Plane Poiseuille flow

. Supriya Karmakar

1. ORR-SOMMEREFLD-SQUIRE EQUATIONS

The linearized Navier-Stokes disturbance equations for plane Poiseuille flow leads to velocity-vorticity formulation $(\widehat{v} - \widehat{\eta})$, which is well known as Orr-Sommerfeld-Squire equations. The equations on $-1 < y < 1$ read

$$\left.\begin{aligned} \left[i\alpha U\left(D^2-k^2\right) - i\alpha U'' - \frac{1}{Re}\left(D^2-k^2\right)^2\right]\widehat{v} = i\omega\left(D^2-k^2\right)\widehat{v} \\ i\beta U'\widehat{v} + \left[i\alpha U - \frac{1}{Re}\left(D^2-k^2\right)\right]\widehat{\eta} = i\omega\widehat{\eta}. \end{aligned}\right\} \tag{1.1}$$

Here "$'$" and D denotes the derivative w.r.t y–coordinate. The boundary condition at the impermeable walls given by $\widehat{v} = D\widehat{v} = \widehat{\eta} = 0$ at $y = \pm 1$.

2. NUMERICAL APPROACH

The variables dependent on y–direction have been numerically implemented through the use of Chebyshev polynomials on a grid of Gauss-Lobatto nodes. Gauss-Lobatto nodes on y–coordinate:

$$y_j = \cos\left(\frac{j\pi}{N}\right), \quad j = 0, 1, 2, \cdots, N \tag{2.1}$$

The Chebyshev polynomials are defined as,

$$T_n(y) = \cos(n\theta), \ \theta = \cos^{-1}(y) \tag{2.2}$$

which (Eq. (2.2)) is the trigonometric form of the Chebyshev polynomials. They can be implemented using the following recurrence relation:

$$\left.\begin{aligned} T_0 &= 1 \\ T_1 &= y \\ T_{n+1} &= 2yT_n - T_{n-1} \end{aligned}\right\} \tag{2.3}$$

We will approximate the dependent variables in Eq. (1.1) on the Gauss-Lobatto points in Eq. (2.1) i.e., expand $\widehat{v}$ and $\widehat{\eta}$ on y in terms of Chebyshev expansions,

$$\widehat{v}(y_j) = \sum_{n=0}^{N} a_n T_n(y_j), \quad \widehat{\eta}(y_j) = \sum_{n=0}^{N} b_n T_n(y_j), \tag{2.4}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$'s are Chebyshev coefficient vector of size $N+1$, and $T_n(y)$ is the n–th Chebyshev polynomial. Note that discretization of $\widehat{v}$ in y–coordinate leads to the following linear system,

$$\underbrace{\begin{pmatrix} \widehat{v}(y_0) \\ \vdots \\ \widehat{v}(y_n) \\ \vdots \\ \widehat{v}(y_N) \end{pmatrix}}_{N+1\times 1} = \underbrace{\begin{pmatrix} T_0(y_0) & \cdots & T_n(y_0) & \cdots T_N(y_0) \\ \vdots & & \vdots & \vdots \\ T_0(y_n) & \cdots & T_n(y_n) & \cdots T_N(y_n) \\ \vdots & & \vdots & \vdots \\ T_0(y_N) & \cdots & T_n(y_N) & \cdots T_N(y_N) \end{pmatrix}}_{N+1\times N+1} \underbrace{\begin{pmatrix} a_0 \\ \vdots \\ a_n \\ \vdots \\ a_N \end{pmatrix}}_{N+1\times 1} \tag{2.5}$$

This is in more compact notation, $\{\widehat{v}\} = [\mathcal{D}_0]\{a\}$. Thus the velocity-vorticity formulation (say $\widehat{q} = [\widehat{v} \quad \widehat{\eta}]^T$) can be written as,

$$\widehat{q} = \begin{pmatrix} \widehat{v} \\ \widehat{\eta} \end{pmatrix} = \begin{pmatrix} [\mathcal{D}_0] & 0 \\ 0 & [\mathcal{D}_0] \end{pmatrix} \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{pmatrix} \tag{2.6}$$

It is also to be noted from that the Eq. (1.1) consists of higher order derivative of the dependent variables (e.g. $D\widehat{v}$,$D^2\widehat{v}, \cdots$ etc.) and hence the corresponding higher order derivative variables,

$$\left.\begin{aligned} \widehat{v}'(y_j) &= \sum_{n=0}^{N} a_n T_n'(y_j) \rightarrow \{D\widehat{v}\} = [\mathcal{D}_1]\{\boldsymbol{a}\} \\ \widehat{\eta}'(y_j) &= \sum_{n=0}^{N} b_n T_n'(y_j) \rightarrow \{D\widehat{\eta}\} = [\mathcal{D}_1]\{\boldsymbol{b}\} \\ \widehat{v}''(y_j) &= \sum_{n=0}^{N} a_n T_n''(y_j) \rightarrow \left\{D^2\widehat{v}\right\} = [\mathcal{D}_2]\{\boldsymbol{a}\} \\ \widehat{\eta}''(y_j) &= \sum_{n=0}^{N} b_n T_n''(y_j) \rightarrow \left\{D^2\widehat{\eta}\right\} = [\mathcal{D}_2]\{\boldsymbol{b}\} \\ \widehat{v}''''(y_j) &= \sum_{n=0}^{N} a_n T_n''''(y_j) \rightarrow \left\{D^4\widehat{v}\right\} = [\mathcal{D}_4]\{\boldsymbol{a}\} \end{aligned}\right\} \tag{2.7}$$

Here the operators $\mathcal{D}_0, \mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_4$ contains the zeroth, first, second, and fourth order derivatives of the Chebyshev polynomial on the Gauss-Lobatto points y_j. Thus we define p–th order derivative of the Chebyshev polynomial $T_n^{(p)}(y)$ by the following recurrence relation,

$$\left.\begin{aligned} &T_0^{(p)}(y_j) = 0 \\ &T_1^{(p)}(y_j) = T_0^{(p-1)}(y_j), \\ &T_2^{(p)}(y_j) = 4T_1^{(p-1)}(y_j) \\ &T_n^{(p)}(y_j) = 2nT_{n-1}^{(p-1)}(y_j) + \frac{n}{n-1}T_{n-1}^{(p)}(y_j),\ n = 3, 4, \cdots \end{aligned}\right\} \tag{2.8}$$

Now all D, D^2, D^4 in Eq.(1.1) are implemented using Eq. (2.7). Hence from Eq. (1.1)

$$\begin{aligned} &\left[\alpha\{U(y_j)\}\left(\mathcal{D}_2 - k^2\mathcal{D}_0\right) - \alpha\{U''(y_j)\}\mathcal{D}_0 - \frac{1}{iRe}\left(\mathcal{D}_4 - 2k^2\mathcal{D}_2 + k^4\mathcal{D}_0\right)\right]\boldsymbol{a} \\ &\qquad\qquad\qquad = \omega\left(\mathcal{D}_2 - k^2\mathcal{D}_0\right)\boldsymbol{a} \end{aligned} \tag{2.9}$$

$$\{\beta U'(y_j)\}\mathcal{D}_0\boldsymbol{a} + \left[\alpha\{U(y_j)\}\mathcal{D}_0 - \frac{1}{iRe}\left(\mathcal{D}_2 - k^2\mathcal{D}_0\right)\right]\boldsymbol{b} = \omega\mathcal{D}_0\boldsymbol{b} \tag{2.10}$$

The boundary conditions are given by,

$$Eqs.\ (2.9) - (2.10)\widehat{v} = 0 \quad \text{at } y = \pm 1 \longrightarrow \sum_{n=0}^{N} a_n T_n(\pm 1) = 0 \text{ i.e., } \mathcal{D}_0(\pm 1)\boldsymbol{a} = 0 \tag{2.11}$$

$$D\widehat{v} = 0 \quad \text{at } y = \pm 1 \longrightarrow \sum_{n=0}^{N} a_n T_n'(\pm 1) = 0 \text{ i.e., } \mathcal{D}_1(\pm 1)\boldsymbol{a} = 0 \tag{2.12}$$

$$\widehat{\eta} = 0 \quad \text{at } y = \pm 1 \longrightarrow \sum_{n=0}^{N} b_n T_n(\pm 1) = 0 \text{ i.e., } \mathcal{D}_0(\pm 1)\boldsymbol{b} = 0 \tag{2.13}$$

The final result from Eqs. (2.9)-(2.10) is a generalized eigevalue problem,

$$\mathcal{A}\boldsymbol{q} = \omega\mathcal{B}\boldsymbol{q}, \quad \text{where } \boldsymbol{q} = [\boldsymbol{a} \quad \boldsymbol{b}]^T \tag{2.14}$$

The boundary conditions (2.11) and (2.12) are implemented by removing first-second and last-second last row of the Orr-Sommerfeld matrix, and similarly (2.13) is implemented by replacing the first and last row of the Squire matrix. (The procedure is outlined in Scmidt and Henningson book). *The boundary rows of $\mathcal{A}$ can be chosen as a complex multiple of the corresponding rows in $\mathcal{B}$. By carefully selecting this complex multiple, the spurious modes associated with the boundary conditions can be mapped to an arbitrary location in the complex plane. Otherwise if $\mathcal{B}$ consists of some zero rows due to zero boundary conditions, then $\mathcal{B}^{-1}$ does not exist, and due to badly conditioned system spurious mode will appear.* The MATLAB code using Chebyshev colocation method is given in Appendix A.

.

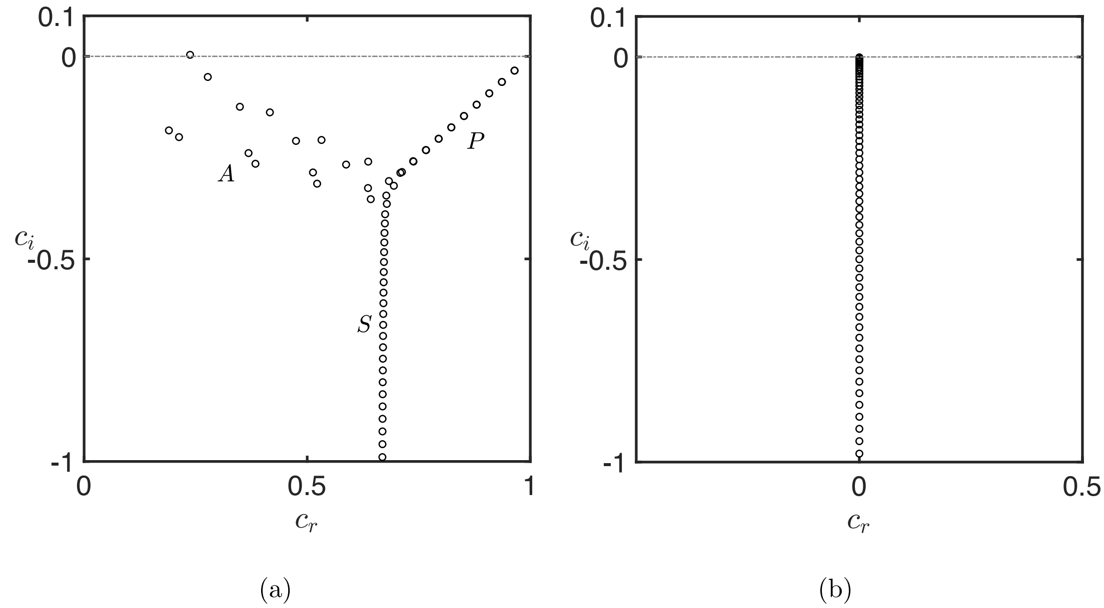

FIG. 1. Orr-Sommerfeld spectrum for plane Poiseuille flow for $Re = 10000$: **(a)** $\alpha = 1,\ \beta = 0$, **(b)** $\alpha = 0,\ \beta = 1$.

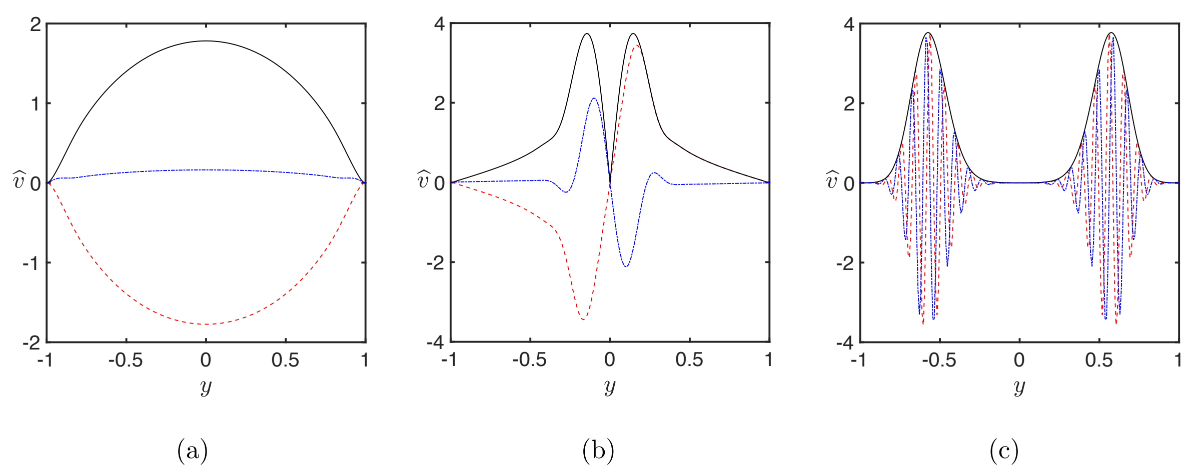

FIG. 2. Orr-Sommerfeld eigenfunction for a eigenvalue at **(a)** A branch (Wall-mode), **(b)** P branch (Center mode), **(c)** S branch of Fig. 1(a).

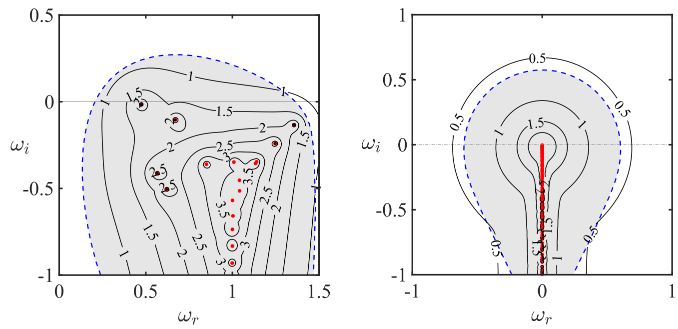

FIG. 3. Caption

Appendix A: MATLAB code for eigenvalue and eigenfunction calculation of plane Poiseuille flow

MATLAB driver program

```
clc
clear
close all

%   Input parameters
    N    =100;
    Re   =5772;
    alp  =1.02;
    beta=0;

%   Chebyshev matrix
    [D0,D1,D2,D3,D4]=Dmat(N);

%   Combined Matrices
    [A,B]=PlanePoiseuille(N,Re,alp,beta,D0,D1,D2,D4);
    os_evals=1:N+1;
    [v,d]=eig(A(os_evals,os_evals),B(os_evals,os_evals));
    eOS=diag(d);

%   Sorting the eigenvalues
    [~,is]=sort(-imag(eOS));
    vs=v(:,is);
    eOS=eOS(is);

    figure(1)
        plot(real(eOS),imag(eOS),'ko','MarkerSize',10);
        set(gca,"fontsize",20)
        ylim([-1 0.1]);
```

```
        xlim([0 1]);
        title('Orr-Sommerfeld eigenvalues', 'FontSize', 18);
        ylabel("$c_i$","interpreter","latex");
        xlabel("$c_r$","interpreter","latex");

 % Eigenfunction plotting of particular input eigenvalue
    yj  =cos(pi*(0:2*N)/(2*N));% Grid points
    ll =input(" Eigenvalue position which to be plotted: ");

    efn=cheb_expansion_soln(yj,vs(:,ll));
    figure(2)
        plot(yj,real(efn),'--r');
        hold on
        plot(yj,abs(efn),'-k');
        hold on
        plot(yj,imag(efn),'.b');
        hold off
        title([" Eigenfunction of eigenvalue:" num2str(eOS(ll))]);
        set(gca,"fontsize",20)
        ylabel("$\widehat{v}$","interpreter","latex");
        xlabel("$y$","interpreter","latex");
```

Chebyshev Differentiation matrix

```
function [D0,D1,D2,D3,D4]=Dmat(N)
    %
    % Function to create differentiation matrices
    %
    % N        = number of modes
    % D0       = zero'th derivative matrix
    % D1       = first derivative matrix
    % D2       = second derivative matrix
    % D4       = fourth derivative matrix
```

```
    % initialize
    num=round(abs(N));
    % create D0
    D0=[];
    vec=(0:1:num)';
    for j=0:1:num
        D0=[D0 cos(j*pi*vec/num)];
    end
    % create higher derivative matrices
    lv=length(vec);
    D1=[zeros(lv,1) D0(:,1) 4*D0(:,2)];
    D2=[zeros(lv,1) zeros(lv,1) 4*D0(:,1)];
    D3=[zeros(lv,1) zeros(lv,1) zeros(lv,1)];
    D4=[zeros(lv,1) zeros(lv,1) zeros(lv,1)];
    for j=3:num
        D1=[D1 2*j*D0(:,j)+j*D1(:,j-1)/(j-2)];
        D2=[D2 2*j*D1(:,j)+j*D2(:,j-1)/(j-2)];
        D3=[D3 2*j*D2(:,j)+j*D3(:,j-1)/(j-2)];
        D4=[D4 2*j*D3(:,j)+j*D4(:,j-1)/(j-2)];
    end
end
```

Computing combined Orr-Sommerfeld Squire matrices

```
function [A,B]=PlanePoiseuille(N,Re,alp,beta,D0,D1,D2,D4)
    % Matrices (A,B) for Generalized eigenvalue problem Ax=cBx
    % N    =number of colocation points
    % Re   =Reynolds number
    % alp = Streamwise wave number
    % beta=Spanwise wave number
    % D0   = Zeroth derivative matrix
    % D1   = First derivative matrix
```

```
% D2  = Second derivative matrix
% D3  = Third derivative matrix
% D4  = Fourth derivative matrix

zi=sqrt(-1);  er=-200*zi;

k2=alp^2+beta^2;
[Nos,Nsq]=deal(N+1);
Z=zeros(Nos,Nsq);
vec=(0:1:N)';

%   Mean Velocity and Derivative of that
u=(ones(length(vec),1)-cos(pi*vec/N).^2);
du=-2*cos(pi*vec/N);

%   Setup Orr-Sommerfeld matrix
A11=-(D4-2*k2*D2+k2^2*D0)/(zi*Re);
A11=A11+alp*(u*ones(1,length(u))).*(D2-k2*D0)+alp*2*D0;
A11=[er*D0(1,:);er*D1(1,:);A11(3:Nos-2,:);er*D1(Nos,:);er*D0(Nos,:)];

B11=(D2-k2*D0);
B11=[D0(1,:);D1(1,:);B11(3:Nos-2,:);D1(Nos,:);D0(Nos,:)];

%   Setup the Squire matrix
A21=beta*(du*ones(1,length(u))).*D0;
A22=alp*(u*ones(1,length(u))).*D0-(D2-k2*D0)/(zi*Re);
B22=D0;
A22=[er*D0(1,:);A22(2:Nsq-1,:);er*D0(Nsq,:)];
A21=[Z(1,:);A21(2:Nsq-1,:);Z(1,:)];  % Boundary Conditions
```

```
    %   Combining all the blocks
    A=[A11 Z;
        A21 A22];
    B=[B11 Z;
        Z B22];
end
```

Chebyshev expansion

```
function val = cheb_expansion_soln(y,a)
    %function that evaluates the Chebyshev expansion for given coeffs
    % y = Gauss-Lobatto points
    % a = vector of coeffs for the Chebyshev expansion
    val = zeros(length(y),1);
    for i=1:length(y)
        for j=1:length(a)
            val(i) = val(i) + a(j)*cheb_basis(y(i),j-1);
        end
    end
end
```

Chebyshev polynomial

```
function val = cheb_basis(y,n)
    % Chebyshev Polynomial of order n evaluated at y
    % Tn(y)=cos(ncos^(-1)y)
    val = cos(n*acos(y));
end
```