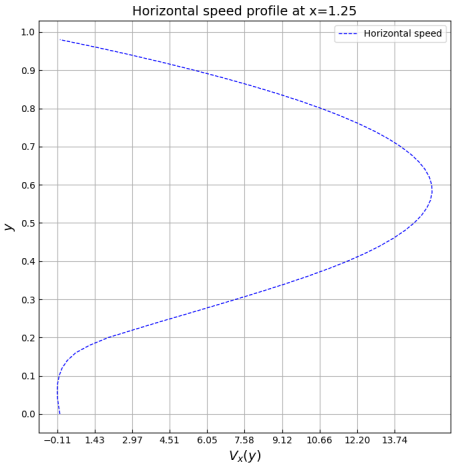
(b) Vertical speed profile at $y = 0.1$



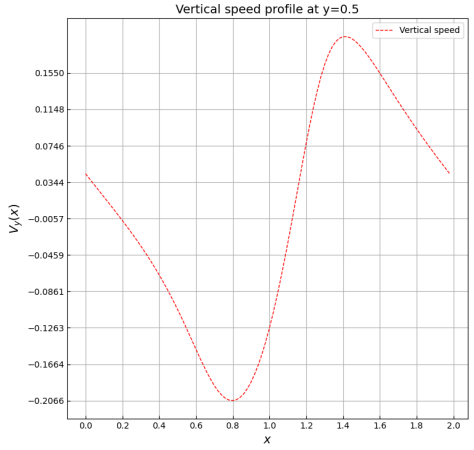
(c) Horizontal speed profile at $x = 1.0$



(d) Vertical speed profile at $y = 0.3$



(e) Horizontal speed profile at $x = 1.4$



(f) Vertical speed profile at $y = 0.7$

Figure 65: Speed profile plots for $F = 10, \mu = 0.05$.

When looking at these plots, we notice the following:

1. The maximum velocity doesn't change notably in the area over the trough. This would be the case without recirculation, as then the cross section of the pipe would increase. Thus, to keep the flow constant, the velocity would have to decrease.

2. The trough itself acts kind of like a driven cavity. If we look to the bottom part of Figure 65c (which is the horizontal speed profile in the trough), we see that it is very similar to the horizontal speed profile in Figure 12. Furthermore, the central part of Figure 65b looks similar to the vertical speed profile in Figure 12. The trough can thus be considered as a driven cavity, but driven by a fluid instead of a moving no-slip surface.

3. When we look at the vertical speed profile right above the trough (Figure 65d), we see that the fluid gets sucked into the trough at the beginning of the trough, and gets pushed out of the trough at the end of the trough.

8 Conclusion

In conclusion, we can see that it is possible to find realistic numerical approximations for solutions to the Navier-Stokes equations using Python. In general, we can see that increasing the Reynolds number leads to a more chaotic fluid flow, where more vortices form. Furthermore, we can observe the quadratic speed profile when considering laminar flow through a pipe. Finally, we can observe recirculations behind obstacles and inside of a trough, as expected.

9 Extras

To finish this report, I would like to add a few figures from systems I have experimented with.

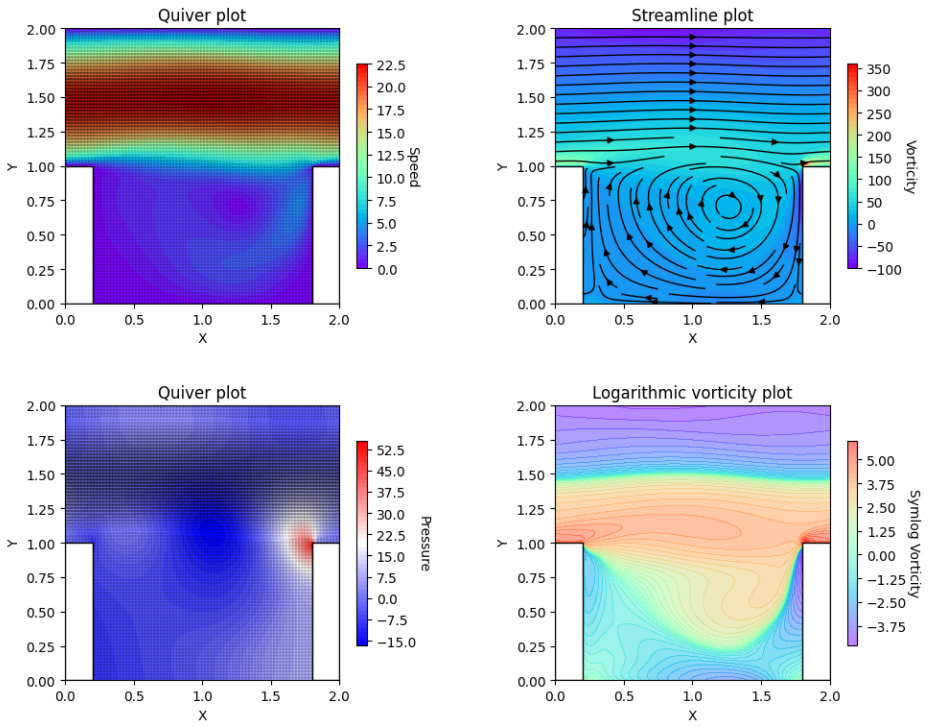


Figure 66: Laminiar flow over a deep trough at a very fine grid. We can clearly see the similaroty to the driven cavity.

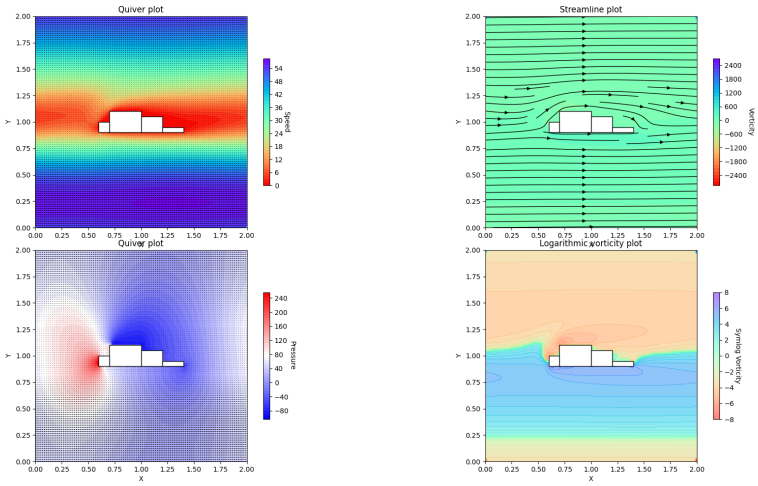


Figure 67: Simulation of a plane wing. We can see that the pressure is lower on top of the wing, which creates lift.

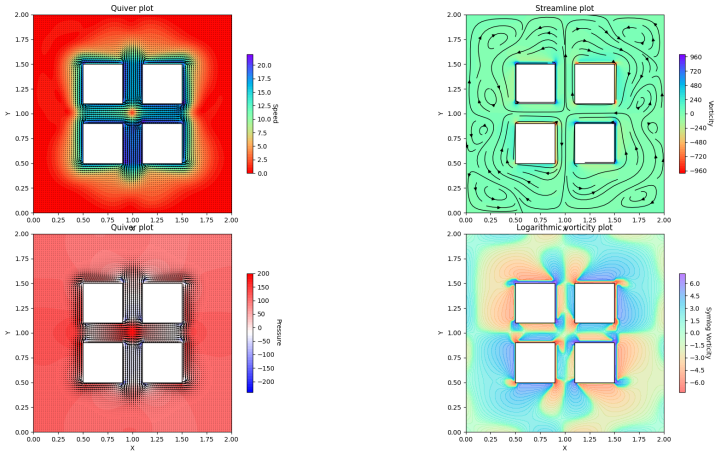


Figure 68: Simulation of 4 spinning belts in a box of fluid.

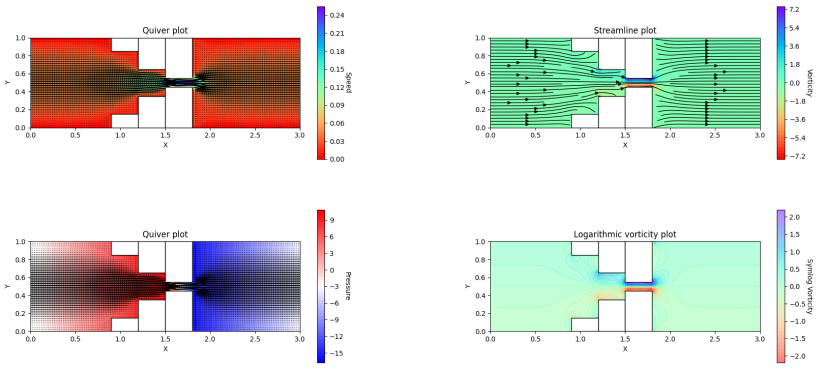


Figure 69: Simulation of a fluid being pushed through a very small hole, so that the velocity becomes very high.

References

[AHA13] S. Adami, X.Y. Hu, and N.A. Adams. A transport-velocity formulation for smoothed particle hydrodynamics. *Journal of Computational Physics*, 241:292–307, 2013.

[VRDL] N. Van Remortel and F. De Lillo. Opdracht Examen Eerste Zittijd 2025.