Pressure-Velocity Coupling: SIMPLE-Based Steady State Solver

Student: 4M5ZS

Cavendish Laboratory, Department of Physics, J J Thomson Avenue, Cambridge. CB3 0HE

Abstract

This text presents a revised SIMPLE algorithm for the pressure-velocity coupling and the results when it is applied for the quadratic lid-driven cavity. We find that the algorithm performs well for a Reynolds number of 100, as compared to the literature. However, for a Reynolds number of 1000, the results deviate significantly, owing to the inability of the algorithm to capture high non-linearity in the governing equations.

1. Introduction

This report discusses the simulation results for the incompressible 2D lid-driven cavity, a case commonly used to the test incompressible fluid dynamics (CFD) solvers. The solver used here employs the finite volume method (FVM), and is built upon an unstructured mesh, implemented in C++. While the solver can accommodate polyhedral cells of any shape, a uniform Cartesian mesh will be used for the simulations. Gauss-Seidel algorithm is used to solve linear equations.

Section 2 presents governing equations for an incompressible fluid (when fluid density is assumed constant), as derived from the Navier Stokes (NS) equations.

Section 3 describes the lid-driven cavity problem and the reason for its popularity in measuring the accuracy of CFD solvers. Furthermore, section 4 presents the implementation details of FVM used within this paper to obtain a solution to the lid-driven cavity.

Section 5 presents the SIMPLE algorithm, originally conceived by Spalding and Patankar, used to solve for the velocity and pressure in a steady-state incompressible system. This paper uses a modified version of said algorithm, which will be detailed in section 6.

The computational set up used to solve the revised form of the algorithm is presented in section 7.

Lastly, section 8 contains comparisons of the associated results obtained herein, whereas section 9 gives a comparison to existing work on this problem.

2. Governing equations

As with most discussions in the field of fluid dynamics, our point of departure is the Navier Stokes equations. Let ρ, $\mathbf{u}$, P, T and e denote the density, velocity, pressure, temperature and energy at a point in the fluid respectively. Let μ and λ denote the dynamic viscosity and thermal conductivity of the heat, respectively. In absence of external forces, the Navier-Stokes equations may be formulated

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

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) - \nabla \cdot \left[\mu(\nabla \mathbf{u} + (\nabla - \mathbf{u})^T)\right] = -\nabla(P + \frac{2}{3}\mu \nabla \cdot \mathbf{u})$$

$$\frac{\partial (\rho e)}{\partial t} - \nabla \cdot \left[\mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \cdot \mathbf{u}\right] + \nabla \cdot (\rho e \mathbf{u}) = -\nabla \cdot \left(\frac{2}{3}\mu(\nabla \cdot \mathbf{u})\mathbf{u}\right) - \nabla \cdot (P\mathbf{u}) - \nabla \cdot (\lambda \nabla T).$$

In order, these are known as the continuity-, momentum- and energy equations. To close the system, an equation of state relating ρ, P and T; and constitutive relations on e with T, and λ, μ with P and T are needed.

In the limit where ρ assumed constant, the temporal derivative disappears in the continuity equation. Dividing by the density then yields the divergence-free condition $\nabla \cdot \mathbf{u} = 0$. Given the divergence-free condition, the second term on the right-hand side of the momentum equation vanishes. Under the assumption of smoothness on the velocity, it may also be seen that $\nabla \cdot (\nabla \mathbf{u})^T = 0$. If the dynamic viscosity, μ, is assumed independent of temperature, the first two equations are decoupled from the third. Lastly, dividing by density, ρ also in the momentum equation, we get the incompressible flow formulation known as the pressure-velocity (PV) coupling:

$$\nabla \cdot \mathbf{u} = 0$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) - \nabla \cdot (\nu \nabla \mathbf{u}) = -\nabla p \tag{1}$$

The dynamic viscosity μ and static pressure P were divided by density to yield the kinematic viscosity ν and the kinematic pressure p.

3. The Lid-Driven Cavity

The lid-driven cavity is often used as a benchmark within incompressible fluid simulations to gauge how well a solver can resolve velocity and pressure in steady-state flow at varying Reynolds number. In two dimensions, the lid-driven cavity takes the simple form of a rectangular domain with three impermeable walls fixed and one (typically the top) moving at some constant velocity, u. A no-slip condition is applied at the walls, such that all velocity components (x and y in the 2D case) take on Dirichlet boundary conditions corresponding to the velocity of the respective walls.

For the form of the problem studied here, a square domain with sides of 0.1 meter and a top boundary moving at a velocity of 1m/s, is assumed. An illustration, taken from [1], can be seen in figure 1.

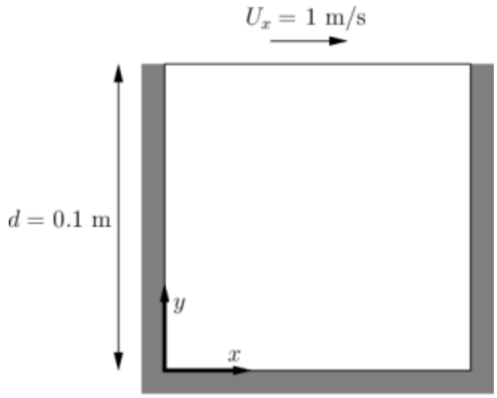

Figure 1: Lid-Driven Cavity, as presented in [1]

These parameters make for a reference length $L = 0.1m$ and reference velocity $u = 1m/s$. Consequently, for a Reynolds number of 100, the constant kinematic viscosity calculates to $\nu = 0.001$, whereas for a Reynolds number of 1000, $\nu = 0.0001$.

Dirichlet conditions on velocity demand Neumann conditions on pressure, in order that the pressure may adjust in accordance with the PV coupling. Here, a zero Neumann boundary condition is applied to the pressure at all boundaries.

Given that a steady state solution is sought, we see immediately that the temporal derivative vanishes from the PV coupling. The system to be solved over the domain is thus reduced to

$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

$$\nabla \cdot (\mathbf{uu}) - \nabla \cdot (\nu \nabla \mathbf{u}) = -\nabla p. \tag{3}$$

4. The Finite Volume Method

The underlying method used in the simulations presented in this text, and indeed in the SIMPLE algorithm, is that of the Finite Volume Method (FVM). The finite volume method relies on a division of the domain of interest into polyhedral cells that can be of varying volume (or area, in the case of 2D). The cells are partitioned by faces, over which the flux in fluid quantities is evaluated. Solutions are calculated in terms of volume-averaged integrals over a cell, *with the assumption that this average is also obtained at a cell's volumetric centroid*.

Thus, under FVM, solutions are generally updated using the following procedure

1. Integrate one or more governing equations over each cell volume
2. Use the divergence theorem to obtain integrals over cell faces
3. Interpolate field values to faces (using available values where necessary to obtain a system of linear equations for one or more variables)
4. Solve the system of linear equations.
5. Substitute updated values into a different governing equation to solve for other variables in a similar fashion
6. Repeat until convergence or until end of time-dependent solution

In a segregated approach, variables are updated separately using available values from other fields. In a coupled approach, a large system of equations is solved for all fields simultaneously.

The first few steps in this procedure are shown below for the steady-state PV momentum equation (3), where we are interested solving for the velocity. Denote by $\mathbf{u}^n$ and $\mathbf{u}^o$ the updated (unknown) and available velocity fields, respectively. Similarly, let p^o denote the available pressure field. An integral representation, of the PV momentum equation over a given cell P with volume V is given by

$$\int_V \nabla \cdot (\mathbf{uu}) dV - \int_V \nabla \cdot (\nu \nabla \mathbf{u}) dV = -\int_V (\nabla p) dV. \tag{4}$$

Applying the divergence theorem over the given cell volume, whose surface S has outward normal $\mathbf{n}$, then yields

$$\int_S (\mathbf{n} \cdot \mathbf{u}) \mathbf{u} dS - \int_S \nu (\nabla \mathbf{u}) \cdot \mathbf{n} dS = -\int_V (\nabla p) dV. \tag{5}$$

Assume that the faces of the polyhedral cell P are indexed by f. Further, let $\mathbf{s}_f$ denote the outward-pointing

area-scaled normal to face f. Consequently, let $\mathbf{u}_f$ and p_f denote the corresponding face values for $\mathbf{u}$ and p. The equation then simplifies to

$$\sum_f (\mathbf{s}_f \cdot \mathbf{u}_f)\mathbf{u}_f - \sum_f \nu(\nabla \mathbf{u})_f \cdot \mathbf{s}_f = -\int_V (\nabla p) dV. \quad (6)$$

In order to achieve a linear system for $\mathbf{u}^n$, the convection and diffusion terms must be discretised; and the right-hand side integral in p, ν_f and the face flux $F_f = (\mathbf{s}_f \cdot \mathbf{u}_f)$ must be estimated using the old cell-centred fields $\mathbf{u}^o$ and p^o. We explain now the low-order discretisation procedure used for simulations in this paper before moving onto the method for evaluating the updated field $\mathbf{u}^n$.

4.1. Linear Interpolation to Faces of Old Values

In a mesh, assume cell P has a face f, across which lies neighbour cell N. Let $\overline{fN}$ denote the distance from the centroid of face f to the centroid of cell N. Similarly, let d_f denote the distance between the centroids of cell P and N. We define the linear interpolation factor $f_x = \overline{fN}/d_f$. For a field γ, a first order accurate estimation to the face value γ_f can be expressed

$$\gamma_f = f_x \gamma_P^o + (1 - f_x)\gamma_N^o.$$

Assume now that f is a face at the domain boundary with centroid a distance d_b from the centroid of P. If f is a Dirichlet boundary in γ with value γ_b, the evaluation is direct: $\gamma_f = \gamma_b$. For a Neumann condition, in which $g_b = \mathbf{s}_f \cdot \nabla \gamma_b$, an interpolation yields $\gamma_f = \gamma_P^o + d_b \gamma_b$.

4.2. Discretisation of Diffusion

We use again the nomenclature from the previous subsection. For a uniform Cartesian grid (as that used in this text), a second-order accurate estimation of the gradient term $(\nabla\gamma) \cdot \mathbf{s}_f$ at face f, across which cell P has neighbour N, is given as

$$(\nabla\gamma) \cdot \mathbf{s}_f = \frac{|\mathbf{s}_f|}{d_f}(\gamma_N - \gamma_P).$$

If f corresponds to Dirichlet boundary, the diffusion calculation follows an equivalent procedure, replacing γ_N with γ_b and d_f with d_b. The trivial Neumann boundary case gives in that the diffusion term can be directly set as $(\nabla\gamma_f) \cdot \mathbf{s}_f = g_b$.

4.3. Discretisation of Convection

Within the convection term $(F_f\gamma_f)$, the calculation of the flux at a face, F_f, is done using some interpolation of the old velocities $\mathbf{u}^o$ and potentially other variables. For SIMPLE, and its revised version, the details of the flux calculation are given in the corresponding sections. Assume in the following that F_f denotes the computed flux at a face f. If cell P has neighbour N across face f, we use the first-order upwind differencing scheme to estimate γ_f, yielding a convection term:

$$F_f\gamma_f = \mathbb{I}(F_f > 0)F_f\gamma_P + \mathbb{I}(F_f < 0)F_f\gamma_N.$$

Here, $\mathbb{I}$ denotes the indicator function. It is worth noting that the upwind scheme is bounded but numerically diffusive in terms of the field γ.

For the case where f corresponds to a Dirichlet boundary, the convection term can be evaluated directly: $F_f\gamma_f = F_f\gamma_b$. For the case of a Neumann condition, the evaluation is again made using interpolation: $F_f\gamma_f = F_f(\gamma_P + d_b g_b)$.

4.4. Assembling a linear system

Having established the methods of discretisation, we return now to the procedure of assembling a linear system for the updated velocity $\mathbf{u}^n$. Consider now a cell P, whose internal faces are indexed by f (with corresponding neighbour N) and whose boundary faces (if any) are indexed by b. Recall that we have specified Dirichlet conditions at all boundaries for the velocities. Discretising each of the terms in equation (4), save the pressure gradient, yields

$$a_P\mathbf{u}_P^n + \sum_N a_N\mathbf{u}_N^n = -\int_V \nabla p dV - \sum_b a_b. \quad (7)$$

where

$$a_P = \sum_f \Big(\mathbb{I}(F_f > 0)F_f + \nu\frac{|\mathbf{s}_f|}{d_f}\Big) + \sum_b \frac{|\mathbf{s}_b|}{d_b}, \quad (8)$$

$$a_N = \mathbb{I}(F_f < 0)F_f - \nu\frac{|\mathbf{s}_f|}{d_f}, \text{ and} \quad (9)$$

$$a_b = \Big(F_b + \nu\frac{|\mathbf{s}_b|}{d_b}\Big)\mathbf{u}_b. \quad (10)$$

The volume integral over the pressure gradient can be discretised using the divergence theorem and estimated using the old pressure field. One such equation for each cell in the domain makes for a linear system of the form $\mathbf{A}\mathbf{u}^n = \mathbf{b}$. The coefficient a_P denotes the diagonal coefficient in row P of $\mathbf{A}$, whereas a_N denotes the coefficient at index (P, N).

5. The SIMPLE Algorithm

The SIMPLE algorithm, proposed by Patankar and Spalding in 1972, was the first numerical means with which to attack the pressure velocity coupling [2]. The algorithm was originally proposed to calculate the steady state velocity and pressure resulting from a given geometry and boundary conditions. However, the algorithm can be modified to accommodate unsteady flows. At this point, it is worth noting that the algorithm as originally proposed calculated velocities and pressures on a *staggered grid*, that is, one in which the velocities are stored at the centroids of faces rather than cells [3].

We have followed here the description of SIMPLE as presented in [4], within which the algorithm is modified to accommodate a collocated grid, with all variables stored at volumetric cell centroids.

The SIMPLE algorithm is iterative, looping through a given list of steps until the pressure and velocity in successive iterations converge. The steps within an iteration are presented below.

5.1. Solving for Velocity

Unsurprisingly, SIMPLE begins by finding a predictor for the velocity, $\widehat{\mathbf{u}}$, calculated by solving the system of equations as implied by (7). In the coefficients a_P, a_N and a_b, the flux terms F_f are taken from the previous iteration (except for the first iteration, for which the initial velocities must be used to estimate these). The integral over the pressure gradient is estimated by applying the divergence theorem and interpolating p^o as in section 4.1. The system is solved with under-relaxation in the form:

$$\frac{a_P}{\alpha_U}\widehat{\mathbf{u}}_P + \sum_N a_N\widehat{\mathbf{u}}_N = -\sum_f p^o_f\mathbf{s}_f - \sum_b a_b + \frac{1-\alpha_U}{\alpha_U}a_P\mathbf{u}^o_P, \tag{11}$$

where $\alpha_{\mathbf{u}} \in (0, 1]$ is some under-relaxation factor on the velocity and $\mathbf{u}^o_P$ is the old velocity. This under-relaxation increases the diagonal dominance in the resulting coefficient matrix for the system of equations. Furthermore, the inclusion of $\mathbf{u}^o_P$ promotes stability in the algorithm by suppressing extreme changes to $\mathbf{u}$ that spill into the pressure calculation [4]. Note that while different velocity components may have different right-hand side (source) terms, the resulting matrix of coefficients is equal for each of them.

5.2. Deriving the Pressure Equation

We are now interested in formulating an equation for the pressure. Note that the velocities, $\widehat{\mathbf{u}}$, computed in the previous step will not be divergence-free. Let $\mathbf{u}_P$ denote an *unknown, divergence-free* cell centred velocity. Using the velocities $\widehat{\mathbf{u}}$ calculated in the previous step, and substituting into 7, yields

$$\mathbf{u}_P = -\frac{1}{a_P}\Big(\sum_N a_N\widehat{\mathbf{u}}_N - \sum_b a_b\Big) - \frac{1}{a_P}\int_V \nabla p dV. \tag{12}$$

The first term on the right-hand side can conveniently be written $\mathbf{H}(\widehat{\mathbf{u}})$ for short. Furthermore, as mentioned in 4, assume now that that the pressure p_P at the centroid of each cell P of volume V satisfies

$$V\nabla p_P = \int_V \nabla p dV \tag{13}$$

Due to the divergence-free assumption on the left-hand side of (12), substituting this equation into the PV continuity equation then yields the **pressure equation**:

$$\nabla\cdot\Big(\frac{V}{a_P}\nabla p\Big) = \nabla\cdot\Big(\frac{1}{a_P}\mathbf{H}(\widehat{\mathbf{u}})\Big).$$

This equation may in turn be integrated over each cell, discretised using the methods described above, and solved in a system of linear equations for a pressure predictor $\widehat{p}$:

$$\sum_f \Big(\frac{V}{a_P}\Big)_f\mathbf{s}_f\cdot(\nabla\widehat{p})_f = \Big(\frac{1}{a_P}\mathbf{H}(\widehat{\mathbf{u}})\Big)_f.$$

Note in particular that ∇p and the coefficient $1/a_P$ are interpolated to each face separately, in order to achieve an effective gradient estimation as in section 4.2.

5.3. Zero fluxes

A second set of predictors for the cell-centred velocities, $\overline{\mathbf{u}}$, may now be found by plugging the predictor $\hat{p}$ back into equation (12). Given the assumption of a divergence-free left-hand side in (12), these velocities will make for conservative net fluxes, with respect to each cell, when interpolated to faces as $F_f = \overline{\mathbf{u}}_f\cdot\mathbf{s}_f$. These fluxes are calculated and stored for the next iteration.

5.4. Updating Fields

While the pressure solutions obtained in $\widehat{p}$ force conservative net fluxes on the part of each cell, these solutions are un-physical, precisely due to the component enforcing divergence-free velocities. The updated pressure, p^n, is thus found by explicit under-relaxation as

$$p^n = p^o + \alpha_p(\hat{p} - p^o), \tag{14}$$

where α_p, is some relaxation factor.

Given this new pressure, p^n, one may now substitute back into equation (12) one final time, using the new pressure and the original velocity predictor, $\widehat{\mathbf{u}}$. Discretising the integral over the pressure gradient in the described manner yields a third velocity predictor, $\widehat{\widehat{\mathbf{u}}}$:

$$\widehat{\widehat{\mathbf{u}}} = \frac{1}{a_P}\mathbf{H}(\widehat{\mathbf{u}}) - \frac{1}{a_P}\sum_f p^n \mathbf{s}_f. \tag{15}$$

In a similar vein to the pressure, the updated velocity $\mathbf{u}^n$ is now computed as a weighted average of the initial predictor $\widehat{\mathbf{u}}$ and the final predictor $\widehat{\widehat{\mathbf{u}}}$:

$$\mathbf{u}^n = \alpha_{\mathbf{u}}\widehat{\widehat{\mathbf{u}}} + (1-\alpha_{\mathbf{u}})\widehat{\mathbf{u}},$$

where $\alpha_{\mathbf{u}}$ is the under-relaxation factor from 5.1.

5.5. Checkerboard Problem

Consider the discretisation of the pressure gradient term as seen in 7. On a uniform cartesian grid, the face interpolation factor f_x on all internal faces f will be exactly $1/2$. Interpolation of p to a face f across which cell P has neighbour N follows $p_f = (1/2)(p_P + p_N)$. Since outward normal vectors at opposing faces mirror each other, it follows that the discretisation may be rewritten as:

$$\int_V \nabla p dV \approx \sum_f p_f \mathbf{s}_f = \frac{1}{2}\sum_f p_N \mathbf{s}_f.$$

In particular, we note that the integral of the pressure-gradient over cell P depends on the neighbouring pressures but *not* on the pressure in cell P itself. This causes a so-called *checkerboard* problem, by which velocities two cells removed from each other tend to exhibit more similarity than adjacent cells. Unless this phenomenon is dealt with in the pressure equation, it will feed forward in the algorithm, causing unphysical results [5].

In the present form of SIMPLE, the remedy comes in the form of the right-hand side of the pressure equation 14. After integration and applying the divergence theorem, $1/a_P\mathbf{H}(\widehat{\mathbf{u}})$ is interpolated as a single vector onto the adjacent faces of each cell. This reestablishes the connection between the velocity in adjacent cells and resolves the checkerboard pattern.

6. A Revised SIMPLE Algorithm

In the present study, a revised, albeit very similar, version of the algorithm will be used to simulate the lid-driven cavity. This version may be described by the following steps:

1. Solve the same system as in (5.1), for a predicted velocity field, $\widehat{\mathbf{u}}$, but with an assumption of a zero pressure gradient:
$$\frac{a_P}{\alpha_U}\widehat{\mathbf{u}}_P + \sum_N a_N\widehat{\mathbf{u}}_N = -\sum_b a_b + \frac{1-\alpha_U}{\alpha_U}a_P\mathbf{u}_P^o \tag{16}$$
2. Calculate (non-conservative) face-fluxes at faces
$$(F_{pre})_f = \mathbf{s}_f \cdot \widehat{\mathbf{u}}_f \tag{17}$$
by interpolating the velocities from step 1.
3. Use these fluxes to solve for a predicted pressure field, $\widehat{p}$, using the modified pressure equation:
$$\nabla \cdot \Big(\frac{V}{a_P}\nabla p\Big) = \nabla \cdot \widehat{\mathbf{u}} \tag{18}$$
4. Use this pressure to calculate divergence-free fluxes by correcting the ones from step 2:
$$(F_{corr})_f = (F_{pre})_f - \Big(\frac{V}{a_P}\Big)_f \mathbf{s}_f \cdot (\nabla\widehat{p})_f \tag{19}$$
5. Calculate the updated pressure values, p^n, using an explicit under-relaxation as in 5.4.
6. Calculate updated velocities using the new pressure field, p^n, by subtracting the discretised gradient integral in line with equation (12):
$$\mathbf{u}^n = \widehat{\mathbf{u}} - \frac{1}{a_P}\sum_f (p^n)_f \mathbf{s}_f \approx \widehat{\mathbf{u}} - \frac{1}{a_P}\int_V \nabla p dV \tag{20}$$

6.1. Notes on the Revised Algorithm

We note that, while the old pressure gradient is not explicitly present on the right-hand side of (16), it is included implicitly in the old velocity $\mathbf{u}^o$, when the equation is under-relaxed. This makes for a weaker coupling of $\mathbf{u}$ and p compared to the original algorithm. Consequently, we expect convergence to be somewhat slower in the revised version.

The fact that the corrected fluxes will be conservative can be seen by considering the corrected net flux over a given cell with faces f:

$$\sum_f (F_{corr})_f = \sum_f \mathbf{s}_f \cdot \widehat{\mathbf{u}}_f - \sum_f \Big(\frac{V}{a_P}\Big)_f \mathbf{s}_f \cdot (\nabla\widehat{p})_f \tag{21}$$

By comparison, the discretisation of (18) over a given cell can be expressed

$$\sum_f \Big(\frac{V}{a_P}\Big)_f \mathbf{s}_f \cdot (\nabla\widehat{p})_f = \sum_f \mathbf{s}_f \cdot \widehat{\mathbf{u}}_f,$$

which shows that the right-hand side of (21) is zero, provided the same interpolation procedure is used in steps 3 and 4. For the initial fluxes F_{pre}, there is no mechanism forcing the velocities $\widehat{\mathbf{u}}$ to be divergence-free.

7. Computational Set-up

The simulation of the 2D lid-driven cavity was conducted for a grid of 200 by 200 cells. While only a single layer of cells was used in the z-direction, each computational cell was specified with a depth of 0.0005 meters. Before the first iteration, the velocities and pressures at cell centroids, as well as the faces fluxes F_f, were set to zero.

The simulation was conducted for a Reynolds number Re of 100 and 1000. Given the length scale of 0.1 meter and the reference velocity of 1m/s at the top wall, this made for kinematic viscosities of $\nu_1 = 0.001$ and $\nu_2 = 0.0001$, respectively. All computations were made with double precision arithmetic.

In each iteration of the revised SIMPLE algorithm, each system of equations (i.e. for velocities x, y, and pressure) were solved using a Gauss-Seidel iterative solver. On each set of equations, 200 sweeps of the Gauss-Seidel solver were applied, after which the residual was calculated.

The under-relaxation parameters used were those as suggested in [5], with $\alpha_{\mathbf{u}} = 0.9$ and $\alpha_p = 0.2$.

We shall refer to the fields $\mathbf{u}^n$, p^n as outer solutions. Similarly, we refer to the fields $\mathbf{u}^n$ and p^n as outer solutions. The vectors $\Delta_i, \overline{\Delta_i} \in \mathbb{R}^3$ were defined as

$$\mathbf{\Delta}_i = \begin{bmatrix} |\mathbf{u}_x^n - \mathbf{u}_x^o|_\infty \\ |\mathbf{u}_y^n - \mathbf{u}_y^o|_\infty \\ |p^n - p^o|_\infty \end{bmatrix}, \quad \overline{\mathbf{\Delta}}_i = \begin{bmatrix} |\widehat{\mathbf{u}}_x^n - \widehat{\mathbf{u}}_x^o|_\infty \\ |\widehat{\mathbf{u}}_y^n - \widehat{\mathbf{u}}_y^o|_\infty \\ |\widehat{p}^n - \widehat{p}^o|_\infty \end{bmatrix}, \tag{22}$$

where $|.|_\infty$ denotes the infinity norm, and computed after each iteration. In other words, these vectors held the maximal absolute change, in each variable, between the solution in the current iteration and the previous one. Each simulation ran with a tolerance $max(\Delta_i) < 10^{-5}$, i.e. until the absolute change in every variable, and in every cell, was less than 10^{-5} between iterations.

8. Results

Diagonal (a_P) and off-diagonal (a_N) coefficients in the discretised momentum equation (16) for the cell at the top left corner of the domain, at the 10th iteration of the revised SIMPLE algorithm for $Re = 100$, are given in table 1.

	$x = 0.00025$	$x = 0.00075$
$y = 0.99975$	**34.0653**	−5
$y = 0.99925$	−5.65578	

Table 1: Coefficients equation (16), top left corner. All numbers order 10^{-7}. Diagonal coefficient in bold.

Similar coefficients for the same cell in the discretised pressure equation (18) were found to be as below.

	$x = 0$	$x = 0.0005$
$y = 0.9995$	**−4.45031**	2.2151
$y = 0.9990$	2.23521	

Table 2: Coefficients equation (18), top left corner. All numbers order 10^{-8}. Diagonal coefficient in bold.

We examine now the coefficients for a cell at the middle of the domain. For the cell with centroid at $x = 0.05025$, $y = 0.05025$, the corresponding table for the discretised momentum equation (16) were found as:

	$x = 0.04975$	$x = 0.05025$	$x = 0.05075$
$y = 0.05075$		−5	
$y = 0.05025$	−5	**22.3503**	−5.10876
$y = 0.04975$		−5.00648	

Table 3: Coefficients equation (16), domain centre. All numbers order 10^{-7}. Diagonal coefficient in bold.

The equivalent coefficients for the pressure equation (18) were found to be

	$x = 0.04975$	$x = 0.05025$	$x = 0.05075$
$y = 0.05075$		3.10691	
$y = 0.05025$	3.10697	**−12.4284**	3.10723
$y = 0.04975$		3.10728	

Table 4: Coefficients equation (18), domain centre. All numbers order 10^{-8}. Diagonal coefficient in bold.

A few remarks are in order regarding the coefficients as shown. We note that due to the changing flux term at each face, F_f, the coefficients for the equation (16) will change as the SIMPLE algorithm progresses from one iteration to the next. Furthermore, we note that, in general, a discretised Laplacian, as in (18), will be symmetric on a uniform cartesian mesh, in the sense that a cell P will get equivalent contributions from opposing neighbours. However, as mentioned in 5.4, the coefficient $1/a_P$ needs to be interpolated to each face surrounding a cell, which makes for the asymmetry.

The number of iterations required to reach convergence for the 200 by 200 cell grid was 1513 with $Re = 100$ and 574 iterations for $Re = 1000$.

Figure 2 demonstrate the evolution of each component in Δ_i and $\overline{\Delta_i}$, as defined in (22), for $Re = 100$ and $Re = 1000$. They show the convergence of the inner $(\widehat{\mathbf{u}}, \widehat{p})$ and outer $(\mathbf{u}^n, p^n)$ solutions as the simulations progress.

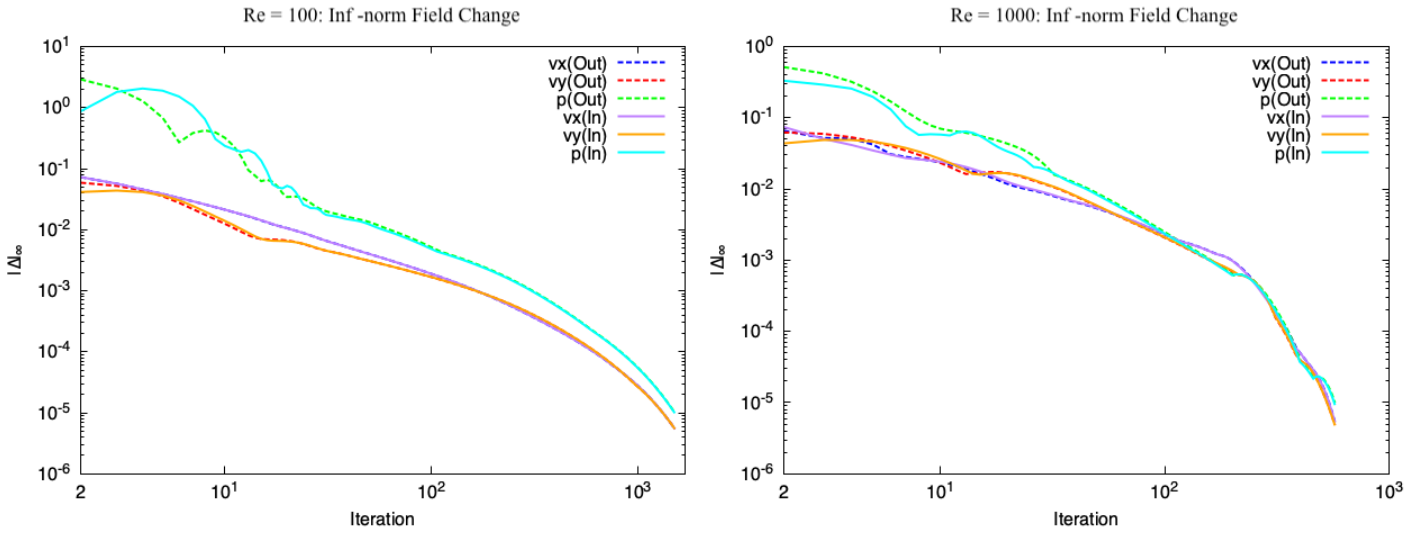

Figure 2: Log-log of Δ_i and $\overline{\Delta}_i$ against iterations. "Out" and "In" denotes element of Δ_i or $\overline{\Delta}_i$, respectively.

Figures 3 show each of the fields over the domain after convergence. It should be noted that the range of values for the pressure plots has been truncated to that indicated by the colour bars in order to better distinguish features away from the top of the domain. For clarity, colour plots of the pressure fields without truncation, are given in Appendix A. The plots of the velocity norm, along with the associated directional vectors, and some streamlines are shown in Appendix B.

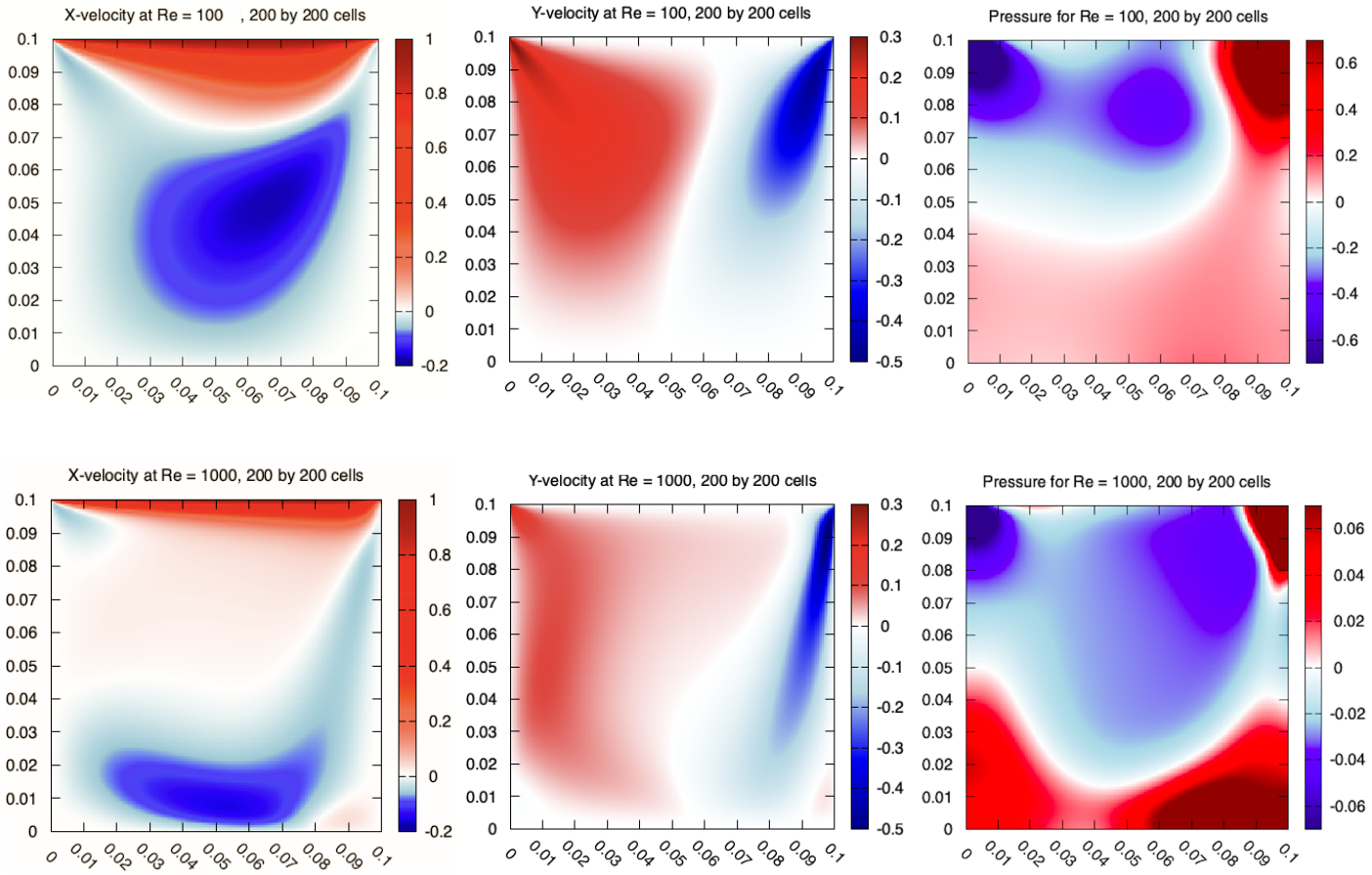

Figure 3: Fields x-velocity, y-velocity, and pressure after convergence

Figure 4 demonstrates the infinity norms of the residual after the inner solution for pressure has been found in each iteration using 200 sweeps of the Gauss-Seidel algorithm. The cases $Re = 100$ and $Re = 1000$ are both included. Residuals for the velocity components are not included, as these were of order 10^{-20} across all iterations on both Reynolds numbers.

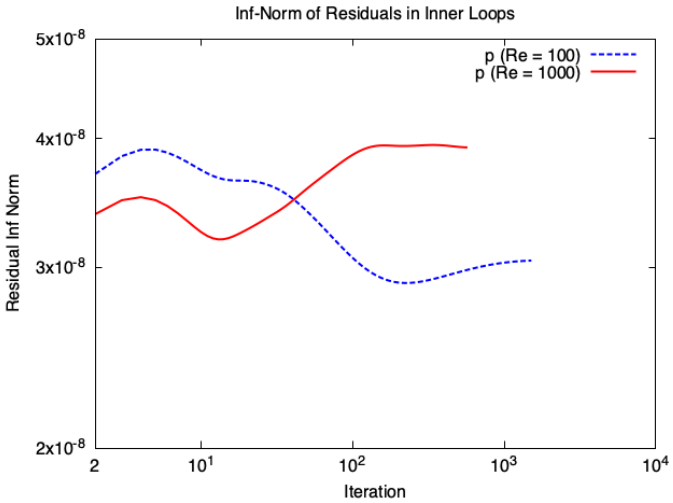

Figure 4: Log-log plots of Residuals in p vs Iterations

To gauge the accuracy of specific values on the domain, we have plotted in figures 5 and 6 the values of x- and y- velocities along the lines $x = 0.05025$ and $y = 0.05025$, respectively. These are compared against the results obtained by [6] as shown. In [6], the domain was $[0, 1] \times [0, 1]$, and x- and y-velocities were sampled along the exact vertical mid-lines $x = 0.5$ and $y = 0.5$, respectively. We have divided their coordinates, in each direction, by 10 to obtain suitable comparisons. Given the equivalent Reynolds numbers used in both studies, the results obtained should be comparable.

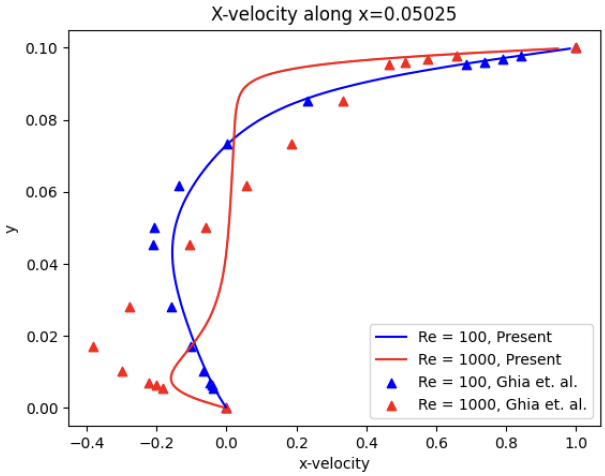

Figure 5: x-velocity along vertical centre line

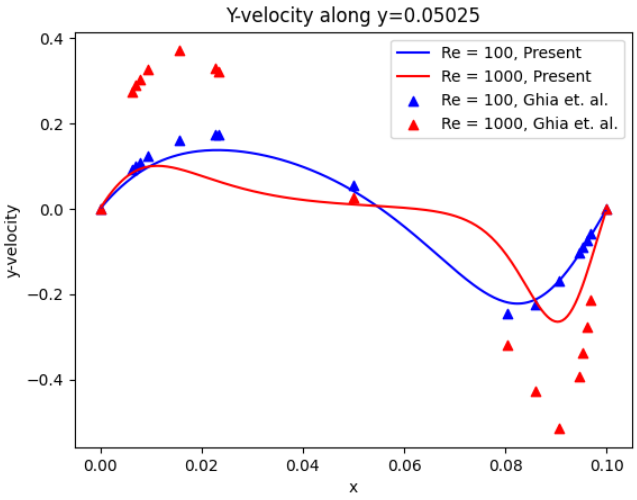

Figure 6: y-velocity along horizontal centre line

9. Discussion

As may be seen from the left panel of figure 2, for $Re = 100$, the infinity norm in the absolute change between iterations decreases steadily beyond about 100 iterations, for all inner and outer solutions. However, the norm in pressure change is unstable for iterations $< 10^2$, demonstrating that the revised SIMPLE algorithm can require a significant number of iterations before reaching a state at which successive iterations monotonically decrease the change in pressure. The similar plot for $Re = 1000$ displays more fluctuations across the other variables in terms of the infinity norm of the change between iterations. However, the pressure is again the variable for which these fluctuations are the largest and for which they persist the longest. Interestingly, even at about 500 iterations, the pressure component of $\mathbf{\Delta_i}$ and $\overline{\mathbf{\Delta}_i}$ display a small peak, during which the infinity norm in the absolute difference between successive iterations were not decreasing. This suggests also that monotonicity of convergence in the pressure solutions are dependent on the Reynolds number. The residuals plotted in figure 4 may offer an explanation as to why this is the case. As seen there, the residuals after 200 Gauss-Seidel sweeps over the pressure equation display a minimum at about 150 iterations for $Re = 100$ and at at about 10 iterations for $Re = 1000$. The residuals for each case appear to settle at levels of about $4 * 10^{-8}$ and $3 * 10^{-8}$, respectively, toward the end of the simulations. Since the pressure equations are not solved to a level of accuracy at machine epsilon, it is possible that the right-hand side of that equation can cause irregularities to the resulting prediction, even if the residuals stays at a constant level.

For the case of $Re = 100$, the computed field plots match very well with those found in [7]. For $Re = 1000$, there is less similarity. Notably, the section of negative x-velocity in the lower part of the domain is shrunken in the present study. Furthermore, the pressure through found at the centre of the domain in [7], for $Re = 1000$, is absent in our result.

Looking at figures 5 and 6, we see that the present velocity profiles match fairly well with the reference results of Ghia et al. for $Re = 100$. However, the results deviate significantly for $Re = 1000$. This suggests that the present form of the SIMPLE algorithm does not deal well with the lid-driven cavity flow at high Reynolds number.

For the case of $Re = 1000$, we note that the kinematic viscosity ν will be reduced by a factor of 10 compared to the case of $Re = 100$. Thus, the contribution from diffusion in the PV momentum equation (3) will be reduced tenfold. This increases the relative contribution from the non-linear convection term. On the other hand, the revised SIMPLE algorithm is based on solving linear systems of equations, devised from linearised discretisations, even of the convection term $\nabla \cdot \mathbf{uu}$. Hence, increased non-linearity of the momentum equation will lead to decrease accuracy in the velocity prediction $\widehat{\mathbf{u}}$, relative to the case of $Re = 100$. In [6], the PV coupling is solved in a coupled approach. Furthermore, while their linear solver is based on the Gauss-Seidel algorithm, increased convergence is achieved in each iteration by using a multi-grid approach which more effectively removes low-frequency errors. In combination with a more sophisticated estimation of the flux terms F_f, they are thus able to handle better the non-linearity induced by higher Reynolds numbers.

As can be seen from B.9, for $Re = 100$, we do not find vortices at the lower corners of the domain, as were found in [6] and [7]. A vortex pattern can, however, be seen in B.10 for $re = 1000$. It is possible that these patterns are not present in our results due to too high a tolerance on the vector $|\mathbf{\Delta_i}|_\infty$ in the case of $Re = 100$.

10. Summary

In this paper, we have presented used a revised version of the SIMPLE algorithm to solve the steady-state pressure-velocity coupling on the reference problem that is the lid-driven cavity. At a Reynolds number of $Re = 100$, we achieve results in close keeping with the litterature at the centre of the domain. However, at the lower corners of the domain, our results do not display the vortices that are present in the literature. Furthermore, at the higher Reynolds number of $Re = 1000$, the algorithm is unable to handle the non-linearities in the governing equations, leading to significantly worse results. A natural extension to this work would include tests for more Reynolds numbers between 100 and 100 to establish a threshold at which the present version of the algorithm becomes untenable. Furthermore, the algorithm could be improved by using a coupled multigrid approach to solve for the initial predictors $\widehat{u}$ and $\widehat{p}$. The multigrid approach would also serve to speed up convergence, which takes several minutes on a MacBook with M1 processor.

References

[1] OpenCFD, Lid-driven cavity flow, Website URL: https://www.openfoam.com/documentation/tutorial-guide/2-incompressible-flow/2.1-lid-driven-cavity-flow, accessed on 21.03.2024 (2024).

[2] H. Jasak, T. Uroić, Modeling in Engineering Using Innovative Numerical Methods for Solids and Fluids, Springer, 2020, Ch. Practical Computational Fluid Dynamics with the Finite Volume Method.

[3] S. Patankar, D. Spalding, A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows, International Journal of Heat and Mass Transfer (1972).

[4] T. Uroić, Implicitly coupled finite volume algorithms, Ph.D. thesis, University of Zagreb (2019).

[5] J. Ferziger, M. Perić, R. Street, Computational Methods for Fluid Dynamics, Fourth Edition, Springer Nature Switzerland AG, 2020.

[6] U. Ghia, K. Ghia, C. Shin, High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method, Journal of Computational Physics (1982).

[7] M. Ge, X. Zhang, K. Brookshire, O. Coutier-Deloshga, Parametric and v&v study in a fundamental cfd process: Revisiting the lid-driven cavity flow, Aircraft Engineering and Aerospace Technology (2022).

Appendix A. Pressure Fields without Truncation in Colour Range

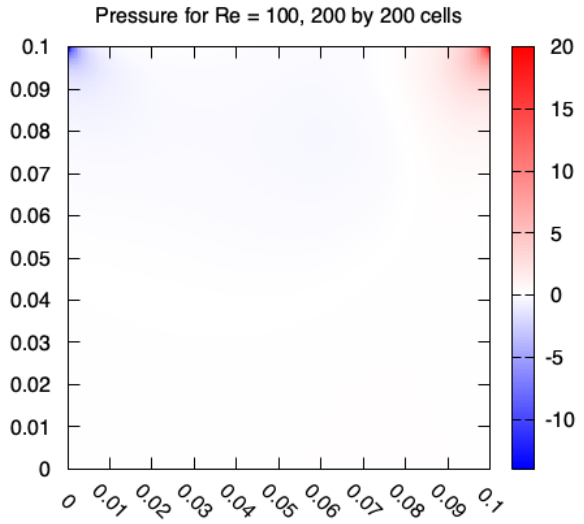

Figure A.7: Pressure profile over domain after convergence. $Re = 100$. No truncation.

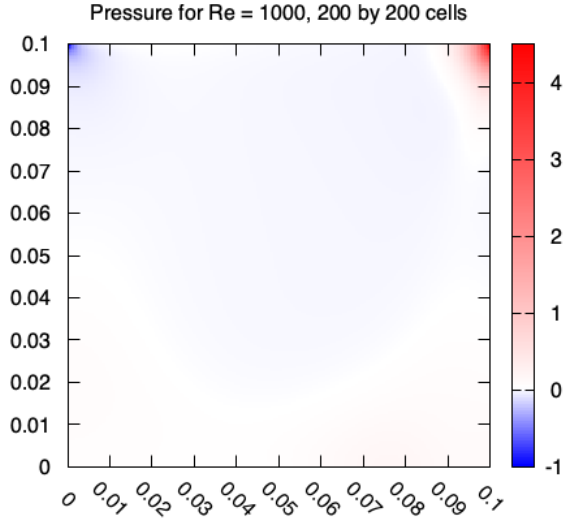

Figure A.8: Pressure profile over domain after convergence. $Re = 100$. No truncation.

Appendix B. Velocity Norm and Streamlines

In the below figures we see the norm of the velocity throughout the domain, along with the associated directional vectors and streamlines starting at $x = 0.07$, $y = 0.07$.

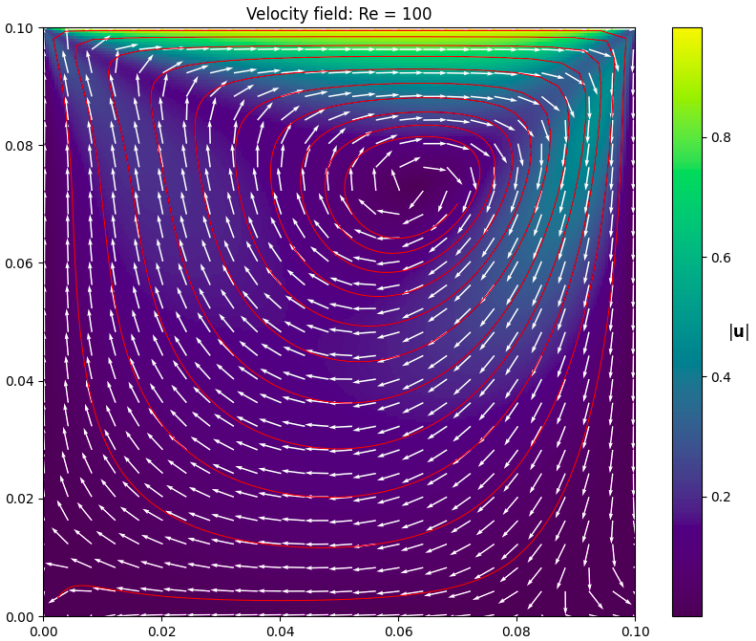

Figure B.9: Colour plot for velocity norm, $Re = 100$. Streamline in red starting at $x = y = 0.07$. Unit-scaled directional vectors on velocity

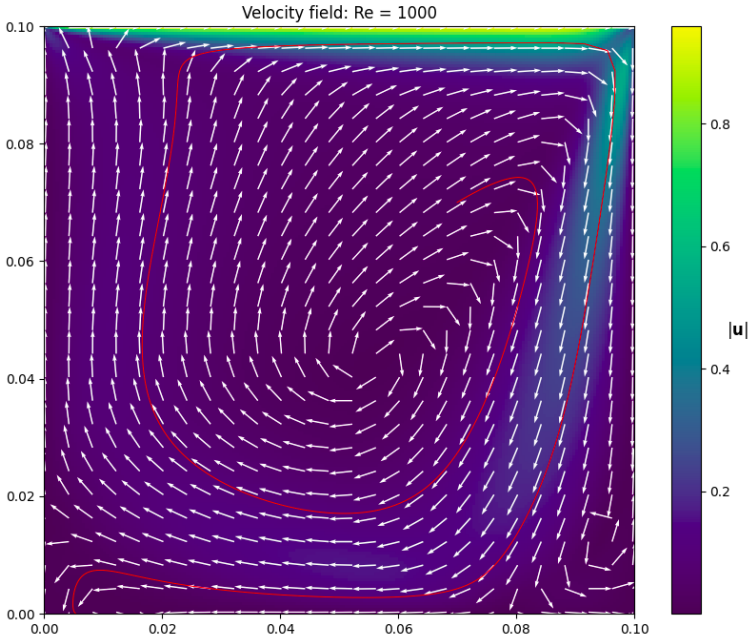

Figure B.10: Colour plot for velocity norm, $Re = 1000$. Streamline in red starting at $x = y = 0.07$. Unit-scaled directional vectors on velocity

The streamlines were generated using Scipy's Runge-Kutta initial value problem solver.