Implementing the 2D Navier-Stokes Equations Using the Finite Volume Method with Hyperdiffusion

Ish Kaul

May 15, 2021

Abstract

We present a numerical scheme to solve the vorticity formulation of the Navier-Stokes equations (NSE) along with their solutions. The NSE is converted to vorticity evolutions and solved using the Finite Volume method with hyperdiffusion along with Strong Stability Preserving 3rd order Runge Kutta time stepping. The velocity is solved using a Poisson equation solver based on the Fast Fourier Transform for the stream potential from vorticity. We present tests of the 1D case with constant velocity for a Gaussian and a Gaussian + discontinuous top hat initial conditions along with periodic boundaries. Additionally, we solve it in the 2D case for a Gaussian monopole, and for the vortex merger case for zero and non-zero viscosity. The enstrophy convergence plot is shown for the zero viscosity case and it approaches unity as expected with a rate of 1.8.

1 Introduction

The Navier Stokes equation is a ubiquitous equation used in many fields of engineering such as fluid dynamics, aerodynamics, weather prediction and even looking at ocean currents. The equation is a partial differential equation (PDE) that describes the dynamics of viscous fluids, mass and momentum conservation for Newtonian fluids. It is essentially modified from Euler Equations for zero viscosity to describe the motion of a viscous Newtonian fluid. We can cast the equations into a vorticity formulation where the quantity called vorticity is evolved in time under velocity and viscosity effects. Now, various numerical methods for implementing the Navier-Stokes have been devised, including popular ones like finite differencing, and advanced ones like finite element and multigrid algorithms. For our case we aim to implement the equations in the vorticity formulation by using the Finite Volume method (FVM), first in one dimension with constant velocity and then in two dimensions for the case of variable velocity. We also include hyperdiffusion built in the equations while solving. The finite volume method solves the spatial part of the equation, and the 3rd order (strong stability preserving) Runge Kutta method is used to time evolve the equation. The code to produce the plots in this paper can be found on Github in the form of IPYTHON notebooks[1].

2 The Vorticity Formulation

We begin with the 2D neutral fluid equation

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v} = \nu \nabla^2 \vec{v} - \nabla p \tag{1}$$

where ν is the viscosity causing the viscous damping term, $\vec{v} = v_x \hat{x} + v_y \hat{y}$ is the 2D velocity of the fluid, p is the scalar pressure whose gradient affects the fluid velocity in the opposite direction. We do not consider the affect of gravity here. For simplicity, we assume that the fluid is incompressible which imposes the condition

$$\nabla \cdot \ \vec{v} \ = \ 0 \tag{2}$$

We introduce the helpful quantities $\omega = (\nabla \times \vec{v}) \cdot \hat{z}$, which is the scalar vorticity, and $\psi : \nabla \psi \times \hat{z} = \vec{v}$, which is called the stream function (or stream potential). With these new definitions we recast the Navier Stokes equation and incompressibility. Consider the 2D case where the vorticity is only in the z direction. Then, the two components (x, y) of the equation would read

$$\frac{\partial v_x}{\partial t} + v_x \frac{\partial v_x}{\partial x} + v_x \frac{\partial v_x}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2} \right) \tag{3}$$

$$\frac{\partial v_y}{\partial t} + v_y \frac{\partial v_y}{\partial x} + v_y \frac{\partial v_y}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2} \right) \tag{4}$$

Now we take the curl of the Navier Stokes equation by taking the x derivative of the y component subtracted by the y derivative of the x component from Equations (3) and (4) as given by the curl formula to yield

$$\begin{aligned} \frac{\partial}{\partial t}\left(\frac{\partial v_x}{\partial x} - \frac{\partial v_y}{\partial y} \right) + v_x \frac{\partial}{\partial x}\left(\frac{\partial v_x}{\partial x} - \frac{\partial v_y}{\partial y} \right) + v_y \frac{\partial}{\partial y}\left(\frac{\partial v_x}{\partial x} - \frac{\partial v_y}{\partial y} \right) \\ + \left(\frac{\partial v_x}{\partial x} - \frac{\partial v_y}{\partial y} \right)\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} \right) = 0 \end{aligned} \tag{5}$$

But notice that under the 2D assumption the vorticity becomes

$$\omega = \nabla \times \vec{v} = \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \tag{6}$$

which is exactly the terms inside the parentheses ocurring in Equation (5). We also have from the stream potential definition

$$v_x = \frac{\partial \psi}{\partial y} \qquad v_y = -\frac{\partial \psi}{\partial x} \tag{7}$$

along with imposing the continuity equation

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} = \frac{\partial^2 \psi}{\partial x \partial y} - \frac{\partial^2 \psi}{\partial x \partial y} = 0 \tag{8}$$

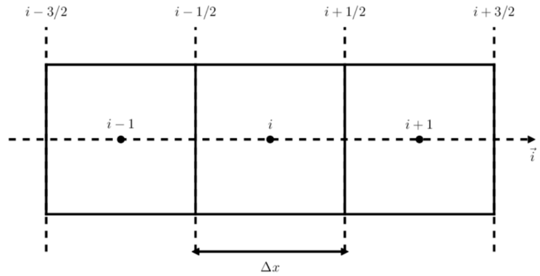


Figure 1: Finite Volume stencil[3]. We want to look at the cell edges.

to finally yield the vorticity evolution equation as

$$\frac{\partial \omega}{\partial t} + v_x \frac{\partial \omega}{\partial x} + v_y \frac{\partial \omega}{\partial y} = \nu \left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2} \right) \tag{9}$$

or cast in a more compact form as it appears in the literature[2]

$$\frac{\partial \omega}{\partial t} = -[\omega, \psi] + \nu \nabla^2 \omega \tag{10}$$

where the Poisson bracket term is the nonlinear advection term defined as follows

$$[\omega, \psi] = \frac{\partial \omega}{\partial x}\frac{\partial \psi}{\partial y} - \frac{\partial \omega}{\partial y}\frac{\partial \psi}{\partial x} \tag{11}$$

along with the incompressibility condition after substituting the stream potential for the velocity

$$\nabla^2 \psi = -\omega \tag{12}$$

Equation (10) is the Navier Stokes equation in the vorticity formulation, while (12) is the incompressibility condition which relates the stream function and the vorticity. We thus aim to implement this coupled system of PDEs using the finite volume method (FVM) as will be explained in the next section.

3 The Finite Volume Method

The FVM is based on looking at the cell averaged quantities of the specified PDE and evolving it in space by looking at the fluxes across the cell boundaries. In essence once a spacial grid is constructed we define quantities called fluxes at the cell edges (as opposed to the cell centers) which are given by the product of the vorticities and velocities at those edges, as shown in Fig. 1.

Now, our goal is to construct an FVM scheme for the coupled differential equation system. Notice that Equation (12) is a Poisson equation which will be dealt separately using pseudo spectral methods later. Here we concern ourselves with a scheme for Equation (10). Now, suppose we construct a spatial grid in x and y where each cell is then indexed by (i, j) for the respective directions. Then, the FVM will work for finding the cell averaged vorticities as defined by

$$\bar{\omega}_{i,j} = \frac{\int_i \int_j \omega \, dx \, dy}{h^2} \tag{13}$$

where h is the grid spacing both in x, and in y. The discussion further assumes that there is unform spacing in x and y, $dx = dy = h$, but all the formulae could be adapted to the case of non-uniform spacing as well. Now, we know the vorticities at each point from the initial conditions, so we get the vorticities at the cell boundaries in both directions by using a 4th order centered scheme

$$\omega_{i+1/2,j} = \frac{7}{12}(\omega_{i+1,j} + \omega_{i,j}) - \frac{1}{12}(\omega_{i-1,j} + \omega_{i+2,j}) \tag{14}$$

$$\omega_{i,j+1/2} = \frac{7}{12}(\omega_{i,j+1} + \omega_{i,j}) - \frac{1}{12}(\omega_{i,j-1} + \omega_{i,j+2}) \tag{15}$$

We also have the velocities at the cell centers, so we find the velocities at the cell edges by using linear interpolation

$$v^x_{i\pm1/2,j} = \frac{v^x_{i\pm1,j} + v^x_{i,j}}{2} \tag{16}$$

$$v^y_{i,j\pm1/2} = \frac{v^y_{i,j\pm1} + v^y_{i,j}}{2} \tag{17}$$

Now we have our expressions that we can use to construct the fluxes at the cell boundaries. As mentioned before the fluxes are simply given as the vorticity and velocity products, however, we also add a hyperdiffusion instead of using limiters for simplicity as follows

$$F^x_{i+1/2,j} = v^x_{i+1/2,j}\omega_{i+1/2,j} - \frac{C_4|v_{max}|h}{48}\left(\frac{3}{12}(\omega_{i+1,j} - \omega_{i,j}) - \frac{1}{12}(\omega_{i+2,j} - \omega_{i-1,j})\right) \tag{18}$$

$$F^y_{i,j+1/2} = v^y_{i,j+1/2}\omega_{i,j+1/2} - \frac{C_4|v_{max}|h}{48}\left(\frac{3}{12}(\omega_{i,j+1} - \omega_{i,j}) - \frac{1}{12}(\omega_{i,j+2} - \omega_{i,j-1})\right) \tag{19}$$

where C_4 is some constant to make the hyperdiffusion term on the order of the first flux term, which we set to 1. v_{max} is the maximum velocity on the grid at that instant in time. With this we can now make the FVM expression for Equation (10). Notice that the Poisson bracket term is actually composed of these fluxes and so the expression just becomes[2]

$$\frac{\partial \bar{\omega}_{i,j}}{\partial t} = \nu \nabla^2 \bar{\omega}_{i,j} - \frac{F^x_{i+1/2,j} - F^x_{i-1/,j} + F^y_{i,j+1/2} - F^y_{i,j-1/2}}{h} \tag{20}$$

Now, we need to look at the viscosity term with the laplacian. For this we use a finite centered difference scheme like

$$\frac{\partial^2 \omega}{\partial x^2} = \frac{\omega_{i+1,j} - 2\omega_{i,j} + \omega_{i-1,j}}{dx^2} \tag{21}$$

$$\frac{\partial^2 \omega}{\partial y^2} = \frac{\omega_{i,j+1} - 2\omega_{i,j} + \omega_{i,j-1}}{dy^2} \tag{22}$$

Thus our final numerical FVM step for the Navier-Stokes equation is given by

$$\frac{\partial \bar{\omega}_{i,j}}{\partial t} = \nu \left(\frac{\omega_{i+1,j} - 2\omega_{i,j} + \omega_{i-1,j}}{h^2} + \frac{\omega_{i,j+1} - 2\omega_{i,j} + \omega_{i,j-1}}{h^2} \right) - \frac{F^x_{i+1/2,j} - F^x_{i-1/,j} + F^y_{i,j+1/2} - F^y_{i,j-1/2}}{h} \quad (23)$$

4 Runge Kutta 3rd and Poisson Solver

Now, we have solved the spatial portion of the equation using the FVM. All we are left with is to evolve the equation in time and find a scheme for the second PDE Equation (12). The time evolution we use here is the 3rd order Runge Kutta method with strong stability preserving (SSP). The standard RK4 method was not used since there is no version where stability is preserved. The SSP-RK3 method is given by the following three steps

$$\omega_1 = \omega_t + \Delta t F(\omega_t, \psi_0) \quad (24)$$

$$\omega_2 = \frac{3}{4}\omega_t + \frac{1}{4}\omega_1 + \frac{1}{4}\Delta t F(\omega_1, \psi_1) \quad (25)$$

$$\omega_{t+dt} = \frac{1}{3}\omega_t + \frac{2}{3}\omega_2 + \frac{2}{3}\Delta t F(\omega_2, \psi_2) \quad (26)$$

where $F(\omega, \psi) = \frac{\partial \bar{\omega}}{\partial t}$. Equation (24) is just a single forward Euler step, and the other two are further approximations to build upon the first step. Notice that at each step we need to input a new ψ value which we need to compute from the velocity of the grid. For this we solve the Poisson equation by inverting the laplacian using the Fast Fourier Transform (FFT). The FFT is the fast version of the Discrete Fourier Transform (DFT)[4] which is given as

$$F_k = \sum_{n=0}^{N-1} f_n e^{-i\frac{2\pi n}{N}k} \quad (27)$$

This is easily implemented through Numpy's FFT package. We construct frequency arrays k_x, k_y in both x and y directions based on the number of points and divide the FFT of the vorticity by the sum of the squares of the frequency arrays. Finally we perform the inverse FFT to give the stream potential. From here we can get the velocities at each spatial grid point by the curl formula in Equation (6) by finite differencing

$$v^x_{i,j} = \frac{\psi_{i,j+1} - \psi_{i,j-1}}{2h} \quad (28)$$

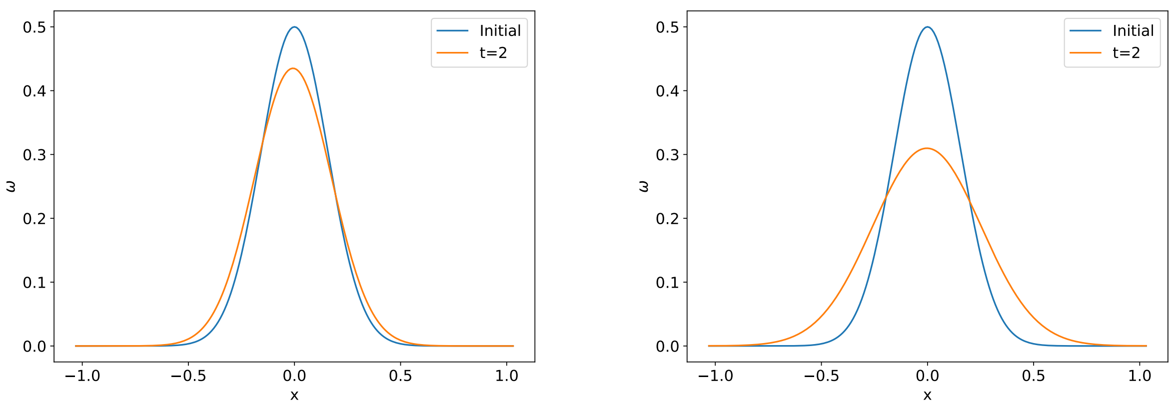
$$v^y_{i,j} = -\frac{\psi_{i+1,j} - \psi_{i-1,j}}{2h} \quad (29)$$

Now, with all the equations set up, we implement these and look at numerically solving the 1D case and the 2D case as will be presented in the next section

5 Results

5.1 1D Case

We apply periodic boundary conditions and consider the case of constant velocity in one dimension. This was implemented along with 3 ghost cells on either side of the boundary. The simulation box is taken to be $-1 \leq x \leq 1$. The initial condition is given with $\omega = \Omega \exp\left(-\frac{x^2}{\sigma^2}\right)$ for $\Omega = 0.5$, $\sigma^2 = 0.05$, $dx = 0.01$ and $v = 1$. The plots are shown in Fig. 2.



(a) $\nu = 0.001$

(b) $\nu = 0.01$

Figure 2: Gaussian initial condition along with the numerical solution at t=2. As we expect the solution should exactly line up with the peak, and the higher viscosity causes more diffusion.

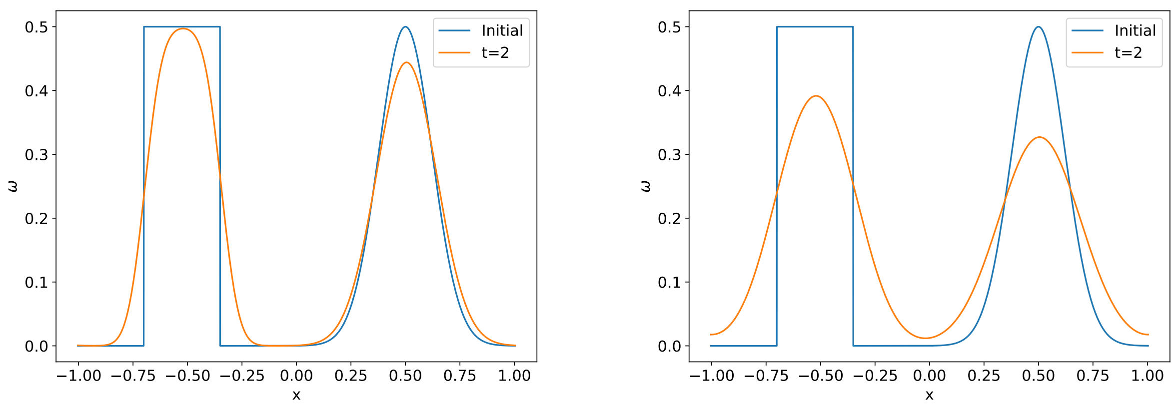
Now we move to the slightly more complicated case of a gaussian with a discontinuous top hat. Here we use $\omega = \Omega \exp\left(-\frac{(x-0.5)^2}{\sigma^2}\right)$ for $\Omega = 0.5$, $\sigma^2 = 0.03$, $dx = 0.001$. The plots are shown in Fig. 3.

5.2 2D Case

Here we consider two cases: the first is a Gaussian monopole with viscosity. Again, we use three ghost cells on either side in both x and y directions along with doubly periodic boundary conditions. Notice that in the 2D case, velocity is continuously evolving and so we use the Poisson solver for the velocity, even for the initial conditions. We take the initial condition on the vorticity as

$$\omega = \exp\left(-\frac{(x-x_1)^2 + (y-y_1)^2}{\sigma^2}\right) \tag{30}$$

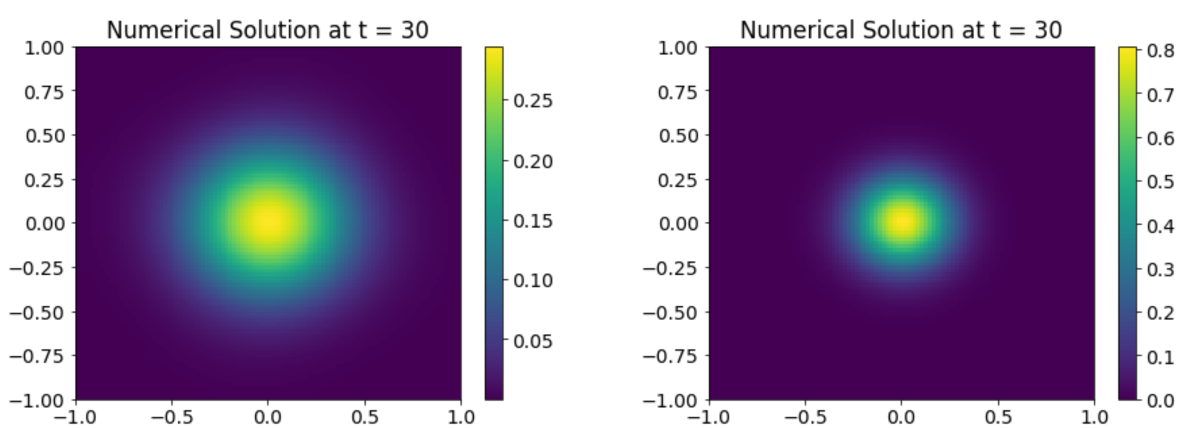
for $x_1 = 0$, $y_1 = 0$, $\sigma^2 = 1/20$ which is a Gaussian monopole in 2D. The solutions are shown in Fig. 4.



(a) $\nu = 0.001$

(b) $\nu = 0.005$

Figure 3: Gaussian + top hat initial condition along with the numerical solution at t=2. As we expect again the solution lines up with the peak, and the higher viscosity causes more diffusion.



(a) $\nu = 0.001$

(b) $\nu = 0.0001$

Figure 4: 2D Gaussian monopole solutions for $h = 0.02$. As expected, increasing viscosity causes more diffusion.

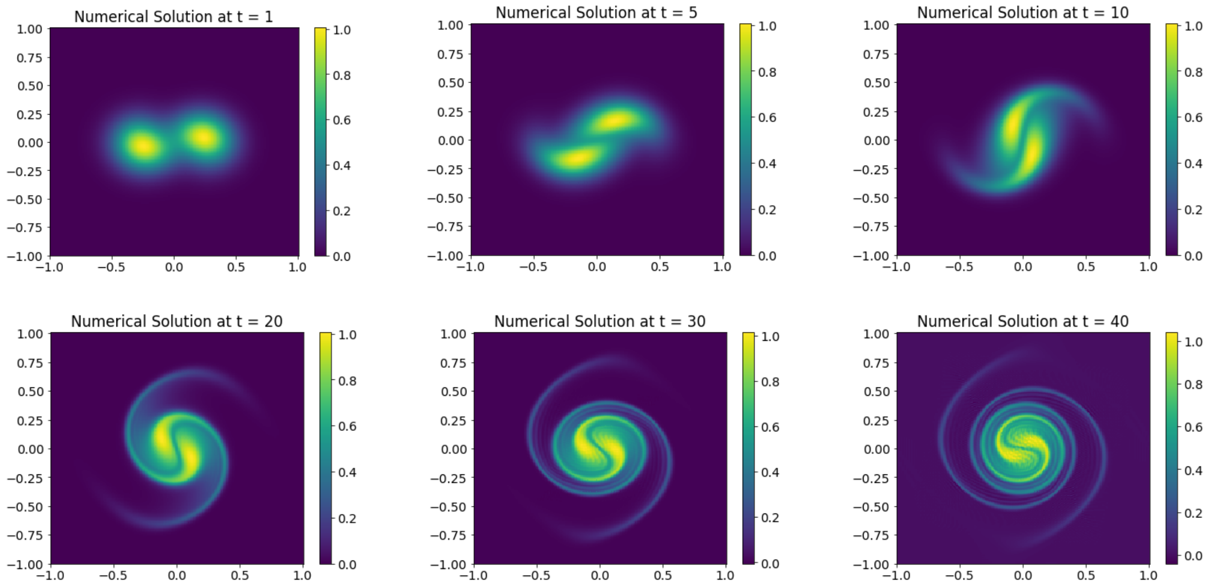


Figure 5: 2D Gaussian dipole solutions for $h = 0.01$ and $\nu = 0$. We see the vortex merger happening with no diffusion, and the spirals are produced as a result of the differential in velocities in the inner and outer boundaries.

Now we consider the second case, which is two Gaussian monopoles with co-rotating velocities, or the 'vortex merger' problem as often called in literature. Here we initialize vorticity as

$$\omega = \exp\left(-\frac{(x-x_1)^2 + (y-y_1)^2}{\sigma^2}\right) + \exp\left(-\frac{(x-x_2)^2 + (y-y_2)^2}{\sigma^2}\right) \tag{31}$$

for $x_1 =$ -0.25, $x_2 = 0.25$, $y_1 = y_2 = 0$, $\sigma^2 = 1/20$. The results are shown in Fig. 5.

Additionally we look at the case with some viscosity in the vortex merger as shown in Fig. 6.

Finally to test our algorithm we look at the enstrophy variation with spatial step size h. The enstrophy is defined as the sum of the squares of the vorticities at each point on the spatial grid, and we should see that the ratio of enstrophy at some time t with the initial enstrophy should approach unity due to enstrophy conservation. Of course this is only valid in the case of zero viscosity where there is no loss. Thus, we plot the enstrophy convergence at time $t = 30$ vs step size in Fig. 7.

The slope in regular scale shows linear convergence with a rate of 1.8. We find this using the formula for enstrophy E

$$p = \frac{\log(e_{i+1}/e_i)}{\log(e_i/e_{i-1})} = \frac{\log((1-E_{i+1})/(1-E_i))}{\log((1-E_i)/(1-E_{i-1}))} \tag{32}$$

The reason this is not up to 4th order (as we would expect since the ω estimate uses the 4th order centered scheme, and the Poisson solver is in essence, exact), is because the cell volume averaged vorticity squared and the cell volume average of the square of the vorticity

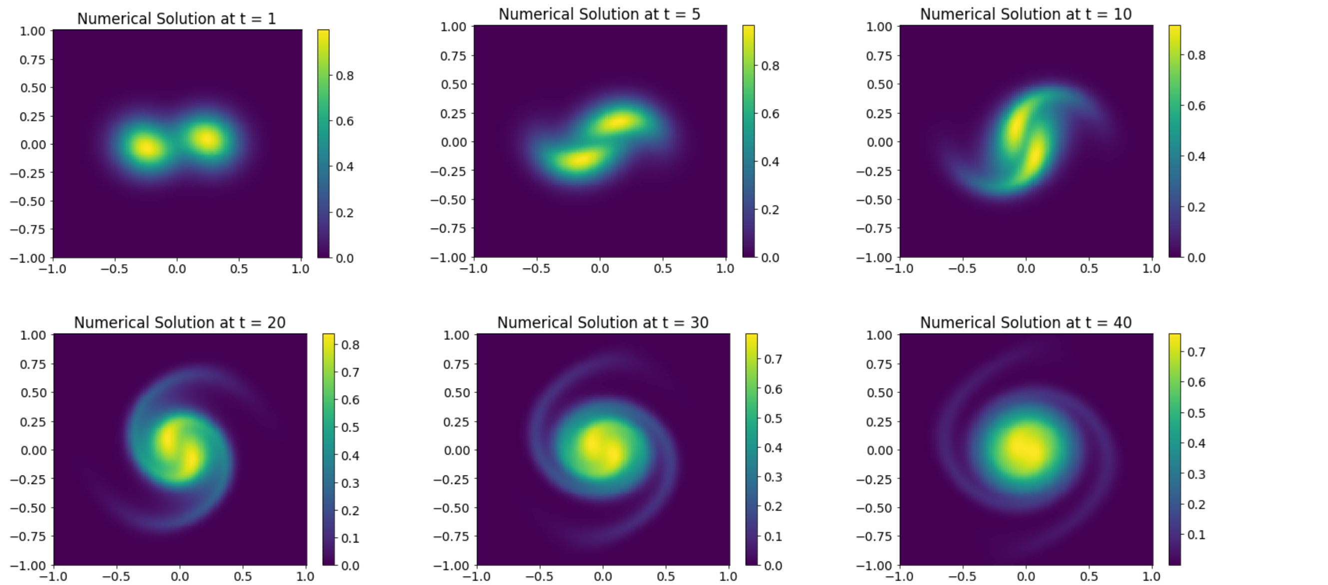


Figure 6: 2D Gaussian dipole solutions for $h = 0.015$ and $\nu = 0.0001$. We see the vortex merger happening with diffusion causing some blurring, and the spirals are produced as a result of the differential in velocities in the inner and outer boundaries.

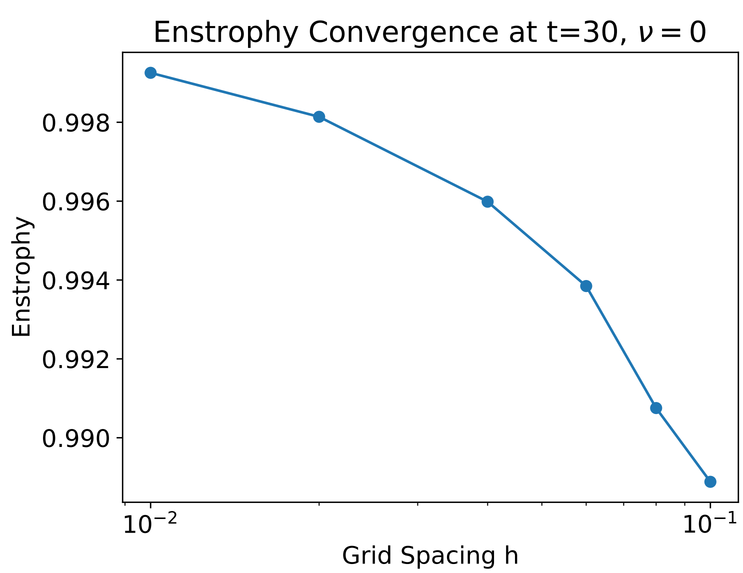


Figure 7: Enstrophy convergence plot, the x axis is on log scale

agree only to 2nd order, and so to be more accurate we would need to construct a 4th order polynomial to approximate the vorticity from the cell averaged vorticity.

6 Conclusion

We construct a numerical scheme for the finite volume method to solve the Navier Stokes equation cast into the vorticity formulation. We solve the NSE using hyperdiffusion along with 3rd order Strong Stability Preserving Runge Kutta method for the time stepping. We solve the equation for the case of a 1D Gaussian with diffusion and the 1D Gaussian plus a discontinuous top hat, and see that the results agree (up to some diffusion). We then solve it for the 2D Gaussian monopole with diffusion to see how the solution becomes more diffuse with higher viscosity. Finally solve it for the vortex merger problem (two co-rotating Gaussian monopoles) with zero and nonzero viscosity. Here we see the development of the spirals as well as the merging phenomena, with blurring caused by the viscosity term. To test the algorithm we provide and enstrophy convergence plot with an estimate to the rate of convergence which we found to be around 1.8. In the future we suggest using a 4th order polynomial to get the vorticity form the average to improve this number and implement limiters instead of hyperdiffusion to get a more accurate result. We also suggest parallelizing the code to gain a speedup.

References

[1] I. Kaul, "finite-volume-Navier-Stokes". https://github.com/ikaul00/finite-volume-Navier-Stokes, 2021.

[2] G. Peterson, J.l. & Hammett, "Positivity preservation and advection algorithms with applications to edge plasma turbulence", *SIAM J. Sci. Comput.* **35** (2013), no. 3, doi:10.1137/120888053.

[3] J. Vanharen, "Theoretical and numerical analysis of block interfaces with mismatched nodes. emphasis on unsteady flows.", 04, 2015. doi:10.13140/RG.2.1.1711.5688.

[4] J. W. Cooley, J.W. & Tukey, "An algorithm for the machine calculation of complex fourier series", *Math. Comp.* **93** (1965) 297–301, doi:10.1090/S0025-5718-1965-0178586-1.