A 2D Navier-Stokes Solver in Python

Student Name: [redacted]

Supervisor Name: [redacted]

Submitted as part of the degree of M.Sc. MISCADA to the

Board of Examiners in the Department of Computer Sciences, Durham University

***Abstract* —** Fluid flows are governed by systems of partial differential equations such as the Navier-Stokes equations (NSE) describing the motion of viscous fluids. The aim of this project is to produce a set of Python scripts that can be used by undergraduate students of the lecture 'Continuum Mechanics' to study such fluid flows. Two different solvers for the 2D NSE for an incompressible fluid are developed for two different flows, the lid-driven cavity and a Kelvin-Helmholtz instability. By varying the Reynolds number, grid size and time step and comparing results for the two problem settings against benchmark values, it is demonstrated that the solvers are robust for a range of parameter settings. The solvers are designed such that the solutions are visualised and saved for later use by the students. Furthermore, a script for the visualisation of stationary velocity fields is developed, which can be used by students to replicate the examples given in the lecture.

***Keywords* —** 2D incompressible viscous Navier-Stokes equations, finite differences, projection method, vorticity stream-function formulation, lid-driven cavity, Kelvin-Helmholtz instability

I INTRODUCTION

Fluid flows are governed by systems of partial differential equations (PDEs) that represent the various conservation laws of momentum, energy and mass. Computational fluid dynamics (CFD) replaces the continuous PDEs by sets of algebraic equations, thus enabling modern computers to discretely model those fluid flows. In this work, the focus will be on the famous 2D Navier-Stokes equations (NSE) which describe the motion of viscous fluids.

The aim of this project is to produce a set of Python scripts to be used by undergraduate students to help them understand basics of CFD in the frame of the lecture 'Continuum Mechanics' (course MATH3101/4081) held at Durham University. Hence, when choosing a type of numerical solver for the equations, emphasis was put on *ease of use* while at the same time providing an adequate degree of *accuracy* to ensure a suitable level of difficulty. The aim is not to present a new method for solving the NSE, but to create an educationally useful demonstration of common techniques used in modern CFD.

In accordance with this aim, two different types of solver are presented. One employs a version of the *projection method* (PM) on a staggered grid as introduced by Chorin (1968) while the other uses the *vorticity stream-function* (VSF) formulation on a collocated grid as introduced by Fromm (1964). Both methods are based on finite difference discretisations.

The flows studied in this project are two-dimensional. This is partly due to the methods chosen to represent them, and partly due to the fact that the intended programming language is Python. Python is a suitable language for this project as it is a robust general purpose programming language and has in recent years been the starting point for many undergraduates when

learning scientific computing. It is straightforward to implement two-dimensional representations at reasonable computational costs. Three-dimensional flow computations however might benefit from being run in parallel to decrease computation time, and while Python is able to run multi-processor operations, this is not the intended outcome of the project.

The performance of the solvers is studied through two examples. The first is the lid-driven cavity problem, which is the standard benchmark case for two-dimensional viscous flow. The second is a shear-flow with a sinusoidal perturbation resulting in a Kelvin-Helmholtz instability. A version of this problem is studied in the lecture and hence it is slightly more relevant to the students than the lid-driven cavity.

Overall, five programmes have been developed. The first two are the implementations of the *lid-driven cavity* and *Kelvin-Helmholtz instability* problems using the projection method approach, while the third and fourth scripts use the vorticity stream-function method to solve the model problems. The final programme is a simple script unrelated to solving the NSE. It contains a routine to plot velocity fields and the trajectories of particles in them. It is intended to be used by students to study simpler examples presented in the lecture.

II RELATED WORK

There are a range of established solution methods for the Navier-Stokes equations (NSE). Over the years, they have been refined, expanded and adjusted in order to give even more precise solutions. This project does not aim to produce the next best mathematical formalism. It strives to employ some more basic numerical techniques in order to be of educational value for an undergraduate course where students potentially have not had much experience in scientific computing. In this section, two mathematical formalisms of the NSE are given that can be approximated numerically with relatively straightforward methods.

A Governing equations

Consider a velocity field $\vec{u}$ for an incompressible fluid. The Navier-Stokes equations describing such a velocity field and the corresponding pressure field p are the *continuity equation*,

$$\nabla \cdot \vec{u} = 0 \quad , \tag{NS1}$$

and the *momentum equation*,

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\, \vec{u} = -\frac{1}{\rho_0} \nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \vec{u} \quad . \tag{NS2}$$

where the former describes the conservation of mass, i.e. the density of the fluid $\rho_0 = const.$, and the latter describes the conservation of momentum. The differential operator $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right)$ represents the first derivatives in two-dimensional Cartesian coordinates (x, y).

The equations above correspond to equations (7.38) and (7.39) in the lecture notes for 'Continuum Mechanics'. There, equation (7.39) is written in terms of the kinematic viscosity $\nu = \mu / \rho_0$ instead of the dimensionless parameter $\mathrm{Re} = \frac{UL}{\nu}$, the Reynolds number (see equation (7.66) in the lecture notes). $U = const.$ represents a characteristic flow speed for the studied problem while $L = const.$ denotes a characteristic length scale for the flow.

Note: In the continuous world the gradient form of the convective term is equivalent to the divergence form of the term, i.e. $(\vec{u}\cdot\nabla)\vec{u} = \nabla\cdot(\vec{u}\vec{u})$. While it is conventional to write the convective term in gradient form when stating the NSE, the divergence form should be used when discretising the equations, as this ensures that the convective term is conservative in nature, see Morinishi et al. (1998). Thus, the two-dimensional equations that need to be discretised in that part of equation (NS2) are

$$\frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} \qquad \text{and} \qquad \frac{\partial v^2}{\partial y} + \frac{\partial uv}{\partial x} \quad . \tag{1}$$

Alternatively, the motion of a two-dimensional flow can be described in terms of a stream-function ψ and a vorticity $\omega = \nabla\times\vec{u}$. This *vorticity stream-function formulation* is obtained by taking the curl of equation (NS2) and using the fact that due to the incompressibility condition (NS1), $\nabla\cdot\omega = 0$. In two dimensions, the governing equations are the *vorticity transport equation*,

$$\frac{\partial\omega}{\partial t} + (\vec{u}\cdot\nabla)\,\omega = \frac{1}{Re}\nabla^2\omega \quad \Longleftrightarrow \quad \frac{\partial\omega}{\partial t} + u\frac{\partial\omega}{\partial x} + v\frac{\partial\omega}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2\omega}{\partial x^2} + \frac{\partial^2\omega}{\partial y^2}\right) \quad , \tag{VTE}$$

with

$$u = \frac{\partial\psi}{\partial y} \qquad v = -\frac{\partial\psi}{\partial x} \quad , \tag{SF}$$

and the *Poisson equation for the stream-function*,

$$\nabla^2\psi = -\omega \quad \Longleftrightarrow \quad \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} = -\omega \quad . \tag{PSE}$$

Through the definition of the stream-function (equation (SF)) continuity is automatically satisfied. Equation (VTE) corresponds to equation (7.59) in the lecture notes.

Either of the two sets of equations above can be used to describe the motion of the flow. In accordance with the goal of this project to introduce common computation techniques in CFD, two sets of solvers are presented in this project, one for each formalism. The individual strengths and weaknesses of the two methods are highlighted as we go along.

B Projection method for the NSE in primitive variables

Consider the formalism in primitive variables u, v and p. Since the density of the fluid is constant, the primary difficulty in obtaining a time-accurate solution to the incompressible NSE is to couple changes in the velocity field to changes in the pressure field while satisfying the incompressibility condition (NS1).

Between the 1960s and the 1980s various 'pressure correction' approaches were developed to tackle this issue. Broadly speaking, in these approaches a Poisson equation for pressure is solved at the current time level such that equation (NS1) will be satisfied at the next time level. That way, if the resulting pressure field is used to update the velocity field, i.e. 'correcting' it for pressure, continuity is ensured in the final velocity field. The Poisson equation for pressure is obtained from the momentum equations.

While this sort of projection method was first developed as the Marker and Cell (MAC) method on a staggered grid by Harlow and Welch (1965), Chorin (1968) and Témam (1969) expanded it to initiate a group of methods more often referred to as 'fractional step methods'.

These are characterised by obtaining an intermediate velocity from the momentum equations and then solving the Poisson equation for pressure which will project the intermediate velocity into a divergence-free field. The method is almost identical to Harlow and Welch's and mostly differs in the definition of the Poisson equation for pressure. Slightly later, Patankar and Spalding (1972) and Patankar (1981) developed the SIMPLE (semi-implicit method for pressure linked equations) and SIMPLER (SIMPLE revised) algorithms, which employ an iterative procedure with repeated correction of the pressure until the solution converges.

Kim and Moin (1985) extended the fractional step method to higher accuracy. They employed second order Adams-Bashforth schemes for the convective terms of equation (NS2) and a semi-implicit Crank-Nicolson scheme for the viscous terms. This approach is commonly employed and there exists a lot of literature on it, both in the form of scientific papers and textbooks. The projection method (PM) solvers developed as part of this project utilise this approach.

C Vorticity stream-function method

The pressure that is obtained through solving equation (PPE) is only an approximation (see Aref and Balachandar (2017)). Furthermore, the projection method requires declaring a value for the pressure at the boundaries of the domain, which we have no information about. A benefit of the vorticity stream-function (VSF) formulation is that it does not make use of p and hence does not require the same sort of coupling of pressure and velocity as the projection method. If knowing the pressure is required, it can always be obtained from the velocity field.

Another benefit of the VSF formulation is that it achieves higher accuracy than the projection method. Depending on temporal and spatial discretisations, especially with regards to boundary values, the formulation using ψ and ω is second-order accurate. Examples of second-order accurate discretisations of this method are easily found in the literature. Some noteworthy resources are U. Ghia et al. (1982), Ghia et al. (1981) and Fromm (1964).

III SOLUTION

Let's reiterate the anticipated primary outcome of the project:

A two-dimensional Navier-Stokes solver that can be used by undergraduates to explore the behaviour of 2D fluids. In particular, a working Python code that can be run on a laptop and used by undergraduate students to help them understand fluid mechanics.

In order to reach this aim, an assortment of solvers for known benchmark problems has to be created. The results that they yield are compared against literature values and based on this comparison we judge which solver best fulfils the aim. The chosen benchmark problems that are implemented to enable students to study fluid flows are the *Lid-Driven Cavity* (LDC) flow and a perturbed shear-layer flow resulting in a *Kelvin-Helmholtz Instability* (KHI). To reiterate, the types of solver that are developed for these flows are a projection method-type solver and a solver based on the vorticity stream-function method.

This sections of the report presents the development of the solvers. To start, a general description of the numerical method used in the discretisation of the continuous equations presented in section II above is given. Subsequently, a theoretical overview of the algorithms underlying the solvers is presented. Finally, the actual code structure and implementation in Python is outlined.

A Finite difference discretisation

To approximate the continuous functions $\vec{u}$, p, ψ and ω, discretisation via *finite differences* is used. The continuous solutions are approximated by finite differences between a discrete number of points on a grid or mesh covering the computational domain Ω. The discrete approximations of the components of the velocity $\vec{u} = (u, v)$ at time level n for example are denoted by $u^n_{i,j}$ and $v^n_{i,j}$. Here, the pairs (i, j) denote the grid points corresponding to x and y coordinates, where the indices $i = 0, 1, 2, ..., N_x$ and $j = 0, 1, 2, ..., N_y$ are integers covering the range of the mesh-size $N_x \times N_y$. For a regular grid, the grid spacing in the x-direction is $\Delta x = 1/N_x$ and the grid spacing in the y-direction is $\Delta y = 1/N_y$. Similarly, the difference between two time steps in the time domain is $\Delta t = 1/T$, and the time index $n = 0, 1, 2, ..., T$.

An in-depth analysis of the drawbacks and benefits of other discretisation approaches than the ones used below exceeds the scope of this report. A general introduction to finite difference methods can be found in Hinch (2020).

B Schematic projection method and temporal discretisation

The underlying algorithm of the projection method scheme is as follows:

1. Evolve the current velocity field $\vec{u}^n$ according to

$$\frac{\partial \vec{u}}{\partial t} = -\left(\vec{u} \cdot \nabla\right) \vec{u} + \frac{1}{\mathrm{Re}} \nabla^2 \vec{u} \quad \Longrightarrow \quad \frac{\hat{\vec{u}} - \vec{u}^n}{\Delta t} = -\left(\vec{u}^n \cdot \nabla\right) \vec{u}^n + \frac{1}{\mathrm{Re}} \nabla^2 \vec{u}^n \tag{2}$$

   to get an intermediate velocity field $\hat{\vec{u}}$. The pressure gradient is not used in this step and $\hat{\vec{u}}$ is not necessarily divergence free.

2. The difference between the true velocity field at the next time level $\vec{u}^{n+1}$ and the intermediate velocity is equal to the pressure gradient: $\frac{\vec{u}^{n+1} - \hat{\vec{u}}}{\Delta t} = -\frac{1}{\rho_0} \nabla p^{n+1}$. Taking the divergence of this, requiring that $\nabla \cdot \vec{u}^{n+1} = 0$ and rearranging, we obtain a Poisson type equation for the pressure:

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t} \nabla \cdot \hat{\vec{u}} \quad . \tag{PPE}$$

   Solving this equation yields an approximation of the pressure field that enforces $\nabla \cdot \vec{u}^{n+1} = 0$.

3. Using the intermediate velocity from step (1) and the pressure approximation from step (2), we obtain a final divergence free velocity field: $\vec{u}^{n+1} = \hat{\vec{u}} - \frac{\Delta t}{\rho_0} \nabla p^{n+1}$.

The simple explicit temporal discretisation presented in these three steps restricts the method to a relatively small time step (Aref and Balachandar (2017)). As indicated in the previous section, the conventionally used temporal discretisation is an explicit second-order Adams-Bashforth method for the non-linear advection term and a second-order implicit Crank-Nicolson scheme for the diffusive/ viscous term. This overall semi-implicit discretisation allows for a larger time-step without losing accuracy. The equation for the intermediate velocity of step (1) changes to

$$\frac{\hat{\vec{u}} - \vec{u}^n}{\Delta t} = -\frac{3}{2} \mathcal{N}^n + \frac{1}{2} \mathcal{N}^{n-1} + \frac{1}{2\mathrm{Re}} \left(\nabla^2 \hat{\vec{u}} + \nabla^2 \vec{u}^n \right)$$

$$\iff \quad \left(1 - \frac{dt}{2\mathrm{Re}}\nabla^2\right)\hat{\vec{u}} = \vec{u}^n + dt \cdot \left(-\frac{3}{2}\mathcal{N}^n + \frac{1}{2}\mathcal{N}^{n-1}\right) + \frac{dt}{2\mathrm{Re}}\nabla^2\vec{u}^n \qquad \text{(intmvel)}$$

where $\mathcal{N} = \vec{u}\cdot\nabla\vec{u}$ is the non-linear term. The two equations for the two velocity components are completely decoupled at this point, which makes solving equation (intmvel) easier. A possible drawback of this method is that it requires us to solve the Helmholtz type equation (intmvel) for the intermediate velocity at each time step, which has a high computational cost, see section E below. Nonetheless, this approach is significantly faster than the fully explicit one as it is stable for larger time-steps.

Since second-order accurate schemes are used for the temporal discretisation of the advection and diffusion terms, it seems that the accuracy of the overall method is also second-order. However, in an overview paper investigating various projection type schemes, Guermond et al. (2006) showed that the method is at best first-order accurate, even if high-order schemes are used for the spatial discretisation.

B.1 Spatial discretisation

The discretisation of the PDEs for the projection method is performed on a *staggered* grid as opposed to a regular grid. The benefit of this is to avoid an oscillation of the pressure field that results from a decoupling at odd and even grid positions as a result of the discretisation. See Hinch (2020) chapter 3.5 or Aref and Balachandar (2017) chapter 8.2 for in-depth analyses of the phenomenon.

Thus, the horizontal velocities, $u_{i,j+\frac{1}{2}}$, are defined on the vertical grid cell edges, while the vertical velocities, $v_{i+\frac{1}{2},j}$, are defined on the horizontal cell edges. The pressure variables, $p_{i+\frac{1}{2},j+\frac{1}{2}}$, are defined at the centres of the cells, see figure 1. Including boundary points, the matrices hence have the following dimensions: $\dim(u) = N_x \times (N_y + 1)$, $\dim(v) = (N_x + 1) \times N_y$, $\dim(p) = (N_x + 1) \times (N_y + 1)$.

The spatial discretisation for the PM-solver on this grid follows the discretisation presented in Seibold (2008). Second-derivatives are approximated using central differences, while first-derivatives are approximated using backward differences. This discretisation places the resulting quantities at beneficial positions on the grid. Consider for example taking the divergence of the intermediate velocity $\hat{\vec{u}}$ for the right-hand side (RHS) of equation (PPE):

$$\begin{aligned}\nabla\cdot\hat{\vec{u}} &= \left(\frac{\partial\hat{u}}{\partial x}\right) + \left(\frac{\partial\hat{v}}{\partial y}\right) \\ &\sim \frac{\hat{u}_{i,j} - \hat{u}_{i-1,j}}{\Delta x} + \frac{\hat{v}_{i,j} - \hat{v}_{i,j-1}}{\Delta y} \qquad (3)\end{aligned}$$

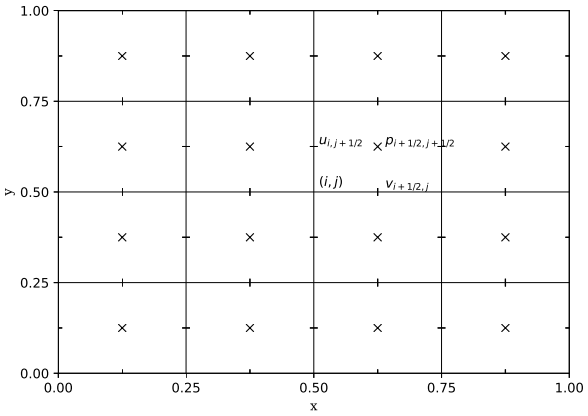

Figure 1: Staggered variable definitions on a 5x5 grid. The horizontal velocities u are defined on the vertical edges of the grid cells at positions $(i, j + \frac{1}{2})$, whereas the vertical velocities v are defined on the horizontal edges at positions $(i + \frac{1}{2}, j)$. The pressure is defined at the centres of the grid cells at positions $(i + \frac{1}{2}, j + \frac{1}{2})$. Note that there are ghost variables outside the border of the domain that are not shown here.

The resulting quantity $b_{i+\frac{1}{2},j+\frac{1}{2}}$ is conveniently defined at the position of the pressure p in the

centres of the grid cells and equation (PPE) can be solved directly. For the full discretisation of all equations we refer to Seibold (2008).

C *Schematic vorticity stream-function method*

For the vorticity stream-function method, the continuous functions ω, ψ and $\vec{u}$ are being discretised on a *co-located* grid where all three variables are defined at the same positions. This is in line with the purpose of this project to investigate a range of techniques.

The vorticity transport equation (VTE) is discretised using the explicit forward-in-time-central-in-space (FTCS) scheme as presented in Salih (2013). This is one of the most elementary illustrations of finite difference discretisations. As the name suggests, it uses central differencing in space and forward Euler differencing in time to approximate the partial derivatives of equation (VTE).

In 2D, the sequence to obtain the flow using the VSF scheme is as follows:

1. Evolve the current vorticity ω using the current velocity field $\vec{u}$ according to

$$\frac{\partial \omega}{\partial t} = -\left(\vec{u}\cdot\nabla\right)\omega + \frac{1}{\mathrm{Re}}\nabla^2\omega \tag{4}$$

$$\implies \quad \frac{\omega^{n+1}-\omega^n}{\Delta t} = -u^n\frac{\partial \omega^n}{\partial x} - v^n\frac{\partial \omega^n}{\partial y} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2\omega^n}{\partial x^2}+\frac{\partial^2\omega^n}{\partial y^2}\right) \quad . \tag{5}$$

2. Solve the Poisson equation (PSE) for the stream-function, $\nabla^2\psi^{n+1} = -\omega^{n+1}$.

3. Update the velocity field using the solution ψ^{n+1} from step (2) according to

$$u^{n+1} = \frac{\partial \psi^{n+1}}{\partial y} \qquad v^{n+1} = -\frac{\partial \psi^{n+1}}{\partial x} \quad , \tag{6}$$

   thereby ensuring a divergence free final velocity field.

As in the projection method, central differences are used to approximate second-derivatives. The difference here is that first-derivatives are also approximated with central differences, as opposed to backward differences. For the full spatial discretisation we refer to Salih (2013).

D *Boundary conditions*

The use of central differences in the projection method solver to approximate second-derivatives means that at some positions one or two velocity values used in the discretisation come from the boundary since

$$\nabla^2\cdot U \quad \implies \quad \frac{U^n_{i+1,j}-2U^n_{i,j}+U^n_{i-1,j}}{\Delta x^2} + \frac{U^n_{i,j+1}-2U^n_{i,j}+U^n_{i,j-1}}{\Delta y^2} \quad . \tag{7}$$

Similarly, the discretisation of the viscous term in the VSF-solver requires one or two boundary values of the vorticity at certain points. The boundary values depend on the specific problem that is studied.

D.1 Model problem 1: Lid-driven cavity

The LDC is the standard problem to assess the performance of two-dimensional solvers in CFD for viscous, incompressible flow. It is good for testing solvers, as it has simple boundary conditions and there exists a vast amount of literature to compare with.

The NSE are solved inside a square cavity with no-slip boundary conditions (B.C.s) on the side and a prescribed tangential/ horizontal velocity along the top. Along the top, bottom, left and right wall (respectively *T, B, L, R*), the B.C.s on the velocity are

$$U_{\mathrm{T}} = U_{top}\ , \quad U_{\mathrm{B}} = U_{\mathrm{L}} = U_{\mathrm{R}} = 0\ , \quad V_{\mathrm{T}} = V_{\mathrm{B}} = V_{\mathrm{L}} = V_{\mathrm{R}} = 0 \quad . \tag{8}$$

For the PM-solver, additionally, B.C.s for the pressure have to be provided and similarly for the stream-function and vorticity in the VSF-solver.

In case of the pressure, conventionally homogeneous Neumann boundary conditions are prescribed wherever no-slip B.C.s are provided for the velocity. For the lid-driven cavity this means that Neumann B.C.s are provided everywhere. The pressure is thus only defined up to a constant, but, since only the gradient of the pressure enters the momentum equation, this is fine. In this work we will prescribe vanishing Neumann B.C.s for the pressure everywhere, i.e. set the constant value of the derivative of p at the boundary to zero. For the interested reader, Hinch (2020) chapter 3.6 explores non-zero derivatives for the pressure.

The ghost values outside the domain are required to define boundary conditions for the viscous term $\nabla^2\vec{u} = L\vec{u}$ in equation(intmvel). Due to the nature of the staggered grid, u has Dirichlet boundary conditions in the horizontal direction where the variable is defined at the cell edge, whereas in the vertical direction the Dirichlet condition is defined in the centre between two data points. Consider for example the y-derivative at the top boundary:

$$Lu_{i,N_y}\big|_y = \frac{u_{i,N_y-1} - 2u_{i,N_y} + u_{i,N_y+1}}{\Delta y^2} = \frac{u_{i,N_y-1} - 3u_{i,N_y}}{\Delta y^2} + \frac{2U_{\mathrm{T}}}{\Delta y^2} \tag{9}$$

Here it was used that $u_{i,Ny+1} = 2U_{\mathrm{T}} - u_{i,Ny} = 2U_{top} - u_{i,Ny}\,, \quad i = 0, ..., N_x\,,$ (see Seibold (2008)). In contrast, at the left boundary in the x-direction, the prescribed boundary value is directly used:

$$Lu_{1,j}|_x = \frac{u_{2,j} - 2u_{1,j} + u_{0,j}}{\Delta x^2} = \frac{u_{2,j} - 2u_{1,j} + U_{\mathrm{L}}}{\Delta x^2} \tag{10}$$

The same procedure needs to be applied for Lv, where the ghost velocity is required in the horizontal direction.

In case of the VSF-solver, since there is no normal velocity component on any of the sides of the cavity, i.e. the flow is parallel along the sides, they represent a streamline of constant value $\psi = c = const.$. Arbitrarily c can be chosen to be zero. In particular, the B.C.s on the velocity components also mean that the second derivatives of ψ in the tangential directions along the walls vanish. This reduces the Poisson equation for the stream-function (PSE) to

$$\left.\frac{\partial^2\psi}{\partial n^2}\right|_{\text{wall}} = -\omega_{\text{wall}} \tag{11}$$

where n is the normal direction. This relationship is used to define boundary conditions for ω. We refer the reader to Salih (2013) and use their equations (31) - (34) for ω at the boundary. To

give an example, ω at the top boundary is given as

$$\omega_{i,N_y} = 2\frac{\psi_{i,N_y} - \psi_{i,N_y-1}}{\Delta y} - \frac{2U_{top}}{\Delta y} \quad . \tag{12}$$

D.2 Model problem 2: Kelvin-Helmholtz instability

The NSE are solved inside a channel. The boundary conditions on the velocity components along the vertical direction at the top and bottom wall are again no-slip conditions. Along the horizontal direction, periodic boundary conditions are prescribed:

$$U_{\mathrm{T}} = u_{i,N_y} = u_{i,N_y-1} \ , \quad U_{\mathrm{B}} = u_{i,0} = u_{i,1} \ , \quad U_{\mathrm{L}} = U_{\mathrm{R}} \tag{13}$$

$$V_{\mathrm{T}} = v_{i,N_y} = v_{i,N_y-1} \ , \quad V_{\mathrm{B}} = v_{i,0} = v_{i,1} \ , \quad V_{\mathrm{L}} = V_{\mathrm{R}} \tag{14}$$

In the case of the PM-solver, the pressure still receives Neumann B.C.s in the vertical direction. Along the horizontal, periodic boundary conditions apply.

In case of the VSF-solver, the B.C.s on the vorticity change entirely. Along the vertical, Neumann B.C.s are imposed, while along the horizontal the same FTCS scheme that is used for the interior points has to be applied. Here it needs to be enforced that $\omega_{0,j} = \omega_{N_x,j}$, i.e. at the left boundary the neighbouring points of $\omega_{0,j}$ are $\omega_{1,j}$ and $\omega_{N_x-1,j}$, and similarly at the right boundary.

In case of the stream-function ψ, periodic B.C.s apply along the horizontal while Dirichlet conditions are imposed along the vertical.

E Solving elliptic partial differential equations - Fast Fourier solvers

The methods introduced above require solving the two-dimensional Poisson equation at various points. In the projection method the Poisson equation is solved to obtain the pressure p, while in the vorticity stream-function method it is solved to obtain the stream-function ψ. In addition, in the semi-implicit discretisation of the projection method the Helmholtz equation is solved for the intermediate velocity. The general Helmholtz equation has the form

$$\left(-\nabla^2 + \kappa\right)\phi = A\phi = f \ , \quad \kappa > 0 \quad . \tag{HH}$$

In the Poisson case, $\kappa = 0$. The Helmholtz equation is an elliptic PDE and there are a number of different approaches to obtain the solution ϕ. In case of a simple Cartesian geometry and Dirichlet, Neumann or periodic boundary conditions, the so-called Fourier method is a suitable and relatively easy method to implement. Furthermore, in terms of computational cost, it is a very efficient method. Direct inversion of the matrix A, for example, has a computational cost of $O(N^6)$ operations on a square $N \times N$ grid. The popular iterative successive over-relaxation (SOR) method reduces this to a cost of $O(N^3)$, but the Fourier method is even better than that at a cost of $O(N^2 \ln N)$, see Hinch (2020).

The general idea of the method is to transform the set of equations from physical space to Fourier space, compute the Fourier coefficients of the solution in Fourier space, and finally transform the solution back into physical space:

Let $\mathcal{F}$ denote the n-dimensional Fourier transform. Given an arbitrary $x \in \mathbb{R}^n$ and a solution $\phi : \mathbb{R}^n \to \mathbb{R}$,

$$\mathcal{F}\left(\frac{\partial}{\partial x_0}\phi\right) = i\xi_0\mathcal{F}(\phi) \ , \quad \mathcal{F}\left(\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_2}\phi\right) = i^2\xi_1\xi_2\mathcal{F}(\phi) \ , \quad (etc.) \ , \tag{15}$$

where $i = \sqrt{-1}$ is the imaginary unit and the ξ_l denote n Fourier coefficients. Applying the Fourier transform to the left- and right-hand-side of equation (HH), we thus obtain

$$\left(\sum_{l=0}^{n} \xi_l^2 + \kappa\right) \mathcal{F}(\phi) = \mathcal{F}(f) \quad \Longleftrightarrow \quad \mathcal{F}(\phi) = \frac{\mathcal{F}(f)}{(\sum_{l=0}^{n} \xi_l^2 + \kappa)} \tag{16}$$

in Fourier space, which is easily solvable. The solution is then obtained via n inverse Fourier transforms:

$$\phi = \mathcal{F}^{-1} \frac{\mathcal{F}(f)}{(\sum_{l=0}^{n} \xi_l^2 + \kappa)} \quad . \tag{17}$$

Depending on the specific boundary conditions, different Fourier approximations of ϕ are suitable. As demonstrated in Swarztrauber (1984), for a periodic solution the discrete Fourier transform (DFT) is appropriate, while for Dirichlet and Neumann problems the discrete sine transform (DST) and the discrete cosine transform (DCT) respectively are sufficient. For a detailed derivation of the relevant two dimensional transforms, see Wiegmann (1999).

F Python implementation

F.1 Discrete Fourier transforms: SciPy's FFT library

Python's SciPy package has a number of fast discrete Fourier Transforms in the Fast Fourier Transform library scipy.fft. When implementing those, it is important to pay attention to normalisation between the forward and the inverse transforms, as well as to be aware of the different 'types' these functions cover. Please see lines 236-270 in the PM-solver and lines 161-169 in the VSF-solver (for the LDC problem).

***Lid-driven cavity problem*:** For the components of the intermediate velocity in the projection method Dirichlet B.C.s are prescribed. Hence, the DST should be used. Due to the nature of the staggered grid though, different DST types should be used for different directions. For u, the DST-I should be used on the x axis while DST-II should be used on the y axis. The axes are switched for the v velocity. The inverse of DST-I is DST-I itself, while DST-III is the inverse of DST-II. For the pressure, the type II/III cosine transform operation should be used in both the x and the y direction. Again, type II- and type III-DCT are inverse to each other. Furthermore, in order to ensure the uniqueness of the solution, the DCT requires setting $\hat{P}_{0,0} = 0$. For the stream-function on the co-located grid in the VSF-solver, the type I-DST should be chosen for both spatial dimensions.

The B.C.s in both solvers are such that the transforms can be performed directly. Due to the shape of the sine and cosine functions, either $U = c$ or $\partial U/\partial n = c$ at the boundary, which is exactly the B.C.s we want for the LDC. For all terms $c = 0$. The only exception to this is the horizontal term in the intermediate velocity. As indicated above, here the top boundary condition has to be enforced in the viscous term.

***Kelvin-Helmholtz instability problem*:** All cosine transforms in the horizontal direction have to be replaced by discrete Fourier transforms to handle periodicity.

F.2 Overall code layout and visualising the solution

The layout of the solvers is the same for all methods. First, all parameters are set and the computational grid is defined. Next, all data arrays are initiated or pre-computed. This is especially relevant for the large coefficient matrices used for the calculations in Fourier space. After that, the main time loop is initialised. In this loop the values stored in the interior of the data arrays are updated as outlined in the sections above and boundary values are directly enforced.

When the code is executed, a number of things happen. In order for students to gain an intuition for how parameter changes affect the solutions, initially the code prints relevant information about the problem settings to the command line, such as Reynolds number, grid size and time settings. Subsequently, a window showing a contour-plot of the velocity (LDC problem) or the vorticity (KHI problem) opens which updates every few time-steps, thus giving the user an idea of the dynamics of the system. When the final time is reached, a directory is created and plots of vorticity and velocity are saved. For the VSF-solver, additionally a plot of $\psi(t_{final})$ is saved. Finally, the value of $\nabla \cdot \vec{u}$ and the physical time needed for the computation are printed. The former to see if equation (NS1) is satisfied and the later for general information.

In terms of efficient programming practices, instead of using nested For-loops to update the elements of the data arrays, convenient array operations are used. This speeds up the execution of code, see e.g. Chudoba et al. (2013). Moreover, the original arrays are overwritten several times in order to save memory.

IV RESULTS AND EVALUATION

In this section, the results obtained with the Navier-Stokes solvers are presented. Problem settings, time step configuration and performance analyses are shown. The first two sections show results and evaluations for the lid-driven cavity problem and the Kelvin-Helmholtz instability problem. Section C contains general remarks about the solvers and the final section gives an overview of the visualisation script. The simulations were performed on a device using an Intel Core i7-4710HQ $2.5\,\mathrm{GHz}$ processor with $8.0\,\mathrm{GB}$ RAM.

A Model problem 1: Lid-driven cavity

Results obtained for the lid-driven cavity problem are compared to the benchmark results presented in U. Ghia et al. (1982) for Reynolds numbers $\mathrm{Re} \in \{100, 400, 1000, 3200\}$. First, qualitative shapes of the solution are shown to demonstrate the qualitative accurateness of the solution. Subsequently, the horizontal and vertical velocities along the vertical/ horizontal line through the geometric centre of the domain are studied for square grids of size $N^2 \in \{32^2, 64^2, 128^2, 256^2\}$ and grid spacing $h_N = \Delta x_N = \Delta y_N$. Lastly, values of the vorticity along the top boundary of the domain are compared. Notably, in case of the horizontal velocity, for Reynolds number $\mathrm{Re} = 3200$ the data point at $y = 0.4532$ given in the literature was omitted in all comparisons, as it does not follow the trend of the overall curve. Similarly, in case of the vertical velocity, for Reynolds number $\mathrm{Re} = 400$ the data point at $x = 0.9063$ was omitted. These deviations from the general trend can possibly be attributed to printing errors in the paper.

The time step used for each computation was chosen as the largest stable step. In case of the explicit vorticity stream-function solver, the time step is subject to stability restrictions. For an outline of the restrictions see Hinch (2020), chapter 2.8. In principal, the numerical schemes are

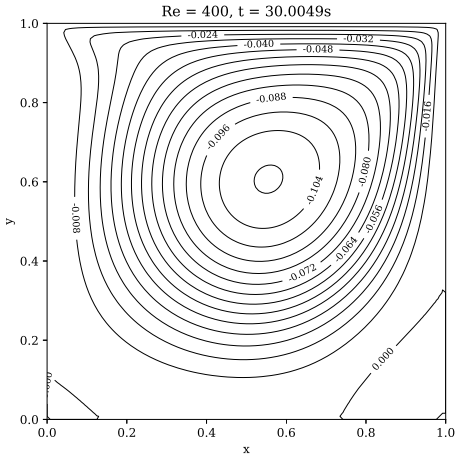

Figure 2: Stream-function profile obtained with the VSF-solver for Re $= 400$ on a grid of size $N^2 = 128^2$.

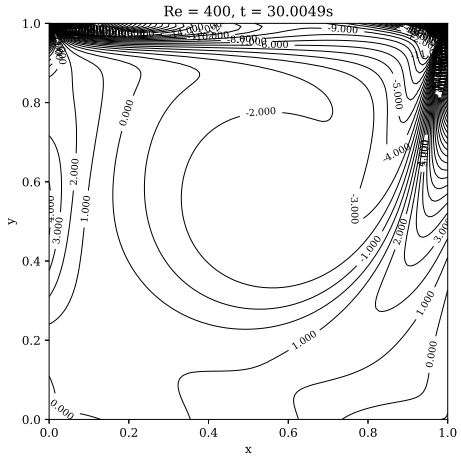

Figure 3: Vorticity profile obtained with the VSF-solver for Re $= 400$ on a grid of size $N^2 = 128^2$.

only stable if the quantities are not transported too far across the grid with each step. Thus, $\Delta t = c * \mathrm{Re}/2 \cdot \frac{\Delta x^2 \Delta y^2}{\Delta x^2 + \Delta y^2}$ was used. Here, the scaling factor c was chosen depending on the Reynolds number, see lines 69-83 in the code (`solve_NSE_liddrivencavity_with_vsf.py`). In case of the semi-implicit projection method solver, the time step can be set larger, in particular due to the implicit treatment of the diffusion term. Ultimately, $\Delta t = c * 1.0 \times 10^{-2}$ was chosen by trial and error with $c = 1$ for low Reynolds numbers, $c = 0.8$ for intermediate Reynolds numbers and $c = 0.5$ for high Reynolds numbers, see lines 100-105 in the code (`solve_NSE_liddrivencavity_with_pm.py`).

The amount of physical time needed to reach a steady state solution depends on the parameters of the problem setting, such as Reynolds number and velocity of the lid. Thaker and Banerjee (2011) suggest to let the lid pass over the top at least ten times to ensure a stationary solution. For good measure, they set $T = 15.0\,\mathrm{s}$ for a lid-velocity of $U_{top} = 1\,\mathrm{ms}^{-1}$ in a box of size $1\,\mathrm{m}^2$ and a time step of $\Delta t = 0.005$. As the methods used for the solvers presented in this report are not fully unconditionally stable, T was varied depending on Reynolds number set.

A.1 Qualitative results

Figures 2 and 3 show the stream-function and vorticity profiles obtained with the vorticity stream-function solver for Reynolds number Re $= 400$. The overall shapes of the profiles match those shown in U. Ghia et al. (1982), and values obtained for specific contour-lines of the solutions match those presented in table III of U. Ghia et al. (1982). Similar profiles are obtained for the other Reynolds numbers and similarly with the projection method solver. These are only a qualitative representation of the solution. The following sections compare concrete values obtained at different points in the domain.

A.2 Velocities through the geometric centre of the domain and vorticity at the top boundary - results

***PM-solver*:** Figures 4 and 5 show the results obtained with the projection method solver for the range of Reynolds numbers. While figure 4 shows the shape of the horizontal velocity along the

vertical line through the geometric centre of the domain, $(x = 0.5, y)$, figure 5 shows the results obtained for the vertical velocity along the horizontal line through the geometric centre of the domain. Literature values are taken from tables I and II of U. Ghia et al. (1982).

***VSF-solver*:** Figures 6 and 7 show the results obtained with the vorticity stream-function solver for the range of Reynolds numbers. Again, the horizontal and vertical velocities are shown separately for a range of Reynolds numbers in comparison to values given in tables I and II of U. Ghia et al. (1982). Here grid sizes $N^2 = 128^2$ and $N^2 = 256^2$ for the VSF-solver are being compared to literature values obtained on a $N^2 = 128^2$ grid.

***Convergence*:** Figure 10 shows the convergence of the error between benchmark values and values obtained with the two solvers for both velocity components as the grid-point separation is changed. The error in the final velocity field was calculated with respect to the results of U. Ghia et al. (1982) on a $N^2 = 128^2$ sized grid, $Error = \max |U - U_{\text{Ghia(N=128)}}|$. The left-hand plot of 10 indicates the error in the horizontal velocity component u, while the right-hand plot indicates the error in the vertical velocity component v. Values for Re $= 400$ are compared, as this value of the Reynolds number is neither the low nor the high extreme of the studied Re-values. Trend lines of order $O(h)$/ $O(h^2)$ are indicated in the plots (red/ blue lines).

***Vorticity*:** Figures 8 and 9 depict results for the vorticity along the top boundary of the domain. Figure 8 shows results obtained with the PM-solver for all Reynolds numbers on a $N^2 = 128^2$ grid. Figure 9 shows results for the VSF-solver for grids of size $N^2 = 128^2$ and $N^2 = 256^2$, again compared to benchmark values on a $N^2 = 128^2$ grid.

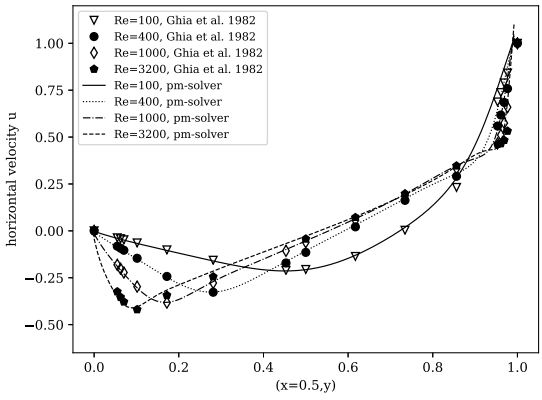

Figure 4: Horizontal velocity along the vertical line through the geometric centre $(x = 0.5, y)$ obtained with the PM-solver. Results for a grid size $N^2 = 128^2$ are compared to literature values.

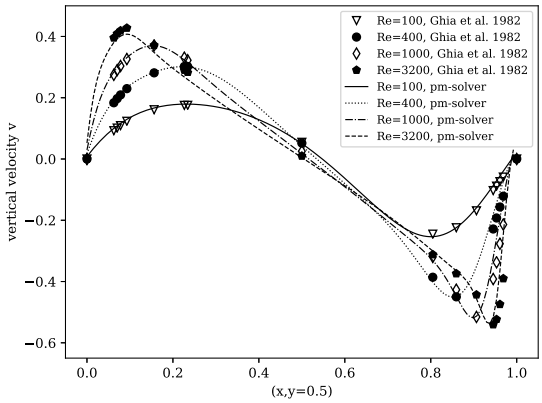

Figure 5: Vertical velocity along the horizontal line through the geometric centre $(x, y = 0.5)$ obtained with the PM-solver. Results for a grid size $N^2 = 128^2$ are compared to literature values.

A.3 Evaluation

Consider the direct comparisons of figures 5 and 7 and figures 8 and 9. While the PM-solver already approximates the literature values for velocity and vorticity quite well at grid resolution $h = 1/128$ for all Reynolds numbers, the VSF-solver only approaches the correct solutions at

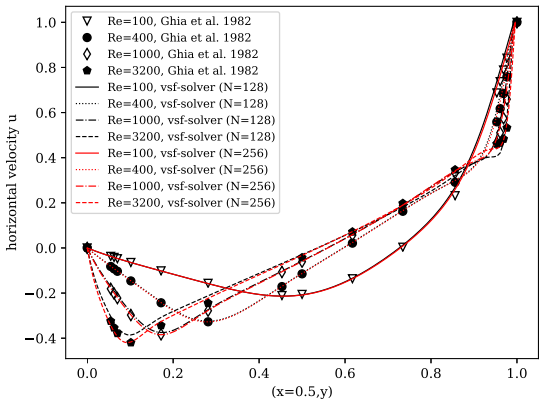

Figure 6: Horizontal velocity along the vertical line through the geometric centre $(x = 0.5, y)$ obtained with the VSF-solver. Results for a grid sizes $N^2 = 128^2$ and $N^2 = 256^2$ are compared to literature values.

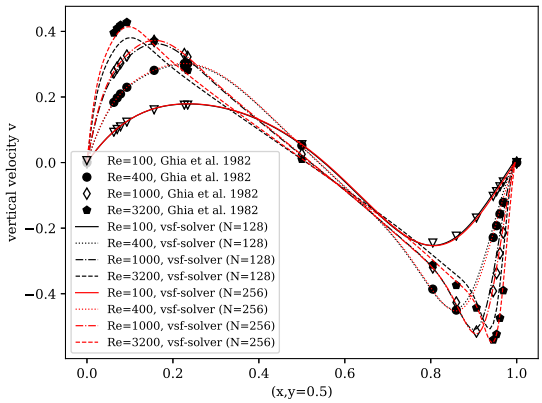

Figure 7: Vertical velocity along the horizontal line through the geometric centre $(x, y = 0.5)$ obtained with the VSF-solver. Results for a grid sizes $N^2 = 128^2$ and $N^2 = 256^2$ are compared to literature values.

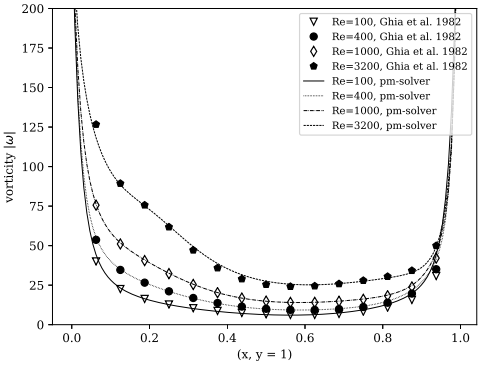

Figure 8: Vorticity along top boundary $(x, y = 1)$ in the LDC, PM-solver for grid spacing $h = \frac{1}{128}$.

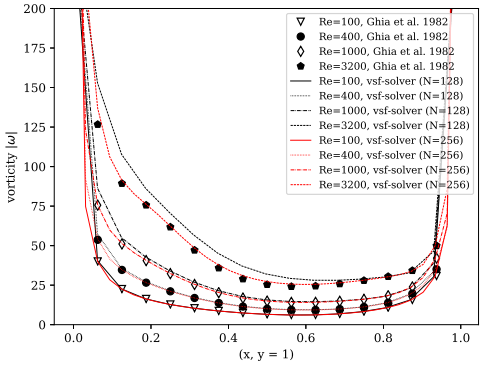

Figure 9: Vorticity along top boundary $(x, y = 1)$ in the LDC, VSF-solver for grid spacings $h = \frac{1}{128}$, $h = \frac{1}{256}$.

grid resolution $h = 1/256$ as the Reynolds number increases (compare red and black lines). Nonetheless, the overall overlap with literature values is good, both for the velocities inside the domain and for the vorticity at the top boundary, even at higher Reynolds numbers. This speaks to the robustness of both solvers. Furthermore, figure 10 indicates mesh-convergence for both solvers: the error with respect to the literature values decreases as the computational mesh is refined. While the PM-solver appears to follow an $O(h)$ trend in case of the horizontal velocity u, for the vertical velocity v the convergence is less than $O(h)$. This behaviour is anticipated, as, as previously mentioned, Guermond et al. (2006) showed that the projection method is at best first-order accurate.

The VSF-solver follows a second-order trend for both velocities. Overall, it displays a lower error than the PM-solver except for $h = 1/32$ in case of the vertical velocity (see the right-hand plot of figure 10). This indicates that it has a higher overall accuracy than the PM-solver.

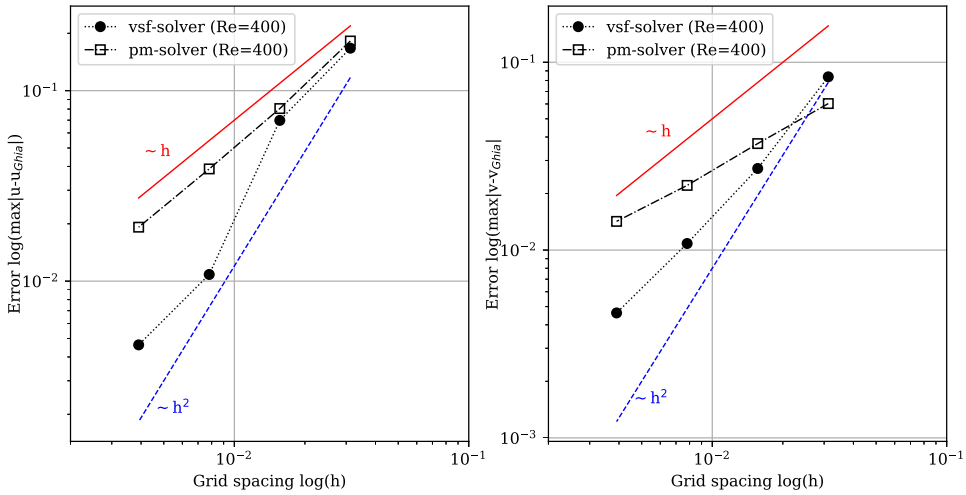

Figure 10: Convergence of solvers based on errors in horizontal velocity u (left) and vertical velocity v (right).

B Model problem 2: Kelvin-Helmholtz instability

Results obtained for the Kelvin-Helmholtz instability problem are compared to the benchmark results presented in Schroeder et al. (2018) for Reynolds numbers $\mathrm{Re} \in \{100, 1000\}$. First, like for the LDC solver, qualitative shapes of the solution are shown to demonstrate the qualitative accurateness of the solution. Subsequently, the evolutions of the kinetic energy K and the enstrophy ϵ of the flow over time is studied for square grids of size $N^2 \in \{32^2, 64^2, 128^2, 256^2\}$.

For all computations, the time step is set at $\Delta t = 3.6 \times 10^{-5}$, as given in the literature. The initial condition for the velocity is set as given in equation (2) of Schroeder et al. (2018): a tanh-shaped profile with a sinusoidal perturbation. In case of the VSF-solver, furthermore an initial vorticity

$$\begin{aligned}\omega_0 =& 16 c_n \pi^2 e^{\frac{-(y-0.5)^2}{\delta_0^2}} \left[4\cos(8\pi x) + 25\cos(20\pi x)\right] + \frac{2c_n}{\delta_0^2} e^{\frac{-(y-0.5)^2}{\delta_0^2}} \left[\cos(8\pi x) + \cos(20\pi x)\right] \\ &- \frac{4c_n(y-0.5)^2}{\delta_0^4} e^{\frac{-(y-0.5)^2}{\delta_0^2}} \left[\cos(8\pi x) + \cos(20\pi x)\right] - \frac{2}{\delta_0} \operatorname{sech}\left(\frac{2y-1}{\delta_0}\right)^2 \end{aligned} \tag{18}$$

is set, which was obtained analytically through the relationship (PSE). Here, $\delta_0 = L = 1/28$ is a characteristic length-scale of the flow and $c_n = 10^{-3}$ is a scaling factor. See figure 11 for a visual impression.

B.1 Qualitative results

Figure 11 shows the vorticity profile of the flow with Reynolds number $\mathrm{Re} = 1000$ at different instances in time. The overall evolution of the profile matches the description given in Schroeder et al. (2018): the initial sinusoidal perturbation develops into four vortexes, which in turn fuse into two vortexes until finally merging into one large vortex. All images were obtained with the projection method solver, except for the first image in the second row, which was obtained with the vorticity stream-function solver. The reason for this is given in section B.3 below.

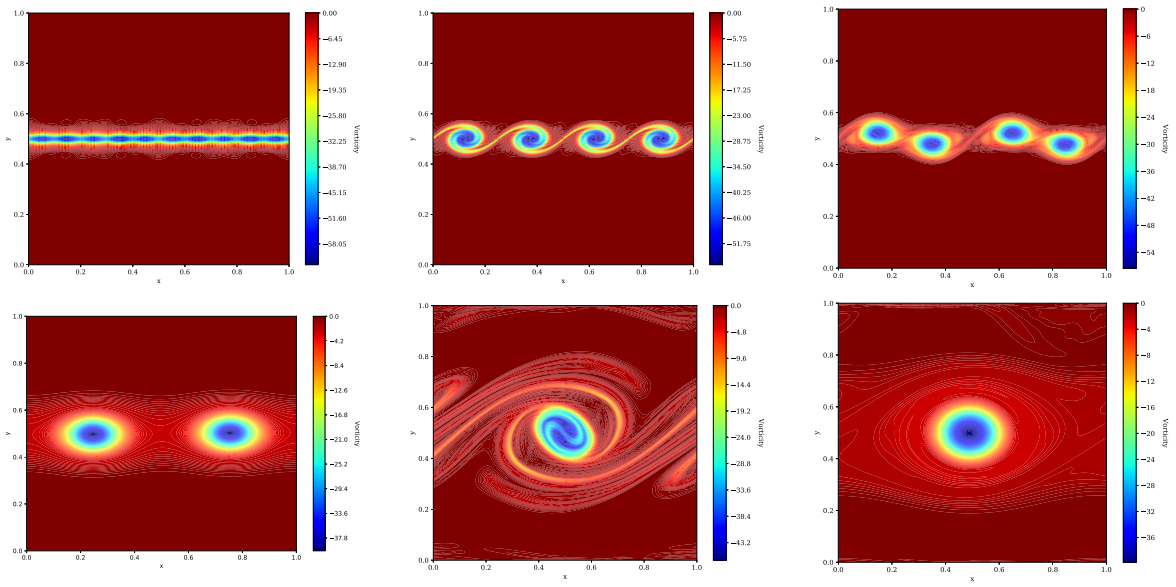

Figure 11: Time evolution of vorticity ω for Re $= 1000$.

B.2 Kinetic energy and enstrophy

***Projection method solver*:** Figures 12 and 14 show the results obtained with the projection method solver for both Reynolds numbers. Here, results obtained on grid sizes $N^2 = 128^2$ and $N^2 = 256^2$ are being compared to literature values obtained on grids of size $N^2 = 128^2$.

***Vorticity stream-function solver*:** Figures 13 and 15 show the results obtained with the vorticity stream-function solver for both Reynolds numbers. Again, results obtained on grid sizes $N^2 = 128^2$ and $N^2 = 256^2$ are being compared to literature values obtained on grids of size $N^2 = 128^2$.

B.3 Evaluation

***Qualitative behaviour*:** While Schroeder et al. (2018) give data for up to $\bar{t} = 400$, the authors state that beyond $\bar{t} = 200$ their values become unreliable. This is based on the observation that the timing of the formation of the final vortex cannot be reliably determined. Indeed, this was seen in the behaviour of both solvers studied in this report. In the simulations performed by Schroeder et al. (2018), the two vortexes that form due to the fusion of the four initial vortices rotate on their own for a while before fusing into the final vortex. The solution given by the projection method solver skipped this stage entirely and the two intermediate vortices merged immediately for both Reynolds numbers studied. In contrast to this, in the vorticity stream-function solver the intermediate vortexes rotated for a prolonged period of time and merging was only complete long after $\bar{t} = 400$. Hence, in the qualitative display, the intermediate two-vortex stage image was obtained with the VSF-solver and not like the other images with the PM-solver.

***Kinetic energy*:** As demonstrated in Schroeder et al. (2018), the physically correct behaviour of the kinetic energy of the flow $\mathcal{K}(\vec{u}) = \frac{1}{2}\int_\Omega |\vec{u}|^2 d\vec{x}$ is to decrease monotonically with time. Globally, this behaviour can be seen in figures 12 and 13 for both solvers.

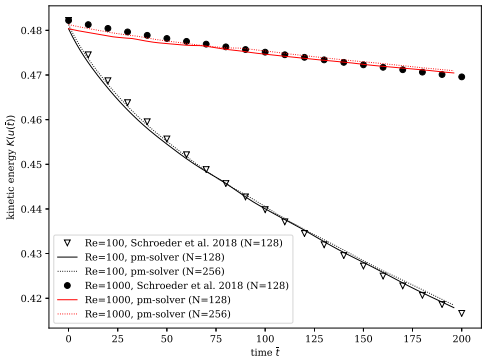

Figure 12: Kinetic energy PM-solver over time.

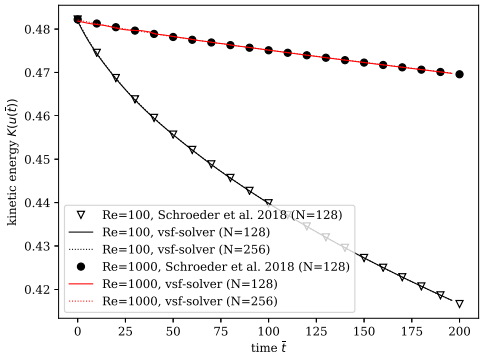

Figure 13: Kinetic energy VSF-solver over time.

***Enstrophy*:** Similarly to the kinetic energy, the enstrophy $\epsilon(\omega) = \frac{1}{2}\int_\Omega |\omega|^2 d\vec{x}$ physically decreases over time. As mentioned in the qualitative evaluation, the timing of the merging processes is a main difference between the solvers. The merging processes can be seen in plateaus in the enstrophy, especially for $\mathrm{Re} = 1000$. Figure 14 shows that at around $\bar{t} = 90$ the enstrophy values given by the PM-solver fall below the values given by Schroeder et al. (2018). This can be attributed to the fact that the intermediate two-vortex stage is skipped.

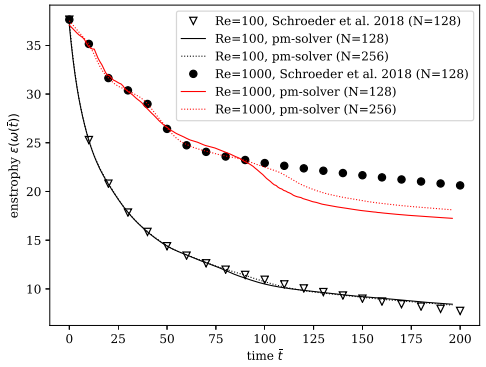

Figure 14: Enstrophy of PM-solver over time.

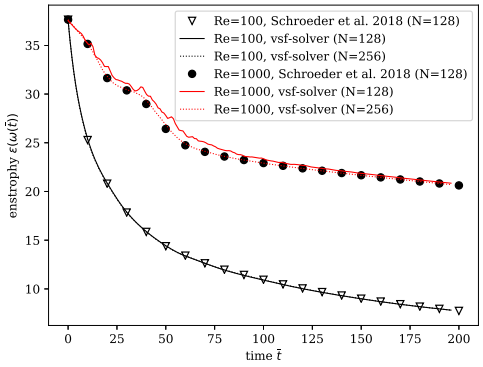

Figure 15: Enstrophy of VSF-solver over time.

***Convergence*:** Figure 16 shows the maximum error in the solutions for kinetic energy and enstrophy with respect to the literature. While in the lid-driven cavity problem convergence was assessed based on the final velocity field, in this case the error is measured over time. Similarly, as seen in the convergence analysis for the previous problem, the error in the PM-solver converges approximately with order $O(h)$. For the VSF-solver results are not as clear as previously. Overall the convergence seems to be at best of order $O(h^2)$ for the kinetic energy, whereas the convergences at finer meshes is more at $O(h^3)$. Visually, 15 shows that a mesh of size $N^2 = 256^2$ used in the solver approximates the benchmark values at mesh size $N^2 = 128^2$ quite well.

The inconclusive convergence behaviour of the error can be attributed to the strong time-dependence of the error. If times of merger-events do not coincide with when they take place in the literature, inevitably the error will appear stronger. Hence, choosing a different approach to determining the error might have been better suited to assess the convergence of the solvers for this case. Alternatively, a higher-order discretisation of ψ when obtaining $\vec{u}$ could help to get a more conclusive behaviour. Nonetheless, the figure overall indicates that both solvers approach the benchmark solution as the computational mesh is refined.

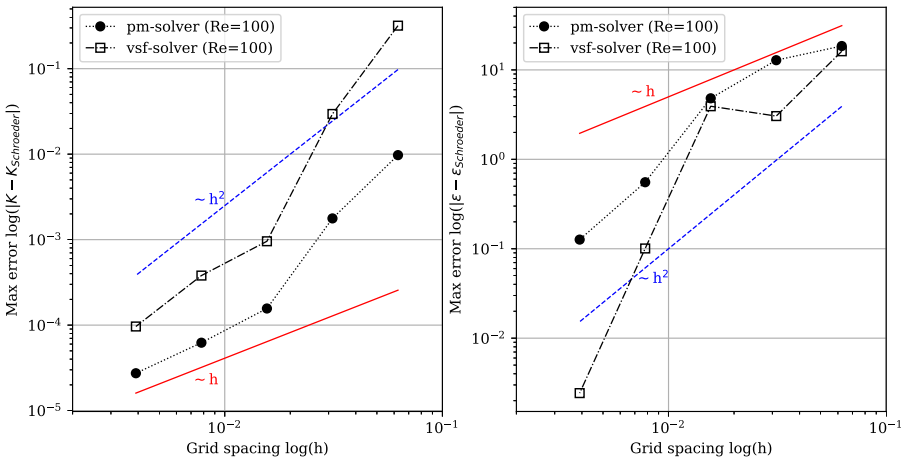

Figure 16: Convergence of solvers based on errors in kinetic energy K (left) and enstrophy ϵ (right). Trendlines for convergence rates of $O(h)$ and $O(h^2)$ are indicated (red/ blue lines respectively).

C *General remarks*

Due to the characteristic length scale of the flow, $\delta_0 = 1/28$, the viscosity of the fluid in the KHI setup is a lot lower than in the lid-driven cavity case. Where previously the solvers were stable for up to $\text{Re} = 3200 \Leftrightarrow \nu = 1/3200$, now flows are studied of viscosity $\text{Re} = 1000 \Leftrightarrow \nu = 1/28 \times 1000$. At those viscosities the solvers might require a smaller time-step or higher order schemes to fully resolve the flow, which could explain the 'wiggly-ness' of the solution for $N = 128$ in figure 15. Schroeder et al. (2018) use a higher order finite element method.

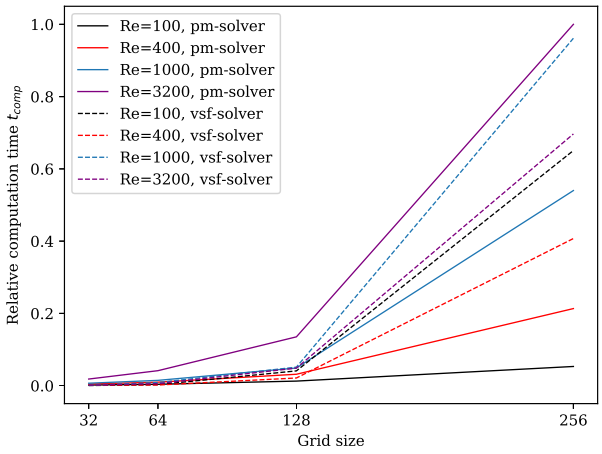

Figure 17: Relative computation times for the LDC problem. Results for the VSF-solver are shown in dashed lines, while results for the PM-solver for the same Reynolds number are shown in solid lines.

Moreover, when assessing the solvers it was monitored whether the continuity-condition (NS1) was fulfilled. For the LDC setting, the value was at most in the order of 10^{-12} for both solvers. For the KHI problem however, the value stayed above 10^{-6} for the PM-solver. It was found that this error can be attributed to an inconsistency at the vertical boundary. Schroeder et al. (2018) use free-slip/ no-penetration B.C.s in the vertical direction, while we have used no-slip B.C.s. Free-slip B.C.s might actually be the correct boundary conditions for this flow, which is why the value of $\nabla \cdot \vec{u}$ close to the boundary does not equal zero in our solvers. Nevertheless, since the overall behaviour matches the literature, the error stemming from this inconsistency does not seem to have a large influence.

Figure 17 shows the relative computation times $t^N_{\text{comp}} = t^N_{\text{this Re}}/t^N_{\text{max, all Re}}$ needed by the solvers for different mesh-sizes N and Reynolds numbers to compute the LDC. Results for the VSF-solver are shown in dashed lines, while results for the PM-solver for the same Reynolds number

are shown in solid lines. At the finest grid-resolution, the PM-solver is faster than the VSF-solver for all Reynolds numbers except for the highest. At Re $= 3200$, the time-step in the PM-solver approaches the time-step used in the VSF-solver. Since overall the VSF-solver performs less operations per time-step, it becomes faster than the PM-solver.

The two solvers each have different strengths. The computation time comparison indicates that the PM-solver is preferable when it comes to stability at larger time-steps. Furthermore, if needed it can be expanded to 3D calculations, which is not possible for the VSF-solver. Despite these benefits, in terms of all other features it performs less well than the VSF-solver. First of all, it does not perform a calculation of the stream-function ψ, which might arguably be a useful quantity to obtain. Second, it is less accurate than the VSF-solver, especially seen in the timing of the mergers for the KHI problem. Finally, it is more difficult to implement, as seen in the overall number of lines of code: 336 for the PM-solver, but only 245 for the VSF-solver.

Ultimately, we believe that the time-step argument is significant though. As long as students are aware of the reduced accuracy, if they want to gain a qualitative understanding of the dynamics of the flow they might be better served with a fast visual presentation of the system, at least for low to intermediate Reynolds numbers.

D Visualisation scripts

Throughout the Continuum Mechanics lecture the students are presented with examples of stationary velocity fields. A secondary goal of this project was to produce Python scripts that enable students to replicate these stationary examples. This sections presents the code layout and its general functions.

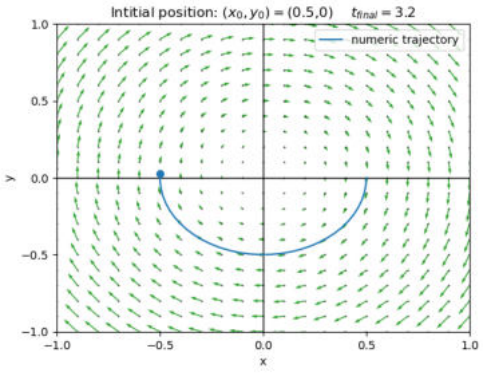

Figure 18: Example result

D.1 Code layout and example result

At the beginning of the programme, the velocity field is defined. Furthermore, the user can specify the initial position of a particle in the field and the amount of time the particle should travel in the field. As the code is executed, first the velocity field is plotted and subsequently the path of the particle from the initial position is shown in time steps of size dt, see figure 18. Optionally, an analytic solution, specified by the user, can be plotted on top of the numerical solution (determined with SciPys odeint function). Thus, the examples presented in the lecture notes are easily replicated and students can verify analytically calculated solutions. The solutions are saved to a directory for later use.

V CONCLUSION

Overall, the aims of the project were fulfilled. Two solvers for the two-dimensional Navier Stokes equations for an incompressible fluid were implemented in Python, one employing the pressure-correction projection method approach and the other using the vorticity stream-function method. The experimental settings (Reynolds number, density, flow speed) for both solvers can be adjusted easily without having to change the overall code, thus enabling students to study the effect of these parameters on simple flows.

The finite difference discretisation to approach the numerical solution with the PM-solver was based on a semi-implicit combination of Adams-Bashforth and Crank Nicolson schemes in time, while the spatial discretisation used standard backward and central differences. The VSF-solver used the FTCS scheme with forward differences in time and central differences in space. While higher accuracy can be achieved with more complex discretisation schemes, for the purely educational purpose of this project, the approaches that were employed are sufficient.

The performance of the solvers was evaluated for two test problems. For the lid-driven cavity problem, both solvers showed good overall agreement with literature values for a range of Reynolds numbers and different time-step settings, thus giving proof of the robustness of the solvers under different settings. For the Kelvin-Helmholtz instability problem, both solvers displayed good alignment with literature values for $\mathrm{Re} = 100$. For $\mathrm{Re} = 1000$, the solvers approached the literature values as the grid was refined. The VSF-solver followed a time evolution of the KHI problem similar to that in the literature, while the PM-solver suffered from residual numerical errors, resulting in a more compressed time-evolution.

Overall, the semi-implicit discretisation for the PM-solver allowed for a larger time step than used in the VSF-method. Nonetheless, for high Re-flow (above $\sim \mathrm{Re} = 3000$), the largest stable time steps of the two methods approach each other. It was seen that the VSF solver becomes faster in this regime, as it involves less operations than the PM-solver.

Apart from the solvers, a simple Python script was created that, when given a stationary velocity field, can visualise that field as well as the paths that particles follow when placed in it.

Future work could focus on establishing more test cases for the NSE-solvers, such as a von-Karmann Vortex street. Alternatively, or in addition, the solvers could be expanded to compute solutions for compressible fluids.

References

Aref, H. and S. Balachandar (Oct. 2017). *A First Course in Computational Fluid Dynamics*. Cambridge University Press. Chap. 8: Multidimensional Partial Differential Equations, pp. 293–374. DOI: https://doi.org/10.1017/9781316823736.

Chorin, Alexandre (Oct. 1968). "Numerical Solution of the Navier-Stokes Equations". In: *Mathematics Of Computation* 22.104, pp. 745–762. URL: https://math.berkeley.edu/~chorin/chorin68.pdf.

Chudoba, R, V Sadílek, R Rypl, and M Vořechovský (2013). "Using Python for scientific computing: Efficient and flexible evaluation of the statistical characteristics of functions with multivariate random inputs". In: *Computer Physics Communications*, pp. 414–427. DOI: https://doi.org/10.1016/j.cpc.2012.08.021.

Fromm, Jacob (1964). *Methods in Computational Physics*. Ed. by B. Alder, S. Fernbach, and M. Rotenberg. Vol. 3: Fundamental Methods in Hydrodynamics. Academic Press. Chap. The Time Dependent Flow of an Incompressible Viscous Fluid, pp. 345–382.

Ghia, U., K. N. Ghia, and C. T. Shin (1982). "High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method". In: *Journal of Computational Physics* 48, pp. 387–411.

Ghia, U, K N Ghia, S G Rubin, and P K Khosla (1981). "Study of incompressible flow separation using primitive variables". In: *Computers & Fluids*, pp. 123–142. DOI: https://doi.org/10.1016/0045-7930(81)90021-9.

Guermond, J., P. Minev, and Jie Shen (2006). "An overview of projection methods for incompressible flows". In: *Computer Methods in Applied Mechanics and Engineering* 195, pp. 6011–6045. URL: https://www.math.tamu.edu/~guermond/PUBLICATIONS/guermond_minev_shen_CMAME_2006.pdf.

Harlow, F. H. and J. E. Welch (1965). "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface". In: *The Physics of Fluids* 8. DOI: https://doi.org/10.1063/1.1761178.

Hinch, E. J. (Apr. 2020). *Think Before You Compute - A Prelude to Computational Fluid Dynamics*. Cambridge University Press. Chap. 2 - Streamfunction-vorticity formulation, pp. 9–34. DOI: https://doi.org/10.1017/9781108855297.

Kim, J. and P. Moin (1985). "Application of a Fractional-Step Method to Incompressible Navier-Stokes Equations". In: *Journal of Computational Physics* 59, pp. 308–323. URL: https://www.ljll.math.upmc.fr/~frey/papers/Navier-Stokes/KimJ.,%20Application%20of%20a%20fractional%20step%20method%20to%20incompressible%20Navier-Stokes%20equations.pdf.

Morinishi, Y, T S Lund, O V Vasilyev, and P Moin (1998). "Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow". In: *Journal of Comp. Physics* 143, pp. 90–124. DOI: https://doi.org/10.1006/jcph.1998.5962.

Patankar, S. V. (1981). "A calculation procedure for two-dimensional elliptic situations". In: *Numerical Heat Transfer* 4, pp. 409–425. DOI: https://doi.org/10.1080/01495728108961801.

Patankar, S. V. and D. B. Spalding (1972). "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows". In: *International Journal of Heat and Mass Transfer* 15, pp. 1787–1806.

Salih, A. (Mar. 2013). "Streamfunction-Vorticity Formulation". In: *Department of Aerospace Engineering, Indian Institute of Space Science and Technology*. URL: https://www.iist.ac.in/sites/default/files/people/psi-omega.pdf.

Schroeder, P W et al. (2018). "On reference solutions and the sensitivity of the2D Kelvin–Helmholtz instability problem". In: *Computers and Mathematics with Applications*. DOI: https://doi.org/10.1016/j.camwa.2018.10.030.

Seibold, B (2008). *A compact and fast Matlab code solving the incompressibleNavier-Stokes equations on rectangular domains*. Massachusetts Institute of Technology. URL: http://math.mit.edu/~gs/cse/codes/mit18086_navierstokes.pdf.

Swarztrauber, P N (1984). *Fast Poisson Solvers*. Ed. by G H Golub. Vol. 24. Studies in Numerical Analysis, pp. 319–370.

Témam, R. (Jan. 1969). "Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (I)". French. In: *Arch. Rational Mech. Anal.* 32, pp. 135–153. DOI: https://doi.org/10.1007/BF00247678.

Thaker, J. and J. Banerjee (Dec. 2011). "Numerical Simulation of Flow in Lid-driven Cavity using OpenFOAM". In: *2nd International Conference in Nirma University, NUiCONE2011*.

Wiegmann, A (1999). *Fast Poisson, fast Helmholtz and fast linear elastostatic solvers on rectangular parallelepipeds*. Tech. rep. Lawrence Berkeley National Laboratory.