AS 5430 Course Project Report

Stability Study of Temporal Mixing Layer

Vikas Dwivedi

$ME15D080$

Contents

List of Figures

1 Introduction

1.1 Theory

The parallel flow of two fluids moving at different velocities constitutes a planar mixing layer. In this project, we study inertial and viscous instability of such flows which is popularly known as Kelvin-Helmholtz (KH) instability. The instability mechanism is a combination of inertia and viscous effects.

1.1.1 Linear inviscid analysis

In the linear stability study, we introduce perturbations to a base flow and study the evolution of these perturbations on the basis of momentum equations which are linearised about that base flow. The rationale behind this type of study is that although NSE are non linear, but when the order of magnitude of perturbations is small compared to base flow variables, non linear terms of perturbations become insignificant. When the equation becomes linear, we can use normal mode analysis to study such flows as linear dynamic systems. As the magnitude of perturbation becomes stronger, we can no longer neglect the effect of non linearity.

The main objective of linear stability study is to find a relationship between a given mode and its corresponding growth rate. In case of planar mixing layers, the procedure followed is as follows[2]:

1. *Analyse the governing law in terms of characteristic scales and neglect insignificant terms.* Ignoring viscous effects compared to inertial ones, we choose the characteristic scales L, V and ρV^2 of the flow as our length, speed, and pressure units. Our governing equations reduce to Euler equations i.e.

$$div(\mathbf{U}) = 0 \tag{1.1}$$

$$\partial_t \mathbf{U} + (\mathbf{U}.\mathbf{gradU})\mathbf{U} = -\mathbf{grad}(P) \tag{1.2}$$

2. *Define base flow.* We consider shear free flow over an unbounded depth. Our base flow is

$$\mathbf{U}(\mathbf{x},t) = 0.5(1 + tanh(y))\mathbf{e_x} \tag{1.3}$$

$$P(\mathbf{x},t) = \overline{P}. \tag{1.4}$$

3. *Linearized perturbation equations*. Generally we assume 3D perturbations, but for parallel shear flows we know that for any 3D unstable mode, there is an associated 2D mode which is more unstable (Squires theorem). Therefore, it is sufficient to consider only 2D perturbations for instability studies. We define perturbation streamfunctions as

   $$\psi = \psi \hat{(y)} e^{i(kx-\omega t)}. \tag{1.5}$$

   Linearized perturbation equation for parallel shear flows are popularly known as Rayleigh's equation i.e.

   $$(\overline{U}-c)(\partial_{yy}-k^2)\hat{\psi} - \partial_{yy}\overline{U}\hat{\psi} = 0 \tag{1.6}$$

   subjected to following BCs

   $$\hat{\psi}(-\infty) = \hat{\psi}(-\infty) = 0.$$

1.1.2 Viscous analysis

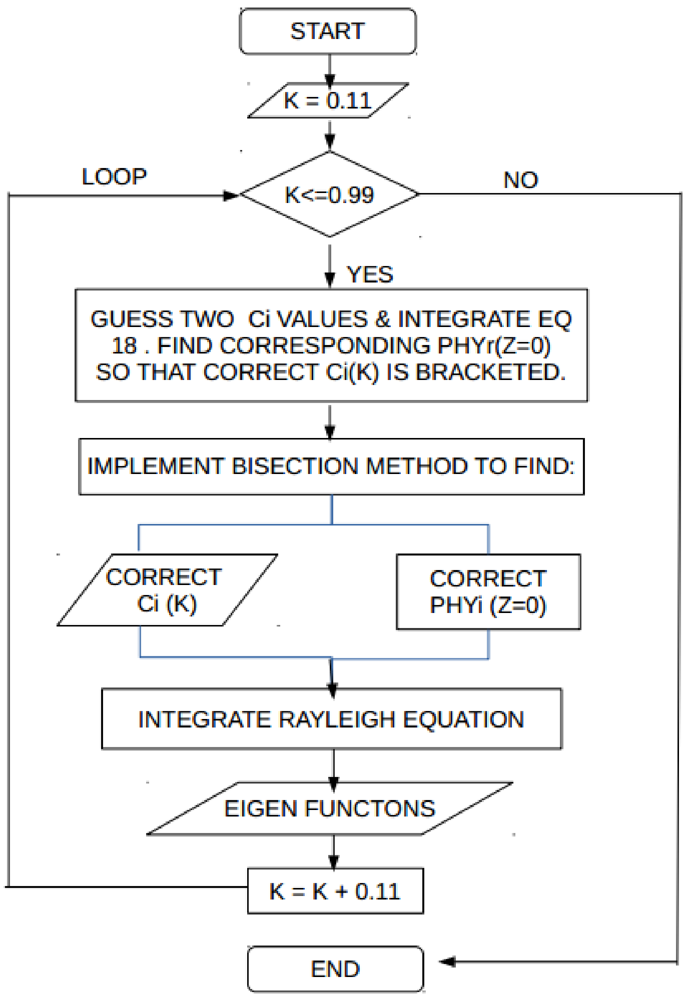
On including viscous effects, an additional fourth order derivative term comes up in the RHS of Rayleigh's equation and this equation is called Orr - Sommerfeld equation.

$$(\overline{U}-c)(\partial_{yy}-k^2)\hat{\psi} - \partial_{yy}\overline{U}\hat{\psi} = \frac{1}{ikRe}(\partial_{yy}-k^2)^2\hat{\psi} \tag{1.7}$$

2 Linear Inviscid Stability of Temporal Mixing Layer

2.1 Numerical formulation

We replicate the numerical procedure given in Michalke [1]. Figure 2.1 shows the flow chart of the numerical procedure.



START
K = 0.11
LOOP
K<=0.99
NO
YES
GUESS TWO Ci VALUES & INTEGRATE EQ 18 . FIND CORRESPONDING PHYr(Z=0) SO THAT CORRECT Ci(K) IS BRACKETED.
IMPLEMENT BISECTION METHOD TO FIND:
CORRECT Ci (K)
CORRECT PHYi (Z=0)
INTEGRATE RAYLEIGH EQUATION
EIGEN FUNCTONS
K = K + 0.11
END

Figure 2.1: Flowchart of numerical procedure

Governing equation is

$$(\overline{U}-c)(\partial_{yy}-k^2)\hat{\psi}-\partial_{yy}\overline{U}\hat{\psi}=0,$$

subjected to $\hat{\psi}(-\infty)=\hat{\psi}(+\infty)=0$.
Base flow is

$$\overline{U}(y)=0.5(1+tanh(y)). \tag{2.1}$$

<u>**In the paper, symbol ϕ has been used for $\hat{\psi}$.**</u> Now setting $\phi=e^{\int\Phi dy}$ and $z=tanh(y)$, Rayleigh equation reduces to Riccati equation i.e.

$$\frac{d\Phi}{dz}=\frac{k^2-\Phi^2}{1-z^2}-\frac{2z}{z-i2c_i} \tag{2.2}$$

subjected to $\Phi(-1)=k$ and $\Phi(+1)=-k$. This is eq (18) in the paper. On separating the real and imaginary parts, it can be shown that real part is symmetric and imaginary part is anti-symmetric. So it can be solved for half domain i.e $[-1,0]$ with the following boundary conditions $\Phi_r(0)=0$ and $\frac{d\Phi_i}{dz}(0)=0$. With this, we can solve the eigen value problem eq(2.2) using bisection technique. In addition, it gives us the correct initial condition for Rayleigh equation i.e.

$$\phi(0)=1 \tag{2.3}$$

$$\phi'(0)=0+i\Phi_i(0) \tag{2.4}$$

On solving the eigen value problem, with correct BCs (eq 2.3 and 2.4) and correct c_i, Rayleigh equation can be simply integrated by expressing it as a system of single order differential equation as follows:

$$\frac{d}{dy}\left\{\begin{array}{c}\phi(1)\\ \phi(2)\end{array}\right\}=\left\{\begin{array}{c}\phi(2)\\ k^2\phi(1)+\frac{\overline{U}_{yy}}{\overline{U}-c}\phi(1)\end{array}\right\} \tag{2.5}$$

where $\phi(1)=\phi$ and $\phi(2)=\frac{d\phi}{dy}$.

In bisection method, we try to find x_r which lies in $[x_{guess1},x_{guess2}]$ and is such that $f(x_r)=0$ by repeatedly bisecting the bracketing interval. Here, $f\rightarrow\Phi_r(0)$ and $x\rightarrow c_i$. In order to find $f(c_i)$ and later integrate Rayleigh equation, we have used Runge-Kutta 2 (Heun's method).

Eigen functions are used to determine the streamlines. With streamfunctions known, vorticity contours can be easily found. Author has studied the flow behavior in terms of streamlines and isovorticity lines corresponding to neutral and most amplified perturbation cases.

Computer codes *Rayleigh_mixinglayer.cpp* numerically solves the eigen value problem according to method described above / in the flowchart . MATLAB script *mixinglayer_figs.m* replicates all the plots. Both the codes are readable as they contain enough comments and reference to equations from the original paper.

2.2 Results

We exactly replicated the results of Michalke in terms of eigen functions, flow structure and calculation of most unstable mode. Summary and conclusions of this linear stability study are as follows:

1. Solved eigen value problem and calculated streamline eigen functions. From the dispersion relation, the most unstable mode was found to be 0.4478. (Figure 2.2 & 2.3)
2. Compared the flow streamlines for the most unstable and the neutrally stable case. It was found that there is no special difference in the two cases. (Figure 2.4)
3. Calculated vorticity eigenfunctions. (Figure 2.5)
4. Compared the flow vorticity contours for the most unstable and the neutrally stable case. It was found that there is appreciable difference in the flow structure for the two cases. So we can conclude that for mixing layer case, it is better to plot vorticity contours for instability studies. (Figure 2.6)

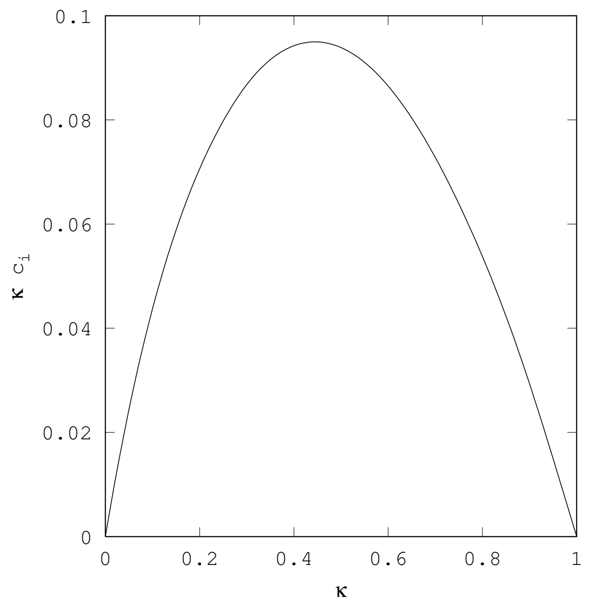
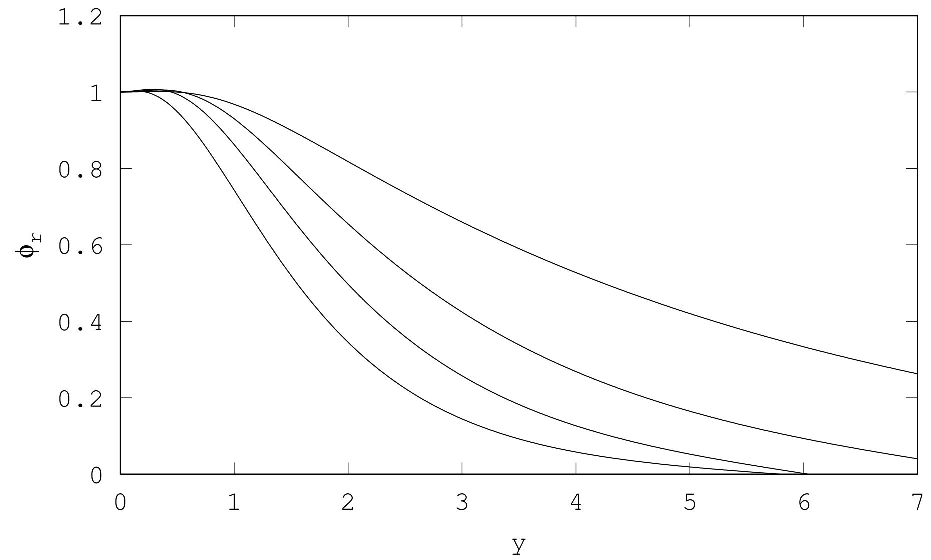
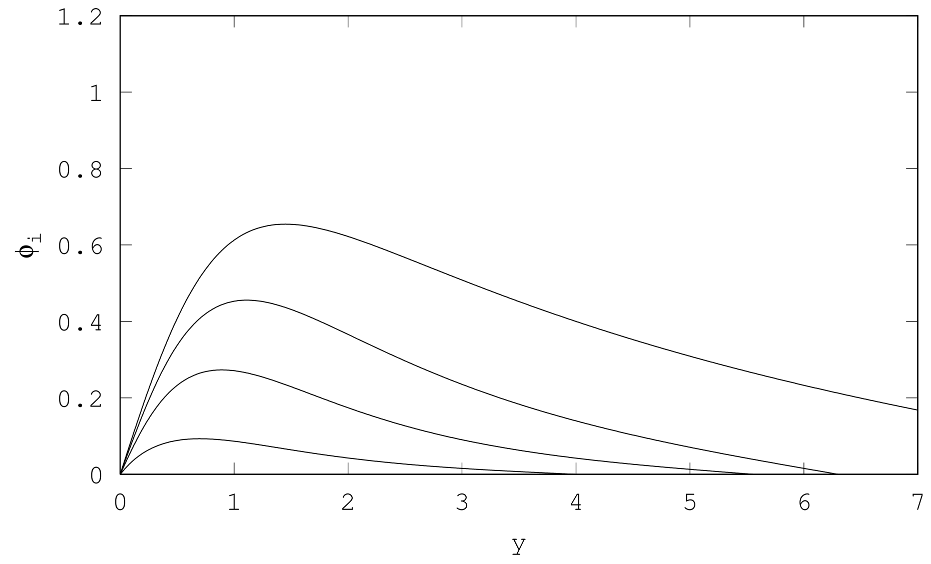


Figure 2.2: Growth rate versus wavenumber (Page 548, Michalke)

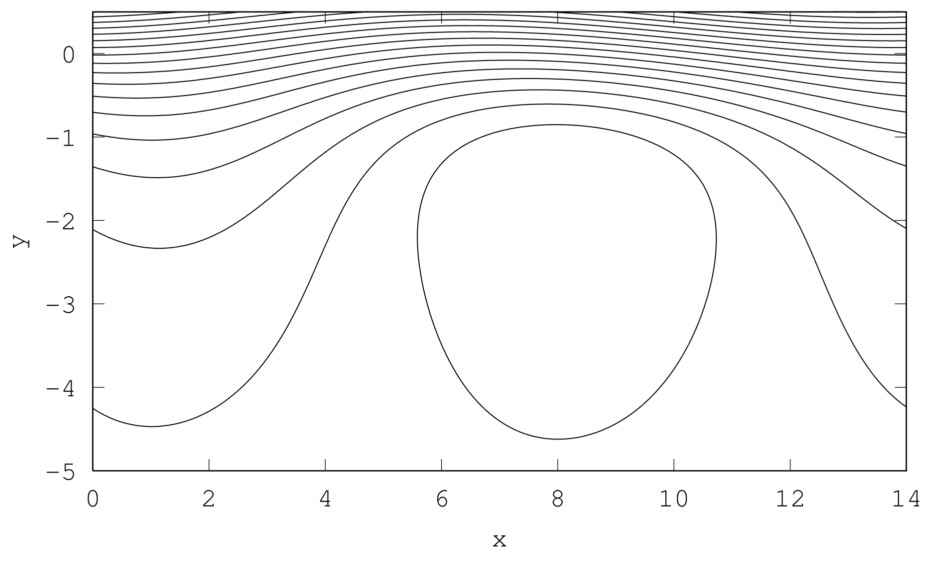


(a) Real part

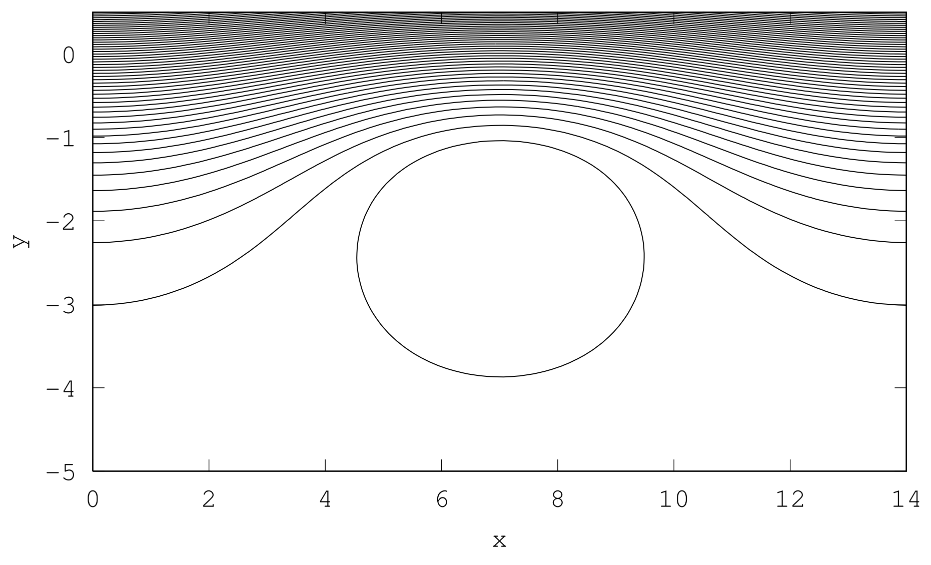


(b) Imaginary part

Figure 2.3: Eigen functions for various wavenumbers (Page 549, Michalke)

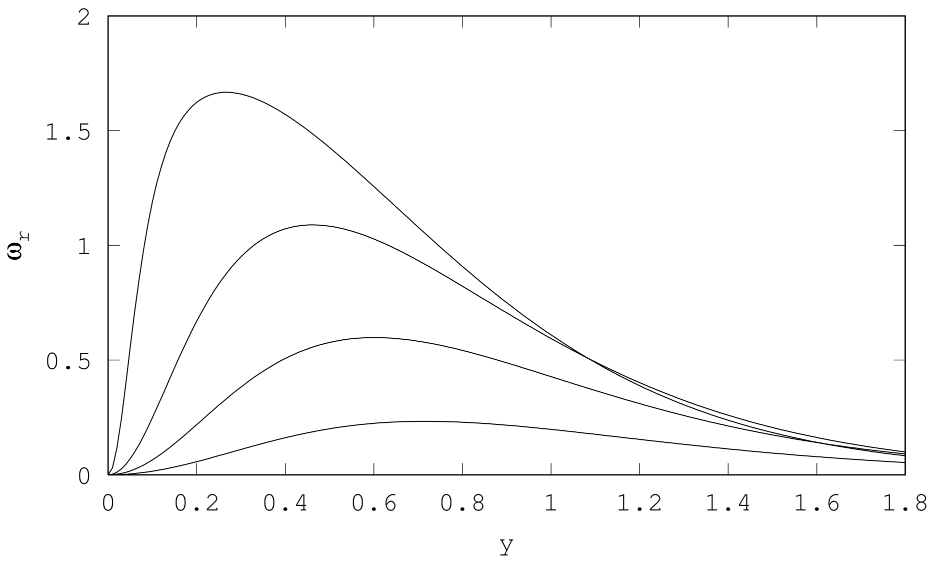


(a) Most amplified case

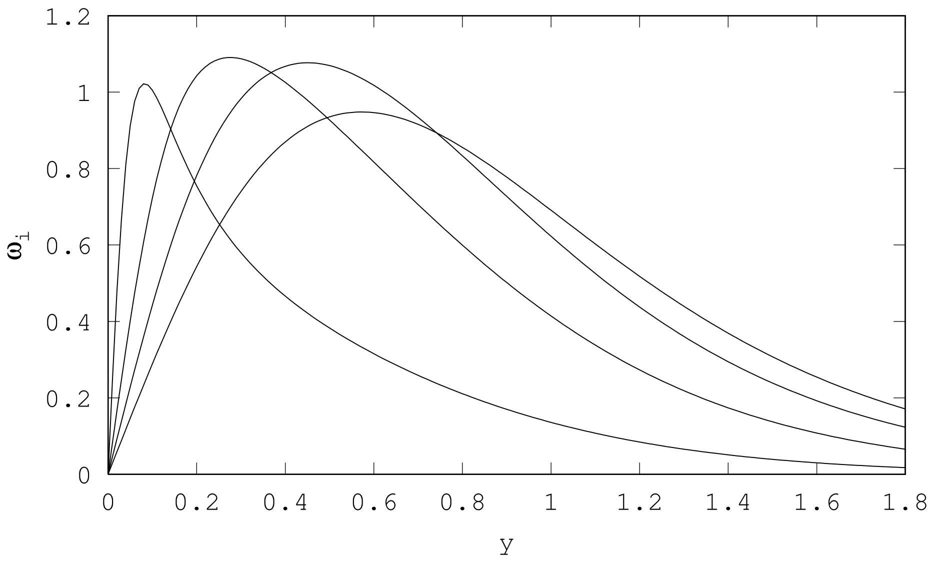


(b) Neutral case

Figure 2.4: Streamlines of the disturbed velocity profile $U(y) = 0.5[1 + tanh(y)]$ for the wave-numbers of maximum amplification $k = 0.4446$ and of the neutral disturbance $k = 1$ at the time $t = 0$ and a disturbance magnitude $\varepsilon = 0.1$(Page 550, Michalke)

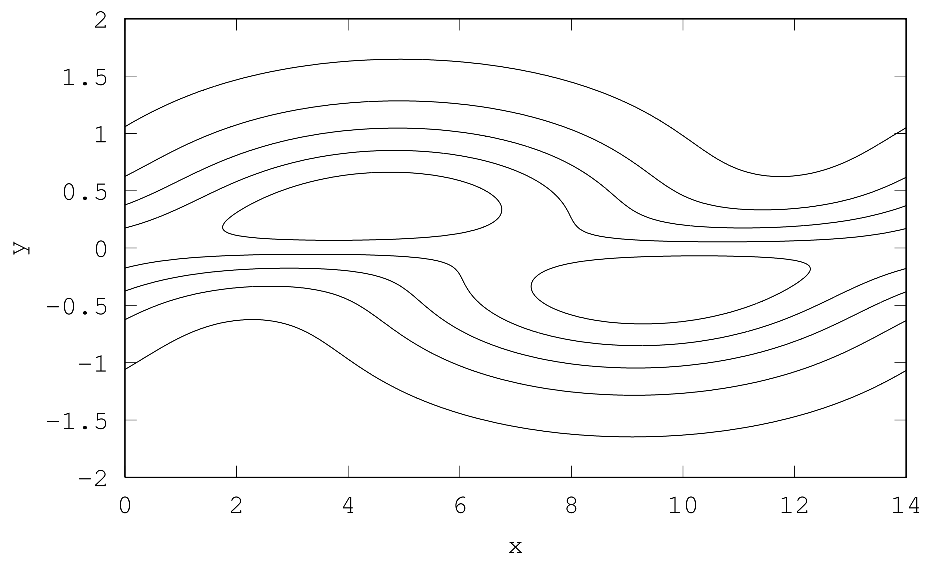


(a) Real part

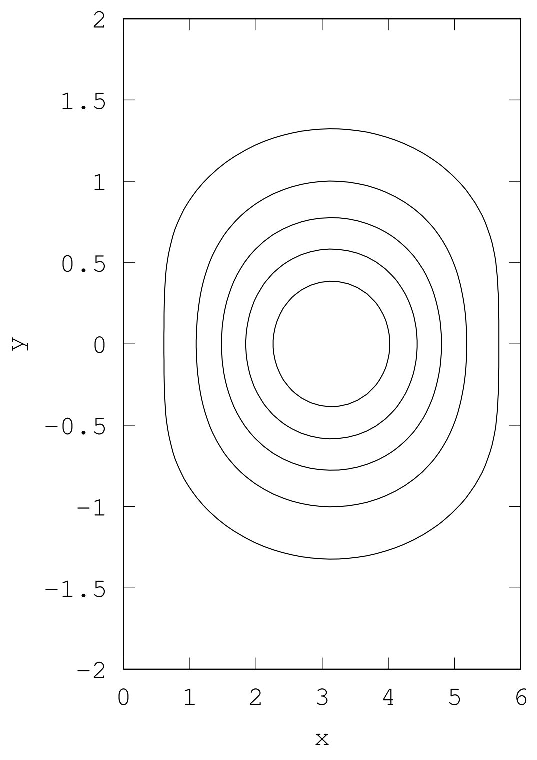


(b) Imaginary part

Figure 2.5: Vorticity eigen functions for various wavenumbers (Page 553, Michalke)



(a) Most amplified case



(b) Neutral case

Figure 2.6: Lines of constant vorticity of the disturbed velocity profile $U(y) = 0.5[1 + tanh(y)]$ for the wave-numbers of maximum amplification $k = 0.4446$ and of the neutral disturbance $k = 1$ at the time $t = 0$ and a disturbance magnitude $\varepsilon = 0.2$(Page 555, Michalke)

3 Viscous Stability of Temporal Mixing Layer

3.1 Numerical formulation

Governing equation is

$$(\overline{U}-c)(\partial_{yy}-k^2)\hat{\psi}-\partial_{yy}\overline{U}\hat{\psi}=\frac{1}{ikRe}(\partial_{yy}-k^2)^2\hat{\psi}$$

Our base flow is

$$\overline{U}(y)=0.5(1+tanh(y)). \tag{3.1}$$

Since the flow is unbounded, we set $z=tanh(y)$ which shrinks the domain to $[-1,+1]$ and changes the profile to

$$\overline{U}(z)=0.5(1+z) \tag{3.2}$$

To simplify the Orr Sommerfeld equation, we use chain rule of differentiation which leads to the following expressions:

$$(\overline{U}(y)-c)(\partial_{yy}-k^2)\hat{\psi}=(\overline{U}(z)-c)((z_y)^2\partial_{zz}+z_{yy}\partial_z-k^2)\hat{\psi} \tag{3.3}$$

$$\partial_{yy}\overline{U}(y)=((z_y)^2\partial_{zz}+z_{yy}\partial_z)\overline{U}(z) \tag{3.4}$$

$$\begin{aligned}(\partial_{yy}-k^2)^2\hat{\psi} &= [(z_y)^4\partial_{zzzz}+6(z_y)^2z_{yy}\partial_{zzz}+(4z_yz_{yyy}+3z_{yy}(z_y)^2-2k^2(z_y)^2)\partial_{zz}\\ &\quad +(z_{yyyy}-2k^2z_{yy})\partial_z+k^4]\hat{\psi}\end{aligned}$$

where

$$z_y = sech^2(y) \tag{3.5}$$

$$z_{yy} = -2tanh(y)sech^2(y) \tag{3.6}$$

$$z_{yyy} = 2(cosh(2y) - 2)sech^4(y) \tag{3.7}$$

$$z_{yyyy} = -2(sinh(3y) - 11sinh(y))sech^5(y) \tag{3.8}$$

Now BCs become

$$\hat{\psi}(-1) = \partial_y \hat{\psi}(-1) = 0 \tag{3.9}$$

$$\hat{\psi}(-1) = \partial_y \hat{\psi}(-1) = 0 \tag{3.10}$$

Chebyschev polynomials are:

$$T_n(y) = cos(n(cos^{-1}y)) \tag{3.11}$$

These polynomials satisfy the following recursive relationship:

$$T_{n+1}(y) = 2yT_n(y) - T_{n-1}(y) \tag{3.12}$$

$$T_n^k(y) = 2nT_{n-1}^{k-1}(y) - \left(\frac{n}{n-2}\right) T_{n-2}^k(y) \tag{3.13}$$

On substituting $\hat{\psi}$ in terms of Chebyschev polynomials

$$\hat{\psi}(y) = \sum_{n=0}^{N-1} a_n T_n(y) \tag{3.14}$$

$$\hat{\psi}(y)^{(k)} = \sum_{n=0}^{N-1} a_n T_n(y)^{(k)} \tag{3.15}$$

at spectral collocation points

$$yC_j = cos\left(\frac{j\Pi}{N-1}\right) \tag{3.16}$$

and using the following property of Chebyschev functions at boundaries
$T_n(+1) = (+1)^n, T_n(-1) = (-1)^n, T_n^{'}(+1) = (+1)^{n-1}n^2, T_n^{'}(-1) = (-1)^{n-1}n^2$,
the Orr Sommerfeld equation reduces to an algebraic system of N equations and N unknown Chebyschev coefficients.

This system can be written as

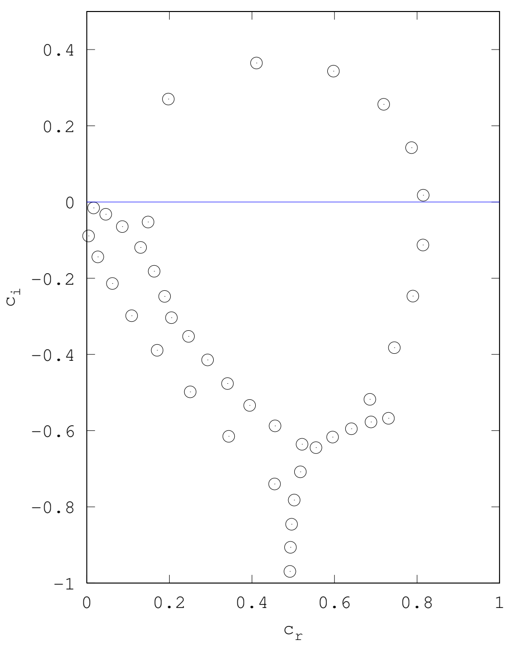
$$Aa = cBa \tag{3.17}$$

where c is the complex eigen value and a is the vector of unknown Chebyschev coefficients.

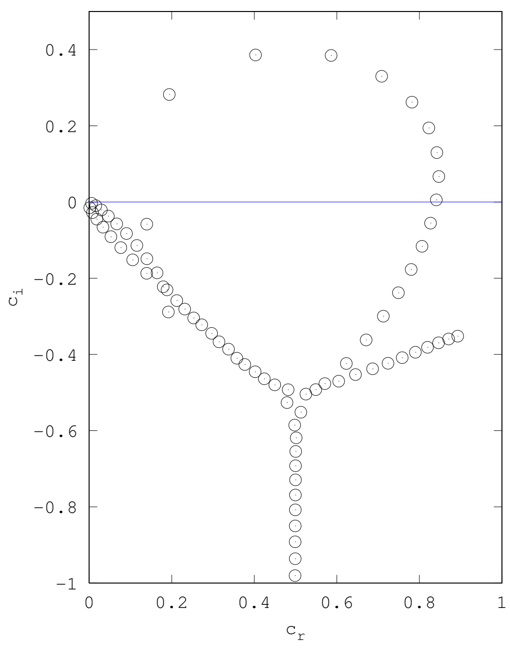
3.2 Results

OSE_mixinglayer.m implements the above mentioned procedure. It is attached at the end of this report.

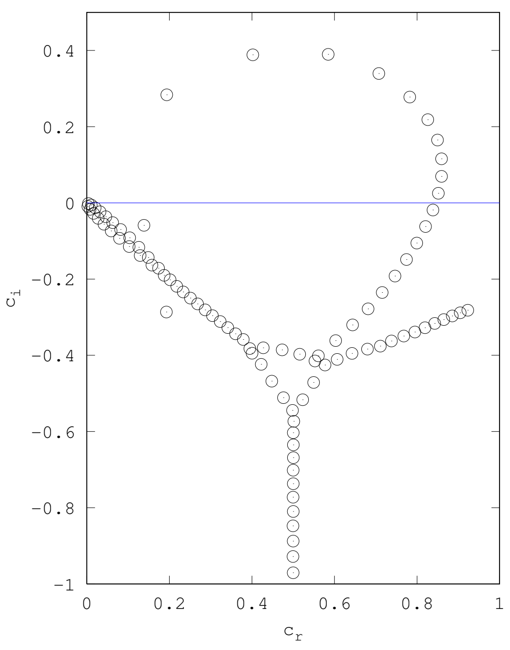
1. Inviscid analysis predicted that the maximum growth at $k = 0.446$ and its value was 0.09. On including the viscous terms, we find that global eigen value spectrum of c_i predicts a greater growth rate and this rate increases as we increase the Re. However, the shape of spectrum remains almost same and the eigen values lie in ordered branches.
2. It can be seen that growth rate doesn't increase unboundedly with increase in K. At $Re = 10000$, the spectrum suggests that temporal mixing layer is stable to very short waves. So the results are not unphysical.



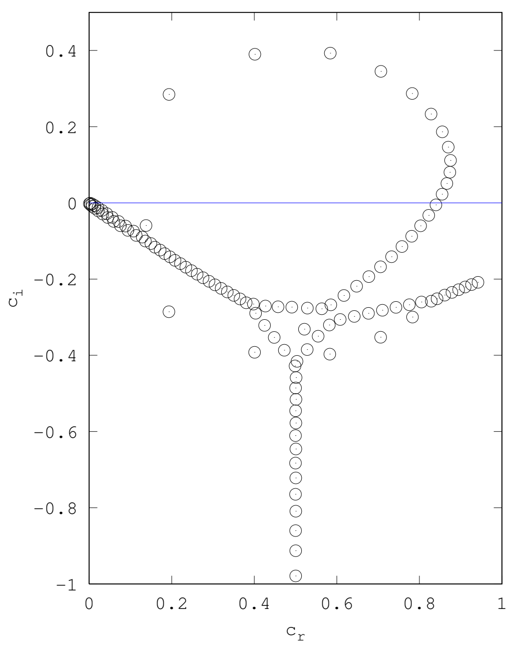
(a) @$Re = 1000$



(b) @$Re = 5000$

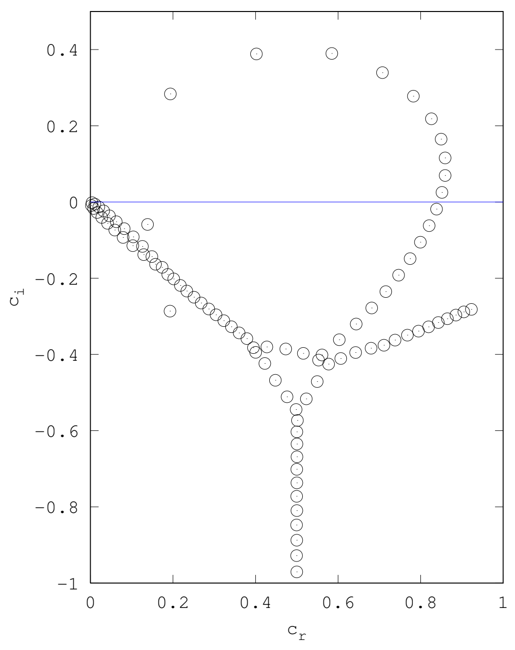


(c) @$Re = 10000$

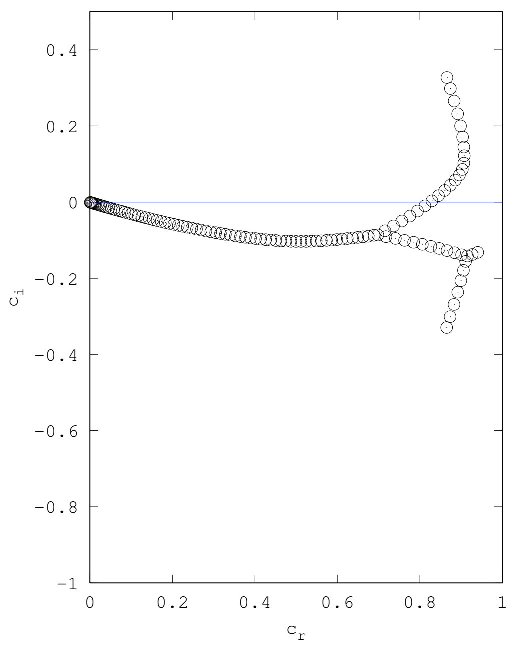


(d) @$Re = 25000$

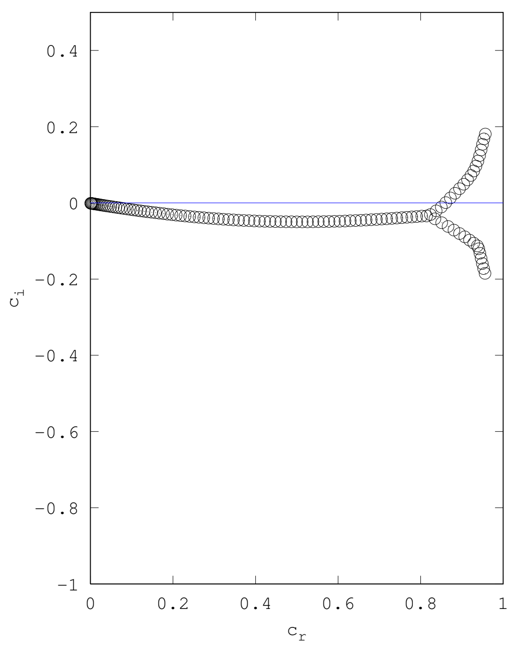
Figure 3.1: Global eigen value spectrum at various Re for $K = 0.446$



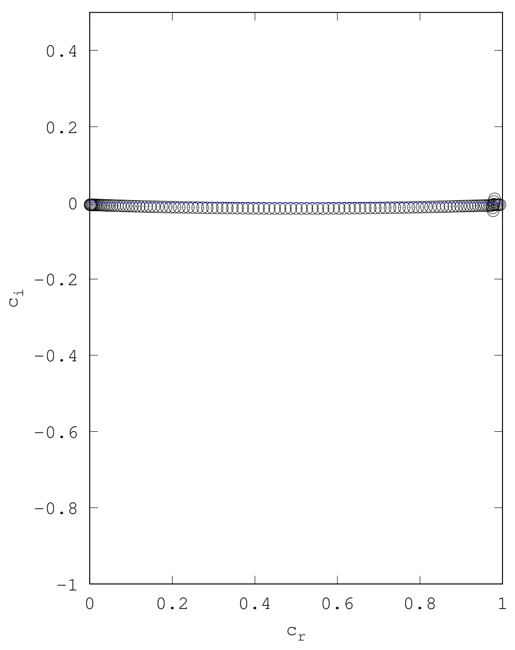
(a) @$K = 0.446$



(b) @$K = 5$



(c) @$K = 10$



(d) @$K = 50$

Figure 3.2: Global eigen value spectrum for various K at $Re = 10000$

Bibliography

[1] Michalke, Alfons. On the inviscid instability of the hyperbolic-tangent velocity profile. Deutsche Versuchsanstalt f. Luft-u. Raumfahrt, 1964.

[2] François Charru. Hydrodynamic Instabilities. Volume 37 of Cambridge Texts in Applied Mathematics, Cambridge University Press, 2011

```
/****************************************************************/
/* AS5430 COURSE-PROJECT                                        */
/* LINEAR INVISCID INSTABILITY STUDY OF TANH VELOCITY PROFILE   */
/* SUBMITTED BY: VIKAS DWIVEDI (ME15D080)                       */
/****************************************************************/

# include <iostream>
# include <cmath>
# include <complex>
# include <cassert>
# include <fstream>
# include <iomanip>
# include <cstdlib>
using namespace std;

double** rk2_riccati (double**, double*, double*, double, double,double,int,int);

int main()
{

 char const* ch1="eig_fcn"; char const* ch2="disp_data";

//-->define iota

 complex<double> ii,iota; ii=-1;
 iota = sqrt(ii);

//------declaration of variables--------//

 int row_PHY=100  ; int col_PHY=2;
 int row_phy=701  ; int col_phy=2;

 double*  k_range           ; k_range = new double [9];
 double*  ci_range          ; ci_range= new double [9];
 double*  Z_span            ; Z_span  = new double [row_PHY];
 double*  y_span            ; y_span  = new double [row_phy];
 double*  PHY_init          ; PHY_init= new double [col_PHY];
 double** PHY = 0           ; PHY     = new double*[row_PHY];
 complex <double>*  phy_init; phy_init= new complex<double> [col_phy];
 complex <double>** phy = 0 ; phy     = new complex<double>*[row_phy];


  for(int i=0;i<9;i++)
    {
     k_range  [i] = 0.11*(1.0+double(i)); // k = [0.11,0.22,...0.99], eigen functions are known for
k=0,1
     ci_range [i] = 0.0;
    }

  for (int i=0;i<row_PHY;i++)
    {
      Z_span[i]=-0.99+(double(i)/(row_PHY-1))*0.99; // Z = [-1,0] because of symmetry. 1st paragraph
Page 547, Michake
    }
  // y_span depends upon the plots we want to replicate
  for (int i=0;i<row_phy;i++)
    {
        y_span[i]= 0.0+(double(i)/(row_phy-1))*7.0;// for eigenfunction plots (Round-1)
      //y_span[i]= 0.0+(double(i)/(row_phy-1))*(-5.0);// for streamfcn plots (Round-1)
      // y_span[i]= -5.0+(double(i)/(row_phy-1))*(7.2); // for streamfcn plots (Round-2)
    }

  // PHY(Z)'s 1st & 2nd column contain real & imaginary part of PHY respectively
  for (int i = 0; i < row_PHY; i++)
    {
      PHY[i] = new double[col_PHY];

      for (int j = 0; j < col_PHY; j++)
      {
        PHY[i][j] =0.0;
      }
    }
  // phy(y)'s 1st & 2nd column contain phy (eig_fcn) & dphydy (first deriv) respectively
  for (int i = 0; i < row_phy; i++)
    {
      phy[i] = new complex<double>[col_phy];

      for (int j = 0; j < col_phy; j++)
      {
        phy[i][j] =0.0;
      }
    }


 double dZ = Z_span[1]-Z_span[0];
 double dy = y_span[1]-y_span[0];
 double k,error;
 double ci_left, ci_right, ci_middle, ci_middle_updated, ci_correct;
 double PHYr_at_0_left,PHYr_at_0_right,PHYr_at_0_middle,PHYi_at_0;
 complex<double> num,deno;

/*-------------------declaration done---------------------------*/


/*---------------wavenumber loop starts here---------------------*/

for (int count=0;count<9;count++)
//for (int count=3;count<4;count++)
{

   k = k_range[count];

   PHY_init[0] = k;PHY_init[1] = 0.0; // See eq (19) Page 546, Michalke

   ci_left = 1e-10; ci_right= 0.5;    // make guess

/*--part-1: Solve EVP (Riccati eqn) using bisection method--*/

// In bisection, we aim to find f(x_root)=0 when x_left < x_root < x_right
// Here, f is PHYr(0) and x is ci. See eq (21) Page 547, Michalke

  PHY = rk2_riccati (PHY, PHY_init, Z_span, k, ci_left, dZ, row_PHY, col_PHY); //find f(ci_left) by 
integrating
  PHYr_at_0_left= PHY[row_PHY-1][0];// f (ci_left)

  PHY = rk2_riccati (PHY, PHY_init, Z_span, k, ci_right, dZ, row_PHY, col_PHY);//find f(ci_right) by 
integrating
  PHYr_at_0_right= PHY[row_PHY-1][0];// f (ci_right)

  assert(PHYr_at_0_left*PHYr_at_0_right < 0); // make sure ci_correct is bracketed

  ci_middle = 0.5*(ci_left+ci_right);// bisect the bracket
  error = 100;

  while(error > 0.5)  // error within 0.5%
  {

    PHY = rk2_riccati (PHY, PHY_init, Z_span, k, ci_middle, dZ, row_PHY, col_PHY);
    PHYr_at_0_middle= PHY[row_PHY-1][0];
    PHYi_at_0 = PHY[row_PHY-1][1];// IC for Rayleigh equation See eq 26 Page 548, Michalke

    if (PHYr_at_0_middle*PHYr_at_0_right<0)
    {
      ci_left = ci_middle;
    }
    else
    {
      ci_right = ci_middle;
    }
    ci_middle_updated = 0.5*(ci_left+ci_right);

    error=100.0*fabs((ci_middle_updated-ci_middle)/ci_middle_updated);

    ci_middle = ci_middle_updated;
  }

  ci_correct = ci_middle;
  ci_range [count] = ci_correct; // c(k) is known

/*--------part-2: integrate Rayleigh's equation (complex)---------------*/

// most amplified wave:
// y Re(phi[0][0])  Im(phi[0][0])  Re(phi[0][1])  Im(phi[0][1])
//-5 0.16464931    -0.070682136    0.084842613   -0.060205098

//-> (Round 1)
phy_init[0] =1.0 ;
phy_init[1] = iota*(PHYi_at_0);

//-> (Round 2)
// phy_init[0] = 0.16464931  -0.070682136*iota;
// phy_init[1] = 0.084842613 -0.060205098*iota;

for (int j=0;j<col_phy;j++ )
{
  phy[0][j]=phy_init[j];
}


/*--intermediate variables--*/
complex<double>* dphydy;
complex<double>* phy_guess;
complex<double>* dphydy_guess;

dphydy       = new complex<double>[col_phy];
phy_guess    = new complex<double>[col_phy];
dphydy_guess = new complex<double>[col_phy];

/*--integrate using Heun's method (RK2)--*/

for (int i=0;i<row_phy-1;i++)
{

//use Eq.3 [Rayleigh equation]

  dphydy[0]= phy[i][1];

  num = tanh(y_span[i])/(cosh(y_span[i])*cosh(y_span[i]));
  deno = 0.5*tanh(y_span[i])-ci_correct*iota;

  dphydy[1]= (k*k-(num/deno))*phy[i][0] ;// See eq 3, pg 544, Michalke

//predictor step1: guess the value of y(i+1)

  for(int j=0;j<col_phy;j++)
  {
   phy_guess[j]=phy[i][j]+dphydy[j]*dy;
  }

//predictor step2: guess the value of dphydx(i+1) using Eq.3 [Rayleigh equation]

  dphydy_guess[0]= phy_guess[1];

  num = tanh(y_span[i+1])/(cosh(y_span[i+1])*cosh(y_span[i+1]));
  deno = 0.5*tanh(y_span[i+1])-ci_correct*iota;

  dphydy_guess[1]= (k*k-(num/deno))*phy_guess[0];

//corrector step: average the slope at two ends

 for( int j=0;j<col_phy;j++)
 {
  dphydy[j]=(dphydy_guess[j]+dphydy[j])*0.5;
 }

//calculate y(i+1) with average slope
```

```
 for( int j=0;j<col_phy;j++)
 {
  phy[i+1][j]=phy[i][j]+dphydy[j]*dy;
 }

}

/*--clean up intermediate variables--*/
delete [] dphydy;
delete [] phy_guess;
delete [] dphydy_guess;

//--> create wavenumber folder
 char command[50];
 sprintf(command,"mkdir /home/vikas/Desktop/AS_5430_project_ME15D080/output/%d",count+1);
 system (command);

//--> write eigen functions

 char fname1[50];
 sprintf(fname1, "/home/vikas/Desktop/AS_5430_project_ME15D080/output/%d/%s",count+1,ch1);
 ofstream fout1;
 fout1.open(fname1);

 for(int i = 0; i < row_phy; i++)
    {
     fout1.precision(8);
     fout1 << y_span[i] << " "<<real(phy[i][0]) << " " << imag(phy[i][0])
                        << " "<<real(phy[i][1]) << " " << imag(phy[i][1]) << "\n";
    }

 fout1.close();

}
/*---------- wavenumber-loop ends here-------------*/

//--> write dispersion data
 char fname2[50];
 sprintf(fname2, "/home/vikas/Desktop/AS_5430_project_ME15D080/output/%s",ch2);
 ofstream fout2;
 fout2.open(fname2);

 for(int i = 0; i < 9; i++)
    {
     fout2.precision(8);
     fout2 << k_range[i] << " " << k_range[i]*ci_range[i] <<" " << ci_range[i] << "\n";
    }

fout2.close();

/*----clean up-------*/

 delete [] k_range ;
 delete [] ci_range;
 delete [] Z_span  ;
 delete [] y_span  ;
 delete [] PHY_init;
 delete [] phy_init;

 for (int  i = 0; i < row_PHY; i++)
    {
     delete [] PHY[i];
    }
 delete [] PHY;
 for (int  i = 0; i < row_phy; i++)
    {
     delete [] phy[i];

    }
 delete [] phy;
```

```cpp
/*--end of program--*/
return 0;
}

/*-----------------------------------*/
/*---- function description----------*/
/*-----------------------------------*/


double** rk2_riccati (double** y, double* y_init, double* x_span, double k, double ci,double dx,int
row, int col)
{

/*--specify initial condition--*/

for (int j=0;j<col;j++ )
{
y[0][j]=y_init[j];
}

/*--intermediate variables--*/
double* dydx;
double* y_guess;
double* dydx_guess;

dydx       = new double[col];
y_guess    = new double[col];
dydx_guess = new double[col];

/*--integrate using Heun's method (RK2)--*/

for (int i=0;i<row-1;i++)
{
//See eq 18 [Riccati equation]  Pg 546, Michalke

  dydx[0]= ((k*k + y[i][1]*y[i][1]-y[i][0]*y[i][0])/(1-x_span[i]*x_span[i]))
           -((2*x_span[i]*x_span[i])/(x_span[i]*x_span[i]+4*ci*ci));

  dydx[1]= -((2*y[i][0]*y[i][1])/(1-x_span[i]*x_span[i]))
             -4*ci*x_span[i]/(x_span[i]*x_span[i]+4*ci*ci);

//predictor step1: guess the value of y(i+1)
  for(int j=0;j<col;j++)
  {
   y_guess[j]=y[i][j]+dydx[j]*dx;
  }

//predictor step2: guess the value of dydx(i+1) using Eq.18 [Riccati equation]

  dydx_guess[0]= ((k*k + y_guess[1]*y_guess[1]-y_guess[0]*y_guess[0])/(1-x_span[i+1]*x_span[i+1]))
                -((2*x_span[i+1]*x_span[i+1])/(x_span[i+1]*x_span[i+1]+4*ci*ci));
  dydx_guess[1]=-((2*y_guess[0]*y_guess[1])/(1-x_span[i+1]*x_span[i]))
                -4*ci*x_span[i+1]/(x_span[i+1]*x_span[i+1]+4*ci*ci);

//corrector step: average the slopes at two ends

 for( int j=0;j<col;j++)
 {
  dydx[j]=(dydx_guess[j]+dydx[j])*0.5;
 }

//calculate y(i+1) with average slope
 for( int j=0;j<col;j++)
 {
  y[i+1][j]=y[i][j]+dydx[j]*dx;
 }

}


/*--clean up intermediate variables--*/
```

```
delete [] dydx;
delete [] y_guess;
delete [] dydx_guess;


return y;
}
```

```
%-----------------------------------------------------------------------------%
% AS5430 COURSE-PROJECT                                                       %
% LINEAR INVISCID INSTABILITY STUDY OF TANH VELOCITY PROFILE                  %
% GNU OCTAVE / MATLAB POST-PROCESSING FILE                                    %
%-----------------------------------------------------------------------------%

%--------plot 1: temporal growth rate vs wavenumber--------------------------%
% data:
file1  = '/home/vikas/Desktop/AS_5430_project_ME15D080/output/disp_data';
disp_data = load(file1,'-ascii');
k  = [0; disp_data(:,1);1]; K = linspace(0,1); % wavenumber
g  = [0; disp_data(:,2);0]; G = spline(k,g,K); % temporal growth

% figure:
f1 = figure(1);
W = 4; H = 4;
set(f1,'PaperUnits','inches');set(f1,'PaperOrientation','portrait');
set(f1,'PaperSize',[H,W])    ;set(f1,'PaperPosition',[0,0,W,H]);
plot(K,G,'k-','LineWidth',1.5);axis([0,1,0,0.1]);
xlabel('\kappa');ylabel('\kappa c_{i}');
print(f1,'-deps','-color','plot 1: disp_data.eps');


%--------plot 2 & 3: phi_eigen_fcns vs y ------------------------------------%
% data
for count = 2:2:8
  index=count/2;
  temp_file  = sprintf('/home/vikas/Desktop/AS_5430_project_ME15D080/output/%d/eig_fcn', count);
  PHI_EIG_DATA {index}=load(temp_file,'-ascii');
end

% figures:
%--------------------plot -2 real_phi ----------------------------------------%
f2 = figure(2);
W = 5; H = 3;
set(f2,'PaperUnits','inches');set(f2,'PaperOrientation','portrait');
set(f2,'PaperSize',[H,W])    ;set(f2,'PaperPosition',[0,0,W,H]);

for i= 1:4
 eig_data=PHI_EIG_DATA{i};
 plot(eig_data(:,1),eig_data(:,2),'k-','LineWidth',1.5);axis([0,7,0,1.2]);
 xlabel('y');ylabel('\phi_{r}');
 print(f2,'-deps','-color','plot 2: phi_r.eps');
 hold on;
end
 hold off;

%--------------------plot -3 imag_phi-----------------------------------------%
f3 = figure(3);
W = 5; H = 3;
set(f3,'PaperUnits','inches');set(f3,'PaperOrientation','portrait');
set(f3,'PaperSize',[H,W])    ;set(f3,'PaperPosition',[0,0,W,H]);

for i= 1:4
 eig_data=PHI_EIG_DATA{i};
 plot(eig_data(:,1),eig_data(:,3),'k-','LineWidth',1.5);axis([0,7,0,1.2]);
 xlabel('y');ylabel('\phi_{i}');
 print(f3,'-deps','-color','plot 3: phi_i.eps');
 hold on;

end

 hold off;
%-----------------plot -4 & 5 omg_eigen_fcns vs y----------------------------%
% data
iota=sqrt(-1);
for i = 1:4
  eig_data = PHI_EIG_DATA{i};
  lambda   = (tanh(eig_data(:,1)).*(sech(eig_data(:,1))).^2)./(0.5*tanh(eig_data(:,1))-iota*disp_data
(2*i,3));
  OMG_DATA{i} = lambda.*(eig_data(:,2)+iota*eig_data(:,3));
end
```

```
%----------------- plot-4 real_omg ----------------------------------------%
f4 = figure(4);
W = 5; H = 3;
set(f4,'PaperUnits','inches');set(f4,'PaperOrientation','portrait');
set(f4,'PaperSize',[H,W])    ;set(f4,'PaperPosition',[0,0,W,H]);

for i= 1:4
 omg_data= OMG_DATA{i};
 plot(eig_data(:,1),real(omg_data),'k-','LineWidth',1.5);axis([0,1.8,0,2]);
 xlabel('y');ylabel('\omega_{r}');
 print(f4,'-deps','-color','plot 4: omg_r.eps');
 hold on;
end
 hold off;
%----------------- plot-5 imag_omg ---------------------------------------%
f5 = figure(5);
W = 5; H = 3;
set(f5,'PaperUnits','inches');set(f5,'PaperOrientation','portrait');
set(f5,'PaperSize',[H,W])    ;set(f5,'PaperPosition',[0,0,W,H]);

for i= 1:4
 omg_data= OMG_DATA{i};
 plot(eig_data(:,1),imag(omg_data),'k-','LineWidth',1.5);axis([0,1.8,0,1.2]);
 xlabel('y');ylabel('\omega_{i}');
 print(f5,'-deps','-color','plot 5: omg_i.eps');
 hold on;
end
 hold off;
%-------------------screen display-----------------------------------------%
% DATA FOR MAX AMPLIFIED WAVE
 wave_number = disp_data(:,1);ww=linspace(disp_data(1,1),disp_data(end,1));
 temp_growth = disp_data(:,2);gg=spline(wave_number,temp_growth,ww);
 init_cond   = disp_data(:,3);ic=spline(wave_number,init_cond,ww);

 [val,ind] = max(gg);
 disp('%---------------------------------------------------------------%')
 disp('MOST AMPLIFIED WAVENUMBER = ');disp(ww(ind));
 disp('MAX GROWTH RATE = ')          ;disp(gg(ind));
 disp('CORRESPONDING Ci= ')          ;disp(gg(ind)/ww(ind));
 disp('%---------------------------------------------------------------%')
%----------plot 6 & 7 streamlines and vorticity contours--------------------%
% data
  max_file  = sprintf('/home/vikas/Desktop/AS_5430_project_ME15D080/k_max_data/k_max/eig_fcn');
  max_k     = load(max_file,'-ascii');
  max_k_grid= max_k;
  for i = 1:701
      max_k_grid(i,:)=max_k(702-i,:);
  end

% make grid
  x =linspace(0,14,700);y =linspace(2.2,-5,701);
  [X,Y]=meshgrid(x,y);
  epsilon1=0.1;k0=0.4478;ci0=0.2120;
  lamb   = (tanh(max_k_grid(:,1)).*(sech(max_k_grid(:,1))).^2)./(0.5*tanh(max_k_grid(:,1))-iota*ci0);
  phi_r= zeros(701,700);phi_i= zeros(701,700);LAMB=zeros(701,700);
  for j=1:700
      phi_r(:,j)=max_k_grid(:,2);
      phi_i(:,j)=max_k_grid(:,3);
      LAMB (:,j)=lamb;
  end
  omg = LAMB.*(phi_r+iota*phi_i);
  str_fcn =   Y + 0.5*log(1+exp(-2*Y))+  epsilon1*(phi_r.*cos(k0*X)-phi_i.*sin(k0*X));
  vort_fcn= -0.5*sech(Y).*sech(Y)     +2*epsilon1*((real(omg).*cos(k0*X))-(imag(omg).*sin(k0*X)));

%----------------- plot-6 streamlines -----------------------------------------%
f6 = figure(6);
W = 5; H = 3;
set(f6,'PaperUnits','inches');set(f6,'PaperOrientation','portrait');
set(f6,'PaperSize',[H,W])    ;set(f6,'PaperPosition',[0,0,W,H]);
contour(X,Y,str_fcn,50,'LineWidth',1.5);axis([0,14,-5,0.5]);
xlabel('x');ylabel('y');
print(f6,'-deps','plot 6: phi_max.eps');
```

```
%------------------ plot-7 vorticity contours ----------------------------------------%

f7 = figure(7);
W = 5; H = 3;
set(f7,'PaperUnits','inches');set(f7,'PaperOrientation','portrait');
set(f7,'PaperSize',[H,W])    ;set(f7,'PaperPosition',[0,0,W,H]);
contour(X,Y,vort_fcn,5,'LineWidth',1.5);axis([0,14,-2,2]);
xlabel('x');ylabel('y');
print(f7,'-deps','plot 7: vort_max.eps');

%---------------------------------------------------------------------------------------%
%  STREAMLINES AND VORTICITY CONTOURS FOR NEUTRAL CASE WITH KNOWN ANALYTIC SOLUTION    %
%---------------------------------------------------------------------------------------%

%------------------ plot-8 streamlines ----------------------------------------%
x =linspace(0,14);y =linspace(0.5,-5);
[X,Y]=meshgrid(x,y);eps_1=0.1;
str_fcn=Y+0.5*log(1+exp(-2*Y))+eps_1*(sech(Y).*cos(k0*X));

f8 = figure(8);
W = 5; H = 3;
set(f8,'PaperUnits','inches');set(f8,'PaperOrientation','portrait');
set(f8,'PaperSize',[H,W])    ;set(f8,'PaperPosition',[0,0,W,H]);
contour(X,Y,str_fcn,50,'LineWidth',1.5);axis([0,14,-5,0.5]);
xlabel('x');ylabel('y');
print(f8,'-deps','plot 6: phi_neutral.eps');

%------------------ plot-9 vorticity contours ---------------------------------%
x =linspace(0,14);y =linspace(2,-2);
[X,Y]=meshgrid(x,y);eps_2=0.2;
vort_fcn=-0.5*sech(Y).^2+ eps_2*(2*(sech(Y)).^3.*cos(X));
f9 = figure(9);
W = 3; H = 4;
set(f9,'PaperUnits','inches');set(f9,'PaperOrientation','portrait');
set(f9,'PaperSize',[H,W])    ;set(f9,'PaperPosition',[0,0,W,H]);
contour(X,Y,vort_fcn,5,'LineWidth',1.5);axis([0,6,-2,2]);
xlabel('x');ylabel('y');
print(f9,'-deps','plot 7: vort_neutral.eps');
```

```
%-----------------------------------------------------%
% Orr Sommerfeld equation for temporal Mixing Layer  %
%-----------------------------------------------------%
clc;clear
N=121;
R=25000;
K=0.446;
% Spectral collocation points Z=cos(m*pi/(N-1)) for m=0..N-1
for m=N-2:-1:2;
Z=cos(m*pi/(N-1));
cheb=zeros(5,N);% contains Chebyschev polynomial and derivs
% Cheb Poly @ Z
cheb(1,1)=1.0;
cheb(1,2)=Z;
for jj=2:N-1;% 1 to n-2
cheb(1,jj+1)=2.0*Z*cheb(1,jj)-cheb(1,jj-1);
end
% Cheb derivs @ Z
for ii =2:5;
cheb(ii,1)=0.0;
cheb(ii,2)=cheb(ii-1,1);
cheb(ii,3)=4.0*cheb(ii-1,2);
for jj =4:N;
cheb(ii,jj)=2.0*(jj-1.0)*cheb(ii-1,jj-1)+(jj-1.0)/(jj-3.0)*cheb(ii,jj-2);
end
end
% Base flow @ Z
U    = 0.5*(1+Z);% Z=tanh(y)
dU   = 0.5;
d2U  = 0.0;
% Mapping variables
y      =  atanh(Z);
dZdy   = (sech(y))^2;
d2Zdy2 = -2*tanh(y)*(sech(y))^2;
d3Zdy3 =  2*(cosh(2*y)-2)*(sech(y))^4;
d4Zdy4 = -2*(sinh(3*y)-11*sinh(y))*(sech(y))^5;
% A and B of Aa=cBa
for jj=1:N;

A(N-m,jj) =   U*((dZdy^2)*cheb(3,jj)+(d2Zdy2)*cheb(2,jj)-K^2*cheb(1,jj))...
            - ((dZdy^2)*d2U+(d2Zdy2)*dU)*cheb(1,jj)...
            -1.0/(i*K*R)*(  cheb(5,jj)*(dZdy^4)...
                           +cheb(4,jj)*(6*d2Zdy2*dZdy^2)...
                           +cheb(3,jj)*(4*dZdy*d3Zdy3+3*d2Zdy2*dZdy^2 -2*K*K*dZdy^2)...
                           +cheb(2,jj)*(d4Zdy4-2*K*K*(d2Zdy2))...
                           +cheb(1,jj)*(K^4));


B(N-m,jj) = ((dZdy^2)*cheb(3,jj)-(d2Zdy2)*cheb(2,jj)-K^2*cheb(1,jj));
end
end
% BCs
for j=1:N;
A(1,j)=1.0;
A(2,j)=(j-1.0)^2.0;
A(N-1,j)=(-1.0)^(j-2.0)*(j-1.0)^2.0;
A(N,j)=(-1.0)^(j-1.0);
B(1,j)=0.0;
B(2,j)=0.0;
B(N-1,j)=0.0;
B(N,j)=0.0;
end
% find eigenvalues c
[V,D]=eig(A,B);
for j=1:N;
real_c(j)=real(D(j,j));
imag_c(j)=imag(D(j,j));
end

disp('max growth rate')
disp(K*max(imag_c(1:end-4)));
```

```
f1 = figure(1);
temp  = sprintf('%f@%f.eps',K,R);
W = 4;H = 5;
set(f1,'PaperUnits','inches');set(f1,'PaperOrientation','portrait');
set(f1,'PaperSize',[H,W])    ;set(f1,'PaperPosition',[0,0,W,H]);
plot(real_c,imag_c,'ko',[0 1],[0 0]);axis([0 1 -1 0.5])
xlabel('c_{r}');ylabel('c_{i}');
print(f1,'-deps','-color',temp);
```