Final project PHYS220:

Vorticity in Atmospheric Science

*Team1: Zeyu Wang, Indaysia Brent, Jailyn Dorsett*

*Division of Physical and Computational Sciences，Delaware State University, Dover, DE 19901, USA*

Part (2)

In order to include matrix B and C, we begin by constructing the operating matrix. As we stacked the rows of each matrix up to the 16th row. From there we were able to create the diagonal elements of the sparse matrix for both B and C. With the B=spdiags([e5 -1\*e2 e0 e3 -1\*e4],[-(m-1) -1 0 1 m-1],n,n); and C=spdiags([e1 -1\*e1 e0 e1 -e1],[-(n-m) -m 0 m n-m],n,n);. Thus, we were able to implement matrix B for (∂u/∂x=1/△x\*B\*u) and matrix C for (∂u/∂y=1/△x\*C\*u).

e0=0000000000000000

e1=1111111111111111

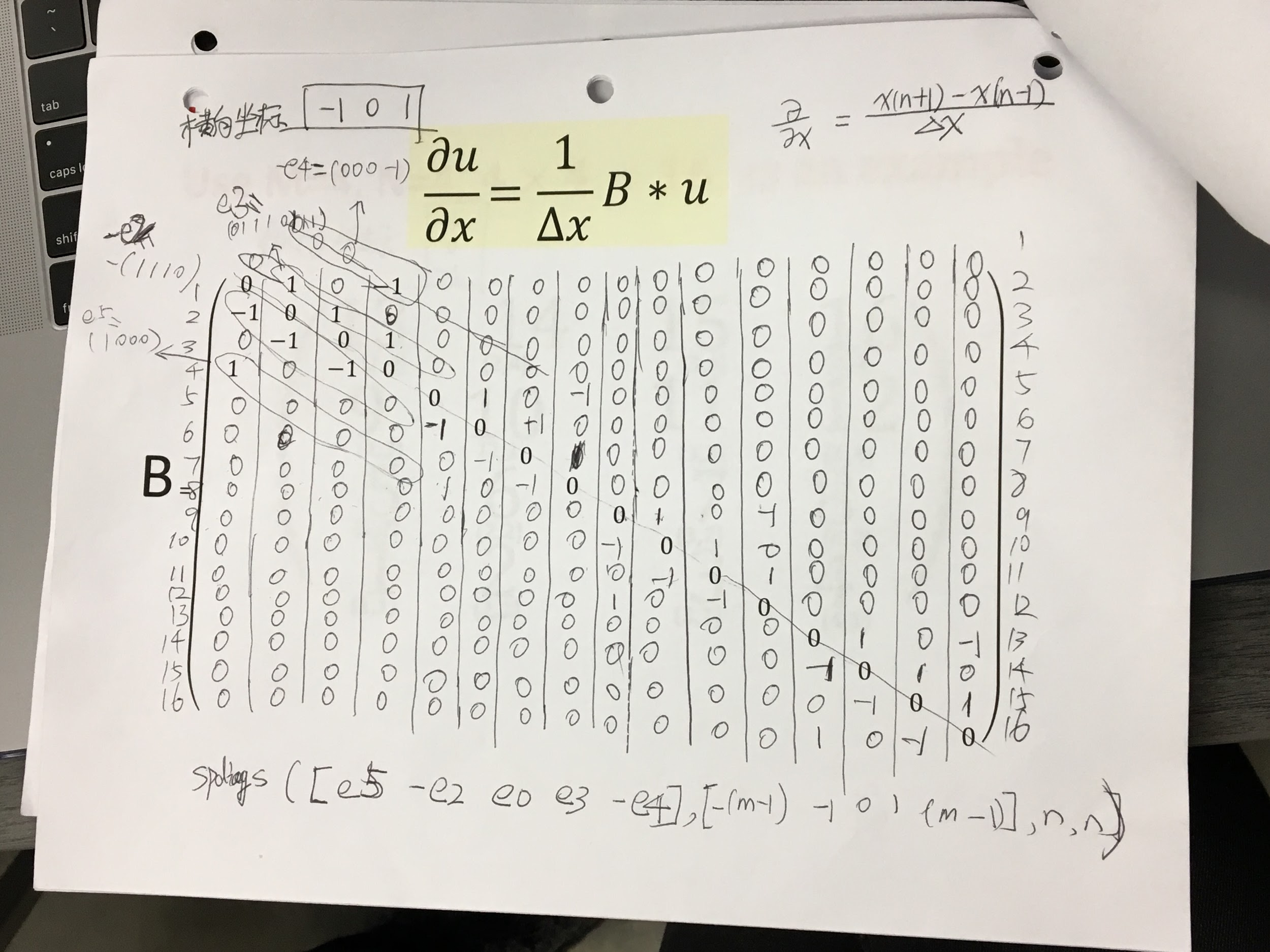
e2=1110111011101110

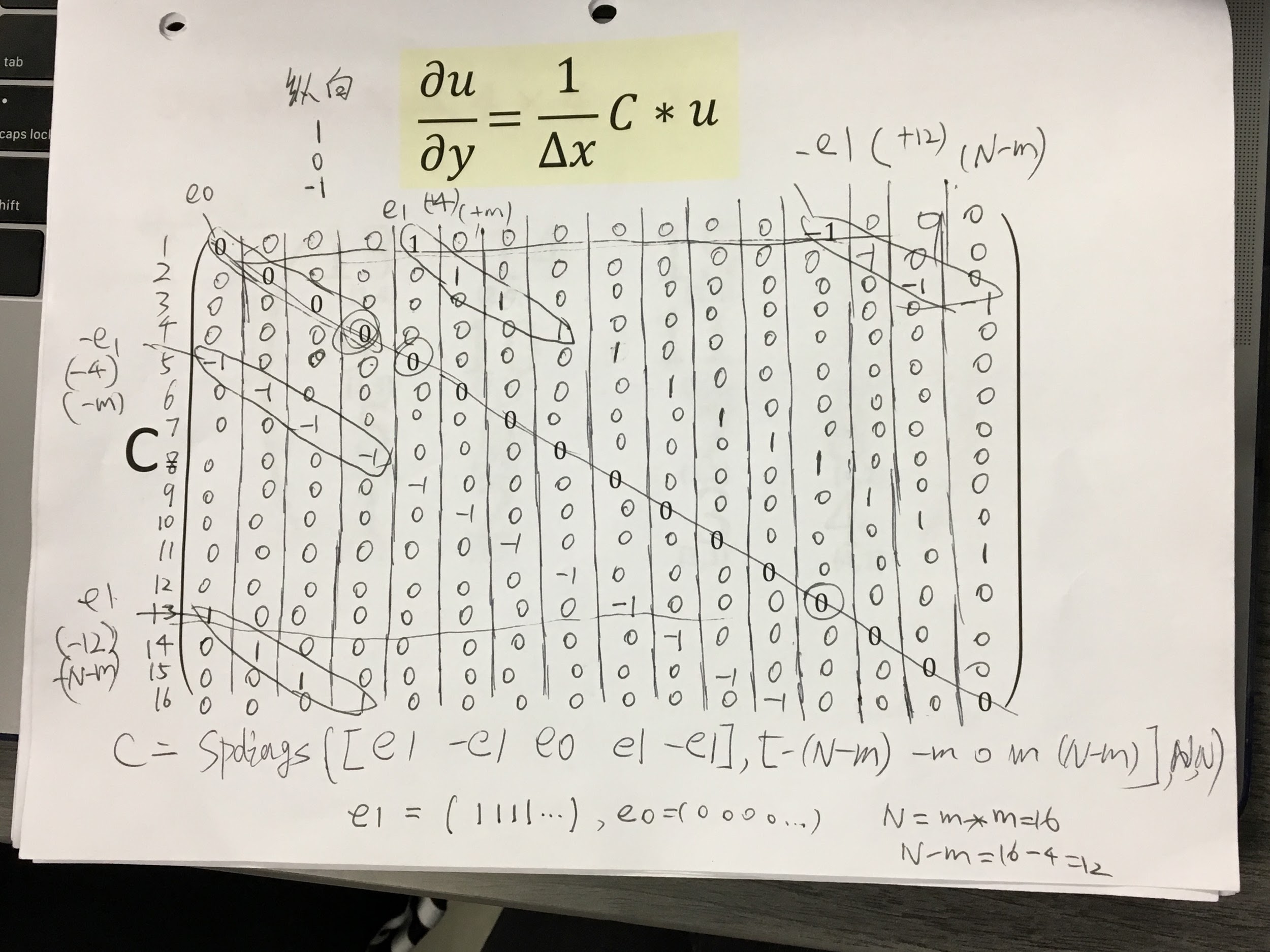
e3=0111011101110111

e4=0001000100010001

e5=1000100010001000

Below are the images of the mapping of Matrix B and Matrix C:





Part (3)

Following our steps in implementing matrix A from the previous class. The group begins by creating the Two Dimensional Diffusion Equation for the new matrix B and C with the new initial and periodic boundary conditions. From there, we were ready to start implementing our code into Matlab. Thus, we followed matrix A that we had previously worked on in class. Where we cleared all variables and figures from Matlab, ensuring that we were starting a clean new script. Next, we named values starting with the N values in both the x and y directions, the total size of the matrix, the vectors of zeros, vectors of ones and stated the related values of e0, e1, e2, and e4 of the spdiags command. Since we needed to explain the values that would be entered in the code and how they all related. Furthermore, we wrote the m^th value with zero and then another one with one. Then, we inserted the shifts that would allow us to correctly build the sparse matrices with the positions listed. Proceeding, we began to structure the matrix of A, B, and C. Using the methods listed in step 2, that allowed us to properly place the elements. Once that was complete, we were able to call the time range of the matrix. Follow by the length/dimension of the matrix. As we proceeded to use the code from class as a guide to complete the overall code. Including the newly wave function of initial conditions. With the values of time step (dx) from the function. Then, we ensured that we made matrix A1 nonsingular to satisfy the matrix structure. Right after, we called the Ordinary Differential Equation(ODE45), with the function, time range and the variables of vorticity. Thereafter, we were able to create the plot.

Part (4)

From completing this project, the group has more of an understanding of complex coding. Instead of dealing with the most basic of functions, we were tasked with working with three matrices and had to show their animations. In this project, we learned how to use 2D/3D PDE equation to solve the Physics Model problem which like Atmospheric Vorticity. We learned how to make a sparse matrix by using the spdiags command and placing the diagonal elements. In addition, we can view the matrix structure by using spy(matrix name) command and we observed the fast direction of expansion by changing the initial condition in x, y or z directions. After that, we also reviewed how to design a loop to overwrite a desired value which satisfied the matrix structure.

Moreover, we develop the plot quality by using a set of functional commands which like linspace, meshgrid, reshape, pcolor and shading etc. The code we had to use held three different conditions of diagonal lines which had to be found before even making the code. We learned how to utilize the elliptic problem to find the stream function by time stepping. It was a pretty straightforward project that was easy to solve after figuring out how to do the mapping out and planning of it.

Part (5)

function fdot=sdf(tspan,ww,dummy,dx,nu,A1,B,C) %define the function

w1=(dx^2)\*ww; %define the vorticity

f=A1\w1; %define the flux

fdot=(nu/dx^2)\*A1\*ww-(1/dx^2)\*(B\*f).\*(C\*ww)+(1/dx^2)\*(C\*f).\*(B\*ww); %define the function

close all;clear all;clc; %clear all variables and figures

m=100; %n value in x and y directions

n=m\*m; %total size of matrix

e0=zeros(n,1); %vector of zeros

e1=ones(n,1); %vector of ones

e2=e1; %copy the one vector

e4=e0; %copy the zero vector

for j=1:m

e2(j\*m)=0; %overwrite every m^th value with zero

e4(m\*j)=1; %overwrite every m^th value with one

end

e3(2:n)=e2(1:n-1,1);e3(1)=e2(n);e3=e3'; %shift to correct Build the sparse matrix A.

e5(2:n)=e4(1:n-1,1);e5(1)=e4(n);e5=e5'; %positions

%build sprase matrix A,B,C

%place diagonal elments

A1=spdiags([e1 e1 e5 e2 -4\*e1 e3 e4 e1 e1],[-(n-m) -m -(m-1) -1 0 1 m-1 m n-m],n,n);

% spy(A1) %view the matrix structure

%place diagonal elements

B=spdiags([e5 -1\*e2 e0 e3 -1\*e4],[-(m-1) -1 0 1 m-1],n,n);

%spy(B) %view the matrix structure

%place diagonal elements

C=spdiags([e1 -1\*e1 e0 e1 -e1],[-(n-m) -m 0 m n-m],n,n);

%spy(C) %view the matrix structure

%Now,we generate the desired initaial condition vector

Time=8;

L=20;

nu=0.001;

nx=m;

x2=linspace(-L/2,L/2,nx+1);

x=x2(1:nx);

ny=m;

y2=linspace(-L/2,L/2,ny+1);

y=y2(1:ny);

[X, Y]=meshgrid(x,y);

W=exp(-2\*X.^2-Y.^2/20);

w=reshape(W,n,1);

% w1=(dx^2)\*w;

dx=x(2)-x(1); %time stepping

A1(1,1)=2; % Make martix A1 nonsingular

% f=A1\w1;

[t,y]=ode45('sdf',[0 Time],w,[],dx,nu,A1,B,C); %solve the ODE function

%Plot

for i=1:250:length(t),

u=reshape(y(i,:),nx,ny);

figure;

pcolor(u); %2D top view

shading('interp')

drawnow

end

% %Animation

% for j=1:1:length(t)

% u=reshape(y(j,:),nx,ny);

% pcolor(u); %2D top view

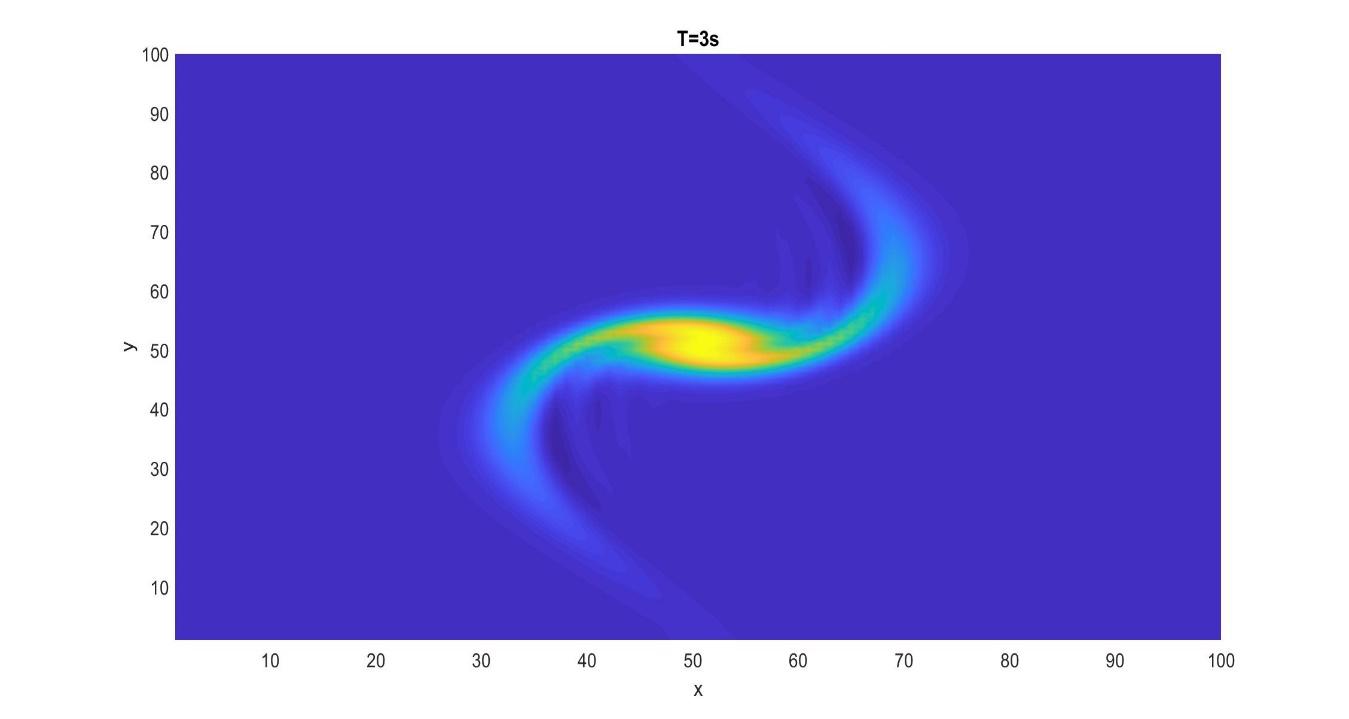
% % surf(u); %3D side view

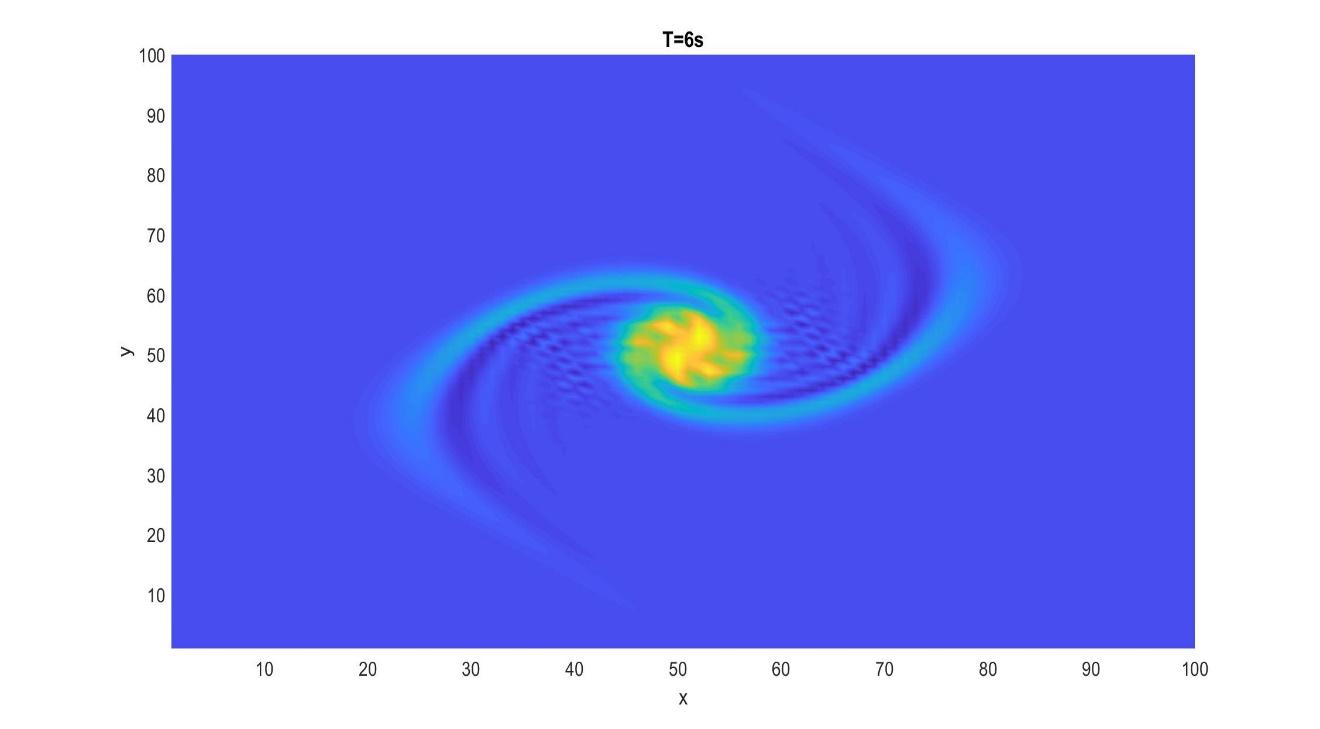
% shading('interp');

% % axis([0 m 0 m 0 1]); %use with surf

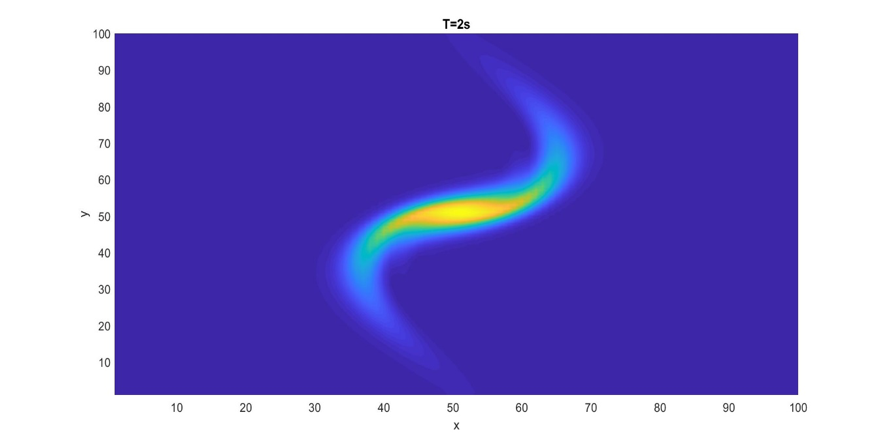
% drawnow

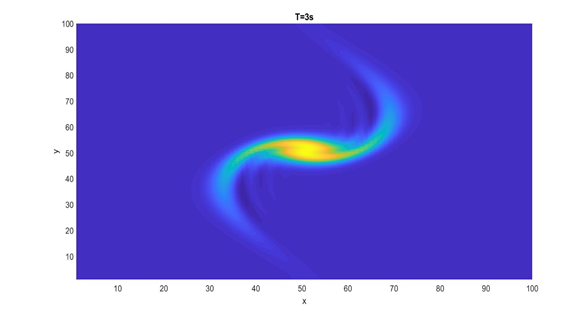
% end

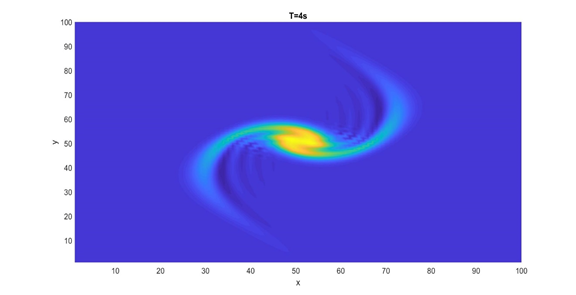


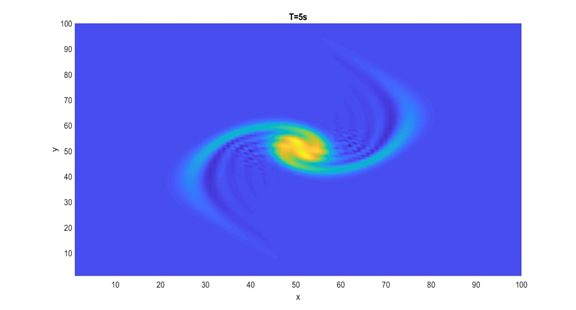


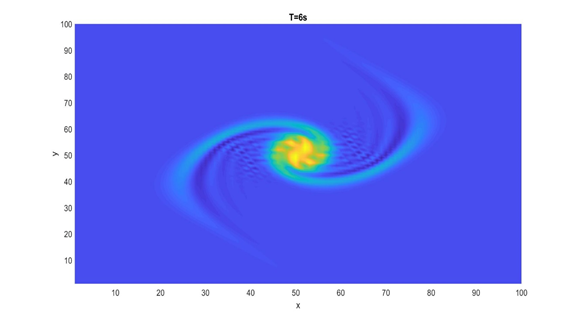


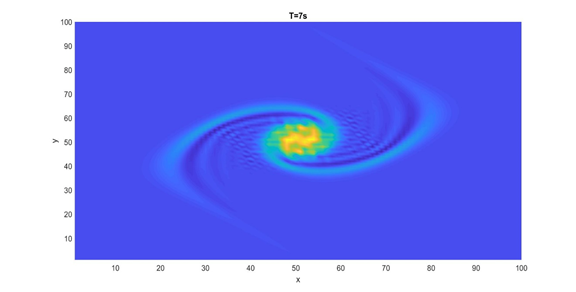


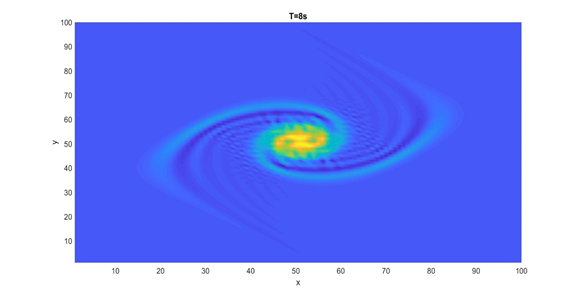












Part (6)

In lecture 8, we learned how to accomplish the 3D animation effects by using the *surf(u)* and *axis([0 m 0 m 0 1])* together, thus we can try it for this project. For the animation part, we will show the top view, bottom view and side view, etc.

for j=1:1:length(t)

u=reshape(y(j,:),nx,ny);

pcolor(u); %2D top view

surf(u); %3D side view

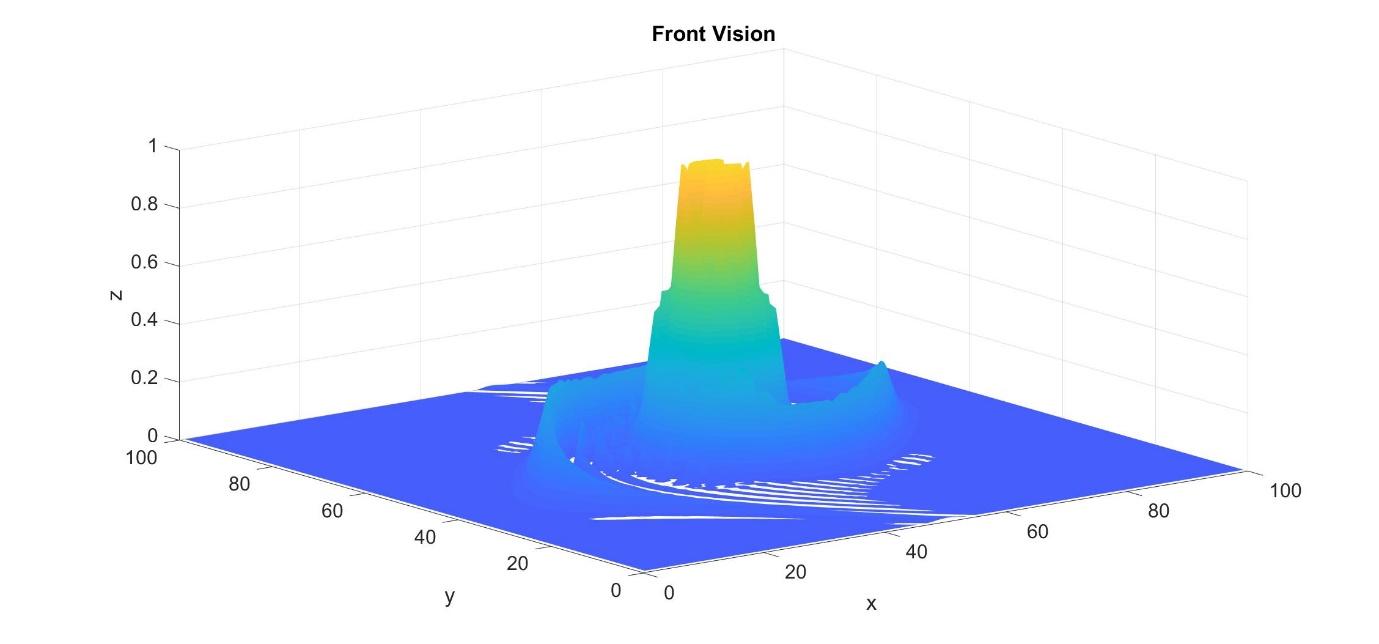
shading('interp');

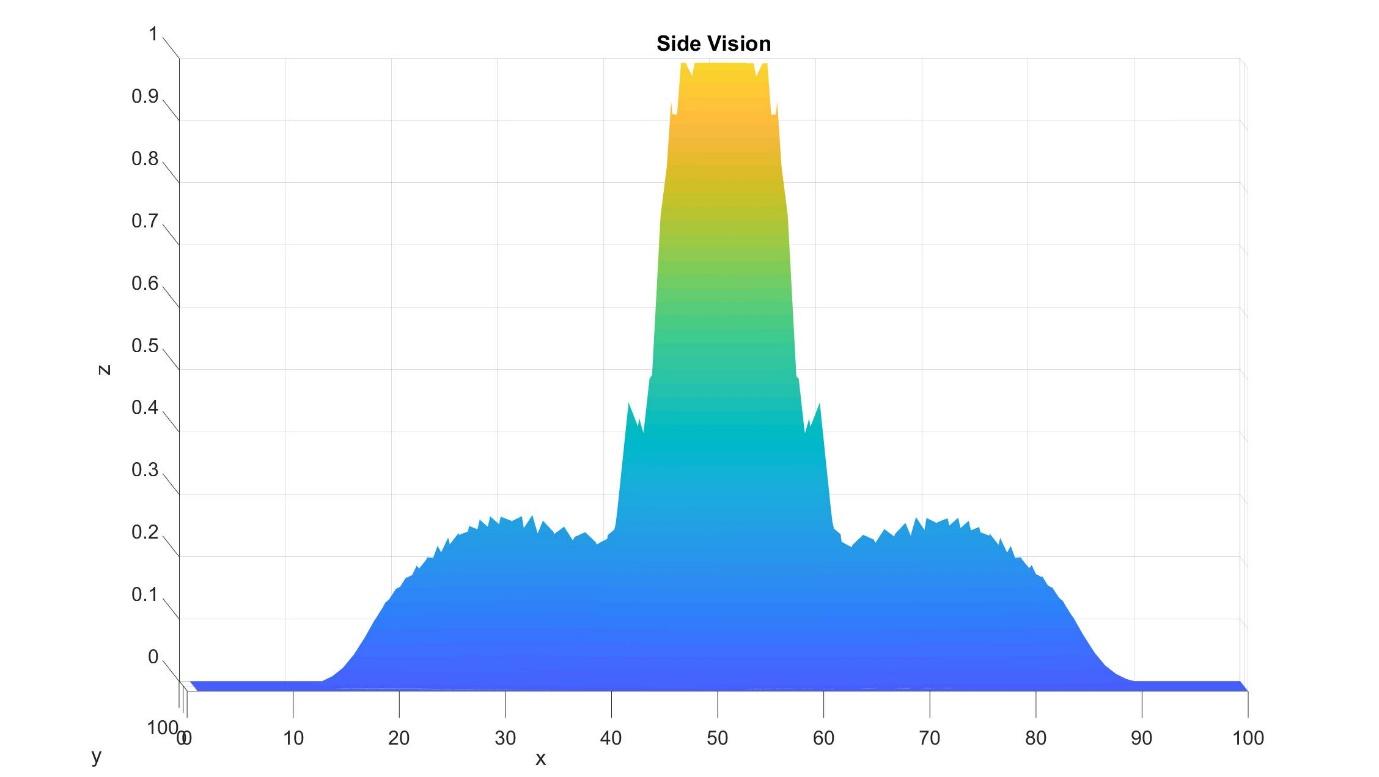
axis([0 m 0 m 0 1]); %use with surf

drawnow

end

The results are shown below:





图片包含 名片, 文字

描述已自动生成

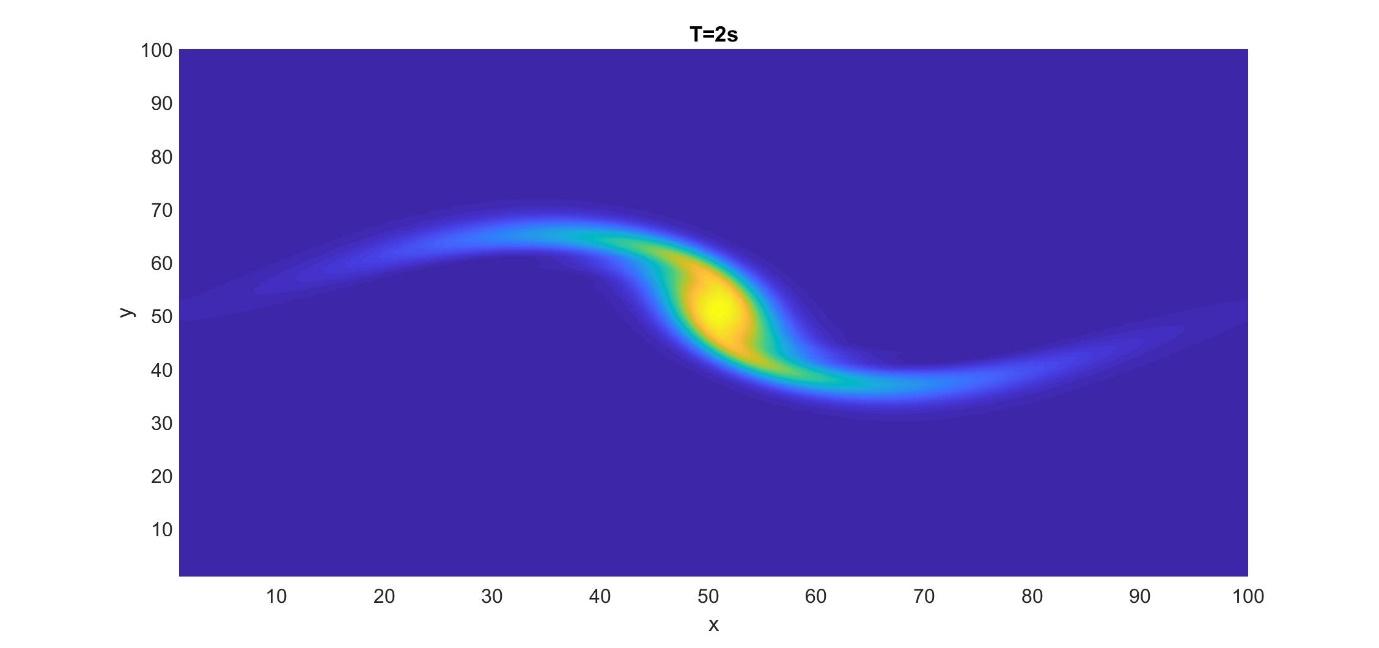
图片包含 名片, 文字

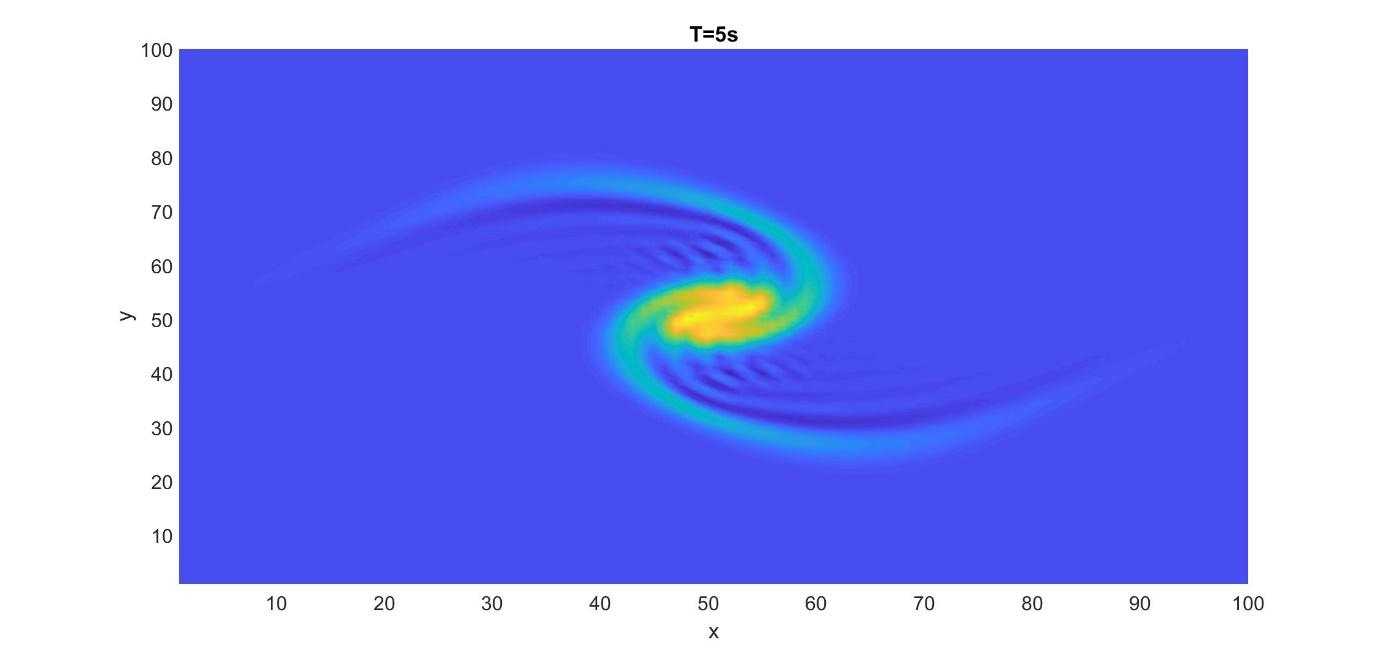
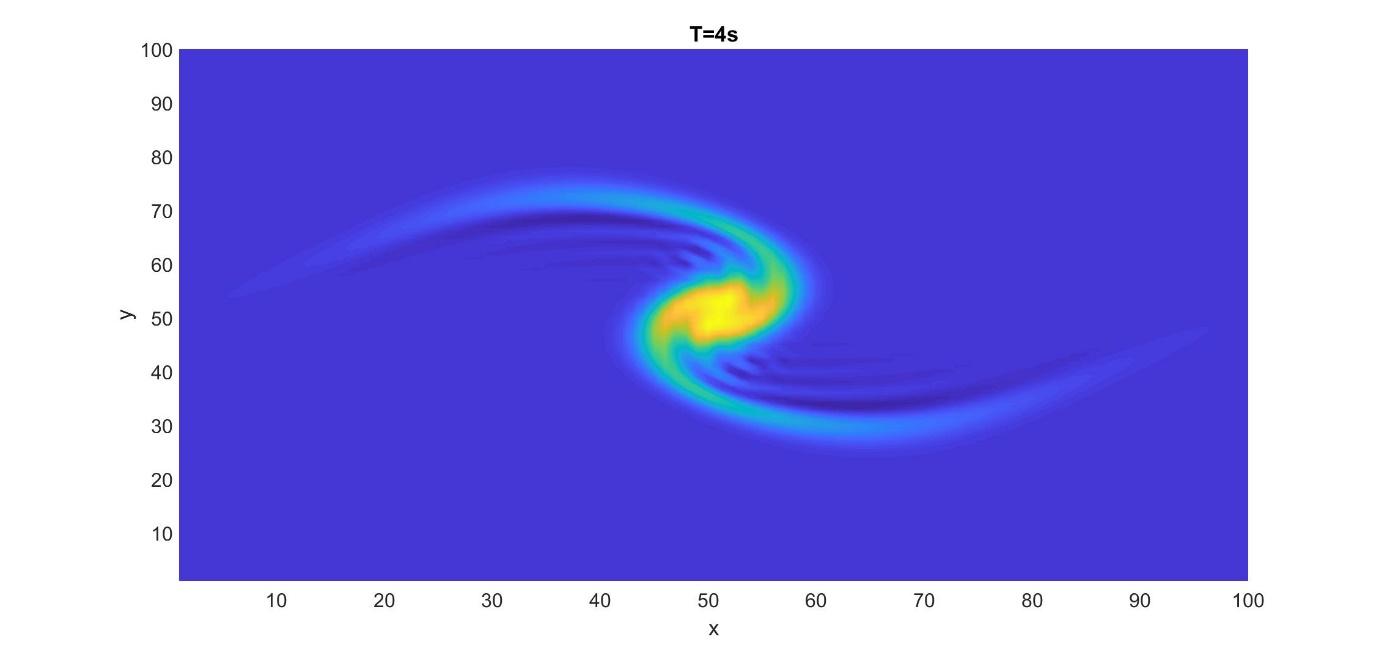
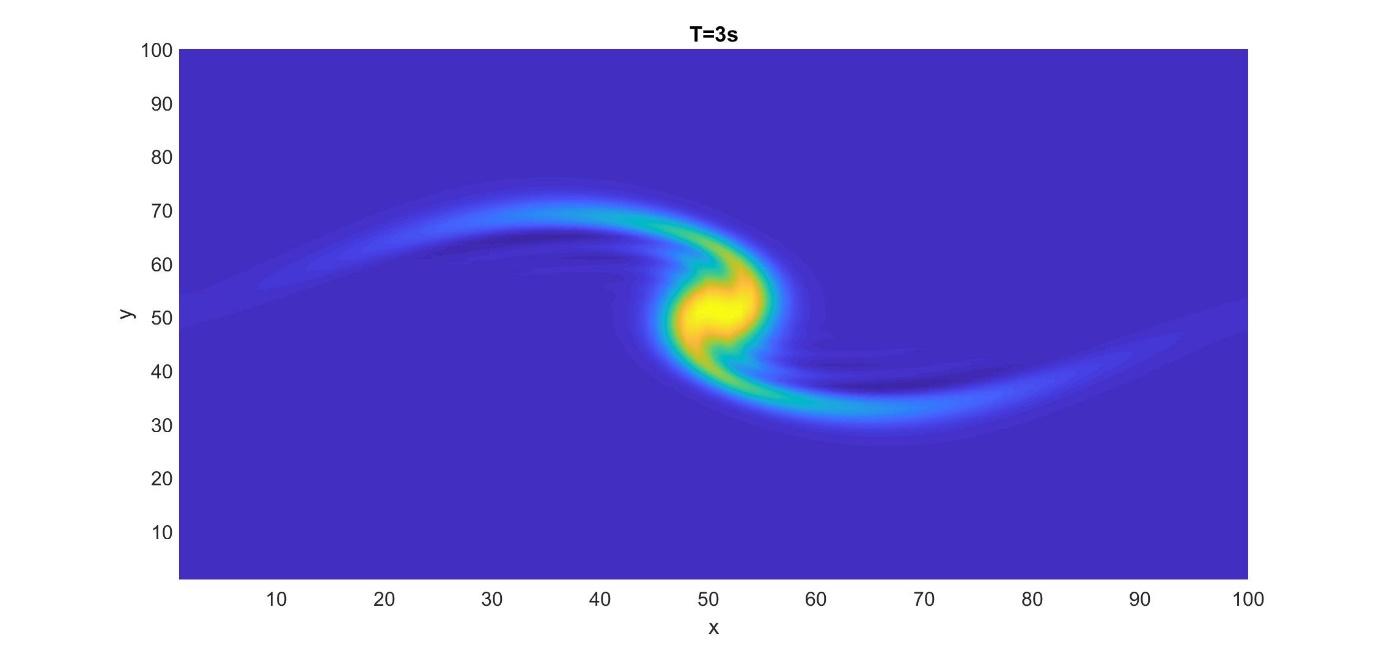
描述已自动生成

After that, we can observe the direction of fast expansion by changing the initial condition from W=exp(-2\*X.^2-Y.^2/20) to W=exp(-X.^2/20-Y.^2\*2). Based on , it’s clear to see the direction of growth is vertical which opposite to the previous one.

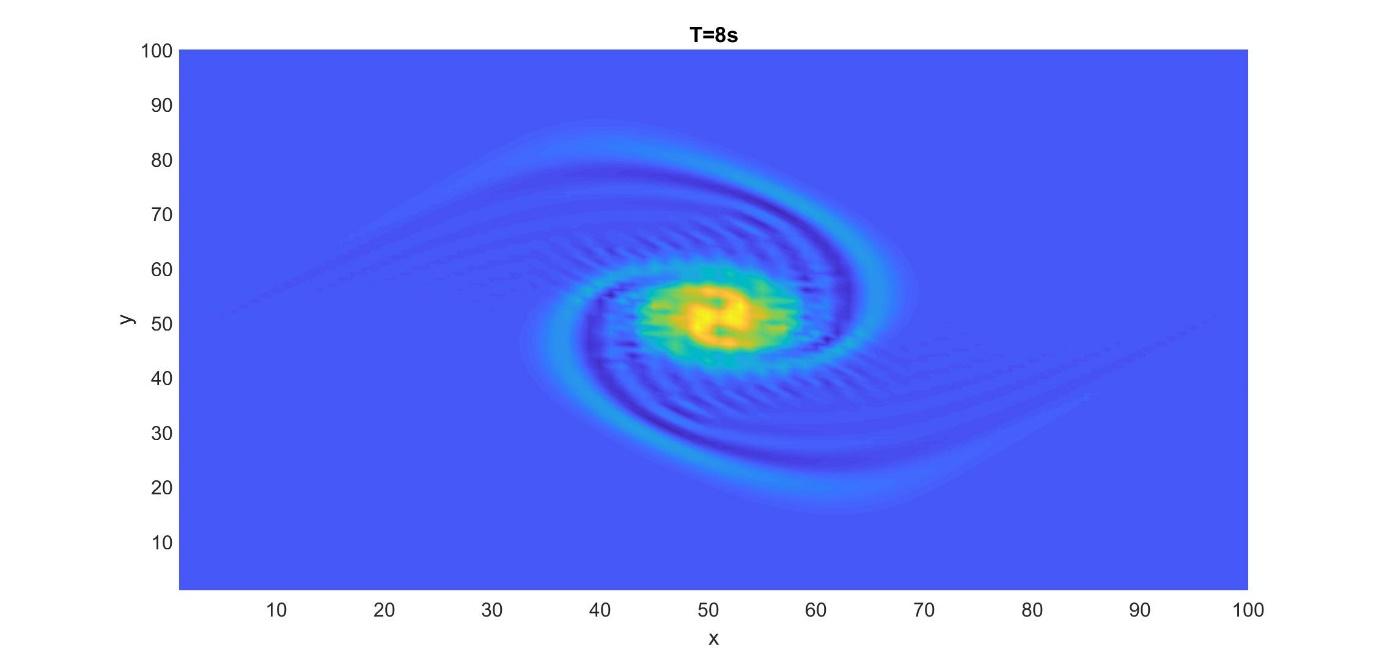
图片包含 屏幕截图

描述已自动生成



图片包含 动物

描述已自动生成图片包含 动物

描述已自动生成

3-D Animation:

