A numerical solution to the lid driven cavity using the finite volume method and the Navier-Stokes equations with possible expansions to astrophysical applications

William Probyn

June 16, 2022

Abstract

The conservative form of the governing equations for compressible fluid flow are derived using Reynold's Transport Theorem, and integrated over a group of fields using the finite volume method to approximate integral terms, the finite difference method is used to approximate any remaining higher order terms. The process of producing a linear set of equations is introduced and the necessary steps to find a solution to a large group of equations are explored. The lid driven cavity is scrutinized, with error values found when results are compared to reputable literature values across an ensemble of systems, producing an average error value for each dimension (across all seven systems introduced) at: $\chi_u^2 = 0.0085$ $\chi_v^2 = 0.0088$, suggesting each dimension has less than a 1% averaged probability of being a random distribution. Possible sources of errors, as well as methods to mitigate them, are explored. Expansions of mathematical theory to include more extreme physics are introduced, while the accessibility of writing this type of code is also mentioned. Conclusions are then drawn on the viability and productivity of pursuing development of these theories numerically, and future work is proposed to make different systems more accessible while attempting to retain productivity.

Contents

List of Figures

1 Introduction

First described by Euler in the 18^{th} century, the Euler equations describe the physical process of advection within a mathematical context. Limited at first to the constraints of incompressibility, work by Navier and Stokes would eventually expand the equation into a general description of the processes which drive continuum mechanics. By realising that fluid flow depends on the conservation of some key properties, a system of equations can be found which can describe a general system. The Reynold's Transport Theorem is used to convert a Lagrangian conservation law into a Eulerian counterpart, the necessary assumptions are described and the equations for compressible flow are derived. Gauss' Theorem can be used to reduce a divergence, ∇, operator into a flux integral across a cell face, the finite volume method is used to reduce surface integrals into finite sums. The equations are integrated across a group of fields, using finite sums, describing a domain in two dimensions. The resulting equations are linear and entirely closed, there are n unknowns and n equations. A simulation following the theories presented within is constructed, and the results are compared to literature using available data. An error estimate is produced, when compared to literature, to provide a quantitative statement for the reliability of the resultant code. Results are produced for a lid driven cavity set of initial conditions and methods are proposed to improve the theories employed. The equations are then explored further with the application of different physical processes, the equations can be used to describe fluids which are under the presence of external forces such as a magnetic field. Some different scenarios in which the finite volume method can be employed to numerically study cosmically rare occurrences are explored. The accessibility of coding these types of simulations are mentioned, with some proposed work which might make future work on such topics more easily accessible to those who might need to follow these methods. With the results found using the theory presented, conclusions are drawn on the confidence of possible results from these applications, and whether or not time should be spent on researching possible scenarios numerically.

2 Governing Equations

As mentioned, the Euler equations describe the advective processes which dominate the movement of inviscid flow:

$$\frac{\partial v_\mu}{\partial t} + (\nabla \cdot v_\mu)v_\mu = -\nabla p + g \tag{1}$$

They describe the trajectory a particle will take given a set of initial conditions under inviscid flow. A distinction is needed between the different ways to describe a system of particles. A Eulerian system aims to describe the system in the context of n number of fixed volume cells within the overall domain. Encoding how the average values of the individual cells are changing over each iteration, this is a very useful system to adhere to when using the Finite Volume Method. On the contrary, a Lagrangian system looks to describe the trajectory of a packet of virtual particles as they traverse through the domain. These two systems are bridged using the Reynold's Transport Theorem (RTT), relating to an intensive (irrespective of mass) property; such as mass flux, momentum flux, or energy flux, to and extensive (dependant on the mass) property; such as mass, momentum, or energy:

$$\frac{d\phi_{ex}}{dt} = \frac{d}{dt}\iiint_V \rho\phi_{in} dV + \iint_A \rho\phi_{in} v_\mu \cdot \hat{n} dA \tag{2}$$

Where ρ is the density, ϕ is a conserved variable, the subscript on ϕ refers to the extensive or intensive property under consideration, the intensive property corresponds to the extensive property divided by the mass. $\hat{n}$ is the unit normal vector pointing out of the cell from the cell face. V and A represent volume of the cell and area of the cell face respectively.

2.1 Conservation of mass

The extensive property ϕ_{ex}, is the mass, m. The intensive property, ϕ_{in}, is the mass flux, $\frac{m}{m} = 1$. The Lagrangian conservation law for mass simply states that the derivative of the mass with respect to time must be zero, substituting into equation 2:

$$\frac{dm}{dt} = \frac{d}{dt}\iiint_V \rho dV + \iint_A \rho v_\mu \cdot \hat{n} dV = 0 \tag{3}$$

This simply describes mass conservation in terms of density and density flux at some point within a defined domain. Crucially, it states that, the divergence of the momentum field must be zero at each step. This makes sense, something cannot appear out of nothing.

2.2 Conservation of momentum

The extensive property ϕ_{ex}, is the momentum, mv_μ. The intensive property, ϕ_{in}, is the momentum flux, $\frac{m}{m} v_\mu = v_\mu$.The Lagrangian law for conservation of momentum is an expression of Newton's second law:

$$\Sigma F_\mu = ma_\mu = m\frac{dv_\mu}{dt} \tag{4a}$$

Substituting this into equation 2 as above:

$$m\frac{dv_\mu}{dt} = \frac{d}{dt}\iiint_V \rho v_\mu dV + \iint_A \rho v_\mu \otimes v_\mu \cdot \hat{n} dA \tag{4b}$$

To produce an equation which can describe the relevant conservation, the necessary mechanics must be described within. The main driving forces for fluid flow are that of surface forces and body forces:

$$m\frac{dv_\mu}{dt} = \iiint_V F_\mu^{Body} dV + \iint_A F_\mu^{Surface} dA \tag{4c}$$

Surface forces can be described using viscous and pressure terms, see e.g: [2, 4]. These can be expressed using the viscous stress tensor, $\tau_{\mu\nu}$, and the unit tensor, $\delta_{\mu\nu}$, with the pressure producing a pressure vector. The latter of which is used to satisfy Gauss' theorem and it's requirements for a inner product between a vector and a unit normal. This will be explored in more detail in the coming sections:

$$\iint_A F_\mu^{Surface} dA = -\iint_A (\tau_{\mu\nu} - p\delta_{\mu\nu}) \cdot \hat{n} dA \tag{4d}$$

Looking for the force to point towards to the surface, a negative is used to invert the vector as the unit normal points out of the cell from the surface. Body forces are generally considered source terms. An example of a possible body force is that of an external field acting on the fluid, such as a magnetic or gravitational field. These can remain as a general F_μ^{Body} term until a scenario arises where the distinction is necessary. comparing equation 4b and equation 4d:

$$\frac{d}{dt}\iiint_V \rho v_\mu dV + \iint_A \rho v_\mu \otimes v_\mu \cdot \hat{n} dA = \iiint_V F_\mu^{Body} dV - \iint_A (\tau_{\mu\nu} - p\delta_{\mu\nu}) \cdot \hat{n} dA \tag{4e}$$

This is momentum conservation in terms of advection forces, diffusive forces, pressure forces, and, external body forces. It is a specific case of the Cauchy tensor, see [4], which can be used to derive the relevant surface forces in a given scenario.

2.3 Conservation of energy

The extensive property ϕ_{ex}, is the energy, $E = mu + \frac{1}{2}mv_\mu^2$. The intensive property, ϕ_{in}, is the energy flux, $\frac{E}{m} = e$. The Lagrangian law for conservation of energy is an expression of the First law of thermodynamics, using this in conjunction with equation 2:

$$\frac{dE}{dt} = \frac{d}{dt}\iiint_V \rho e dV + \iint_A \rho e \cdot \hat{n} dA = \dot{Q} + \dot{W} \tag{5a}$$

Change in heat across the system can be described as the total heat flux coming into the system across the boundaries, where again a minus is chosen to compensate for the unit normal:

$$\dot{Q} = -\iint_A q_\mu \cdot \hat{n} dA \tag{5b}$$

The majority of work done on the system is due to pressure and viscous heating, see: [3, 4]. Although other sources exist, it is assumed these terms are negligible for this case. Where the work done by pressure is the force multiply the distance plus the viscous stress tensor, $\tau_{\mu\nu}$, describing viscous heating:

$$\Sigma W = \tau_{\mu\nu} + F_\mu \cdot d_\mu \tag{5c}$$

The force felt across the surface is the pressure multiplied by the unit normal across the surface. As seen previously, the unit tensor, $\delta_{\mu\nu}$, is used to produce a vector to satisfy Gauss' Theorem:

$$dF_\mu = -p\delta_{\mu\nu} \cdot \hat{n} dA \tag{5d}$$

The distance which this pressure moves into the cell is the velocity of the fluid multiplied by the time step considered:

$$d_\mu = v_\mu \delta t \tag{5e}$$

equations 5d, 5e can be substituted into equation 5c to find a term for the rate of change of the work done, once again considering sign convention and noting that the δt term is reduced when considering a rate of change:

$$\dot{W} = -\iint_A (\tau_{\mu\nu} - p\delta_{\mu\nu}) v_\mu \cdot \hat{n} dA \tag{5f}$$

Conversely, work done due to viscous heating can be described in terms of a stress tensor, $\tau_{\mu\nu}$, much the same way as for the conservation of momentum. Bringing equations 5b, and 5f into equation 5a a conservation law for energy is found:

$$\frac{dE}{dt} = \frac{d}{dt}\iiint_V \rho e dV + \iint_A \rho e v_\mu \cdot \hat{n} dA = -\iint_A (q_\mu - \tau_{\mu\nu} - p\delta_{\mu\nu}) v_\mu \cdot \hat{n} dA \quad (5g)$$

Finally, equations 3, 4e, 5g, show the conservation of mass momentum and energy in terms of fluxes across the face of the cell.

2.4 Gauss' Theorem

Partial Differential Equations (PDEs) can be expressed both in terms of integrals or derivatives. Each have their own use cases, for example, using integral notation is a more natural path when considering control volumes and surfaces. On the other hand, the differential equations are more succinct and in cases easier to interpret. Each conservation equation found so far has a constituent differential version. Each of which can be found using the Divergence/Gauss' Theorem, relating a volume integral and its property vector, Ψ_μ to a surface integral and its property vector and unit normal vector, $\hat{n}$. The differential form of the equation is integrated over the domain, this can be worked back to produce the differential form:

$$\nabla \cdot \Psi_\mu = \iiint_V \nabla \cdot \Psi_\mu dV = \iint_A \Psi_\mu \cdot \hat{n} dA \quad (6)$$

Applying this condition to a given conservation law can produce the analogous differential equation.

2.5 Conservation of mass in differential form

Applying equation 6 to equation 3 produces a differential conservation of mass equation:

$$\iint_A \rho v_\mu \cdot \hat{n} dA = \iiint_V \nabla \cdot \rho v_\mu \implies \frac{d\rho}{dt}\iiint_V (\rho + \nabla \cdot \rho v_\mu) dV \quad (7a)$$

$$\therefore \frac{d\rho}{dt} + \nabla \cdot \rho v_\mu = 0 \quad (7b)$$

This is the continuity equation, as mentioned previously, this condition is used to constrain the mass of the system to a constant value.

2.6 Conservation of momentum in differential form

Following the same steps above, instead using equation 6 and equation 4e:

$$\iint_A \rho v_\mu \otimes v_\mu \cdot \hat{n} dA + \iint_A (\tau_{\mu\nu} + p\delta_{\mu\nu}) dA \tag{8a}$$

$$\implies \iiint_V \nabla \cdot \rho v_\mu \otimes v_\mu dV + \iiint_V \nabla \cdot \tau_{\mu\nu} dV + \iiint_V \nabla \cdot p\delta_{\mu\nu} dV \tag{8b}$$

$$\frac{d}{dt} \iiint_V (\rho v_\mu dV + \nabla \cdot \rho v_\mu \otimes v_\mu) dV - \iiint_V (F_\mu^{Body} + \nabla \cdot \tau_{\mu\nu} + \nabla \cdot p\delta_{\mu\nu}) dV = 0 \tag{8c}$$

$$\therefore \frac{d\rho v_\mu}{dt} + \nabla \cdot \rho v_\mu \otimes v_\mu = F_\mu^{Body} - \nabla \cdot \tau_{\mu\nu} - \nabla \cdot p \tag{8d}$$

Noting that the gradient of a unit tensor, $\nabla \cdot \delta_{\mu\nu}$, reduces to 1.

2.7 Conservation of energy in differential form

As above, using equation 6 and equation 5g:

$$\iint_A \rho e v_\mu \cdot \hat{n} dA + \iint_A (q_\mu + \tau_{\mu\nu} + p\delta_{\mu\nu}) v_\mu \cdot \hat{n} dA \tag{9a}$$

$$\implies \iiint_V \nabla \cdot \rho e v_\mu dV + \iiint_V \nabla \cdot q_\mu dV + \iiint_V \nabla \cdot (\tau_{\mu\nu} + p\delta_{\mu\nu}) dV \tag{9b}$$

$$\frac{d}{dt} \iiint_V (\rho e + \nabla \cdot \rho e v_\mu) dV + \iiint_V (\nabla \cdot q_\mu + \nabla \cdot \tau_{\mu\nu} + \nabla \cdot p) dV = 0 \tag{9c}$$

$$\therefore \frac{d\rho e}{dt} + \nabla \cdot \rho e v_\mu = \nabla \cdot q_\mu - \nabla \cdot \tau_{\mu\nu} - \nabla \cdot p \tag{9d}$$

equations 7b, 8d, 9d, represent the general differential forms of each of the conservation equations introduced. Formally known as the continuity and general compressible Navier-Stokes equations.

2.8 The stress tensor and equation of state

The equations found so far describe the completely general physics of fluid motion for all possible situations, as long as the correct mechanics are considered. To specialise the model, an equation of state and thermodynamic properties must be used to relate the 3 unknown variables, ρ, v_μ, e, to the temperature across the system and hence pressure term. A constitutive relation must be used relating the stresses within the fluid to each of the three

unknown variables. Finally, a constitutive relation must also be used to express the heat flux across the boundary in terms of the 3 unknown variables and the pressure. Each of these are outlined below for the compressible NS equations. Once applied, the equations can easily be manipulated to consider incompressible flows instead. This is explored in more detail in the coming sections. For the equation of state, the ideal gas law in molar form can be used:

$$pV = nRT \rightarrow V = \frac{m}{\rho} \rightarrow n = \frac{m}{M} \tag{10a}$$

$$\therefore p = \frac{\rho RT}{M} \tag{10b}$$

Where, R is the gas constant, T the temperature, and M the mean molecular weight. A viscous stress tensor is required to model the necessary stresses encountered in fluid flow.

For a Newtonian fluid, the stress tensor, $\tau_{\mu\nu}$ must be linear with respect to strain rates. It's divergence must be zeros for a fluid at rest. The system must be act independent of the direction it is working, it must be isotropic. It is commonly accepted, see here: [2, 3, 6], that the only solution to these set of boundaries is:

$$\tau_{\mu\nu} = 2\mu\epsilon_{\mu\nu} + \lambda\nabla \cdot (v_\mu)\delta_{\mu\nu} \tag{11a}$$

Where ϵ is the rate of strain tensor:

$$\epsilon_{\mu\nu} = \frac{1}{2}[\nabla \cdot v_\mu + \nabla \cdot (v_\mu)^T] \tag{11b}$$

$(\nabla \cdot v_\mu)$, is the rate of expansion of flow. μ and λ are proportionality constants associated to the linearity of the strain rates. Named the first and second constants of viscosity respectively. The divergence free nature of $\tau_{\mu\nu}$ leads to the common approximation that $\lambda \approx -\frac{2}{3}\mu$, see here for a detailed derivation [2, 3, 4, 6]. Pulling equations 11a and 11b together produces a good approximation for the viscous stresses involved in compressible fluid flow:

$$\tau_{\mu\nu} = \mu[\nabla \cdot v_\mu + \nabla \cdot (v_\mu)^T - \frac{2}{3}(\nabla \cdot v_\mu)\delta_{\mu\nu}] \tag{11c}$$

Should the viscosity be assumed constant across the field, the equation can be further reduced, see e.g [2, 3, 4]:

$$\implies \mu[\nabla^2 \cdot v_\mu + \frac{1}{3}\nabla \cdot (\nabla \cdot v_\mu)] \tag{11d}$$

Gauss' Theorem, equation 6, can then be applied to convert the divergence operator into a surface integral:

$$\implies \iint_A \mu[\nabla \cdot v_\mu + \frac{1}{3} \cdot (\nabla \cdot v_\mu)] \cdot \hat{n} dA \tag{11e}$$

Finally, the heat flux across the boundaries of the domain are related using Fourier's law of thermal conductivity:

$$q_\mu = -k\nabla \cdot T \tag{12}$$

2.9 The compressible Navier-Stokes equations

Bringing equations 11e and 12 into the previously derived conservation equations, 4e and 5g. The compressible viscous fluid field equations can then be found for momentum and energy respectively:

$$\begin{aligned} &\frac{d}{dt}\iiint_V \rho v_\mu dV + \iint_A \rho v_\mu \otimes v_\mu \cdot \hat{n} dA = \iiint_V F_\mu^{Body} dV \\ &- \iint_A \mu[\nabla \cdot v_\mu + \frac{1}{3} \cdot (\nabla \cdot v_\mu)] \cdot \hat{n} dA - \iint_A p\delta_{\mu\nu} \cdot \hat{n} dA \end{aligned} \tag{13a}$$

$$\begin{aligned} &\frac{d}{dt}\iiint_V \rho e dV + \iint_A \rho e v_\mu \cdot \hat{n} dA = - \iint_A k\nabla T \cdot \hat{n} dA \\ &- \iint_A \mu[\nabla \cdot v_\mu + \frac{1}{3} \cdot (\nabla \cdot v_\mu)] \cdot \hat{n} dA - \iint_A p\delta_{\mu\nu} \cdot \hat{n} dA \end{aligned} \tag{13b}$$

equations 13a, 13b, are the two equations, along with equation 3, which can be used to fully describe the fields associated with the flow of a compressible, viscous, Newtonian, fluid in terms of area and surface integrals.

2.10 Incompressibility

By assuming density remains constant throughout the domain, the set of equations can be reduced into just 2 unknown equations, with the number of unknowns varying by the number of dimensions being considered. Conservation of energy is not needed in this case as the thermal field has no physical impact on the mechanical field as there is no variation in density. There is therefore no need find ρ using the conservation of energy and equation of state. ρ can be divided through to cancel the term out, pressure and stress terms will have an additional $\frac{1}{\rho}$ term, this eventually leads to the dimensionless, incompressible, Navier-Stokes equation. Starting with the continuity equation, realising a static density means a vanishing derivative.

$$\iint_A v_\mu \cdot \hat{n} dA = 0 \tag{14a}$$

$$\implies \nabla \cdot v_\mu = 0 \tag{14b}$$

Crucially, this term appears in the stress tensor used to describe compressible flow. Source and ρ terms have been dropped, assuming they are $0ms^{-1}$ and $1kgm^{-3}$, for simplicity:

$$\frac{d}{dt}\iiint_V v_\mu dV + \iint_A v_\mu \otimes v_\mu \cdot \hat{n} dA = -\iint_A \mu \nabla \cdot v_\mu \cdot \hat{n} dA - \iint_A p\delta_{\mu\nu} \cdot \hat{n} dA \tag{14c}$$

This is a simplified conservation of momentum equation for compressible, viscous flow known as the incompressible equations. These are the assumptions, in two dimensions, which will be explored quantitatively. They are heavily researched and comparison to literature is widely available. The methods needed to manipulate these equations have already been presented, in later sections ways to apply an arbitrary domain over the equations are explored. The methods to reduce integral and derivative terms into linear terms are proposed. Limitations are mentioned and then different possible expansions are qualitatively explored.

3 Domain Discretization

The simplest ways to solve non-linear Partial Differential Equations implement the ideas of domain discretization, allowing the set of general equations to instead be described in the context of a finite volume (or domain). This volume can be split into n number of cells, forming an nxn grid where all cells together can describe each point within the volume. Each method utilises the concept of domain discretization. The ability to take a defined volume (or domain) and reduce it into a system of control volumes (or cells) in which the average value of the system can be stored. Any virtual particle which might find themselves within a specific cell would be subject to the physics being averaged across the same cell.

Fundamentally, a space must be partitioned into an arbitrary number of cells, where information (momentum, etc) is stored locally. These cells can be non uniform and of arbitrary shape; a cell whose centre doesn't align with that of it's neighbours, would require correct interpolation for the vector between the two cell centre points and the cell face. This would need to be applied individually across the system of cells at each iteration. This is described in detail within [1].

For a simplified system; a uniform grid, with identical square cells, is considered. The domain is assigned an arbitrary size which can be split into any number of cells. A system such as this can trivially be described using a matrix.

3.1 The Finite Difference Method

By applying the rules of differentiation directly to derivative terms within a partial differential equation (PDE) one can reduce them into a set of linear equations applied across a domain split into cells. Achieved by realising that a derivative is a comparison between two points which are infinitely close together. Relaxing the constraints, allowing the distance between two points to instead be finite, one can reduce the first order derivatives within a PDE into linear algebra. This can be seen practically for some vector, Ψ_μ, a forward and backwards difference are shown below:

$$\frac{\partial \Psi_\mu}{\partial x} = \frac{\Psi_\mu(x+\delta x) - \Psi_\mu(x)}{\delta x} = \frac{\vec{\Psi}_{i+1,j} - \vec{\Psi}_{i,j}}{\delta x} + \mathcal{O}(\Delta x) \tag{15a}$$

$$\frac{\partial \Psi_\mu}{\partial x} = \frac{\Psi_\mu(x) - \Psi_\mu(x-\delta x)}{\delta x} = \frac{\vec{\Psi}_{i,j} - \vec{\Psi}_{i-1,j}}{\delta x} + \mathcal{O}(\Delta x) \tag{15b}$$

Where the indexing, i, j represent the index within the system and ∂x the finite difference between two points.

Second derivatives can be approximated using a Taylor expansion, some choices must be made about the comparison at hand. For example, when considering a diffusive process a centered difference around a centre point is a suitable method to use:

$$\vec{\Psi}_{i+1,j} = \vec{\Psi}_{i,j} + \delta x \frac{\partial \Psi_\mu}{\partial x} + \frac{\delta x^2}{2!}\frac{\partial^2 \Psi_\mu}{\partial x^2} + \frac{\delta x^3}{3!}\frac{\partial^3 \Psi_\mu}{\partial x^3} + \mathcal{O}(\Delta x^4) \tag{15c}$$

$$\vec{\Psi}_{i-1,j} = \vec{\Psi}_{i,j} - \delta x \frac{\partial \Psi_\mu}{\partial x} = \frac{\delta x^2}{2!}\frac{\partial^2 \Psi_\mu}{\partial x^2} - \frac{\delta x^3}{3!}\frac{\partial^3 \Psi_\mu}{\partial x^3} + \mathcal{O}(\Delta x^4) \tag{15d}$$

Taking the sum of these expansions produces the necessary linear algebra, noting any terms beyond the third order are likely to be negligible:

$$\frac{\partial^2 \Psi_\mu}{\partial x^2} = \frac{\Psi^t_{\mu i+1,j} - 2\Psi^t_{\mu i,j} + \Psi^t_{\mu i-1,j}}{\delta x^2} + \mathcal{O}(\Delta x^2) \tag{15e}$$

Using the Finite Difference Method (FDM) can reduce Partial Differential Equations into a linear algebra problem, an action carried out inherently well

by computers. This methodology has been employed frequently and is well described in sources such as [6, 7, 8]. It is ultimately a simple reduction of a complicated system consisting of partial differential equations.

Although FDM systems are simple to describe, they have their limitations. There is no guarantee of a physically correct solution, due to errors associated with Taylor expansion, finite difference between points, and, physical memory constraints, a lot of resources must be spent to retain a physical solution. The ideas and assumptions being used within the FDM can always be applied to first order derivatives, other methods tend to focus on reducing higher order terms in more robust ways which can then be approximated using the Finite Difference assumptions if needs be.

3.2 The Finite Volume Method

The Finite Volume method builds upon the mathematical expressions of the conservation laws. A volume (domain) is defined, V, which contains all of the information for the system being considered. The volume is split into an arbitrary quantity of control volumes (cells) which each face having- some surface area, A. The ultimate goal is to consider the flux across each face, by applying the conservation equations, one already knows the necessary constraints which would require the system to remain physical.

The method aims to reduce integrals under the key assumption that every cell has a finite number of faces which can be summed across, see e.g: [1], [6, 7, 8], ultimately any surface integrals can be reduced into a sum of algebraic terms which can be used to simplify the set of equations:

$$\iint_A \Psi_\mu \cdot \hat{n} dA = \sum_{Faces} \Psi_\mu \hat{n} A \tag{16}$$

For each iteration, each cell will need to find an intermediate momentum value using the neighbouring cells, one which is not divergence free, and use this guess to produce a linear system of equations, of the form $Ax = b$. This can be solved numerically using a direct or iterative method. This result, along with the previously computed intermediate momentum, can be used to find the true divergence free solution. It is this method which will be explored throughout the presented work and steps to reduce a PDE into linear algebra using this philosophy will be shared in more detail in the coming sections.

4 Application of the Finite Volume Method

Previously touched upon, this method aims to reduce the integrals within the conservation equations. Assuming a surface integral can be replaced by a sum across the faces of the cell. equations 3, 4e, 5g, can be reduced into linear algebra terms. The continuity equation can used to build the necessary components for the linear system of equations, $Ax = b$, to be solved.

4.1 A staggered grid

Caused by the decoupling of pressure and momentum, numerical instability can become a problem if not properly appreciated. This can lead to a system which doesn't behave as expected. Consider a system of 4 cells and whose pressure values form a repeating pattern across the cells, such that the gradient across them would numerically be considered zero. This would cause no net movement in the field, something which is nonphysical if there is a defined boundary placing constant velocity into it. Many sources employ the staggered grid method, seen here [7, 13]. The momentum values are instead referenced on the faces of the cells, their values are therefore dependent on the cells around the cell wall, this allows the checker boarding effect to be resolved.

In two dimensions, the x-momentum is integrated over a set of cells shifted to the right of the pressure cells, to describe momentum across the face. The y-momentum is integrated over a set of cells shifted below the pressure cells, in the same way. This leads to nuance which must be carefully considered when expressing terms across the domain as information is no longer describing the same location across system variables. This also leads to the needs to interpolate variables between the three fields being considered.

4.2 Source and time dependent terms

all properties are considered constant across each cell volume and cell face, it then follows:

$$\iiint_V F_\mu^{Body} dV = F_\mu^{Body} \iiint_V dV = F_\mu^{Body} V \tag{17a}$$

$$\frac{d}{dt} \iiint_V v_\mu dV = \frac{dv_\mu}{dt} \iiint_V dV = \frac{dv_\mu}{dt} V \tag{17b}$$

The derivative left in equation 17b can then be approximated using the

Finite Difference Method in equation 15a:

$$\implies \frac{v_\mu(t+\delta t) - v_\mu(t)}{\delta t} V = \frac{v_\mu^{t+1}{}_{i,j} - v_{\mu i,j}^t}{\delta t} V \tag{17c}$$

4.3 Advective and pressure terms

As stated, the integral across an area can be replaced by the sum across the faces of the cell, it then shows for the advective term:

$$\iint_A v_\mu \otimes v_\mu \cdot \hat{n} dA = \sum_{Faces} v_\mu \otimes v_\mu \cdot \hat{n} A \tag{18a}$$

The pressure term is handled in much the same way:

$$\iint_A p\delta_{\mu\nu} \cdot \hat{n} dA = \sum_{Faces} p\delta_{\mu\nu} \cdot \hat{n} A \tag{18b}$$

4.4 The viscous term

The same again can be done for the viscous term:

$$\iint_A \mu \nabla \cdot v_\mu \cdot \hat{n} dA = \sum_{Faces} \mu \nabla \cdot v_\mu \cdot \hat{n} A = \sum_{Faces} \mu \frac{dv_\mu}{dx_\mu} \cdot v_\mu \cdot \hat{n} A \tag{19a}$$

As above, the remaining derivative can be reduced using FDM, equation 15a. Two dimensions are considered such that, $v_\mu = (v_1, v_2) = (u, v)$ and $x_\mu = (x_1, x_2) = (x, y)$. Should the dimensions in the derivative match, a forward or backwards difference is applied depending on which face is being considered:

$$\frac{du}{dx} = \frac{u_{i+1,j} - u_{i,j}}{\delta x} \rightarrow \frac{du}{dx} = \frac{u_{i,j} - u_{i-1,j}}{\delta x} \tag{19b}$$

A $\pm$ is used to show the two possible comparison points for a cell in each dimension. As this is a tensor terms are needed for each possible combination of forces, this includes terms which account for both dimensions. As a staggered grid is employed, care needs to be taken when referencing cells between each momentum field, for two dimensions this produces four unique equations, starting with the u field:

$$\frac{du}{dy} = \frac{u_{i,j+1} - u_{i,j}}{\delta y} \rightarrow \frac{dv}{dx} = \frac{v_{i,j+1} - v_{i-1,j+1}}{\delta x} \tag{19c}$$

The v field would then be:

$$\frac{du}{dy} = \frac{u_{i+1,j} - u_{i+1,j-1}}{\delta y} \rightarrow \frac{dv}{dx} = \frac{v_{i+1,j} - v_{i,j}}{\delta x} \tag{19d}$$

The two dimensions are staggered in two distinct directions. When referencing each field with respect to the other, the indexing must reflect the necessary transformation between fields. In this case, the v_2 field can be referenced from the v_1 field by considering which direction the domain has been shifted, the j index must be shifted one cell upwards to counter the extra cell at the bottom of the domain in the v_2 field. For the v_1 field to be referenced within the v_2 field, the i index must be moved on to the right, for the reason outlined above.

4.5 The discrete continuity equation

As above, the continuity equation for incompressible flow, equation 14a, also needs to be reduced into a linear algebra problem. The steps follow those laid out already:

$$\iint_A v_\mu \cdot \hat{n} dA = \sum_{Faces} v_\mu \cdot \hat{n} A = 0 \tag{20}$$

4.6 A linear set of equations

This can all be brought together, with respect to the fact only two dimensions are considered. The system of cells is homogeneous, each one can be handled identically. The following variables are therefore known across the proposed domain:

$$V = \Delta x \Delta y \rightarrow A_{\pm i} = \Delta y \rightarrow A_{\pm j} = \Delta x \tag{21a}$$

Where each term, δ, has been replaced with, Δ, its known finite term derived from the dimensions of the domain and cells. Since the proposed case is in two dimensions, each term can be reduced by one order. Such that, the volume becomes the area of the cell while the area becomes the length of the cell face.

Each integral is replaced with a sum across the faces. In two dimensions, each sum is replaced with 4 terms representing the 4 faces of each cell. Following this logic through will then produce a linear equation for

each momentum field, source terms have been considered negligible:

$$\sum_{Faces} p\delta_{\mu\nu} \cdot \hat{n}A = p\delta_{\mu\nu} \cdot \delta_{\mu\nu}\Delta x_\mu + p\delta_{\mu\nu} \cdot -\delta_{\mu\nu}\Delta x_\mu$$

$$\sum_{Faces} v_\mu \otimes v_\mu \cdot \hat{n}A = (v_\mu v_1 \cdot \hat{i} + v_\mu v_1 \cdot -\hat{i})\Delta y + (v_\mu v_2 \cdot \hat{j} + v_\mu v_2 \cdot -\hat{j})\Delta x$$

$$\sum_{Faces} \mu \cdot v_\mu \cdot \hat{n}A = \mu[(\nabla v_\mu \cdot \hat{i} + \nabla v_\mu \cdot -\hat{i})\Delta y + (\nabla v_\mu \cdot \hat{j} + \nabla v_\mu \cdot -\hat{j})\Delta x] \tag{21b}$$

Where the pressure term is dependant on dimension, when considering a specific dimension, the unit tensor devolves into the unit vector for the same dimension. The area is taken with respect to which face is being considered, in the x-dimension it would cross the right/left face, the area of the face would therefore be Δy. The sign of the unit normal vector describes the necessary direction and dimension for comparison.

Placing this into the conservation of momentum equation, 14c, an equation for the derivative of the momentum in purely linear terms can then, finally, be found:

$$\begin{aligned}\frac{{v_\mu^{t+1}}_{i,j} - v^t_{\mu i,j}}{\Delta t}\Delta x\Delta y = [&-p\delta_{\mu\nu} \cdot \delta_{\mu\nu}\Delta x_\mu + p\delta_{\mu\nu} \cdot -\delta_{\mu\nu}\Delta x_\mu] \\ &-[(v_\mu v_1 \cdot \hat{i} + v_\mu v_1 \cdot -\hat{i})\Delta y + (v_\mu v_2 \cdot \hat{j} + v_\mu v_2 \cdot -\hat{j})\Delta x] \\ &-\mu[(\nabla v_\mu \cdot \hat{i} + \nabla v_\mu \cdot -\hat{i})\Delta + (\nabla v_\mu \cdot \hat{j} + \nabla v_\mu \cdot -\hat{j})\Delta x]\end{aligned} \tag{21c}$$

$$\begin{aligned}v^{t+1}_{\mu,i,j} = v^t_{\mu,i,j} + [&-\frac{p\delta_{\mu\nu} \cdot \delta_{\mu\nu}\Delta x_\mu + p\delta_{\mu\nu} \cdot -\delta_{\mu\nu}\Delta x_\mu}{\Delta x\Delta y} \\ &-\frac{(v_\mu v_1 \cdot \hat{i} + v_\mu v_1 \cdot -\hat{i})}{\Delta x} - \frac{(v_\mu v_2 \cdot \hat{j} + v_\mu v_2 \cdot -\hat{j})}{\Delta y} \\ &-\frac{\mu(\nabla v_\mu \cdot \hat{i} + \nabla v_\mu \cdot -\hat{i})}{\Delta x} - \frac{\mu(\nabla v_\mu \cdot \hat{j} + \nabla v_\mu \cdot -\hat{j})}{\Delta y}]\Delta t\end{aligned} \tag{21d}$$

The velocities are expressed on the cell edges of the pressure field, the relevant quantities must be interpolated to the cell faces, assuming the property

is constant across the cell volume:

$$\begin{aligned}
v_\mu v_1 \cdot \hat{i} &= \frac{v_{\mu,i+1,j} + v_{i,j}}{2} \rightarrow v_\mu v_1 \cdot -\hat{i} = \frac{v_{\mu,i,j} + v_{\mu,i-1,j}}{2} \\
v_\mu v_2 \cdot \hat{j} &= \frac{v_{\mu,i,j+1} + v_{\mu,i,j}}{2} \rightarrow v_\mu v_2 \cdot -\hat{j} = \frac{v_{\mu,i,j} + v_{\mu,i,j-1}}{2} \\
v_1 v_2 \cdot \hat{j} &= \frac{v_{i,j+1} + v_{i-1,j+1}}{2} \rightarrow v_1 v_2 \cdot -\hat{j} = \frac{v_{i,j} + v_{i-1,j}}{2} \\
v_1 v_2 \cdot \hat{i} &= \frac{v_{i,j+1} + v_{i-1,j+1}}{2} \rightarrow v_1 v_2 \cdot -\hat{i} = \frac{v_{i,j} + v_{i-1,j}}{2} \\
v_2 v_1 \cdot \hat{j} &= \frac{v_{i+1,j} + v_{i+1,j-1}}{2} \rightarrow v_2 v_1 \cdot -\hat{j} = \frac{v_{i,j} + v_{i,j-1}}{2}
\end{aligned} \tag{21e}$$

Where the subscript μ refers to the dimension and i, j the placement in the domain. Equations 21d is used to find the intermediate velocity field for each cell. The discrete continuity equation, 20, can also be handled in this way. The steps are the same as those above, the equation is integrated over the p field. Velocity values are interpreted at the cell faces, according to the p field. As before care needs to be taken on how values are referenced with respect to a staggered grid.

$$\sum_{Faces} v_\mu \cdot \hat{n} A = (v_\mu \cdot \hat{i} + v_\mu \cdot -\hat{i})\Delta y + (v_\mu \cdot \hat{j} + v_\mu \cdot -\hat{j})\Delta x = 0 \tag{21f}$$

Crucially, each term in equation 21f is a value which is known for each cell. Equation 21d can then be substituted into equation 21f with respect to each directional comparison. The viscous and advective terms have been reduced into one, $\vec{F}$, term for simplicity:

$$\begin{aligned}
v_\mu \cdot \hat{i} = u^{t+1}_{i+1,j} &= u^t_{i+1,j} + \Delta t \vec{F}^t_{i+1,j} - \frac{\Delta t}{\Delta x}(p_{i+1,j} - p_{i,j}) \\
v_\mu \cdot -\hat{i} = u^{t+1}_{i,j} &= u^t_{i,j} + \Delta t \vec{F}^t_{i,j} - \frac{\Delta t}{\Delta x}(p_{i,j} - p_{i-1,j}) \\
v_\mu \cdot \hat{j} = v^{t+1}_{i,j+1} &= v^t_{i,j+1} + \Delta t \vec{F}^t_{i,j+1} - \frac{\Delta t}{\Delta x}(p_{i,j+1} - p_{i,j}) \\
v_\mu \cdot \hat{j} = v^{t+1}_{i,j} &= v^t_{i,j} + \Delta t \vec{F}^t_{i,j} - \frac{\Delta t}{\Delta x}(p_{i,j} - p_{i,j-1})
\end{aligned} \tag{21g}$$

Substituting into equation 21f and collecting the pressure and momentum

terms on respective sides:

$$\begin{aligned} & p_{i,j}(\frac{2\Delta t\Delta y}{\Delta x}+\frac{2\Delta t\Delta x}{\Delta y}) \\ -\frac{\Delta t\Delta y}{\Delta x}(p_{i+1,j}+p_{i-1,j})-\frac{\Delta t\Delta x}{\Delta y}(p_{i,j+1}+p_{i,j-1}) & \\ =[(u^t_{i+1,j}+\Delta t\vec{F}^t_{i+1,j})-(u^t_{i,j}+\Delta t\vec{F}^t_{i,j})]\Delta y & \\ +[(v^t_{i,j+1}+\Delta t\vec{F}^t_{i,j+1})-(v^t_{i,j}+\Delta t\vec{F}^t_{i,j})]\Delta x & \end{aligned} \tag{21h}$$

This is a matrix equation, specifically of the form $Ax = b$. In this case the b vector is the right hand side values. The x vector represents the unknown pressure values. The A coefficient matrix is described by the concatenation of all the possible coefficient which appear in front of a specific cell, depending on which term is being compared. There a 9 unique cell types depending on where they reside within the domain. Each corner and each side has a unique combination of coefficients, while the interior cells can be compared in every direction.

5 Boundary and initial conditions

Some cells within the domain do not have a natural way of comparing neighbouring cells. An edge cell will need to be handled independently, this leads to the 9 possible combinations. Where a cell does not have a neighbour to compare to, some artificial value will instead be taken. This value can be interpreted as a boundary condition. There are many ways in which boundary conditions are implemented, some of the common methods are described below.

5.1 Dirchlet boundary condition

By simply assuming the solution to the out of bounds cell, this value can instead be used in place of the usual comparison value. This is a simple however a powerful way to manipulate the system equations to describe a range of scenarios. This can be expressed mathematically:

$$\Psi_{\mu}^{Boundary} = \phi \tag{22}$$

Where ϕ is a prescribed value. The velocities are interpolated onto the edge cell using this logic. The values are then used in comparison elsewhere and propagate through the system during each iteration.

5.2 Von Neumann boundary condition

A Von Neumann boundary condition, on the other hand, attempts to describe the boundary with the use of the derivative and tangent vector on the boundary being considered. Generally this can be seen as:

$$\frac{\partial \Psi_\mu}{\partial x_\mu} = \phi \tag{23}$$

Where ϕ is once again a prescribed value. This type of boundary condition can be useful in certain applications of domain symmetry, seen in more detail here [7]. When a structured grid with cells of varying size is employed, a boundary condition must be imposed across the change of cell size. This can be done using the Neumann condition in equation 23. This boundary can also be used to impose a pressure condition around the domain of a system.

5.3 No slip conditions

When a virtual particle is attached to a solid boundary, in this case a wall, it will posses no inherent tangential or parallel velocity to the wall. This can be reduced into a slip condition should the parallel velocity be allowed to move past zero. This can be expressed mathematically as such:

$$v_\mu^{Boundary} = 0 \tag{24}$$

This is an application of the Dirchlet boundary condition, equation 22, introduced previously.

5.4 The lid driven cavity

Consisting of an enclosed, isolated box with a no slip condition applied to each edge. Velocity, in the form of a Dirchlet boundary condition, is added parallel to the northern wall. The system is then iterated on until convergence. This is a well documented system and its characteristics are widely researched. This system can then be used to interpret the errors associated with the assumptions and numerical instability imposed using the Finite Volume Method.

6 A system of linear equations

As mentioned, the methods employed require the solution of a linear system of equations to be found. The system is built using the information found

throughout the domain, including any boundary conditions which have been applied. There are several ways in which this can be achieved, but there remain two distinct methods. These will be mentioned in the coming sections.

6.1 Building the system of equations

The system of equations must be built. Following from equation 21h, the coefficients and b vector can be built for each cell across the system. This is a structured grid and so there exists some structure within the coefficients matrix. The A matrix describes the coefficients of the pressure vector, x. A banded diagonal coefficient matrix is found with 5 distinct bands. Where the main diagonal describes the coefficient at the given cell, each respective band otherwise describes the 4 possible neighbours for each cell (a_{i+}, a_{i-}, a_{j+}, a_j). An example for a 3x3 system can be seen below:

$$A = \begin{bmatrix} a & a_{i+} & 0 & a_{j+} & 0 & 0 & 0 & 0 & 0 \\ a_{i-} & a & a_{i+} & 0 & a_{j+} & 0 & 0 & 0 & 0 \\ 0 & a_{i-} & a & 0 & 0 & a_{j+} & 0 & 0 & 0 \\ a_{j-} & 0 & 0 & a & a_{i+} & 0 & a_{j+} & 0 & 0 \\ 0 & a_{j-} & 0 & a_{i-} & a & a_{i+} & 0 & a_{j+} & 0 \\ 0 & 0 & a_{j-} & 0 & a_{i-} & a & 0 & 0 & a_{j+} \\ 0 & 0 & 0 & a_{j-} & 0 & 0 & a & a_{i+} & 0 \\ 0 & 0 & 0 & 0 & a_{j-} & 0 & a_{i-} & a & a_{i+} \\ 0 & 0 & 0 & 0 & 0 & a_{j-} & 0 & a_{i-} & a \end{bmatrix}$$

In other applications of the finite volume method, where the domain is not structured, the resulting coefficient matrix can be much more complicated as each cell might have more or less than 4 neighbours. These grids require more sophisticated handling when trying to find a solution. This leads on to the two methodologies which can be used to find a solution to the system of equations numerically.

6.2 Direct solvers

Direct solvers aim to use the fundamental logic behind linear algebra. Looking for a direct solution immediately by means of matrix factorisation.

6.2.1 LU decomposition

LU decomposition, see e.g [7, 13], breaks the coefficient matrix into two triangular matrices. The two can then be used to find independent solutions

to the problem.

$$Ax_\mu = b_\mu \rightarrow LUx_\mu = b_\mu \tag{25a}$$

By definition this then implies:

$$\implies Ly_\mu = b_\mu \rightarrow Ux_\mu = y_\mu \tag{25b}$$

This method is an efficient method when solving smaller systems of cells. As the coefficients matrix is $n^2 \text{x} n^2$, the overall system of information is larger than that of the actual system size. This is nonetheless an attractive method as it is able to find an exact solution in one iteration over the matrix. Most time is spent computationally, in decomposing the matrix into the upper and lower triangular parts. For the work presented an external library, UMFPACK and its dependencies, are used to solve the linear system using LU decomposition, the documentation can be found here: [9].

6.3 Iterative solvers

On the other hand, iterative solvers aim to converge to a solution. This allows the system to contain less information at any one time, at the cost of taking longer to find a valuable solution.

6.3.1 Conjugate gradient

When considering systems which might be describing entire structures, the resulting grid is expected to contain a large quantity of individual cells. Direct solutions quickly become slow and require large amounts of computational resources, especially when all of the information in the system is stored. By instead considering an iterative solution, such as the conjugate gradient method, less information can be stored while maintaining reasonable convergence at higher cell counts. As described within [10, 13], the conjugate gradient method is a minimizing algorithm, it aims to find the minima for a given function. By following the gradient until the resulting vectors are orthogonal, comparing to the residual and adjusting as necessary, a good solution can be found in a relatively short computation.

7 Application of theory

The methods introduced within can be solved individually by hand and, devoid of human error, should produce a coherent result. This is slow as there are potentially thousands of equations which need to be solved at each iteration. Instead, time is spent constructing the necessary steps within an

environment which can interpret these inputs, this naturally leads to the implementation within a computer program. The basis of most proposed algorithms tend to derive from the SIMPLE algorithm, see: [8], a set of steps which use the decoupling of the momentum equations to interpret a divergent momentum value. Substituting into the continuity equation a linear system of equations is found and solved using previously mentioned methods. The resulting pressure field can be used with equation 21d, in conjunction with the intermediate, divergent, momentum to find a true divergence free solution for any given cell for the next time iteration. The steps are outlined in the list below:

1. Set the relevant boundary conditions, including any sinks or sources.
2. Interpolate velocities to cell walls and find divergent momentum values.
3. Construct system of equations.
4. Solve the system of equations using a given method.
5. Find next iteration values using known momentum and pressure values.
6. Check average kinetic energy for convergence:

   $\Delta KE >$ **Tolerance:** Go to step 2.
   $\Delta KE \leq$ **Tolerance:** System has converged.

7.1 Possible computational approaches

Several computer languages could be suited to executing the resulting code. Python is a strong candidate for its strong scientific community with many open source modules which are readily updated and well maintained. Python is however, an interpreted language, such that each line of code is executed one at a time by a global interpreter. This is a limit which is hard coded into the logic of the Python language. Python also aims to reduce the need for memory management, instead depending on the interpreter to manage memory allocation at runtime. Unfortunately, this means Python can struggle when encountering very large problems and cause convergence to take an unreasonable amount of time. In light of this, a lower level, compiled language, in this case C++, is used. Although a more hands on language, the speed of loops allows for simpler descriptions of the equations when compared to vectorized version (which might be found in Python code). Being a compiled language, memory management is cut out of the runtime and instead handled specifically within the code itself. This makes the code less

accessible, however many of the same core concepts envelop both languages and each resulting code would describe an identical process. Although computation is handled within the C++ environment, CSV files and the use of Matplotlib, a Python library for data visualisation, is used for post processing the resultant converged solution.

7.2 Data handling and memory management

Physical memory is needed to store all of the information about the system within RAM. The coefficients matrix requires an especially large memory footprint, as its size of $n^2 \text{x} n^2$. Luckily, the matrix itself consists mostly of zeros. This means most of the information taken up in system memory will only represent zeros. By instead considering only the non zero inputs, a sparse matrix can instead be constructed. A sparse matrix can be formatted in several ways. Each method is derived from the coordinate list philosophy, this is the method which is introduced. Each valid (non zero) entry within the coefficients matrix is placed into a new vector containing all non zero entries. Two vectors accompany with the matrix i and j values in the same place as the non zero value. Each method beyond attempts to reduce the memory requirement further by referencing a row/column in one vector for a set of individual column/row indices.

7.3 GPGPU computation

A General-Purpose Graphics Processing Unit (GPGPU) can be used in these types of applications. Application Programming Interfaces (APIs) built into C and C++, such as CUDA or OpenCL, can be used to spread workload across multiple graphical processing units. A Cental Processing Unit (CPU) is a highly general unit which excels at nothing highly specific, but can execute almost any task on a general basis. On the other hand, a General Processing Unit (GPU) is designed assuming simpler, homogeneous, computations are needed across a group of pixels. Formally designed for texture manipulation, the striking resemblance to the problems encountered within allow the presented systems to be applied in a parallel manner. This method comes with its own overhead (during shifting of values from global RAM to video RAM), something which a CPU won't encounter. However, as the system grows larger and larger the increased time taken for the GPU cores is smaller as each processing unit will still only be doing slightly more computation, as opposed to the 1 processing unit having to do it all. Unfortunately, the power of GPUs has only just become viable to employ these methodologies on a wider scale; it's implementation is complicated and further reduces

the accessibility of the code, introducing more memory management and the possible complications it brings.

8 Results

The lid driven cavity is used as a benchmark test. This is a widely researched topic and papers with the corresponding data are widely available. Results found using the theories laid out are compared to this data, which can be used to produce a quantitative error value. The Reynolds number is used to distinguish the different systems. A dimensionless number which describes the dominant forces acting on the fluid, a lower number suggest viscous forces dominate, it can be found using the dynamic viscosity:

$$Re = \frac{\rho v' L}{\mu} \implies \mu = \frac{\rho v' L}{Re} \tag{26}$$

Where v' is the velocity added at the boundary. The boundary conditions are those laid out in section 5.4, a constant $1ms^{-1}$ is added parallel to the northern wall. No slip boundaries, i.e., zero flow, is implemented across each wall in each dimension that isn't otherwise providing velocity to the system. A line of cells down the centre of each dimensional velocity is taken and plotted against its position in the domain. The resulting velocity profile is then contrast against the equivalent data found within [5]. A χ^2 best fit function is used on the two curves to produce an error value:

$$\chi^2 = \frac{\sum_i (y_{\text{present}} - y_{\text{observed}})^2}{k+1}. \tag{27}$$

8.1 Lid driven cavity

Figures 1 and 2 show the results found for the lid driven cavity of a system with Re of 100. All systems are considered on a 129x129 grid, as seen in [5]. The tolerance considered for convergence is taken as $1e^{-10}$, this is a reasonable compromise between accuracy and steps needed for convergence, where the machine precision, most commonly $2e^{-16}$, would be the natural error introduced due to numerical computation and the limits imposed due to physical memory usage.

A stream plot and pressure gradient is produced in figure 1 and the velocity profiles are seen in figure 2. Error values are found to be $\chi_u^2 = 0.0126$, $\chi_v^2 = 0.0024$, when compared to those found within [5]. Qualitatively, flow behaves as expected. Positive velocity is added parallel to the north wall, the flow then moves down the right side, causing a high pressure gradient

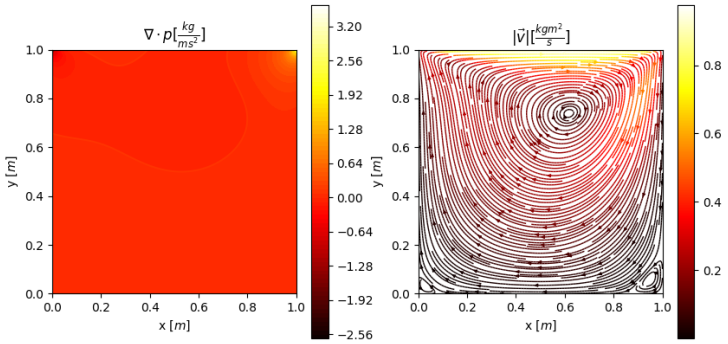


Figure 1: Pressure contour (left), showing variations in the pressure field, and momentum stream plot (right), showing average volumetric flow throughout the domain, graphs for a lid driven cavity at Re = 100, found using method presented throughout. A central vortex is seen to be forming, viscous forces dominate at lower Re numbers and so this eddy dominates the system through stiffer movement, much smaller eddies can be seen forming at the corners of the southern wall.

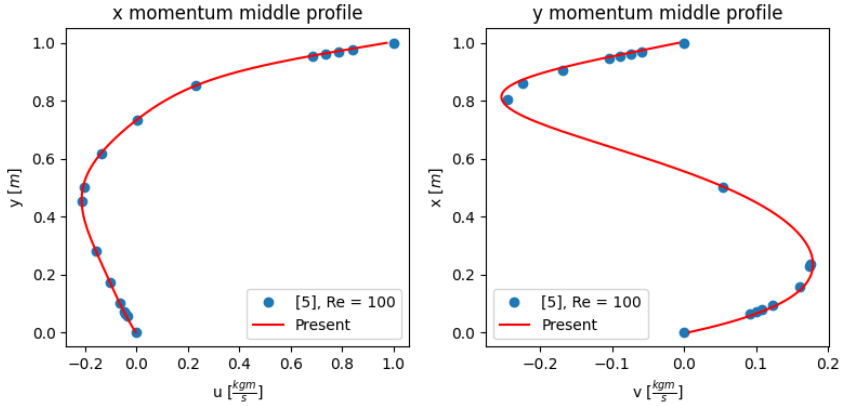


Figure 2: Analysis graphs produced for a lid driven cavity at Re = 100, the cell at the center of the domain is taken for each dimension and their velocities are plotted against their distance from the domain wall, producing a velocity profile. A χ^2 function is used with published values, seen in [5], to produce an error estimate for the system, these are found to be $\chi^2_u = 0.0126$, $\chi^2_v = 0.0024$

in the top right of the domain relative to the rest, with an equivalent low pressure zone at the top left. A distinct eddy in the centre of the domain is seen. At lower Re the flow is expected to be dominated by viscous forces, more mechanical energy is required to overcome these forces and the system tends to remain in a uniform motion around the central eddy. At the southern corners two separate vortexes can be seen forming due to the backwash caused by tangential flow, a result of no slip boundaries at the walls of the domain. The overall pressure in the system is comparatively large when contrasted against following results, due to the larger quantities of mechanical energy required to move the fluid resulting in a larger pressure contribution to balance the conservation. The same conditions are then explored for a system with a Reynold's number at 1000.

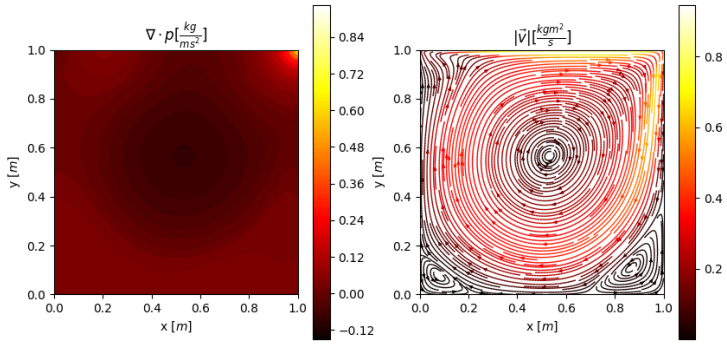


Figure 3: Pressure contour (left) and momentum stream plot (right) graphs for a lid driven cavity at Re = 1000. Viscous forces are becoming overshadowed by the bulk movement of the fluid. Less mechanical energy is required to change the overall state of the system. A more defined central eddy is seen, with similar eddies around the southern wall. Flow is beginning to separate around the left corner of the norther wall, the fluid is becoming less dominated by friction and so more areas of flow separation are expected to be seen forming. As less energy is required to change the system, the overall steps needed for convergence begins to grow.

Figures 3 and 4 show the resultant fields found after convergence of a system at a Re number of 1000. An error estimated to be $\chi_u^2 = 0.0093$ and $\chi_v^2 = 0.0100$ is found in both dimensions. The central eddy becomes more distinct along with corresponding high and low pressure zones as velocity is added at the top boundary. As the overall contribution of the pressure field diminishes, variations become more pronounced. Less mechanical energy is

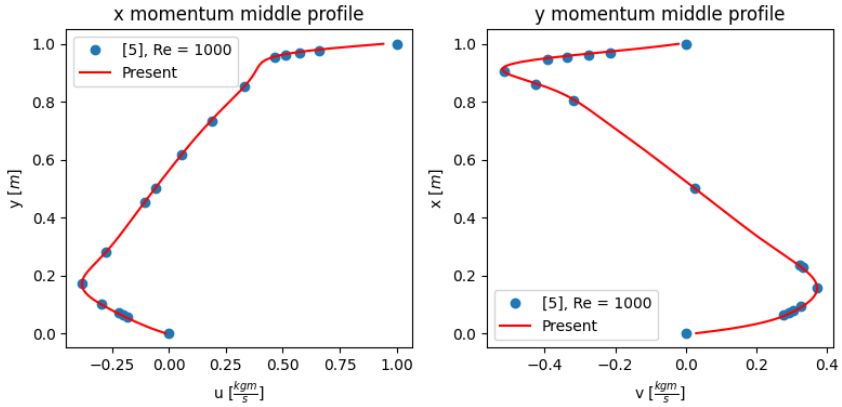


Figure 4: Analysis graphs produced for a lid driven cavity at Re = 1000. The same middle cells are considered as above, an error of $\chi_u^2 = 0.0093$ and $\chi_v^2 = 0.0100$ is found for the u and v momentum dimensions respectively. Variations are seen due to slight differences in boundary condition handling, where [5] uses physical cells to describe the boundaries, boundaries implemented presently are interpolated to the edge cells causing some discrepancy during comparison.

added to the system to incur movement and so the overall contribution by the pressure terms are diminished. Advection is becoming the dominant force, as there exists less friction between each individual particle, possible places within the domain for separation of flow to occur begin to grow. Seen already along the southern wall where two distinct eddies are observed in each corner, the larger one being closer to the injection of momentum across the northern wall to the right. A third eddy is beginning to form around the left corner of the northern wall, this is to be expected as friction between individual particles drops allowing more independent reaction and movement throughout the system. Momentum and pressure values are seen in a key to the right of their respective plots, they suggest a reasonable output whose maximum momentum doesn't exceed that which is introduced along the northern wall. Deviations seen in figures 2 and 4 are from the different methods used to apply boundary conditions, where a physical cell is found within [5]. Instead, the value is interpolated onto a staggered grid describing the flux at the walls of the pressure domain. This leads to variations around the boundaries of each system.

Figures 5 and 6 visualise the resultant fields produced at convergence for a system at a Re number of 5000, as before the dimensions of the domain are a 129x129 grid split between a $1m$x$1m$ domain. A central vortex is still seen,

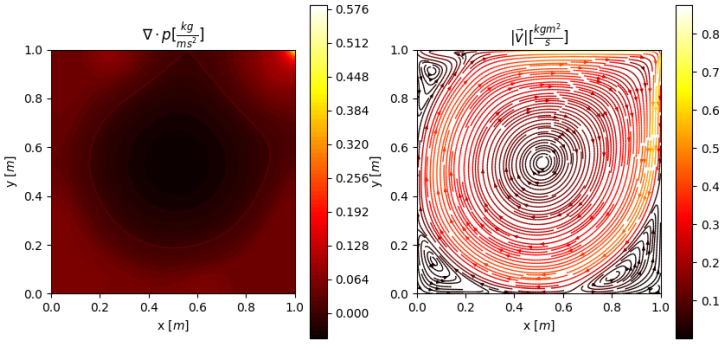


Figure 5: Pressure contour (left) and momentum stream plot (right) graphs for a lid driven cavity at Re = 5000. Following from previous systems, the central eddy becomes ever more defined, being more dominated by bulk movement of flow as the Re number is increased. More distinct variations in overall pressure field are found. Flow has separated around each corner apart from the right corner around the northern wall, the eddies around the southern wall have become fully defined while the third eddy is found the left corner of the northern wall is becoming more defined.

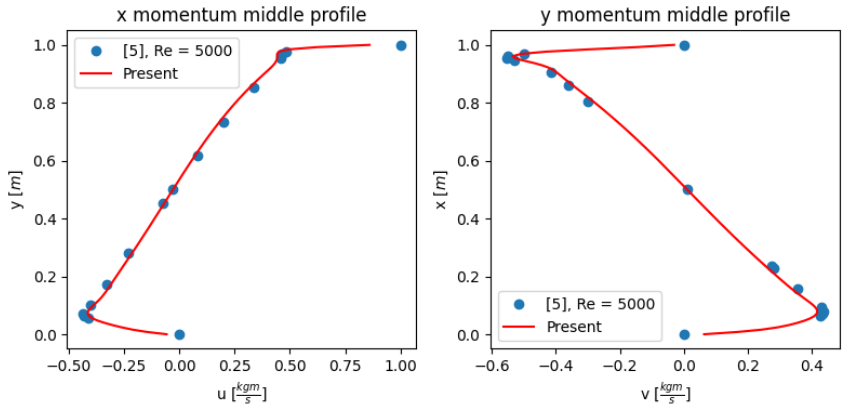


Figure 6: Analysis graphs produced for a lid driven cavity at Re = 5000. As before, the same middle cells are taken from the domain for each momentum dimension, plotted against the distance from the domain boundary. An error estimate is found for each dimension: $\chi_u^2 = 0.0065$ and $\chi_v^2 = 0.0084$.

as before, this central eddy provides the turbulent energy required for the smaller eddies in the domain to maintain their tangential flow. Each distinct eddy is becoming well defined at each corner they are present as the overall movement of flow becomes a more distinct force throughout the system. The pressure contour qualitatively confirms this process, distinct pressure zones are seen with the greatest variations occurring around each observable eddy. An error estimate is found for each momentum dimension as: $\chi_u^2 = 0.0065$ and $\chi_v^2 = 0.0084$.

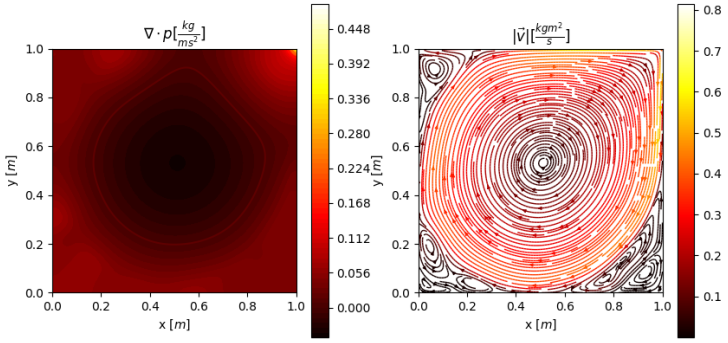


Figure 7: Pressure contour (left) and momentum stream plot (right) graphs for a lid driven cavity at Re = 10000. Advection is very much the driving force for propagation in this system. Lower friction between virtual particles allows separation of flow to occur more frequently. A central eddy still maintains the overall structure of the system, chaos is beginning to creep in around the corners of the domain where smaller eddies are beginning to feed off the now well defined corner eddies.

Figures 7 and 8 show the fields produced at convergence for a system at Re number of 10000. This system is beginning to show signs of turbulence and by extension chaos. A central eddy is still seen, the definition of which remains similar to those observed. Curiously, flow has begun to separate along each of the existing separation points along the southern wall. No longer is there one defined eddy at each corner, instead a selection of eddies can be seen, with a more distinct eddy seen around the left corner of the northern wall. This is a clear example of the mechanisms which dictate the introduction of turbulence, as flow becomes less dominated by the friction between particles the ability for smaller vortices to feed off larger ones becomes ever more possible. This leads to a cascade of turbulence as the fluid reacts more and more to its own movement. Each system exhibits similar, although

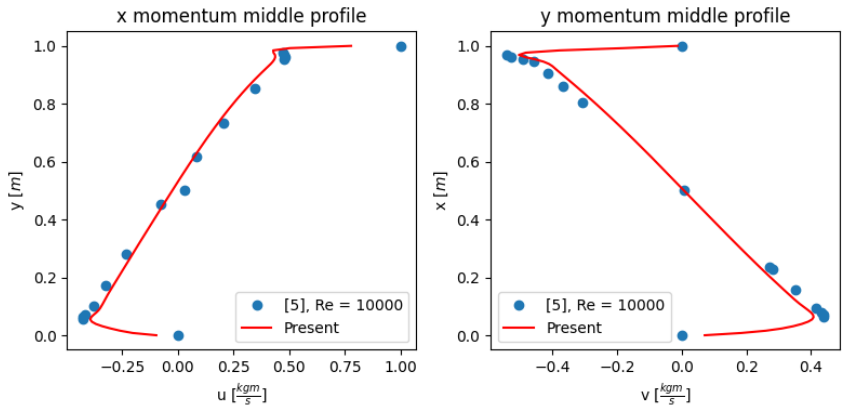


Figure 8: Analysis graphs produced for a lid driven cavity at Re = 10000. A momentum profile is found and an error estimate for each dimension produced: $\chi_u^2 = 0.0086$ and $\chi_v^2 = 0.0091$

distinguishable, characteristics from those seen within [5]. These differences and the limitations of the methods implemented are explored in the next section.

Figure 9 shows the CSV output, for the system at Re = 10000, used to communicate between C++ computation and Python post-processing. It also contains relevant information which needs to be shared to find necessary system variables. Python decodes this format back into its original fields and then matplotlib is used to visualize the data set. Curiously, the system converged in 1929868 steps, something which took around 3.7 days to compute for a total simulated time of around 50 minutes, when the system converged. This is merely a simple manipulation of the governing equations and its resultant systems are considerably less complicated than those which arise in the real world, if the system were to instead diverge; days of possible simulation is lost and the parameters which setup the system must be reconsidered, all for it to begin computation over a large time frame once more. This is of course impractical in a real world sense, where an extra dimension requires similar treatment of the third axis of movement theoretically requiring at least 33.3% more computational resources, if the contributions of energy conservation are also considered this very quickly becomes unmanageable; Moore's law can only take us so far. This is a problem which has been considered before and the possible expansions to the theories proposed are mentioned in later sections.

All of the systems tested have incorporated a Dirichlet boundary condition to apply velocity to the northern wall of the domain with no slip bound-

```
ITERATION: 1929868
BOX DIMENSIONS , 1 , 1
MATRIX DIMENSIONS , 129 , 129
RE , 10000
| X | Y | U | V | P |
0 , 0 , 0 , 0 , 0.0564878
0.0078125 , 0 , -0.000305502 , 0 , 0.0564889
0.015625 , 0 , -0.000735483 , 0 , 0.056502
0.0234375 , 0 , -0.00112215 , 0 , 0.0565259
0.03125 , 0 , -0.00145119 , 0 , 0.0565592
```

Figure 9: An example of the CSV output used to describe the system at convergence, python reads the same .txt file and decodes the CSV format placing relevant values back into their respective fields. This particular example shows the beginnings of the system seen in figure 7, the system took 1929868 steps to reach convergence. This highlights the problems of Direct Numerical Simulation and requires some thought on the possible methods which can be used to reduce this problem into a less computationally expensive problem.

aries along each wall. An example of a Von Neumann boundary condition can be seen below, by rearranging the momentum equation 8d to describe the pressure gradient and multiplying by the unit normal, by then imposing a no slip condition at the boundary, a boundary condition for the pressure field can also be applied, this process is well described in [7]:

$$\hat{n}\nabla p = \rho\hat{n}(-\frac{\partial u}{\partial t} - v_\mu \nabla v_\mu + \mu\nabla^2 v_\mu) \implies \frac{\partial p}{\partial x_\mu} = \rho\mu\frac{\partial^2 v_\mu}{\partial x_\mu^2} \tag{28}$$

This can be approximated using a Taylor expansion, seen in equation 15e. By applying this at certain cells, an artificial pressure boundary can be imposed. Figures 10 and 11 shows the possible uses of these types of boundary conditions, where the shape of an object can be describe throughout a Cartesian domain, the system can be set up to handle and interpret the flow around such an object, similar approaches can be used on a non structured grid to better describe the shape of an object. For the work presented a circle and square are qualitatively explored, noting the difficulty in quantitatively verifying a system such as this. Flow can be seen moving around the object, where the central eddy is no longer able to converge in the middle of the system, the eddy will instead form near the right corner of the northern wall. This is of course a simple manipulation but the theory can be applied in

many useful ways.

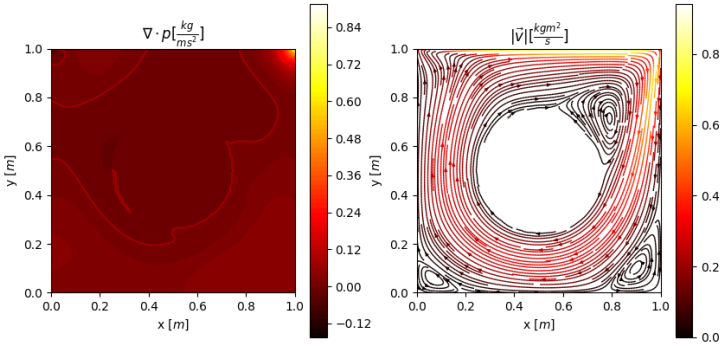


Figure 10: Showing the system produced at convergence for a system at Re = 1000 with a Von Neumann pressure boundary, seen in equation 28. The necessary cells are found using the Cartesian description of a circle. The overall dynamics of the system remain however, the placement of the object disrupts the formation of the central vortex.

8.2 Possible sources of errors and limitations

Numerical errors can become exemplified quickly throughout approximation of differential equation. Each iteration will add its own deviation from the exact solution, the hope is to reduce this at each step. There is a natural limit to precision when describing numbers within the memory of a computer, known as machine precision, this can be used as a baseline error value. The largest source of error comes in the overall time step used, presently a first order method is used to approximate spatial and time derivatives, more accurate methods can be used which aim to use more points along the function curve to determine the gradient of a variable. Otherwise, minutia in the methods employed to describe boundary conditions between those presented and those seen within [5] leave noticeable variation in the resultant fields, mostly around the boundaries. Where [5] use a determined boundary defined within the system, presently a ghost cell is is interpolated onto the outer edge of the pressure field. This change is not significant globally but sometimes produced a noticeable local variation in the resultant velocity profiles. In terms of physical deviation from the solution, a staged Reynold's number is not a physically accurate phenomena. In reality the Re number of flow will change as the characteristic flow and shape of the domain change.

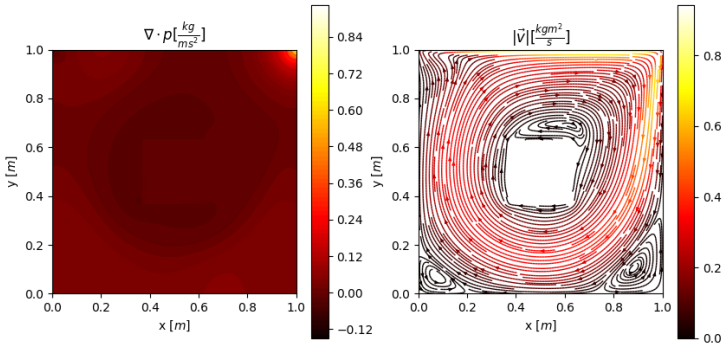


Figure 11: Showing the system produced at convergence for a system at Re = 1000 with a Von Neumann pressure boundary, seen in equation 28. The necessary cells are found using the Cartesian description of a square. A similar vortex is seen forming along the top of the object, as mass comes around the edge a low pressure zone is formed resulting in separation of flow along the surface of the object.

In the work presented the Re number is used to find the dynamic viscosity at each step. Another possible application would be the opposite, using a constant dynamic viscosity to extrapolate a Re number across the system at each iteration. This was avoided as the values compared to literature also pin the system Re number, recognising its uses in succinctly describing different systems.

9 Applicability and expansion

The equations laid out within can describe more than just the flow of a conventional fluid. This specific physics is encoded within the stress tensor and can be manipulated to describe different situations where conservation laws apply.

9.1 Turbulence

Turbulence occurs when flow lines separate and begin to interact tangentially to the mean flow. Turbulence can quickly lead to chaos which is inherently difficult to model accurately, this problem leads to the non linear nature of the governing equations. Direct Numerical Simulation (DNS), such as the

methods introduced within, can still simulate the physics on all scales. However, the information required to accurately model turbulence on all scales quickly makes this unfeasible beyond small scale simulations. As seen in figure 9 systems which are dominated by turbulent processes require large computational resources to resolve, this very quickly becomes impractical in a real world application where extra dimensions and possibly extra contributions require further resources to find. Methods are then needed to simplify the computation and instead model the effects of turbulence to provide a broad picture of a given system, an in depth breakdown can be found in e.g: [6], [7].

9.1.1 Reynold's averaged Navier-Stokes

Introduced by Osborne Reynolds, [11], this method aims to reduce a variable into a time averaged and fluctuating component, producing a mean equation for each conservation law being considered. When stresses are also modeled and averaged a turbulence equation is found. The flow is assumed to be incompressible, the differential form of the equations are considered, seen in equation 8d, this equation can then be time averaged, the full derivation can be found in Appendix E:

$$(\nabla \cdot \overline{v_\mu})\overline{v_\mu} = -\nabla \overline{P} - (\nabla \cdot v'_\mu)v'_\mu + \mu \nabla^2 \overline{v_\mu} \tag{29}$$

Crucially, should the system be in a state where no turbulence processes are occurring, the equations reduce to the normal Navier-Stokes equations. A new term is introduced during the decomposition. Conventionally these are known as Reynold stresses as they posses units of Nm^{-3}. Consequentially, there are not enough equations to describe all the inherent variables. A model for the behaviour of these stresses must be determined to produce a valid result. Recently, machine learning has been employed to find these relationships allowing much more accurate modelling, see [12]. The turbulent kinetic energy is used, alongside a turbulence model such as the k-ϵ model to find a close the system of equations. The turbulent kinetic energy can be seen below, with a detailed derivation found in e.g: [6], [7]:

$$\begin{aligned} \rho\frac{\partial k}{\partial t} + \rho\bar{v_\nu}\frac{\partial k}{\partial x_\nu} = -\rho v'_\nu v'_\mu \frac{\partial \bar{V}_\mu}{\partial x_\nu} - \rho\frac{1}{2}\frac{\partial (v'_\mu \bar{v'_\nu} v'_\nu)}{\partial x_\nu} \\ -\frac{\partial (\bar{v'_\mu} p')}{\partial x_\mu} - \mu\frac{\partial u'_\mu}{\partial x_\nu}\bar{\frac{\partial u'_\mu}{\partial x_\nu}} + \frac{\partial}{\partial x_\nu}(\mu\frac{\partial k}{\partial x_\nu}) \end{aligned} \tag{30}$$

$-\rho v'_\nu v'_\mu \frac{\partial \bar{V}_\mu}{\partial x_\nu}$ is then modelled using various different theories. Some of the more common examples can be found here [7, 6, 13]. All of these equations

together produce a closed system of equations for the RANS equation 29.

Ultimately, the method aims to study the local effects of turbulence on the average flow seen across the domain. This method remains a viable solution when one realises that across an ensemble of experiments, the solution should converge to the Reynolds averaged solution. Although this doesn't provide intricate detail across the system, a large system where direct simulation of turbulence at each point is no longer attractive, this solution is a cheap method to draw conclusions on the average effects on a larger system.

9.1.2 Large eddy simulation

Large Eddy Simulations begin to explore the modelling of turbulence on scales smaller than those inherently described within the domain. A minimum of 4 cells are required to described a vortex:

$\rightarrow$	$\downarrow$
$\uparrow$	$\leftarrow$

Table 1: Minimum scale required to describe turbulent processes

Large scale turbulence can be resolved using RANS simulation, propagation across smaller scales can then be modelled. By realising that turbulent flow tends to follow predictable characteristics, dependant on large scale turbulence, smaller scales can be interpreted instead of simulated. The detailed derivations can be found here: [6, 7]. Providing increased levels of detail for turbulent terms can provide a more robust stand alone solution. However, more detail is needed in the domain using possibly more complicated geometry, a precursor RANS simulation is usually employed to resolve the necessary scales within the mesh, dependant on the turbulent kinetic energy within the region. This can very quickly snowball into a slow and expensive process when considering all simulations needed to find a coherent answer. However, the detail which is possible using this method is desirable.

9.2 Magneto-hydrodynamics

By combining Maxwell's equations, with the general conservation equations, 3, 4e, 5g, a general solution for the flow of a plasma can be found. Once stresses are accounted for and following through with the necessary discretization, then provides a system of linear equations which can describe the motions of a plasma in the context of an electric or magnetic field. These equations are likely to be used in situations dealing with plasma physics, such

as those related to the movement of fluid within a star. This type of physics is generally hard to closely examine, the most practical path for research is therefore effective use of the governing equations for simulation. More detail can be found in e.g.,: [16, 17].

9.3 Relativistic fluid flows

When fluid flows approach the speed of light or a very large magnetic field is applied, the classic assumptions pioneered by Newton begin to fall apart. Past this limit, extra physics is required to describe the unique processes which occur. These types of equations are useful when exploring fluids which can reach these types of speeds. Such as gas orbiting close to black holes or under the influence of extreme magnetic fields. Research in these areas is fascinating and readers are encouraged to find more in e.g [18, 19], and, [20]

9.4 Astronomy and Cosmology

When modelling a universe the mechanics are assumed to be that of a perfect fluid, with the correct considerations about the necessary parts which build the model, simulation of astronomical occurrences is possible. This would involve manipulation of the stress tensor to instead describe the mass-density properties of a given universe, a relevant equation of state would be necessary dependant on the composition of said universe. Simulations of possible processes can provide a more complete picture of the overall processes occurring. This can be used to aid in the correct exploration for necessary observations.

Astronomy inherently attempts to describe the physics in all things found throughout the universe. Many of these physical processes can be likened to the flow of a fluid under some external forces. The combination of Magneto-hydrodynamics and relativistic fluid flow can be used to explore, for example, the behaviour of quasars, a phenomenon which is likely a rare occurrence in the modern universe. Although many have since been found, the only visible clues for their existence lie in images taken from million of light years away. Should the equations derived to explain the physics suitably reflect those which are observed. When neutron stars collide, the rare probability means direct observation is difficult, although systems such as LIGO aim to change this, requires the modelling of both Einstein and Maxwell's contributions in the conservation equations, see e.g.: [18, 19, 21]. These equations can be manipulated in many ways, possibly including turbulence models of their own, see in [22].

10 Improvements and considerations

Some improvements to the methods used throughout to approximate terms could be implemented to allow the numerical solution to better represent the actual solution. The ease of writing and manipulating the code needed for a complete simulation is explored and systems to provide a more flexible code base which could possibly be used to more easily manipulate the base equations is mentioned.

10.1 Improvements to numerical stability

First order methods are employed within the proposed methodology, this is done to reduce the number of terms considered and keep the code manageable. An example of a higher order forward difference is given below, where the full derivation can be found here, [7]:

$$\frac{\partial \Psi_\mu}{\partial x} = \frac{-3\Psi_\mu(x) + 4\Psi_\mu(x+\delta x) - \Psi_\mu(x+2\delta x)}{2\partial x} + \mathcal{O}(\Delta x^4) \qquad (31)$$

This type of approximation can be used around boundaries of the domain where information may not be available in every direction. Similar steps can be taken during interpolation of velocities to the cell walls. First order methods are ultimately the least accurate possible implementation, although still producing a viable result, it requires careful considerations of individual cell sizes and the time step which must be taken at each iteration. Higher order methods generally tend to use more points along the function curve to approximate necessary variables, most clearly seen in the implementation of the finite difference method, seen in equations 15a and 15c, but also applicable to interpolation methods employed to interpret values at cell walls, seen in equation 21e. Each of these methods can be expanded to include more points along the curve, this provides a better overall approximation at the expense of a more complicated mathematical setup. The exploration of these methods and the effects they pose on the overall stability of a given system is of significance, finding a medium between accuracy gained and complexity during implementation is important to remain productive during experimentation.

10.2 Accessibility of implementation

A compiled language, in C++, is used in the proposed work for its wide use and freely available development environments. Although not necessarily important, the increased computational speed and simpler mathematical syntax (in the form of loops as opposed to vectorized code) was preferred

over interpreted languages such as Python. These benefits allow a simpler implementation of the mathematical aspects of the code, this allowed a much easier debugging environment as the code tended to follow the same syntax as the maths which has been derived, allowing for easy distinction between mathematical and code related errors (examples of important parts of the code structure are outlined within the appendix, where clear comparisons between the maths derived and that implemented in code can be seen). However, C++ is designed for computer science applications, built as an Object Orientated implementation of the C language, much more of the code is focused on the managing of memory and secure implementation of information, something which is only introduced due to the low level access C++ gives to the developer. More time must be spent on setting up the correct environment and ensuring all pieces are in place to allow the program to build into an executable. Linking extra modules can require compiling external code, something which can inherently cause problems. All of these add up to increase time spent on development which can hinder the progress of an experiment, this can be alleviated by repeated use of a language to learn its nuances, eventually the positives of using a lower level language will begin to reveal itself in more complicated situations. Work should be done on code which allows simple implementation of different, more complicated initial conditions, along with higher order methods to allow for smoother convergence. Qualitatively, an output at each step could be implemented for the visual confirmation of a given system, something which could be done comparatively easily as an extension to GPGPU computation. An extension of the code to accommodate for several different types of physical scenarios would allow for quick experimentation of a seemingly infinite number of different scenarios. Although the time taken to produce and corroborate any work like this would be considerable, the time gained across sustained use would eventually pay off. Even after these suggestions there are mechanisms which professional codes consider a base, such as a non structured adaptive grid, choice between interpolation methods and system of equation solvers along with a host of implemented models and expansions to conservation equations. There are examples of commercial software which implement these methods, see StarCCM+ and Ansys Fluent. These large scale applications run on old logic with ageing documentation, something which raises the entry requirements for those wishing to use them.

11 Conclusion

Lagrangian conservation laws are converted into their Eulerian counterpart using Reyonold's Transport Theorem, see equation 2, the Navier-Stokes equations are derived to describe compressible fluid flow, seen in equation 4e. The methods to reduce integrals and derivatives into linear algebra terms are explored and applied to the necessary components. A simulation is built on top of the proposed theory, an overall error value is found for both dimensions considered; $\chi_u^2 = 0.0085$, and, $\chi_v^2 = 0.0088$ respectively. The definition of a best fit function leads to the conclusion that the resultant velocity profiles have a 0.85% and 0.88% averaged probability of being a random distribution in the u and v dimensions respectively. These values suggest a successful application of the theory presented. Improvements are suggested for limitations identified above, in the form of higher order finite different approximations and interpolation schemes as Well as improvements in the mesh quality and composition. The concepts behind turbulence modelling are introduced, the RANS equations 29 are mentioned and ways to approximate Reynold stress models are mentioned. Expansions to more extreme physics are examined, different possible use cases are then proposed for possibly useful research areas, such as cosmology and astronomy. Ultimately, a reasonable error value has been found using the theories mentioned within. Work in researching exotic events, such as neutron star or black hole mergers, should be encouraged due to the rarity of being able to physically observe such an event. Although computational methods could not, and really should not, replace observational results, their use can help in the fundamental understanding of a seemingly endless number of scenarios, should the right physics be expressed. Any further work beyond those mentioned, should focus on the implementation of a specific system mesh. A unstructured and polygonal mesh could help spread computational resources more efficiently across a given domain. The necessary information for this mesh can be resolved using a RANS simulation. The implementation of a turbulence model would provide a computationally cheaper method to resolve smaller scales providing a balance between accuracy and computation needed to converge. Machine learning can be used to model the viscous contributions needed to close the RANS equations, this can be expanded to include models for the possible applications in General Relativistic fluid flows or Magneto-Hydrodynamics, or both. Higher order interpolation schemes have been proposed to produce a more accurate result at the expense of a more complicated system of equations. Methods to better share computational work load across hardware are mentioned in the form of GPGPU computation which can easily be extended to produce output at each iteration by interpreting the fields as textures,

something which a GPU is designed to manipulate. The systems described are ultimately a simple manipulation of the governing equations, one which only scratches the surface of the many applications of this type of mechanics, future work should focus on a more general and accessible application of theory, allowing a simple manipulation and interpretation of a resultant system. Ultimately, work should be done on these types of computational implementations to provide a more accessible and productive way to produce coherent output for a variety of ranges, one which can be easily manipulated in its simplicity while also providing detailed analysis and output for those who might require it.

References

[1] Jasak H. Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows. Department of Mechanical Engineering Imperial College of Science, Technology and Medicine; 1996

[2] Batchelor G. An Introduction to Fluid Dynamics. Cambridge University Press; 1967.

[3] Stokes G. On the theories of the internal friction of fluids in motion, and of the equilibrium and motion of elastic solids. [S.l.]: [s.n.]; 1845.

[4] Serrin J. Mathematical Principles of Classical Fluid Mechanics. Berlin: Springer; 1959.

[5] Ghia U, Ghia K, Shin C. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. Journal of Computational Physics. 1982.

[6] Blazek J. Computational fluid dynamics. Amsterdam: Elsevier; 2006.

[7] Zikanov O. Essential computational fluid dynamics. John Wiley & Sons, Inc. 2010

[8] Patankar S. Numerical Heat Transfer and Fluid Flow. 1st ed. 1980.

[9] Davis T. UMFPACK User Guide. 2019;.

[10] Saad Y. Iterative methods for sparse linear systems, second edition. Philadelphia: Society for Industrial and Applied Mathematics; 2003.

[11] Reynolds O. IV. On the dynamical theory of incompressible viscous fluids and the determination of the criterion. Philosophical Transactions of the Royal Society of London (A). 1895.

[12] Kaandorp M, Dwight R. Data-driven modelling of the Reynolds stress tensor using random forests with invariance. Computers Fluids. 2020.

[13] Ferziger J, Peric M. Computational methods for fluid dynamics. Berlin: Springer; 2002.

[14] Kim D. Fluid Engine Development. Boca Raton: Taylor Francis, a CRC Press, Taylor Francis Group; 2017.

[15] Pitt-Francis J, Whiteley J. Guide to scientific computing in C++. London: Springer; 2012.

[16] Freret L, Ivan L, De Sterck H, Groth C. High-Order Finite-Volume Method with Block-Based AMR for Magnetohydrodynamics Flows. Journal of Scientific Computing. 2018;79(1):176-208.

[17] Fambri F. A novel structure preserving semi-implicit finite volume method for viscous and resistive magnetohydrodynamics. International Journal for Numerical Methods in Fluids. 2021;93(12):3447-3489.

[18] Ceylan T, LeFloch P, Okutmustur B. A Finite Volume Method for the Relativistic Burgers Equation on a FLRW Background Spacetime. Communications in Computational Physics. 2018;23(2).

[19] Le Floch P, Makhlof H. A Geometry-Preserving Finite Volume Method for Compressible Fluids on Schwarzschild Spacetime. Communications in Computational Physics. 2014;15(3):827-852.

[20] Frauendiener J. Miguel Alcubierre: Introduction to 3 + 1 numerical relativity. General Relativity and Gravitation. 2011;43(10):2931-2933.

[21] Darbha S, Kasen D, Foucart F, Price D. Electromagnetic Signatures from the Tidal Tail of a Black Hole—Neutron Star Merger. The Astrophysical Journal. 2021;915(1):69.

[22] Radice D. Binary Neutron Star Merger Simulations with a Calibrated Turbulence Model. Symmetry. 2020;12(8):1249.

[23] GitHub - WeGoingProbyn/Dissertation: QMUL Physics BSc Dissertation [Internet]. GitHub. 2022 [cited 7 April 2022]. Available from: https://github.com/WeGoingProbyn/Dissertation

A Code excerpts

The general ideas behind the methods employed have been derived and built using sources outlined above, specifically related to the mathematical logic behind the derivations the reader is pointed to sources such as: [1, 6, 7, 8, 13]. For more C/C++ specific implementation of some numerical methods employed sources such as [14, 15] are encouraged. A GitHub repository is used to keep track of the project, it can be found using [23]. For clarity some excerpts are provided to contrast the mathematical derivations provided within. The relevant figure references the equivalent equation.

Beyond the mathematical implementation, a class hierarchy is used to attempt a more modular design. The base class defines all the necessary parameters such as the the size of the domain and number of cells. System variables such as the Reynolds number are set, velocity at a specific wall can be added parallel to the same wall using functions defined within the base class. A lot of thought is spent on how to reference one dimensional arrays in two dimensions, something which comes about due to the complexities involved in C++, it is necessary to provide functions which can move between the two referencing schemes consistently. A physics class is derived from the base class, where most of the excerpts given below originate. This class handles the mathematical logic behind the system, using functions to access the system variables defined within the base class. The driver class is then derived from this physics class, where the logic behind the propagation of the system is implemented, this ultimately allows the simulation to be set up and run in only a few lines of code.

A.1 Velocity interpolation

When contrast to equation 21e a clear similarity can be seen, between figure 12, in the mathematical notation and logical implementation. Row-major ordering of arrays in C++ poses some challenges during development where careful consideration of memory referencing must be employed to produce a viable result. Row-major and Column-major ordering can be inverted by transposing a given matrix. A container class with functions that handle the complications of referencing 1D arrays in 2D allow for easier referencing of values within a field, similar functions are used to place the values in the correct place in a given array. Simple structures are used to bound groups of information together into a single variable, this allows single parts of the physics which build the system to be calculated for all neighbouring cells at the same time.

```
vec6 Physics::InterpolateVelocities(int i, int j, const char* dim) {
    // Find the correct dimension
    if (dim == "x") {
        // These values remain in the domain, no artificial boundary is required
        double UEAST = 0.5 * (GetMatrixValue(i + 1, j).u + GetMatrixValue(i, j).u);
        double UWEST = 0.5 * (GetMatrixValue(i, j).u + GetMatrixValue(i - 1, j).u);
        double VNORTH = 0.5 * (GetMatrixValue(i - 1, j + 1).v + GetMatrixValue(i, j + 1).v);
        double VSOUTH = 0.5 * (GetMatrixValue(i - 1, j).v + GetMatrixValue(i, j).v);
        if (j == 0) {
            double UNORTH = 0.5 * (GetMatrixValue(i, j + 1).u + GetMatrixValue(i, j).u);
            // At the bottom of the domain there is no lower point, an artificial boundary is needed
            double USOUTH = GetVelocityBoundary().E;
            // Return all the information within a 6 component vector structure
            return vec6(UEAST, UWEST, UNORTH, USOUTH, VNORTH, VSOUTH);
        }
        if (j == GetSPLITS().y - 1) {
            double USOUTH = 0.5 * (GetMatrixValue(i, j).u + GetMatrixValue(i, j - 1).u);
            // At the top of the domain there is no higher point, an artificial boundary is needed
            double UNORTH = GetVelocityBoundary().W;
            // Return all the information within a 6 component vector structure
            return vec6(UEAST, UWEST, UNORTH, USOUTH, VNORTH, VSOUTH);
        }
        else {
            // Otherwise there is no bordering boundary and no artificial step is needed
            double UNORTH = 0.5 * (GetMatrixValue(i, j + 1).u + GetMatrixValue(i, j).u);
            double USOUTH = 0.5 * (GetMatrixValue(i, j).u + GetMatrixValue(i, j - 1).u);
            // Return all the information within a 6 component vector structure
            return vec6(UEAST, UWEST, UNORTH, USOUTH, VNORTH, VSOUTH);
        }
    }
```

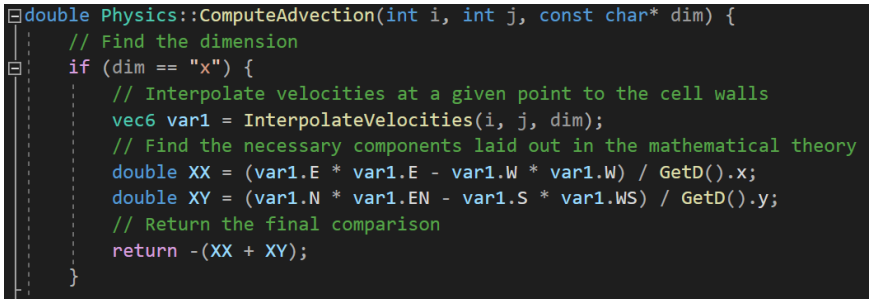
Figure 12: A section of code implemented to interpolate x dimension momentum values to the cell walls of the domain. These equations are the logical implementation of equation 21e.

A.2 Advective forces

Figure 13 is seen in the advective terms in equation 21b, velocities are interpolated using functions seen in figure 12. Each terms is then found and returned as a variable which can be used in further computation. The coding philosophy employed attempted to reduce the complicated equations into more manageable linear steps. This helped throughout debugging, providing a more precise point of error. The same logic is applied to the other, v, dimension.

A.3 Diffusive forces

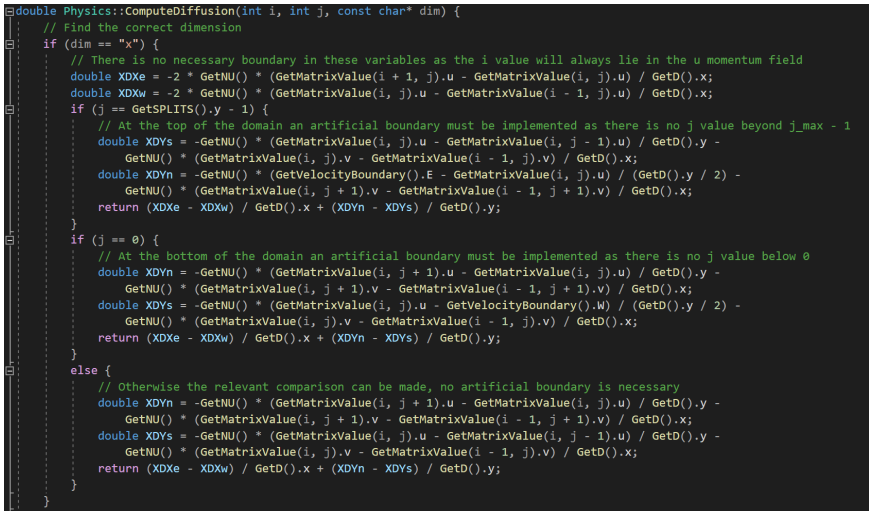
Figure 14 is seen as diffusive terms in equation 21b. The mathematical build up is more complicated than the previously introduced advective terms, the same philosophy is applied, the equations have been split into their constituent parts and the necessary value is returned upon successful execution. This allows for easy identification of problems during code execution and debugging. Boundary conditions must be applied at the correct points throughout the domain, this is reflected in the use of if statements. When a cell meets the boundary condition, an artificial velocity provided by the user is applied.



```
double Physics::ComputeAdvection(int i, int j, const char* dim) {
    // Find the dimension
    if (dim == "x") {
        // Interpolate velocities at a given point to the cell walls
        vec6 var1 = InterpolateVelocities(i, j, dim);
        // Find the necessary components laid out in the mathematical theory
        double XX = (var1.E * var1.E - var1.W * var1.W) / GetD().x;
        double XY = (var1.N * var1.EN - var1.S * var1.WS) / GetD().y;
        // Return the final comparison
        return -(XX + XY);
    }
```

Figure 13: A section of code implemented to compute advective forces for x dimension momentum values at specific points in the domain. These equations are the logical implementation of the advective terms seen in equation 21b.

As the domain is extended in the x direction, the comparisons across this dimension are describable at every point throughout the domain. Problems occur when a y dimensional comparison is made, artificial information must be added to the system to allow it to remain closed.



```
double Physics::ComputeDiffusion(int i, int j, const char* dim) {
    // Find the correct dimension
    if (dim == "x") {
        // There is no necessary boundary in these variables as the i value will always lie in the u momentum field
        double XDXe = -2 * GetNU() * (GetMatrixValue(i + 1, j).u - GetMatrixValue(i, j).u) / GetD().x;
        double XDXw = -2 * GetNU() * (GetMatrixValue(i, j).u - GetMatrixValue(i - 1, j).u) / GetD().x;
        if (j == GetSPLITS().y - 1) {
            // At the top of the domain an artificial boundary must be implemented as there is no j value beyond j_max - 1
            double XDYs = -GetNU() * (GetMatrixValue(i, j).u - GetMatrixValue(i, j - 1).u) / GetD().y -
                GetNU() * (GetMatrixValue(i, j).v - GetMatrixValue(i - 1, j).v) / GetD().x;
            double XDYn = -GetNU() * (GetVelocityBoundary().E - GetMatrixValue(i, j).u) / (GetD().y / 2) -
                GetNU() * (GetMatrixValue(i, j + 1).v - GetMatrixValue(i - 1, j + 1).v) / GetD().x;
            return (XDXe - XDXw) / GetD().x + (XDYn - XDYs) / GetD().y;
        }
        if (j == 0) {
            // At the bottom of the domain an artificial boundary must be implemented as there is no j value below 0
            double XDYn = -GetNU() * (GetMatrixValue(i, j + 1).u - GetMatrixValue(i, j).u) / GetD().y -
                GetNU() * (GetMatrixValue(i, j + 1).v - GetMatrixValue(i - 1, j + 1).v) / GetD().x;
            double XDYs = -GetNU() * (GetMatrixValue(i, j).u - GetVelocityBoundary().W) / (GetD().y / 2) -
                GetNU() * (GetMatrixValue(i, j).v - GetMatrixValue(i - 1, j).v) / GetD().x;
            return (XDXe - XDXw) / GetD().x + (XDYn - XDYs) / GetD().y;
        }
        else {
            // Otherwise the relevant comparison can be made, no artificial boundary is necessary
            double XDYn = -GetNU() * (GetMatrixValue(i, j + 1).u - GetMatrixValue(i, j).u) / GetD().y -
                GetNU() * (GetMatrixValue(i, j + 1).v - GetMatrixValue(i - 1, j + 1).v) / GetD().x;
            double XDYs = -GetNU() * (GetMatrixValue(i, j).u - GetMatrixValue(i, j - 1).u) / GetD().y -
                GetNU() * (GetMatrixValue(i, j).v - GetMatrixValue(i - 1, j).v) / GetD().x;
            return (XDXe - XDXw) / GetD().x + (XDYn - XDYs) / GetD().y;
        }
    }
```

Figure 14: A section of code implemented to compute diffusive forces for x dimension momentum values at specific points in the domain. These equations are the logical implementation of the diffusive terms seen in equation 21b.

A.4 Linear System

Figure 15 shows how the necessary arrays and matrices for the linear system of equations are built. Specifically, this example focuses on the interior cells of the domain, neighbouring coefficients are predictable and are set when the matrix is designated. Specific cells can then be specialised. For example, the top left corner of the domain has only 2 neighbouring cells, there is no information about these cells resulting in zeros placed in intervals throughout the bands of the resultant coefficient matrix.

```
void Physics::BuildInterior() {
    // Declare variable
    double var;
    // Need to loop over all cells which don't border a boundary
    for (int j = 1; j < GetSPLITS().x - 1; j++) {
        for (int i = 1; i < GetSPLITS().y - 1; i++) {
            // Find the value for the B vector using the theories derived
            var = -GetD().y * (GetMatrixValue(i + 1, j).u + GetDT() * GetInterimValue(i + 1, j).x) +
                   GetD().y * (GetMatrixValue(i, j).u + GetDT() * GetInterimValue(i, j).x) -
                   GetD().x * (GetMatrixValue(i, j + 1).v + GetDT() * GetInterimValue(i, j + 1).y) +
                   GetD().x * (GetMatrixValue(i, j).v + GetDT() * GetInterimValue(i, j).y);
            // Set to the correct part of the linear system matrix
            SetLinearValue(i, j, var, "b");
            // Find the necessary coefficients for the coefficients matrix
            var = (2 * GetDT() * (GetD().y / GetD().x)) + (2 * GetDT() * (GetD().x / GetD().y));
            // Set to the correct part of the linear system matrix
            SetLinearValue(i, j, var, "a");
        }
    }
```

Figure 15: A section of code implemented analogous to equation 21g.

A.5 Post Processing

Python is used over C++ due to the scientific nature of its modules. Numpy and Matplotlib allow easy manipulation of data sets, matplotlib allows the visualisation of a wide range of data: streamplots, contours, and, linear graphs are employed within however this only really touches the surface of its capabilities. CSV files, see figure 9, are used to communicate between C++ and Python, this is limited to the writing of the final solution to a .txt file in the program directory as computational cost for these processes can quickly become unreasonable at large matrix sizes. Necessary information and the 3 resultant fields are written to file, Python decodes and places the correct values back into the relevant fields. Figure 16 shows how the momentum profile plots are handled. The correct points are taken out of the domain by comparing those taken in [5], as the domains are the same size the points should match identically. These selected points are placed into another array which is compared to literature values freely available within [5]. A χ^2 best fit can then be executed across the two functions using equation 27.

```
# Finding the correct points in the results
for index,dx in enumerate(x):
    for _index,ghia in enumerate(xghia):
        # If the points within xghia and x
        # are the same value place into array
        # for chi squared comparison
        if abs(dx)-abs(ghia) < 2e-16:
            holderx[_index] = x[index]
            holderu[_index] = u_profile[index]
```

Figure 16: A section of code implemented in Python to visualise the fields produced. Matplotlib is chosen for its easy use, with CSV files used to communicate between the two languages. A χ^2 function is used to compare the points between literature points and produced points.

B Other tested systems

B.1 Re = 400

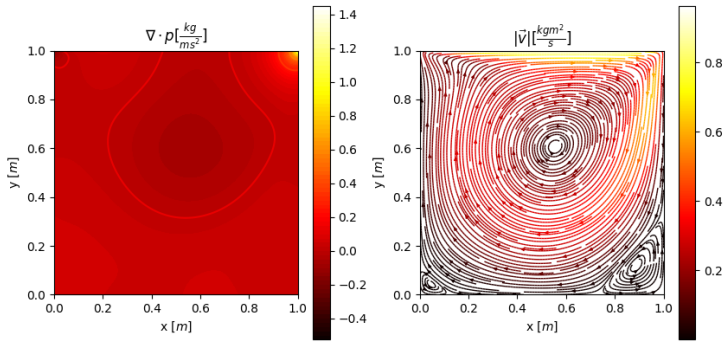


Figure 17: Pressure contour (left) and momentum stream plot (right) graphs for at Re=400. Graphs are found using equations presented within.

Computed error values are found to be $\chi^2_u = 0.0073$, $\chi^2_v = 0.0123$. A central eddy is seen. Advection has become a more dominant force within the system, seen in more distinct variations in the pressure gradient across observed eddies, although the values are smaller as less work is done to get the system moving. A more distinct vortex can be seen in the south east corner

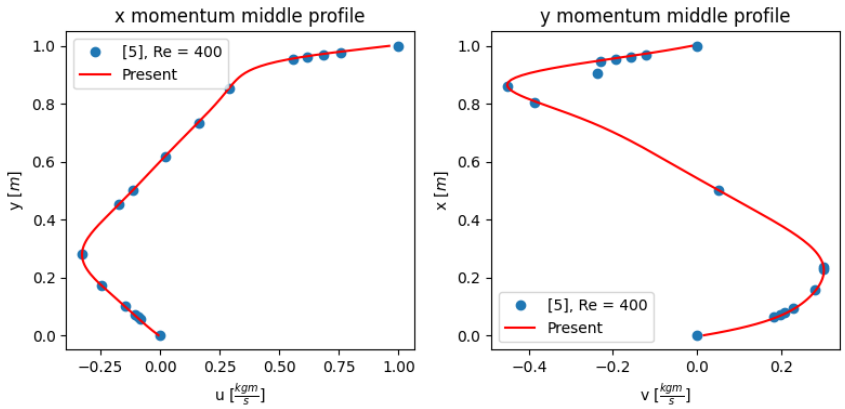


Figure 18: Analysis graphs produced for a lid driven cavity at Re = 400. The same central cells are taken for analysis. error values are found at $\chi_u^2 = 0.0091$, $\chi_v^2 = 0.0123$

of the domain, something which would be expected in a flow becoming less dominated by viscous forces.

B.2 Re = 3200

Following similar patterns seen previously, figures 19 and 20 show the plots produced when applying the finite volume method to a lid driven cavity at a Re number of 3200. A well defined central vortex dominates the movement of the system. A third eddy forms in the left corner of the northern wall due to the ever easier conditions needed for separation of flow, the system is now considered turbulent with an Re $>$ 2000, with the two previously seen eddies along the southern wall becoming ever more distinct in their total contribution to the movement of the system.

B.3 Re = 7500

Figures 21 and 22 show the system found at convergence when the Reynold's number is set to 7500. Following the patterns seen in previous systems, a central eddy dominates the structure throughout the domain, flow is separating in all of the places seen previously. An extra eddy is seen around the right of the southern wall. Absolute pressure values are becoming smaller throughout the field resulting in more distinct areas of variation, this leads to more readily available places within the domain where the conditions which require flow separation are met.

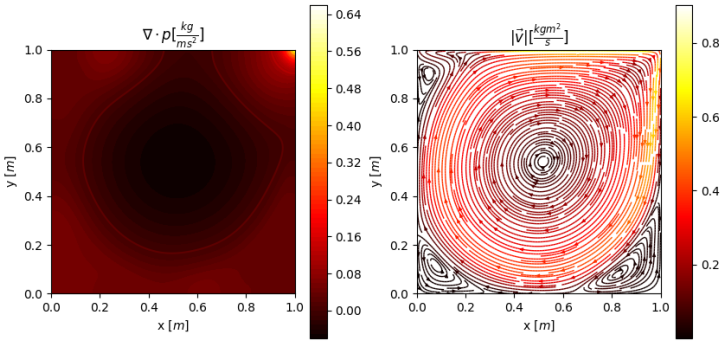


Figure 19: Pressure contour (left) and momentum stream plot (right) graphs for a lid driven cavity at Re = 3200. Following from previous systems, the central eddy becomes more defined, becoming more dominated by bulk movement of flow as the Re number is increased. More distinct variations in overall pressure gradient is found. Flow has separated around each corner apart from the right corner around the northern wall, the eddies around the southern wall have become fully defined while a third smaller eddy is found around the left corner of the northern wall. Turbulent processes are assumed to occur around Re = 2000, the systems which succeed this one are all becoming more dominated by turbulent processes.

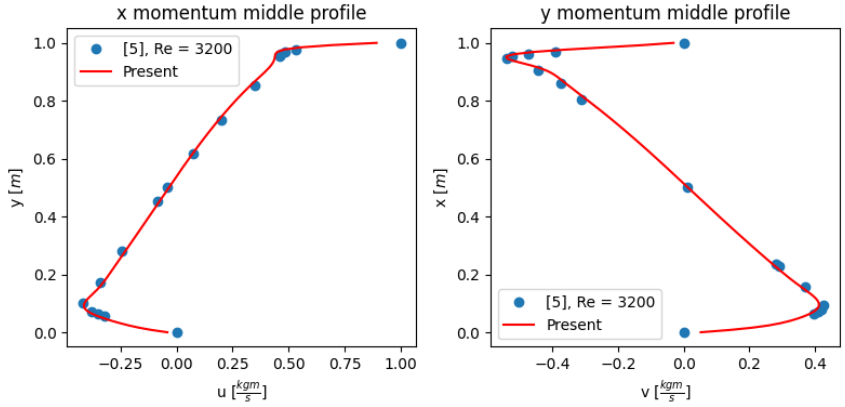


Figure 20: Analysis graphs produced for a lid driven cavity at Re = 3200. As before, the same middle cells are taken from the domain for each momentum dimension, plotted against the distance from the domain boundary. An error estimate is found for each dimension: $\chi_u^2 = 0.0073$ and $\chi_v^2 = 0.0083$

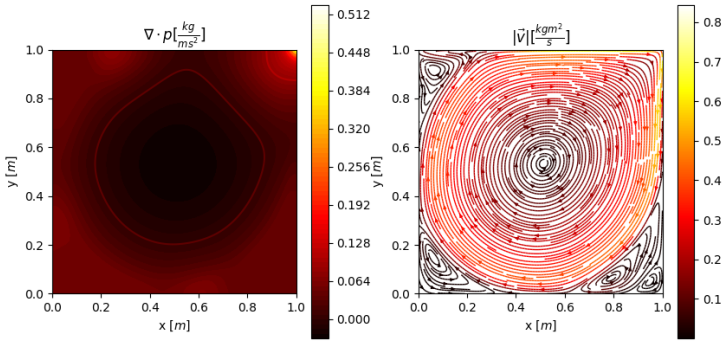


Figure 21: Pressure contour (left) and momentum stream plot (right) graphs for a lid driven cavity at Re = 7500. The system is dominated by the central vortex, distinct eddies are seen along the southern corners and around the left of the northern wall. Another distinct eddy can be seen forming below the previously seen eddy around the right of the southern wall.

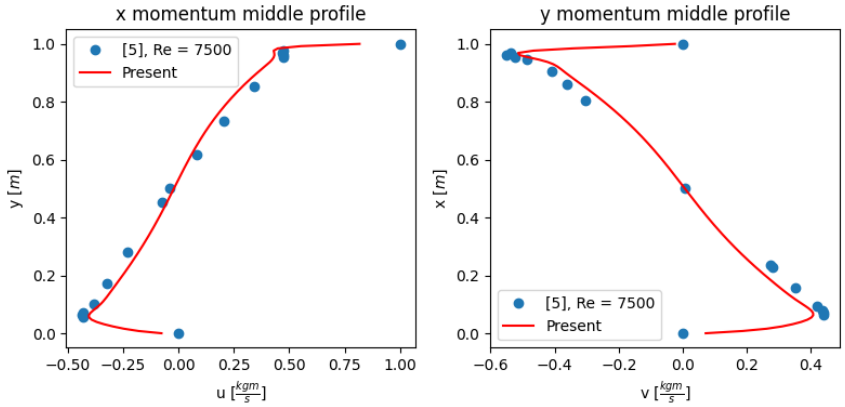


Figure 22: Analysis graphs produced for a lid driven cavity at Re = 7500. The middle profiles are taken and a best fit χ^2 function produces an error value for each dimension: $\chi^2_u = 0.0061$ and $\chi^2_v = 0.0108$

C RANS derivation

$$v_\mu = \overline{v_\mu} + v'_\mu \rightarrow \overline{v_\mu} = \lim_{t\to\infty} \int_0^t v_\mu dt \tag{32a}$$

Where a bar above a variable denotes an average, for example $\overline{v_\mu}$ is the average component and v'_μ is the locally fluctuating component. Some identities

are needed before proceeding, more detail on their origins can be found here [6], [7]:

$$\overline{v'_\mu} = 0 \rightarrow \overline{\overline{v_\mu}} = \overline{v_\mu} \rightarrow \overline{v+u} = \overline{v} + \overline{u} \rightarrow \overline{\overline{v} \cdot u} = \overline{v} + \overline{u} \rightarrow \overline{\frac{\partial v_\mu}{\partial x_\mu}} = \frac{\partial \overline{v_\mu}}{\partial x_\mu} \quad (32b)$$

32a can now be substituted into the 7b and 8d. The time average is then taken by integrating across an arbitrarily large amount of time:

$$\nabla \cdot v_\mu = \nabla \cdot (\overline{v_\mu} + v'_\mu) \rightarrow \int_0^t \nabla \cdot (\overline{v_\mu} + v'_\mu) dt \implies \nabla \cdot \overline{v_\mu} = 0 \quad (33a)$$

Where the average of a fluctuating term is zero while the average of an average is the same average, the reverse is then possible such that:

$$\nabla \cdot (\overline{v_\mu} + v'_\mu) - \nabla \cdot \overline{v_\mu} = 0 \implies \nabla \cdot v'_\mu = 0 \quad (33b)$$

Both the mean and fluctuating terms must be adhere to the constraints of the continuity equation. A similar, albeit more involved, substitution must be done for the momentum equation, 8d:

$$\frac{\partial}{\partial t}(v_\mu + v'_\mu) + [\nabla \cdot (\overline{v_\mu} + v'_\mu)](\overline{v_\mu} + v'_\mu) = -\frac{\partial}{\partial x_\mu}(\overline{P} - p') + \mu \nabla^2 \cdot (\overline{v_\mu} + v'_\mu) \quad (34a)$$

As before, the time average is taken. The previously stated identities are applied where they arise. Working this through produces the averaged momentum equation seen in equation 29.