ASSIGNMENT-1

DESAI MANISH **21101022**

Question 2

Loading the data

```
clc
clear all
close all
load PIVdata.dat
d = PIVdata;
```

converting the data into 3D matrix: 120x120 grids (400 screenshots in each layer)

```
u = permute(reshape(d(:,3),120,120,400),[2 1 3]);
v = permute(reshape(d(:,4),120,120,400),[2 1 3]);
x = permute(reshape(d(:,1),120,120,400),[2 1 3]);
y = permute(reshape(d(:,2),120,120,400),[2 1 3]);
```

finding the embedded mean

```
mu = mean(u , 3);
mv = mean(v,3);
```

finding the fluctuations in u and v velocities

```
for i  = 1:400
    varu(:,:,i) = u(:,:,i)-mu;
    varv(:,:,i) = v(:,:,i)-mv;
end
```

finding the reynolds stress at x = 15.3 mm location

```
for i = 1:120
    r11 = 0;
    r12 = 0;
    r22 = 0;
    for k = 1:400
        r11 = varu(i,50,k)*varu(i,50,k)+r11;
        r22 = varv(i,50,k)*varv(i,50,k)+r22;
        r12 = varu(i,50,k)*varv(i,50,k)+r12;
    end
```

```
    stress11(i)= r11/400;
    stress12(i)= r12/400;
    stress22(i)= r22/400;

end
```

plotting the stresses:

```
plot(flip(stress11),'r')
hold on
plot(flip(stress12),'b')
hold on
plot(flip(stress22),'g')
```

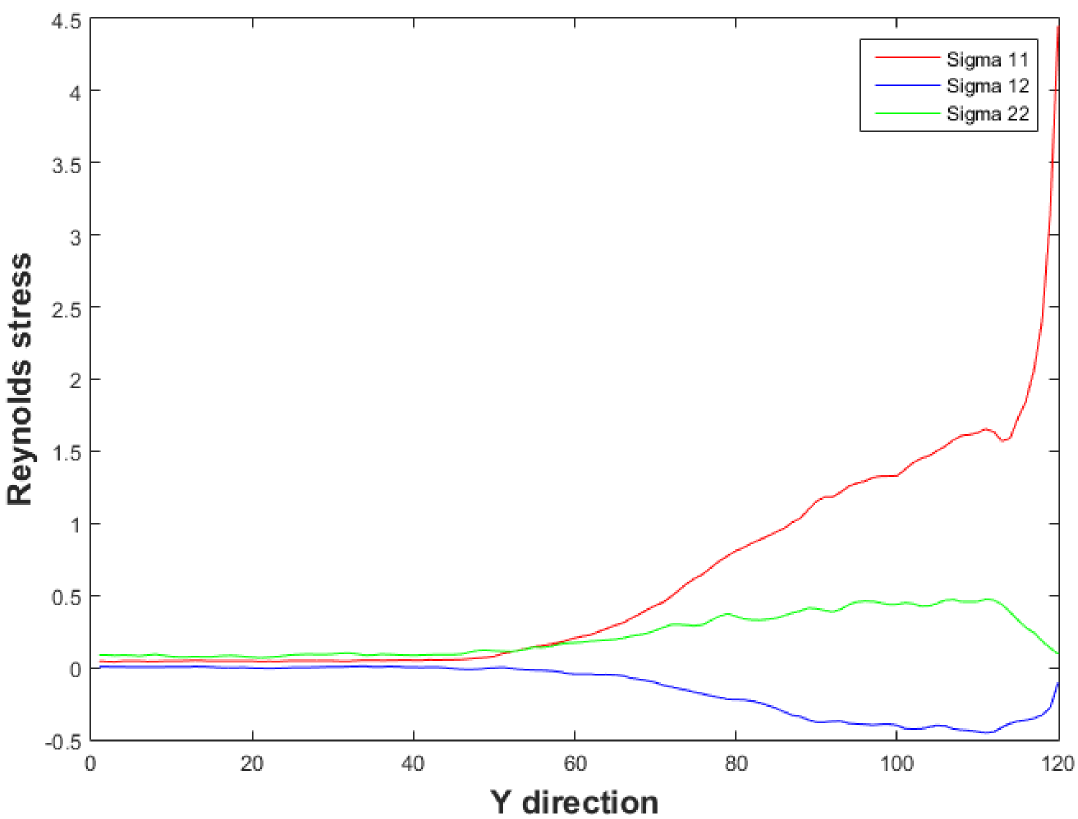

Fig1:Reynolds stress plot

finding correlation considering x0 ≈ 15.3 mm and y0 ≈ 3.9 mm

```
ms = 0
for i = 1:400
    ms = varu(104,50,i)*varu(104,50,i)+ms;
end
rms = ms/400;

for i = 1:120
    for j = 1 :120
        r = 0;
        for k = 1:400
            r = varu(i,j,k)*varu(104,50,k)+r;
        end
        R(i,j) = r/(rms*400);

    end
end
%%

contourf(flipud(R),300,'LineColor','non')
colorbar
xlabel('X direction','FontSize', 15,'fontweight',
'bold' ...
    );
ylabel('Y direction','FontSize',15,'fontweight',
'bold') ;
```

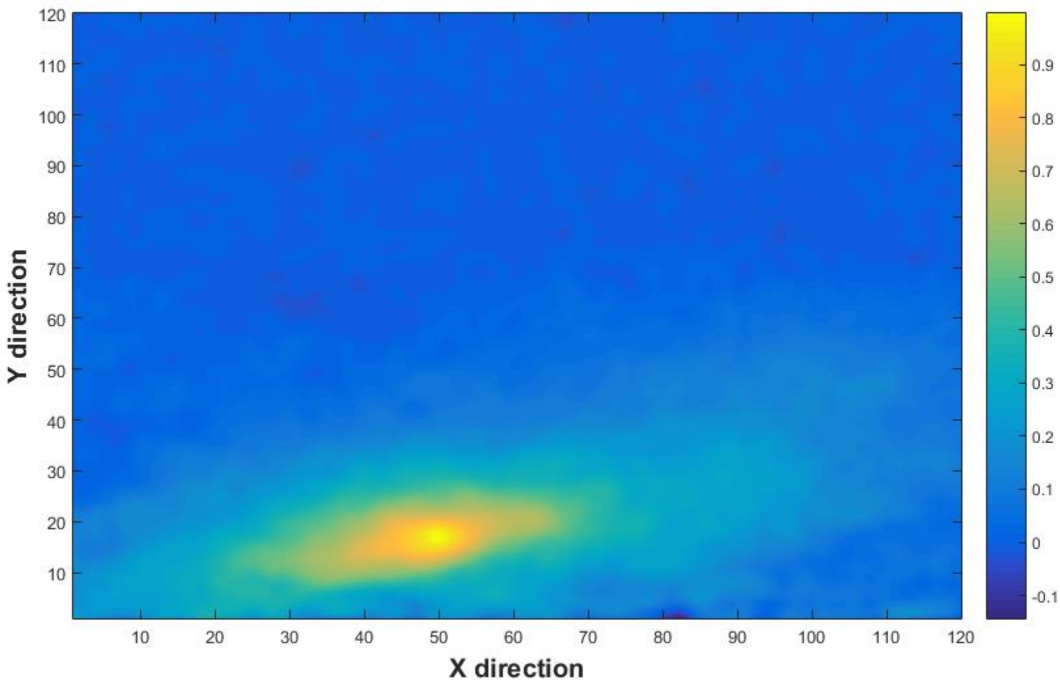

Fig2: Correlation function plot

```
Longitudinal correlation
loc = 104
xmat = [y1(loc,1)^2, y1(loc,1), 1 ;y1(loc+1,1)^2,
y1(loc+1,1), 1 ; y1(loc-1,1)^2, y1(loc-1,1), 1]
r1mat = [Rl(loc);Rl(loc+1);Rl(loc-1) ]

r1con = inv(xmat) * r1mat
r1x = y1(loc-7,1):-0.001 :y1(loc+7,1)
r1tms = r1con(1) * r1x.^2 + r1con(2) * r1x + r1con(3)
plot(r1x,r1tms)
hold on


p4 = plot(y1(:,50),Rl,'LineWidth',3)
xlabel('x direction','FontSize', 15,'f
ontweight', 'bold' ...
    );
ylabel('Longitudnal
corelation','FontSize',15,'fontweight', 'bold') ;
```

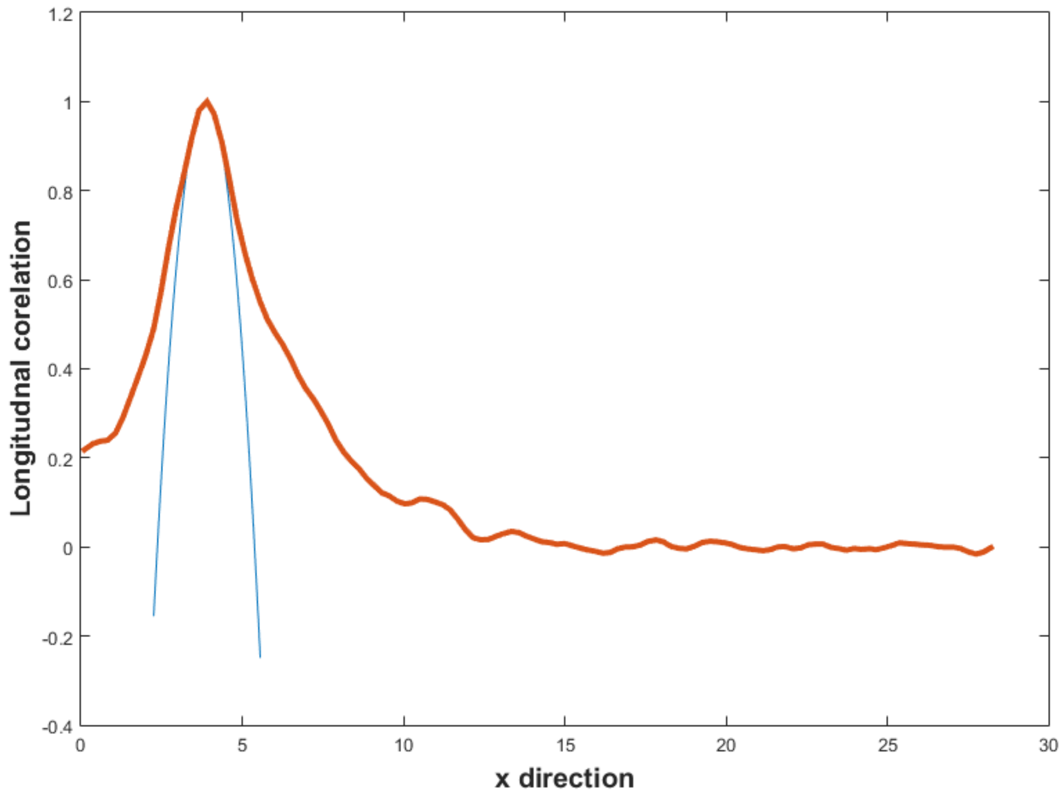

Fig3: Longitudinal Correlation plot

Transverse correlation:

```
loc = 50
xmat = [x1(1,loc)^2, x1(1,loc), 1 ;x1(1,loc+1)^2,
x1(1,loc+1), 1 ; x1(1,loc-1)^2, x1(1,loc-1), 1]
r1mat = [Rt(loc);Rt(loc+1);Rt(loc-1) ]

r1con = inv(xmat) * r1mat
r1x = x1(1,loc-7):0.001 :x1(1,loc+7)
r1tms = r1con(1) * r1x.^2 + r1con(2) * r1x + r1con(3)
plot(r1x,r1tms)
hold on


p4 = plot(x1(104,:),Rt,'LineWidth',3)
xlabel('y direction','FontSize', 15,'fontweight',
'bold' ...
    );
```

```
ylabel('Transverse
corelation','FontSize',15,'fontweight', 'bold') ;
```

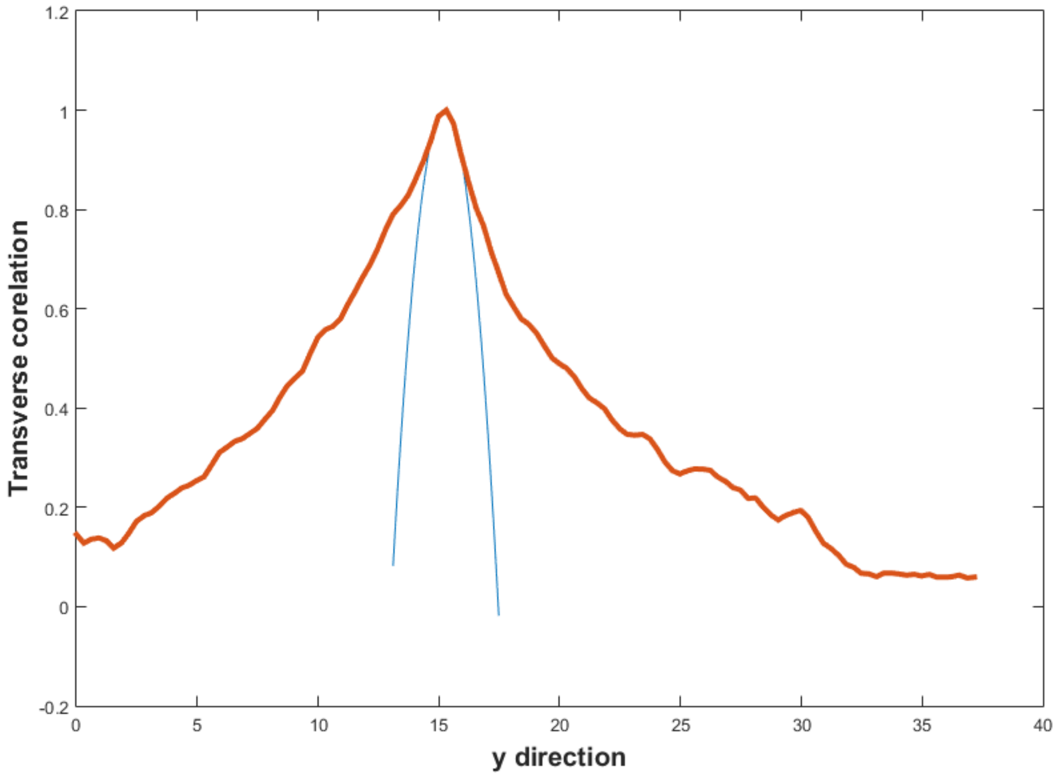

Fig4: Transverse correlation plot

Longitudinal micro scale: 2.7809 mm
Transverse micro scale: 1.8525 mm

Question 3

```
clear all;
% File which includes three columns, y U U''
data= load ('velocity1.dat');  % load your velocity
profile
%----------------------------------------

% Number of Chebyshev modes
N = 99; N1 = N+1;
j = (0:N)'; X = cos(pi*j/N);
%-------------------------------------

% Scaling for mapping from -1 to 1
L = data(end,1);
scal=L/2; % Physical domain => [0, L]
Y=X*scal + scal; % Mapping from Physical to Chebyshev
domain
%----------------------------------

% Velocity and its second derivative are being
evaluated at scaled Y
U = interp1(data(:,1), data(:,2), Y);
U2 = interp1(data(:,1), data(:,3), Y);
fac = 1/scal;
%-------------------------------------

%%For singel value of alpha and Re, for example,
% alpha = 0.3;
% length(alpha)
% Rey = 600;
%%-------------------------------
%%
% For a range of alpha and Reynolds number, for
example,
alpha = 0.1:0.01:0.75;
length(alpha)
Rey = 10:100:1500;
%-----------------------------------------
```

```
% Main program; try to understand before you run it
%ci = zeros(length(alpha),length(Rey));
%cr = zeros(length(alpha),length(Rey));
for ii = 1:length(alpha)
    ii
    for jj = 1:length(Rey)
        al = alpha(ii);
        R = Rey(jj);
        zi = sqrt(-1); a2 = al^2; a4 = a2^2; er = -
200*zi;
        [D0,D1,D2,D3,D4] = Dmat(N); % Read about it in
the book.
        D1 = fac*D1;
        D2 = (fac^2)*D2;
        D4 = (fac^4)*D4;
        B = (D2-a2*D0);
        A = (U*ones(1,N1)).*B-(U2*ones(1,N1)).*D0-(D4-
2*a2*D2+a4*D0)/(zi*al*R);
        A = [er*D0(1,:); er*D1(1,:); A(3:N-1,:);
er*D1(N1,:); er*D0(N1,:)];
        B = [D0(1,:); D1(1,:); B(3:N-1,:); D1(N1,:);
D0(N1,:)];
        d = (inv(B)*A);
        [vv, c] = eig(d);                  % eigenvalues
are being evaluated using eig function
        [mxci,I] = max(imag(diag(c)));  % useful for
contour plots
        ci(ii,jj)= mxci;
        %cr(ii,jj) =real(c(I));

    end;
end
%%
%save('ci')
%%
%ci = load('ci.mat')

contourf(ci)
title('Neutral stability curve')
```

```
xlabel('Reynolds number','FontSize', 15,'fontweight',
'bold' ...
    );
ylabel('alpha','FontSize',15,'fontweight', 'bold') ;
%%
%plot(data(:,2),data(:,1))
```

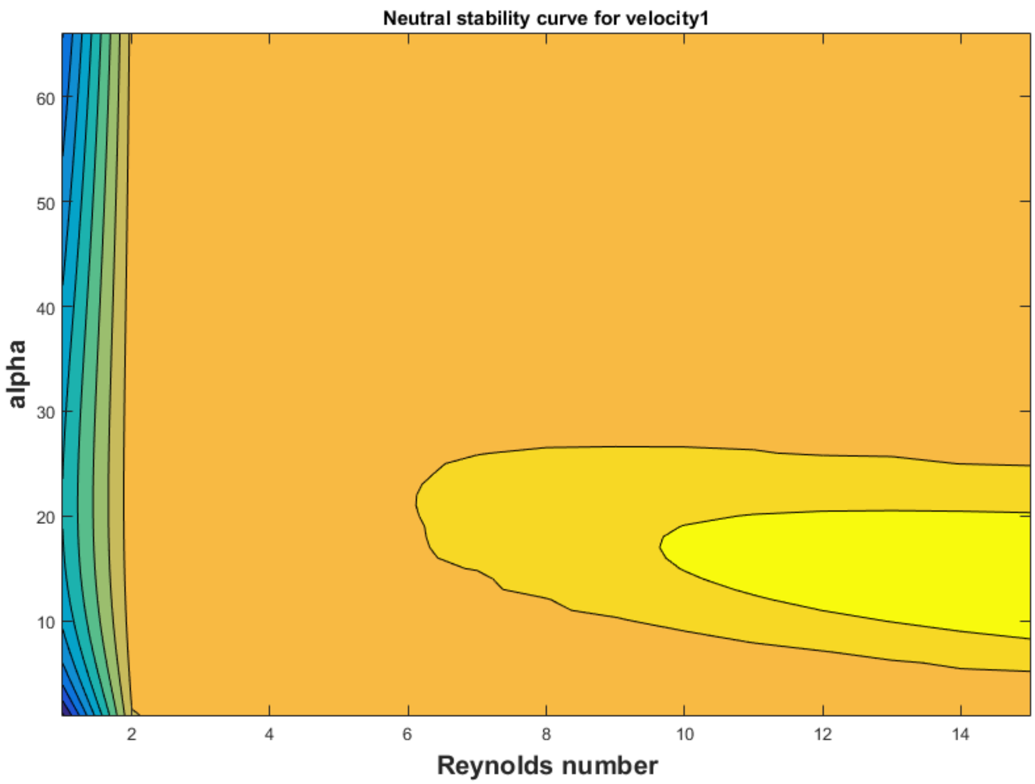

Fig5: Neutral stability curve for velocity 1

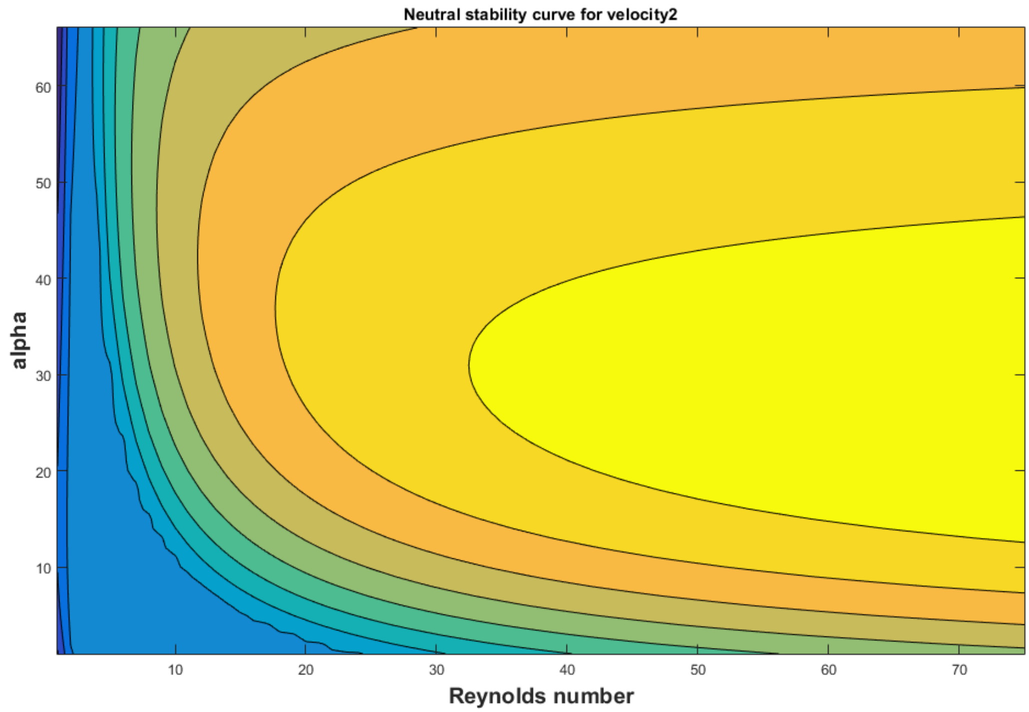

Fig6 : Neutral stability curve for velocity 2

```
clear all;
% File which includes three columns, y U U''
data= load ('velocity1.dat');  % load your velocity
profile
%-----------------------------------------

% Number of Chebyshev modes
N= 99; N1 = N+1;
j = (0:N)'; X = cos(pi*j/N);
%--------------------------------------

% Scaling for mapping from -1 to 1
L = data(end,1);
scal=L/2; % Physical domain => [0, L]
Y=X*scal + scal; % Mapping from Physical to Chebyshev
domain
%-----------------------------------
```

```
% Velocity and its second derivative are being
evaluated at scaled Y
U = interp1(data(:,1), data(:,2), Y);
U2 = interp1(data(:,1), data(:,3), Y);
fac = 1/scal;
%--------------------------------------

%%For singel value of alpha and Re, for example,
alpha = 0.3;
length(alpha)
Rey = 600;
%%-------------------------------
%
% For a range of alpha and Reynolds number, for
example,
% alpha = 0.1:0.01:1.5;
% length(alpha)
% Rey = 10:0.5:3000;
%------------------------------------------

% Main program; try to understand before you run it
%ci = zeros(length(alpha),length(Rey));
%cr = zeros(length(alpha),length(Rey));
for ii = 1:length(alpha)
    ii
    for jj = 1:length(Rey)
        al = alpha(ii);
        R = Rey(jj);
        zi = sqrt(-1); a2 = al^2; a4 = a2^2; er = -
200*zi;
        [D0,D1,D2,D3,D4] = Dmat(N); % Read about it in
the book.
        D1 = fac*D1;
        D2 = (fac^2)*D2;
        D4 = (fac^4)*D4;
        B = (D2-a2*D0);
        A = (U*ones(1,N1)).*B-(U2*ones(1,N1)).*D0-(D4-
2*a2*D2+a4*D0)/(zi*al*R);
        A = [er*D0(1,:); er*D1(1,:); A(3:N-1,:);
er*D1(N1,:); er*D0(N1,:)];
```

```
        B = [D0(1,:); D1(1,:); B(3:N-1,:); D1(N1,:);
D0(N1,:)];
        d = (inv(B)*A);
        [vv, c] = eig(d);                  % eigenvalues
are being evaluated using eig function
%          [mxci,I] = max(imag(diag(c)));  % useful for
contour plots
%          ci(ii,jj)= mxci;
%          cr(ii,jj) =real(c(I));

    end;
end
maxc = max(imag(diag(c)))

for i = 1:100
    if imag(c(i,i)) ==maxc
        loc = i
    end
end

maxv= vv(:,loc);
values = abs(maxv);

plot(values,Y,'b','LineWidth',1.5)


%%
uu = vv/(sqrt(-1)*alpha);
maxu = uu(:,loc);
valuesu = abs(maxu);
plot(flip(valuesu),Y,'b','LineWidth',1.5)
```

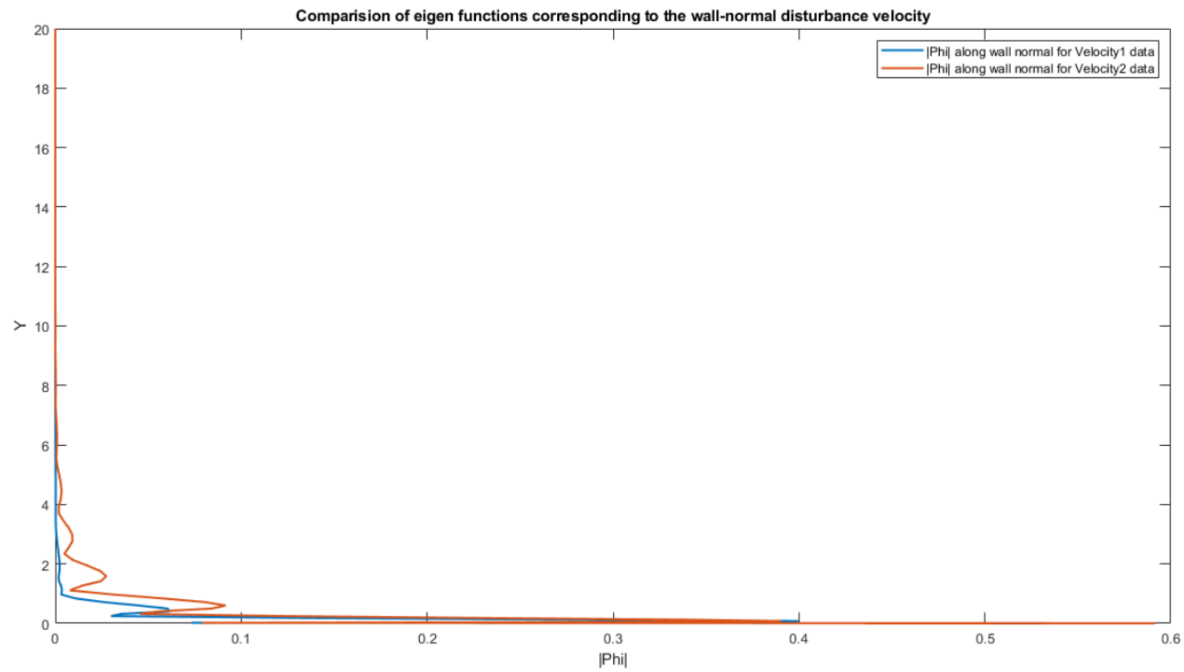

Fig7: wall normal disturbance velocity

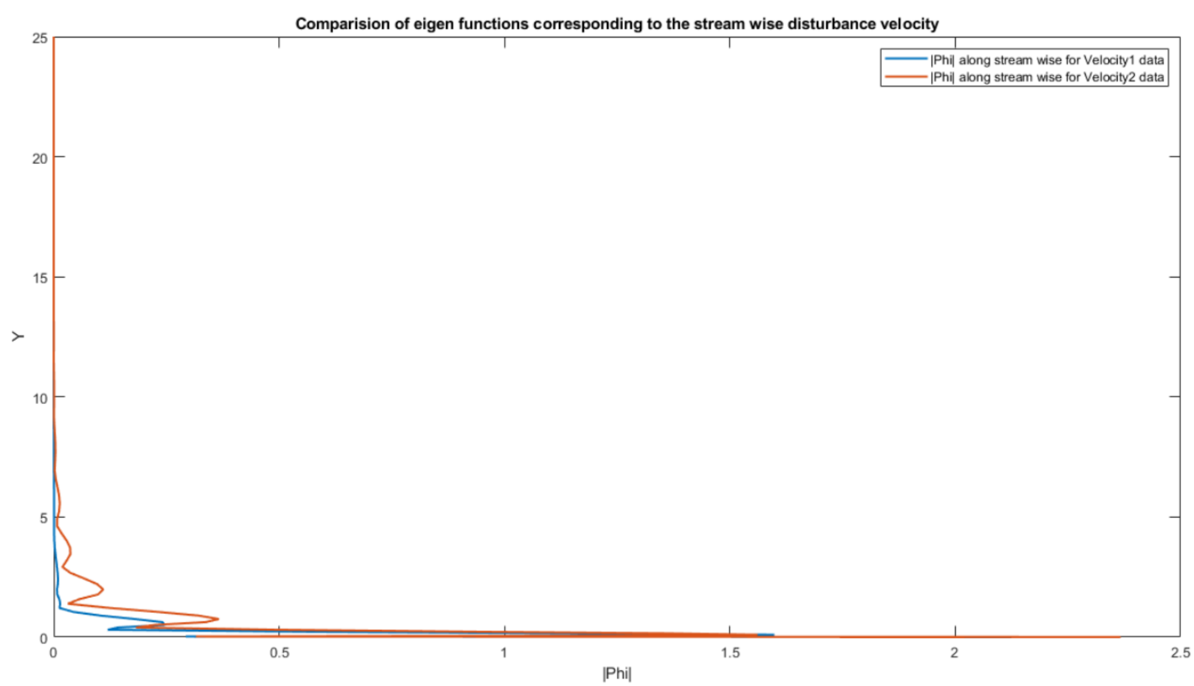

Fig8: streamwise disturbance velocity

Question 1:
This part was done in Python
Importing libraries
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Importing data
```
data = np.genfromtxt('hotwire.dat')
t = data[:,0]
v  = data[:,1]
vm = np.mean(v)
n = len(v)
vmean = np.ones(len(v))*vm
var = v-vm
dem = 0
```
Finding rms
```
for a in range(n):
    dem = (var[a])**2 + dem

dem = dem/n
```

Finding the Correlation

```
R = np.zeros(30)
for i in range(30):
    r = 0
    for j in range(n-i):
        r = var[j]* var[j+i] +r
    cor = r/(n-i)
    R[i] = cor/dem
  #  print(n-i,i,cor,R[i])



plt.plot(t[:30],R)
```

Finding the parabola

```
pa = np.array([[t[0]**2, t[0]
,1],[t[1]**2,t[1],1],[t[2]**2 , t[2], 1]])

pai = np.linalg.inv(pa)
pb = np.array([R[0], R[1], R[2]])
px = pai @ pb

tp = np.linspace(0,t[4],10)
y = np.zeros(len(tp))
for i in range(len(tp)):
    y[i]= px[0]*tp[i]**2 + px[1]*tp[i]+px[2]


plt.plot(tp,y)
```

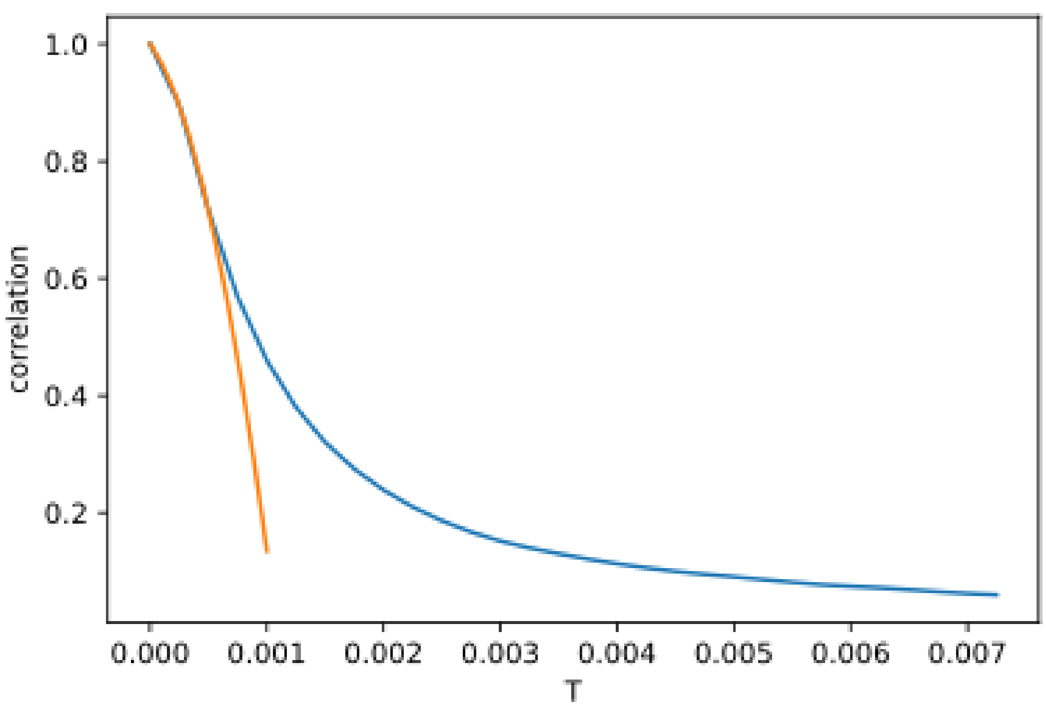

Fig9: correlation vs time difference plot here blue line shows the correlation plot and orange line shows the parabola

Finding taylor microscale and time scale

```
from sympy import symbols, solve

x = symbols('x')
expr = px[0]*x**2+px[1]*x+px[2]


sol = solve(expr)
print(sol)
#%%

area = np.trapz(R)
print(area)
```