2D Incompressible Navier Stokes Equation in Vorticity Streamfunction Formulation using Spectral Method

Anushka Subedi

ASU ID: 1225812200

Contents

1 Introduction **4**
1.1 Vorticity-Streamfunction Formulation 4
1.2 Geometry, Boundary and Initial Conditions 5

2 Numerical Method **6**

3 Results **7**

4 Discussion and Conclusion **10**

Appendices **13**

Nomenclature

u : Velocity in the x-direction
$\hat{u}$: Fourier Velocity Component in the x-direction
v : Velocity in the y-direction
$\hat{v}$: Fourier Velocity Component in the y-direction
P : Pressure
ν : Viscosity
ω : Vorticity
$\hat{\omega}$: Fourier Vorticity Component
ψ : Streamfunction
$\hat{\psi}$: Fourier Vorticity Component
K_x : Wave Number in x-direction
K_y : Wave Number in y-direction
N_x : Number of Grid in x-direction
N_y : Number of Grid in y-direction
$\circledast$: Convolution Sum
FFT : Fast Fourier Transform
$IFFT$: Inverse Fast Fourier Transform
$2D$: Two Dimensional
ODE : Ordinary Differential Equation

1 Introduction

In this project, 2D incompressible Navier Stokes equations in vorticity streamfunction formulation is solved. Fourier Methods has been implemented to solve the 2D Navier Stokes equation.

1.1 Vorticity-Streamfunction Formulation

In the Vorticity-Streamfunction Formulation, three equations reduce to two equations and solving 2D Navier-Stokes equation is easier and also an ideal way to go about the problem. The two dimensional incompressible Navier-Stokes equations are:

- Continuity Equation

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{1}$$

- X-Momentum Equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{2}$$

- Y-Momentum Equation

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{3}$$

Differentiating (2) with y and (3) with x, we get,

$$\frac{\partial^2 u}{\partial y \partial t} + \frac{\partial u}{\partial y}\frac{\partial u}{\partial x} + u\frac{\partial^2 u}{\partial y \partial x} + \frac{\partial v}{\partial y}\frac{\partial u}{\partial y} + v\frac{\partial^2 u}{\partial y^2} = -\frac{1}{\rho}\frac{\partial^2 P}{\partial y \partial x} + \nu\left(\frac{\partial^3 u}{\partial y \partial x^2} + \frac{\partial^3 u}{\partial y^3}\right) \tag{4}$$

$$\frac{\partial^2 u}{\partial x \partial t} + \frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + u\frac{\partial^2 v}{\partial x^2} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial y} + v\frac{\partial^2 v}{\partial x \partial y} = -\frac{1}{\rho}\frac{\partial^2 P}{\partial x \partial y} + \nu\left(\frac{\partial^3 u}{\partial x^3} + \frac{\partial^3 u}{\partial x \partial y^2}\right) \tag{5}$$

Now, subtracting equation (4) from equation (5), we get,

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) + u\frac{\partial}{\partial x}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) + v\frac{\partial}{\partial y}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) + \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) = \\ \nu\left[\frac{\partial^2}{\partial x^2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) + \frac{\partial^2}{\partial y^2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)\right]\end{aligned} \tag{6}$$

From the continuity equation (1), $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$, so the last term in LHS of (6) becomes zero, yielding,

$$\frac{\partial}{\partial t}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) + u\frac{\partial}{\partial x}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) + v\frac{\partial}{\partial y}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) = \nu\left[\frac{\partial^2}{\partial x^2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) + \frac{\partial^2}{\partial y^2}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)\right] \tag{7}$$

The vorticity is defined as:

$$\overline{\omega} = \nabla \times \overline{V} \tag{8}$$

The two dimensional form in xy-plane is:

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{9}$$

Subsituting equation (9) in equation (7), we get,

$$\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = \nu\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right) \tag{10}$$

This is the vorticity transport equation to be solved instead of the momentum equations above. The velocities in x and y direction in terms of stream-function, ψ is given by:

$$\begin{aligned} u &= \frac{\partial \psi}{\partial y} \\ v &= -\frac{\partial \psi}{\partial x} \end{aligned} \tag{11}$$

Substituting equation (11) in the Continuity Equation (1), we get,

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\omega \tag{12}$$

Therefore, three sets of equations reduce to two equations (10) and (12) to be solved.

1.2 Geometry, Boundary and Initial Conditions

The Navier-Stokes equations in vorticity stream-function formulation i.e. equations (10) and (12) are solved in a square domain of length 2π on all sides, with **periodic boundary** conditions. The **initial condition** is a random gaussian number for each Fourier component of ω, similar in modality of [1] which uses random guassian realization for each Fourier component of ψ for solving the vorticity transport equation (10). The co-ordinates at which the random fourier component $\hat{\omega}$ is employed as initial condition are taken from [2]. The paper goes into much detail study about coherent structures and inverse cascading in turbulence, but for the scope of this project, their coordinates, mentioned only in their code, has been used. The stream-function can then be calculated by solving the Poisson Equation in (12). The geometry with the boundary conditions are as shown in Figure (1.2).

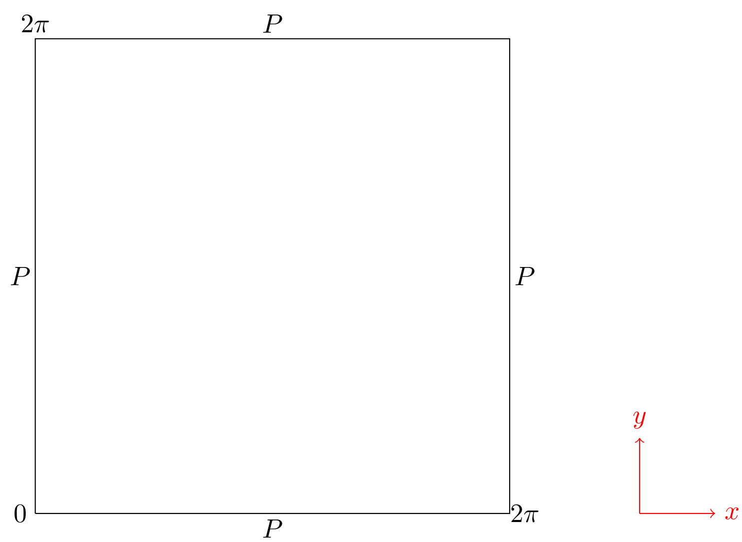

Figure 1: Geometry

The values of ω and ψ completely define the two equations (10) and (12):

$$\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = \nu\left(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}\right)$$

and

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\omega$$

The velocity fields can then be found using equations (11),

$$u = \frac{\partial \psi}{\partial y}$$

and

$$v = -\frac{\partial \psi}{\partial x}$$

To find the pressure term, which is cancelled out in the vorticity stream-function formulation, pressure Poisson Equation must be solved, which isn't the scope of this project.

2 Numerical Method

Fourier Method has been used to solve the problem mentioned in Section 1. The Fourier transform of any field, u_{ij} from spectral to physical space is given by:

$$u_{ij} = \sum_{m=\frac{-N_x}{2}}^{\frac{N_x}{2}-1} \sum_{n=\frac{-N_y}{2}}^{\frac{N_y}{2}-1} \widehat{u_{mn}} e^{iK_m X_i + iK_n Y_j} \tag{13}$$

Here, $K_m = \frac{2\pi m}{L_x}$ and $K_n = \frac{2\pi n}{L_y}$ are the wave numbers. In matlab FFT routines, they are in the order, $K_x = [0, 1, 2, ... \frac{Nx}{2} - 1, \frac{-Nx}{2}, ... - 2, -1]$ and $K_y = [0, 1, 2, ... \frac{Ny}{2} - 1, \frac{-Ny}{2}, ... - 2, -1]$. The inverse fourier transform of (13) is given by:

$$\widehat{u_{mn}} = \frac{1}{N_x N_y} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} u_{ij} e^{-(iK_m X_i + iK_n Y_j)} \tag{14}$$

The X_i and Y_j in equations (13) and (14) are the collocation points given by: $X_i = \frac{iL_x}{N_x}$ and $Y_j = \frac{iL_y}{N_y}$ where, $i = 0, 1, 2, ...N_x$ and $j = 0, 1, 2, ...N_y$. The last collocation point is not used in the above equations because with fourier methods, the boundary conditions are periodic and hence, $x_N = x_0$ and $y_N = y_0$.
The n^{th} order derivative of equation 13 is given by:

$$\frac{\partial^n u_{ij}}{\partial x^n} = \sum_{m=\frac{-N_x}{2}}^{\frac{N_x}{2}-1} \sum_{n=\frac{-N_y}{2}}^{\frac{N_y}{2}-1} \widehat{u_{mn}} (ikx)^n e^{iK_m X_i + iK_n Y_j} \tag{15}$$

As we can see in equation 15, the derivative of the fourier transform is the same as the transform with a complex wave number $(ikx)^n$ multiplying the RHS, instead of the use of differential schemes which are subject to more errors. This makes the Pseudo-Spectral or Collocation Fourier method so powerful.
Now, in order to implement this numerical method into our problem, expressing equation (10) in terms of the expansions in equations (13) and (15), we get,

$$\frac{\partial \widehat{\omega_{mn}}}{\partial t} + (u\hat{_{mn}} \circledast iK_x \widehat{\omega_{mn}}) + (v\hat{_{mn}} \circledast iK_y \widehat{\omega_{mn}}) = \nu(-({K_x}^2 + {K_y}^2))\widehat{\omega_{mn}} \tag{16}$$

This is obtained using orthogonality, where all the summations and exponential terms are eventually cancelled out, yielding the above equation.
The equation to be solved is,

$$\frac{\partial \widehat{\omega_{mn}}}{\partial t} = -(u\hat{_{mn}} \circledast iK_x \widehat{\omega_{mn}}) - (v\hat{_{mn}} \circledast iK_y \widehat{\omega_{mn}}) - \nu({K_x}^2 + {K_y}^2)\widehat{\omega_{mn}} \tag{17}$$

A time marching scheme, like Adams-Bashforth can be used to solve equation (17). The ode113 routine in MATLAB performs the Adams-Bashforth scheme and the same has been used in the project to time march.

However, first, there are two things that need to be figured out:
1. How to perform convolution sum?
2. What are $u\hat{_{mn}}$ and $v\hat{_{mn}}$?
The convolution sum of $(f \circledast g)$ is calculated by first performing the inverse transform of f and g indvidually and then, performing the forward transform of the product of the two inverse transforms i.e.

$$(f \circledast g) = FFT(IFFT(f) * IFFT(g)) \tag{18}$$

Next, to find $u\hat{_{mn}}$ and $v\hat{_{mn}}$, they can be expressed in terms of either $\widehat{\omega}$ or in terms of streamfunction in equations (11). In this project, it has been expressed in terms of $\widehat{\omega}$ because the initial conditions are also taken for $\widehat{\omega}$, it has been deduced such that,
From Poisson Equation in equation (12) expressed in terms of expansion of the Fourier Methods from equations (13) through (13), we get, streamline,

$$\psi_{mn} = \frac{\omega_{mn}}{K_x^2 + K_y^2}$$

The relation between ψ and u and v is given by equation (11), which gives,

$$\begin{aligned} \widehat{u_{mn}} &= \frac{iK_y \omega\hat{_{mn}}}{K_x^2 + K_y^2} \\ \widehat{v_{mn}} &= \frac{-iK_x \omega\hat{_{mn}}}{K_x^2 + K_y^2} \end{aligned} \tag{19}$$

With all these requirements figured out, equation (17) can now be solved, except, the aliasing problem in Fourier Methods still needs to be mitigated.
The dealiasing treatment has been done using the 3/2 rule [3]. In this method, the IFFT has been performed in the convolution term for M=3/2N points, instead of the Nx/Ny points. The coefficients are then padded zero for any $|m| > N_x$ and $|n| > N_y$ i.e.
For $|m| > N_x$ and $|n| > N_y$,

$$\widehat{\omega_{mn}} = 0$$

For all other m and n,

$$\widehat{\omega_{mn}} = \widehat{\omega_{mn}}$$

Therefore, solving equation (17) by first mitigating the aliasing problem, solving for the convolution sum and using the $\widehat{u_{mn}}$ and $\widehat{v_{mn}}$ and time-marching using the Adams-Bashforth method gives the solution for $\widehat{\omega_{mn}}$. The ω_{mn} can then be calculated by performing the IFFT of $\widehat{\omega_{mn}}$. Alternatively, as done in this project for easier coding, the final FFT of the convolution sum can be held off, the last term in the equation (17) can be inverse transformed, which makes the ODE in equation (17) for ω_{mn}, instead of $\widehat{\omega_{mn}}$. After this, regular Adams-Bashforth method is implemented to find ω_{mn}.

3 Results

The vorticity distribution at time=0, time=1, time=5, time=10, time=15 and time=30 are attached here. The results of vorticity distribution shows that as time moves forward, the vorticity is concentrated within a confined portion of the spatial domain in the flow structure. This vorticity concentration seems to adopt an axisymmetric form and endure through passive advection by larger scales in the flow, as seen in the Figure 7 at time=30. Due to limited computational capability, the simulation could not be ran further in this project, but higher time would show even better vorticity concentration in the domain.

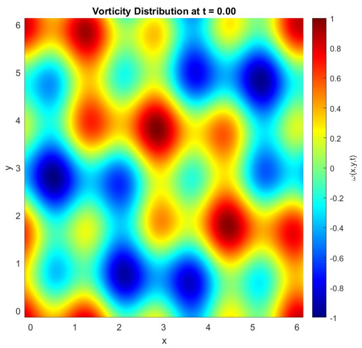

Figure 2: Vorticity @ t=0

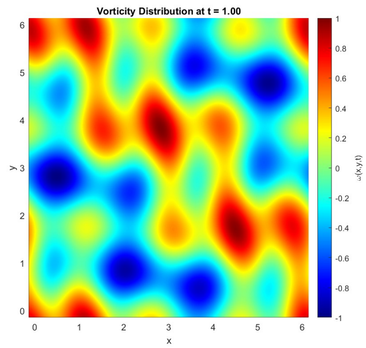

Figure 3: Vorticity @ t=1

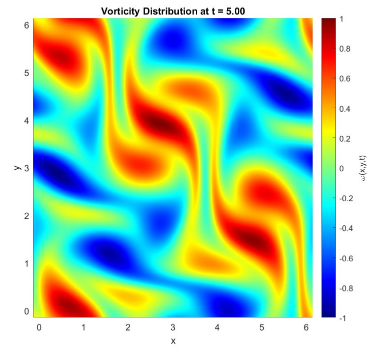

Figure 4: Vorticity @ t=5

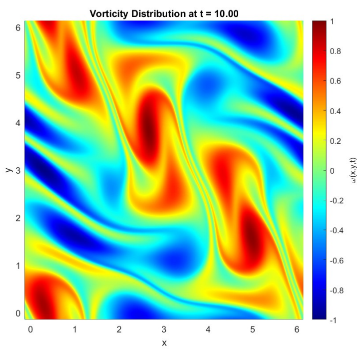

Figure 5: Vorticity @ t=10

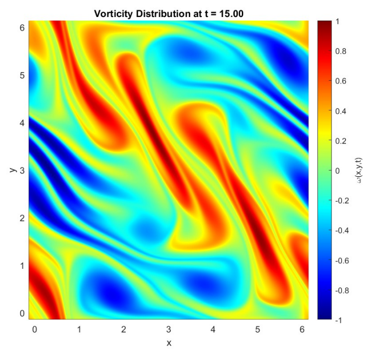

Figure 6: Vorticity @ t=15

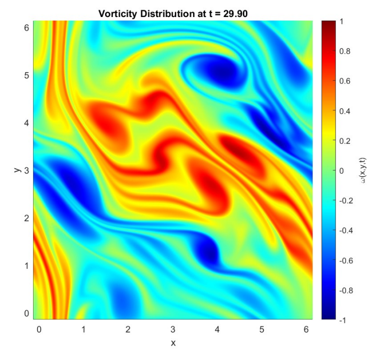

Figure 7: Vorticity @ t=30

Similarly, the streamline distribution at time=0, time=1, time=5, time=10, time=15 and time=30 are attached here. The results of streamfunction distribution shows that as time moves forward, the streamfunction is concentrated within a confined portion of the spatial domain in the flow structure. This streamfunction concentration also seems to adopt an axisymmetric form and endure through passive advection, as seen in Figure 13 at t=30.

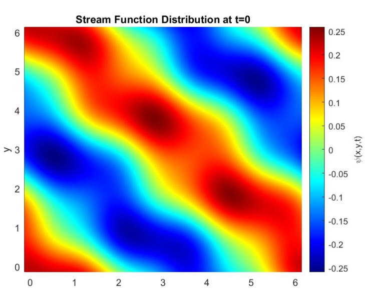

Figure 8: Streamline @ t=0

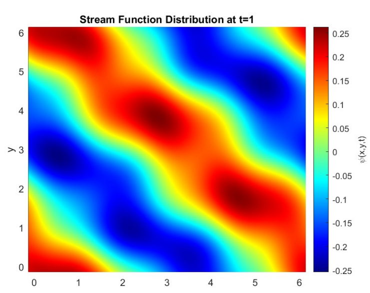

Figure 9: Streamline @ t=1

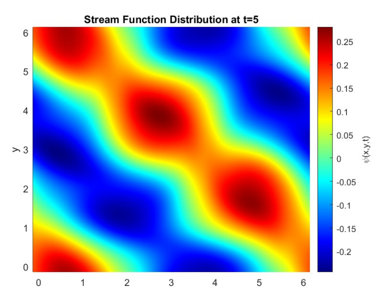

Figure 10: Streamline @ t=5

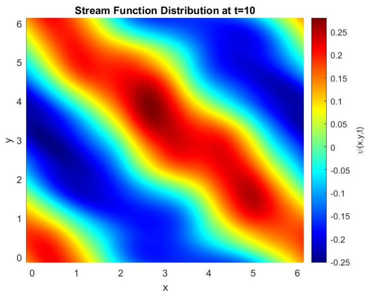

Figure 11: Streamline @ t=10

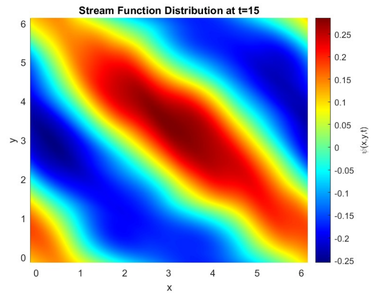

Figure 12: Streamline @ t=15

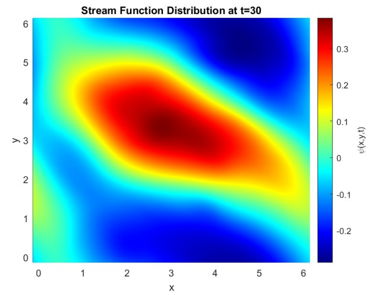

Figure 13: Streamline @ t=30

The u velocity distribution at time=0, time=1, time=5, time=10, time=15 and time=30 are attached here. The u velocity distribution also depicts similar results as that of vorticity and streamfunction.

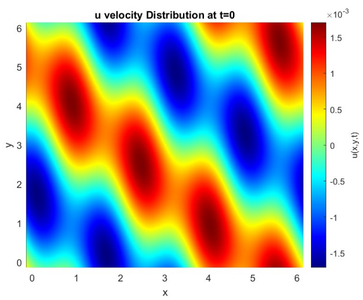

Figure 14: u velocity @ t=0

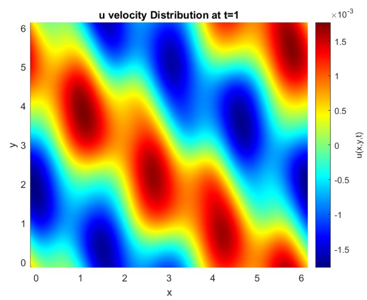

Figure 15: u velocity @ t=1

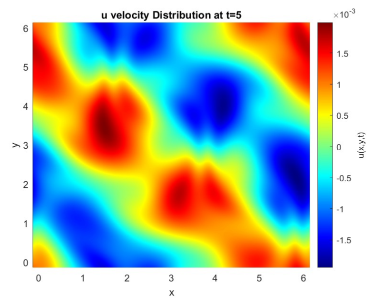

Figure 16: u velocity @ t=5

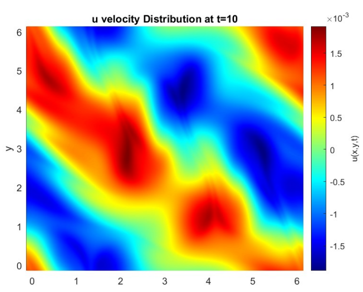

Figure 17: u velocity @ t=10

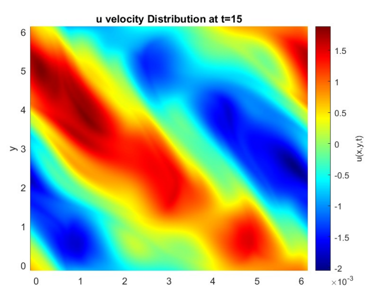

Figure 18: u velocity @ t=15

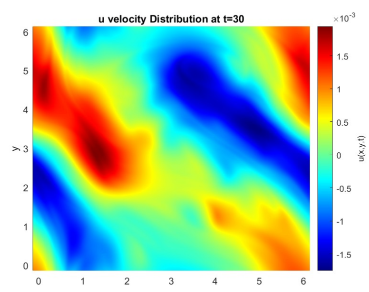

Figure 19: u velocity @ t=30

Likewise, the v velocity distribution at time=0, time=1, time=5, time=10, time=15 and time=30 are attached here. The v-velocity also has similar outcomes as that of all other flow parameters mentioned above.

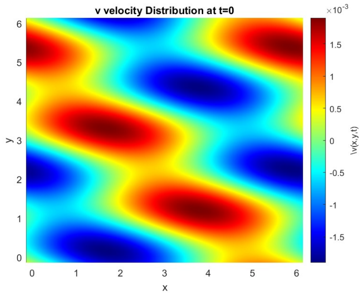

Figure 20: v velocity @ t=0

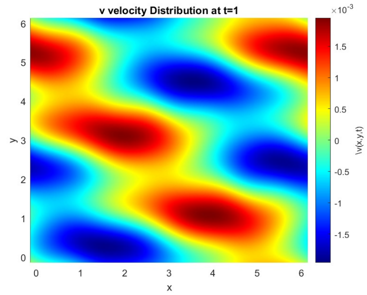

Figure 21: v velocity @ t=1

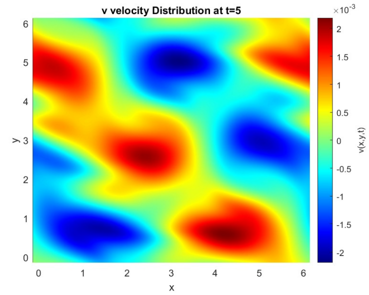

Figure 22: v velocity @ t=5

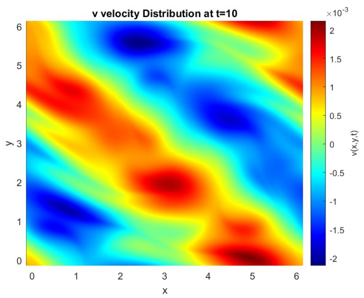

Figure 23: v velocity @ t=10

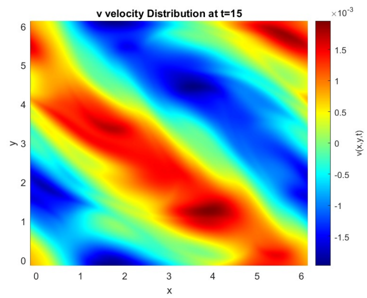

Figure 24: v velocity @ t=15

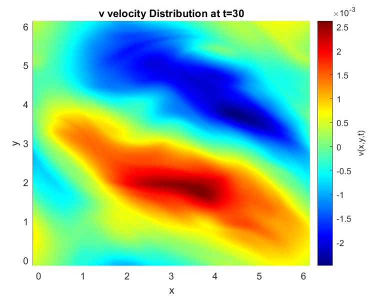

Figure 25: v velocity @ t=30

4 Discussion and Conclusion

The exact analytical solution for the problem solved here, with random number vorticity fourier component initial conditions couldn't be found. Nevertheless, the grid convergence for the problem is attached here.

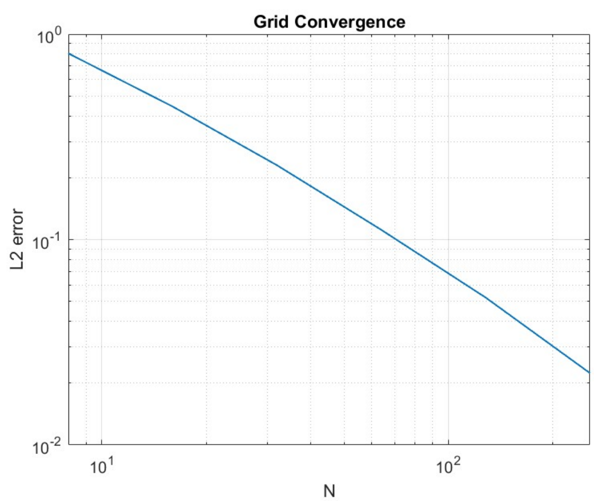

Figure 26: Grid Convergence

The L2 error shows the solution is converged with increasing N. The value of Nx=Ny=N=1024 has been used for running all the simulations at different final times in this project.

Since the initial conditions in [1] involved random gaussian numbers for $\hat{\psi}$, the two results couldn't be quantitatively compared. However, since the modality of the problem is similar, we can sure perform a qualitative comparison. In [1], the results of vorticity distribution shows that with time moving forward, the vorticity is concentrated within a confined portion of the spatial domain in the flow structure, even though there is no vorticity at the beginning of the simulation. This occurs due to homogeneous, isotropic and high reynold's number turbulence. The property of the vorticity parameter and it's evolution with time seems to be the same as that observed in this project, concentrated to a confined region in the domain, having an axisymmetric form and enduring the advection by larger scales in the flow.

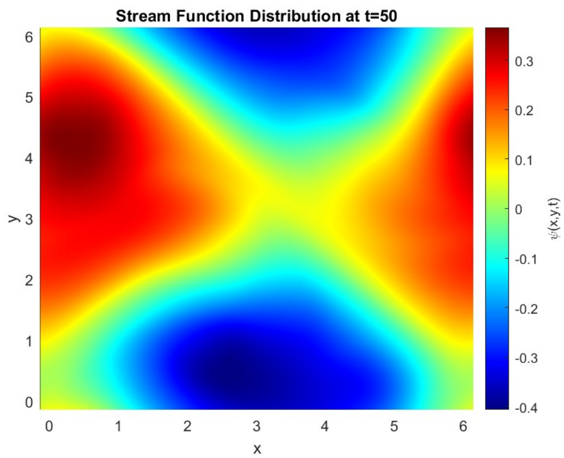

Figure 27: Streamfunction @ t=30

Figure 28: Streamfunction @ t= 16.5 in [1]

The streamfunction comparison from the project and the mentioned paper (two different times) are attached in Figure 27 and Figure 28. We can see that, eventually, in both the cases, parameters start taking the same form with decaying turbulence. Turbulence Kinetic Energy and Turbulence Dissipation Rate calculations would probably give us better comparisons, but those parameters have become out of the scope of the project with limited time.

References

[1] J. C. McWilliams, “The emergence of isolated coherent vortices in turbulent flow,” *Journal of Fluid Mechanics*, pp. 21–43, 1984.

[2] J. Wang, J. Sesterhenn, and W.-C. Müller, “Coherent structure detection and the inverse cascade mechanism in two-dimensional navier–stokes turbulence,” *Journal of Fluid Mechanics*, vol. 963, May 2023, ISSN: 1469-7645. DOI: `10.1017/jfm.2023.313`. [Online]. Available: `http://dx.doi.org/10.1017/jfm.2023.313`.

[3] C. Canuto, *Spectral Methods in Fluid Dynamics*. Springer Nature, 198.

Appendices

Code:

```
clc;clear;

%% With Nx=Ny=1024
Nx = 1024;
Ny = 1024;
lx = pi;
ly = pi;
x=(0:Nx-1)'*(2*lx/Nx);
y=(0:Ny-1)'*(2*ly/Ny);
kx = [Nx/2:Nx 1-Nx:(-Nx/2-1)]'*pi/lx;
ky = [Ny/2:Ny 1-Ny:(-Ny/2-1)]'*pi/ly;
%3/2 dealising
dea_x = (Nx/4+2:Nx/4*3);
kx(dea_x) = 0;                    %
dea_y = (Ny/4+2:Ny/4*3);
ky(dea_y) = 0;
%parameters
t= 1;
t_final = 5;
dt = 0.1;
nu = 1E-4;
%initial condition for vorticity
seed = RandStream('dsfmt19937','Seed',5);       %make the same random numbers each time
RandStream.setGlobalStream(seed);
w_cap = zeros(Nx, Ny);
%Jiahan Wang used these values for these co-ordinates and they work, all zeros do not
w_cap(2,2) = randn(1) + 1i*randn(1);
w_cap(1,5) = randn(1) + 1i*randn(1);
w_cap(4,1) = randn(1) + 1i*randn(1);
w = real(ifft2(w_cap));
w_act = w/(max(w);

figure(1)
[X,Y] = meshgrid(x,y);
surf(X, Y, w_act);
view([0 90]);
colormap("jet")
shading flat;
cc = colorbar;
title ('Vorticity Distribution: N=1024');
xlabel('x');
xlim([0 2*lx]);
ylabel('y');
ylim([0 2*ly]);
xlabel(cc, '\omega(x,y,t)');


%%with varying Nx and Ny values
```

```
Nx = 256;
Ny = 256;
lx = pi;
ly = pi;
x=(0:Nx-1)'*(2*lx/Nx);
y=(0:Ny-1)'*(2*ly/Ny);
kx = [Nx/2:Nx 1-Nx:(-Nx/2-1)]'*pi/lx;
ky = [Ny/2:Ny 1-Ny:(-Ny/2-1)]'*pi/ly;
%3/2 dealising
dea_x = (Nx/4+2:Nx/4*3);
kx(dea_x) = 0;
dea_y = (Ny/4+2:Ny/4*3);
ky(dea_y) = 0;
%parameters
t= 1;
t_final = 5;
dt = 0.1;
nu = 1E-4;
%initial condition for vorticity
seed = RandStream('dsfmt19937','Seed',5);        %make the same random numbers each time
RandStream.setGlobalStream(seed);
w_cap = zeros(Nx, Ny);
%Jiahan Wang used these values for these co-ordinates and they work, all zeros do not
w_cap(2,2) = randn(1) + 1i*randn(1);
w_cap(1,5) = randn(1) + 1i*randn(1);
w_cap(4,1) = randn(1) + 1i*randn(1);
w = real(ifft2(w_cap));
%Normalizing to order one
w = w/max(w);

figure(2)
[X,Y] = meshgrid(x,y);
surf(X, Y, w);
view([0 90]);
colormap("jet")
shading flat;
cc = colorbar;
title ('Initial Vorticity Distribution');
xlabel('x');
xlim([0 2*lx]);
ylabel('y');
ylim([0 2*ly]);
xlabel(cc, '\omega(x,y,t)');
%% Grid Convergence

%L2 convergence
Ltwoall=0;
 for j= 1:Nx
     for p=1:Ny
     Ltwoall=Ltwoall+(abs(w(j,p)-w_act(j*1024/Nx,p*1024/Ny)).^2);
     end
end
```

```
Ltwo1=(Ltwoall.*(1/(Nxy.*Ny))).^(1/2);
%plotting the error
N=[ 8, 16, 32,64, 128, 256];
Ltwo1= [0.80566,0.44402,0.2299,0.11146,0.05218,0.02237];
figure(3)
loglog(N,Ltwo1,LineWidth=1)
grid on
xlabel('N')
ylabel('L2 error')
title('Grid Convergence')

%%vorticity calculation
%calculating omega
vid=VideoWriter('video','MPEG-4');
open(vid);
while (t < t_final)
    w_vec = ode113(@(w) RightHand(w, Nx, Ny, kx, ky), 0:dt:dt, w);  %RHS solves the right hand side of the equation
    w=w_vec;
    t = t + dt;
    figure(4)
    [X,Y] = meshgrid(x,y);
    surf(X, Y, w);
    view([0 90]);
    colormap("jet")
    shading flat;
    cc = colorbar;
    title ('Vorticity Distribution at t=');
    xlabel('x');
    xlim([0 2*lx]);
    ylabel('y');
    ylim([0 2*ly]);
    xlabel(cc, '\omega(x,y,t)');
    F=getframe(figure(4));
    writeVideo(vid, F);
end
close(vid);
%%streamfunction calculation
%calculating streamfunction by solving poisson equation
%in x-direction
 for p=1:Nx
     Q_x=fft(-w);
 end
 Q_x=Q_x/Nx;
 for i =1:Nx
    Q_x_shifted(1:Nx,i)=fftshift(Q_x(1:Nx,i));
 end
%in y-direction
 for m=1:Ny
         Q_y(m,:)=fft(Q_x_shifted(m,:));
 end
 Q_xy=Q_y/Ny;
 for i =1:Ny
   Q_xy_shifted(i,1:Ny)=fftshift(Q_xy(i,1:Ny));
```

```
 end
%phi_cap(m,n)
 for  m=1:Nx
     for n=1:Ny
         k(m)=(2.*pi.*(m-(Nx/2+1)))./(2*lx);
         l(n)=(2.*pi.*(n-(Ny/2+1)))./(2*ly);
         phi_cap(m,n)=-Q_xy_shifted(m,n)./((k(m).^2)+(l(n).^2));
     end
 end
 phi_cap(Nx/2+1,Ny/2+1)=0;
 for i =1:Ny
     phi_cap(i,1:Ny)=ifftshift(phi_cap(i,1:Ny));
 end
%inverse 2D FFT for phi(j,p)
%inverse in y dir:
for m=1:Ny
            phi_y(m,1:Ny)=(ifft(phi_cap(m,1:Ny).*Ny));
end
for i= 1:Ny
       phi_y(1:Ny,i)=fftshift(phi_y(1:Ny,i));
end
%in x-direction
 for p=1:Ny
     phi_x=real((ifft(phi_y.*Ny)));
 end
%plotting the streamfunction
figure(5)
[X,Y] = meshgrid(x,y);
surf(X, Y, phi_x);
view([0 90]);
shading flat;
colormap("jet")
cc = colorbar;
title ('Stream Function Distribution at t=');
xlabel('x');
xlim([0 2*lx]);
ylabel('y');
ylim([0 2*ly]);
xlabel(cc, '\psi(x,y,t)');

%%velocity reconstruction
%calculating and plotting the u-velocity
[u,v] = gradient(phi_x) ;
figure(6)
surf(X, Y, v);
view([0 90]);
shading flat;
colormap("jet")
cc = colorbar;
title ('u velocity Distribution at t=');
xlabel('x');
xlim([0 2*lx]);
ylabel('y');
ylim([0 2*ly]);
```

```
xlabel(cc, 'u(x,y,t)');

%calculating and plotting the v-velocity
[u,v] = gradient(phi_x) ;
figure(7)
surf(X, Y, v);
view([0 90]);
shading flat;
colormap("jet")
cc = colorbar;
title ('v velocity Distribution at t=');
xlabel('x');
xlim([0 2*lx]);
ylabel('y');
ylim([0 2*ly]);
xlabel(cc, 'v(x,y,t)');

%%RHS function
%function to calculate the RHS of vorticity transport equation
function rhs = RightHand (w, Nx, Ny, kx, ky)
    nu = 1E-4;
    [Kx, Ky] = meshgrid(kx,ky);
    Ksq = Kx.^2 + Ky.^2;
    %avoiding 1/0 of Ksq
    Ksqinv = zeros(size(Ksq));
    if Ksq ~= 0
        Ksqinv=1./Ksq;
    else
        Ksqinv=0;
    end
    w_cap = fft2(w);
    %inverse of the all terms in convo sum
    %first term rhs
    viscousterm=real(ifft2(-nu*Ksq.*w_cap));
    %second term rhs
    u_cap = real(ifft2(1i*Ky.*Ksqinv.*w_cap));
    wx_cap = real(ifft2(1i*Kx.*w_cap));
    %third term rhs
    v_cap = real(ifft2(-1i*Kx.*Ksqinv.*w_cap));
    wy_cap = real(ifft2(1i*Ky.*w_cap));

    %directly calculate w and not w_cap
    rhs = viscousterm - u_cap.*wx_cap - v_cap.*wy_cap;
end
```