Ch 8. Chebyshev Series and the FFT

Solving homework using Python

2020.12.02(전산천문학)

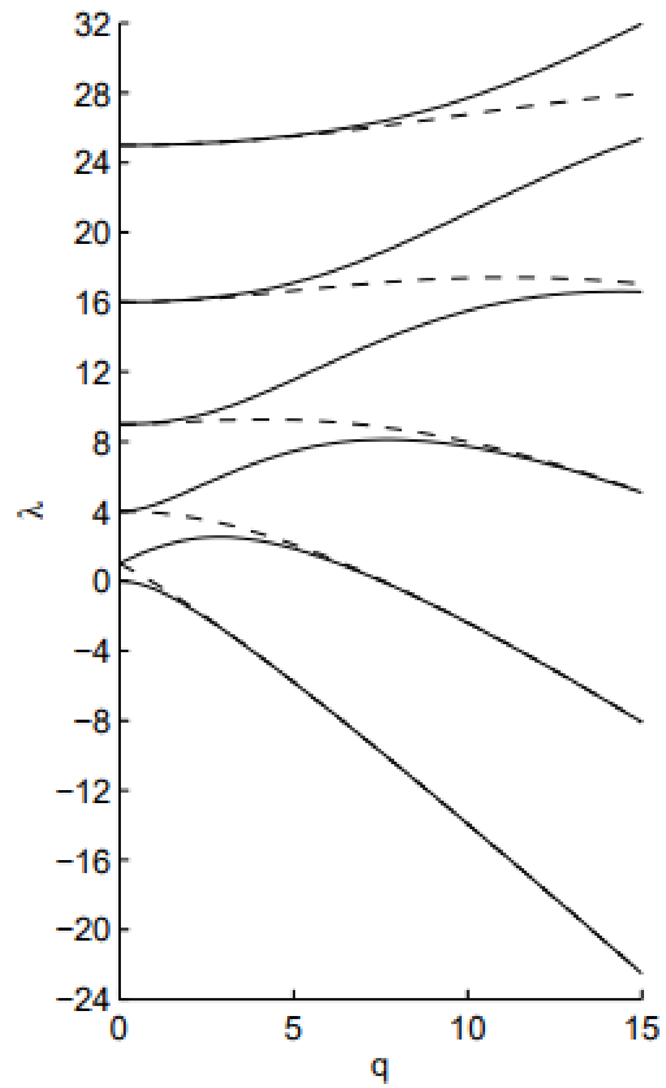
202050135 홍지민

Program1. Matlab Code

1. Implement Program 21 and produce a plot similar to Output 21.

Program 21

```
% p21.m - eigenvalues of Mathieu operator -u_xx + 2qcos(2x)u
%         (compare p8.m and p. 724 of Abramowitz & Stegun)

  N = 42; h = 2*pi/N; x = h*(1:N);
  D2 = toeplitz([-pi^2/(3*h^2)-1/6 ...
                -.5*(-1).^(1:N-1)./sin(h*(1:N-1)/2).^2]);
  qq = 0:.2:15; data = [];
  for q = qq;
    e = sort(eig(-D2 + 2*q*diag(cos(2*x))))';
    data = [data; e(1:11)];
  end
  clf, subplot(1,2,1)
  set(gca,'colororder',[0 0 1],'linestyleorder','-|--'), hold on
  plot(qq,data,'linewidth',.8), xlabel q, ylabel \lambda
  axis([0 15 -24 32]), set(gca,'ytick',-24:4:32)
```



Program1. Python Code

Periodic

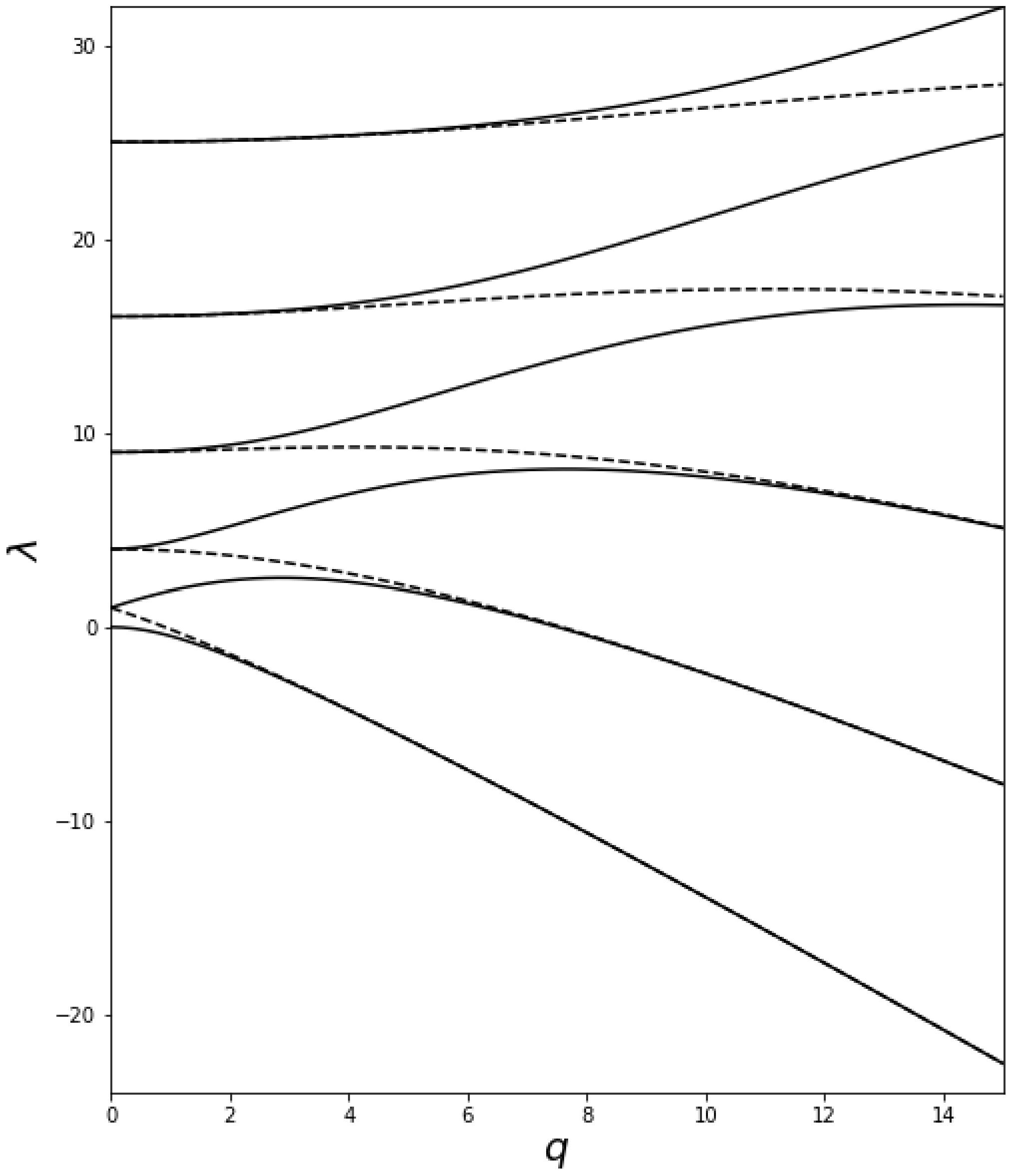
$$S_N''(x_j) = \begin{cases} -\dfrac{\pi^2}{3h^2} - \dfrac{1}{6} & j \equiv 0 \pmod{N}, \\ -\dfrac{(-1)^j}{2\sin^2(jh/2)} & j \not\equiv 0 \pmod{N}. \end{cases}$$

1-order spectral differentiation can be written in the m

$$D_N^{(2)} v = \begin{pmatrix} \ddots & \vdots & \\ \ddots & -\frac{1}{2}\csc^2\left(\frac{2h}{2}\right) & \\ \ddots & \frac{1}{2}\csc^2\left(\frac{h}{2}\right) & \\ & -\frac{\pi^2}{3h^2} - \frac{1}{6} & \\ & \frac{1}{2}\csc^2\left(\frac{h}{2}\right) & \ddots \\ & -\frac{1}{2}\csc^2\left(\frac{2h}{2}\right) & \ddots \\ & \vdots & \ddots \end{pmatrix} v.$$

```python
from scipy.linalg import toeplitz
from numpy.linalg import eigvals

N = 42 ; h = 2*np.pi/N ; x = h*np.arange(1,N +1,1)

A = (-np.pi**2)/(3*h**2)-1/6 ; B = -0.5*(-1)**np.arange(1,N,1)/np.sin(h*np.arange(1,N,1)/2)*
array = np.append(A,B)

D2 = toeplitz(array)
qq = np.arange(0,15 +0.2,0.2) ; data = []
for q in qq:
  e = sorted(eigvals(-D2 + 2*q*np.diagflat(np.cos(2*x))))
  data = np.append(data,e[0:11])

data = np.reshape(np.real(data),(-1,11))
```

$$L_N = -D_N^{(2)} + 2q\,\mathrm{diag}(\cos(2x_1), \ldots, \cos(2x_N)),$$



In Python

This equation is Mathieu eq.

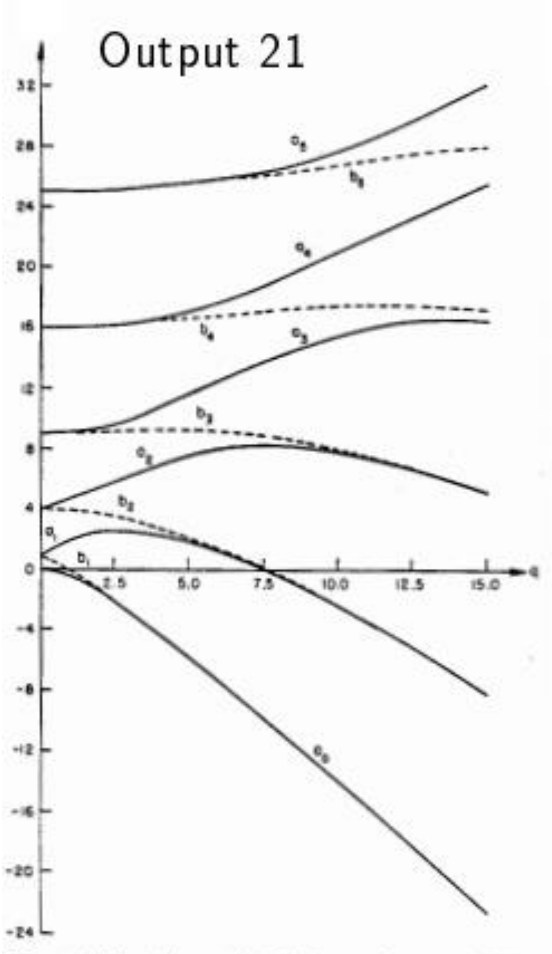
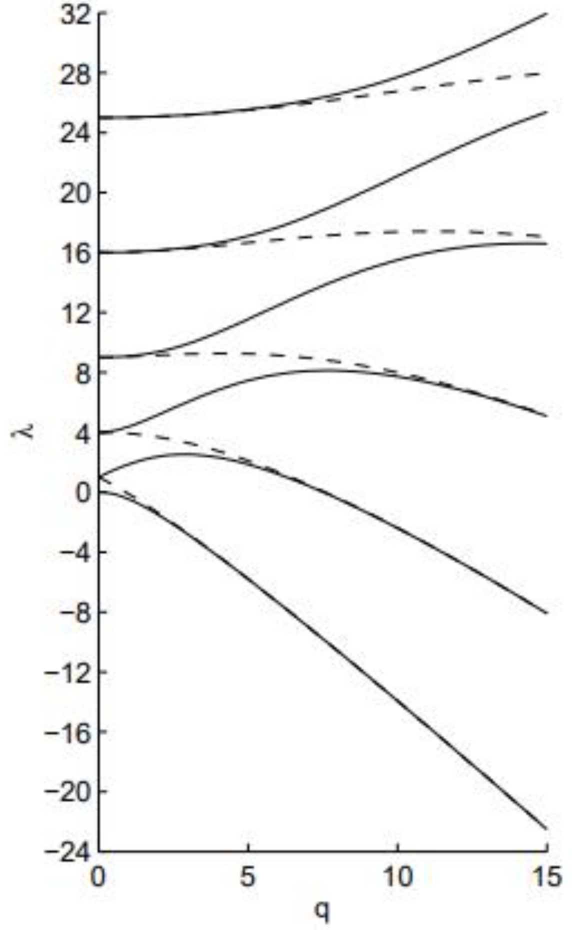


FIGURE 20.1. *Characteristic Values a_r, b_r* $r=0,1(1)5$



In Matlab

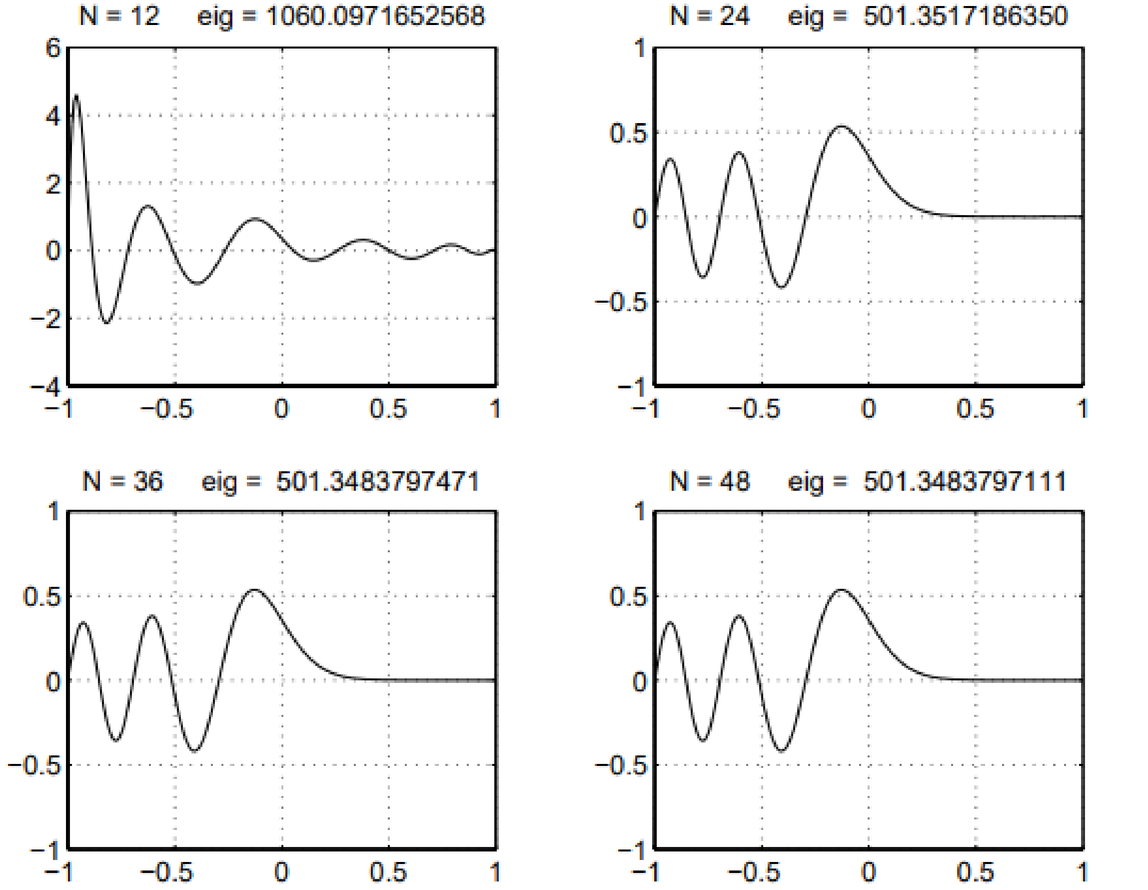
Program2. Matlab Code

2. Implement Program 22 and produce a plot similar to Output 22.

Program 22

```
% p22.m - 5th eigenvector of Airy equation u_xx = lambda*x*u

  clf
  for N = 12:12:48
    [D,x] = cheb(N); D2 = D^2; D2 = D2(2:N,2:N);
    [V,Lam] = eig(D2,diag(x(2:N)));      % generalized ev pr
    Lam = diag(Lam); ii = find(Lam>0);
    V = V(:,ii); Lam = Lam(ii);
    [foo,ii] = sort(Lam); ii = ii(5); lambda = Lam(ii);
    v = [0;V(:,ii);0]; v = v/v(N/2+1)*airy(0);
    xx = -1:.01:1; vv = polyval(polyfit(x,v,N),xx);
    subplot(2,2,N/12), plot(xx,vv,'linewidth',1), grid on
    title(sprintf('N = %d     eig = %15.10f',N,lambda))
  end
```

Output 22



Program2. Python Code

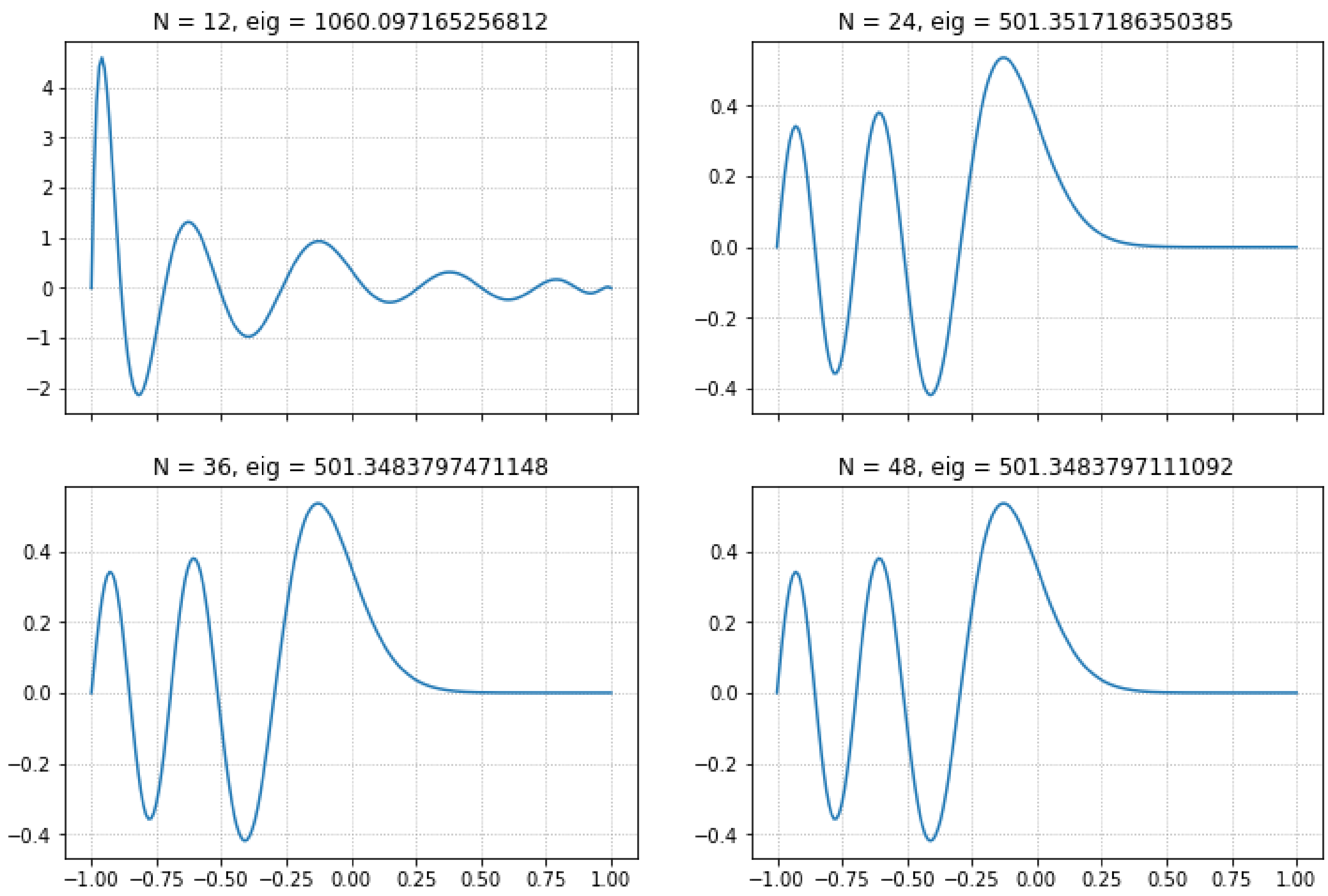
Set Tilda(?) Function

```
for N in np.arange(12,48 +12,12):
  D,x = cheb(N); D2 = np.delete(np.delete((D **2),[0,N],0),[0,N],1)
  X = np.diagflat(x[1:N])
  w,v = eig(D2,X)
  ii = np.where(w>0)[0] ; w = w[w > 0]
  v = v[:,ii] ; index = np.argsort(w)
  ii = np.where(index == 4)[0][0]
  v = np.append(np.insert(v[:,ii],0,0),0) ; w = w[ii]
  v = v/v[int(N/2)]*airy(0)[0]
  xx = np.arange(-1,1+0.01,0.01)
  vv = np.polyval(np.polyfit(x,v,N),xx)
```

$$Au = \lambda Bu, \qquad A = \tilde{D}_N^2, \quad B = \mathrm{diag}(x_0, \ldots, x_N).$$

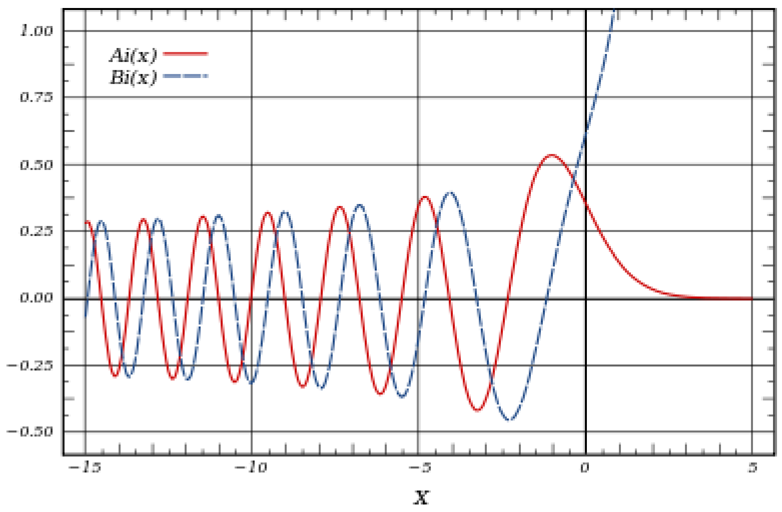
Calculate this text(?)

eigenvalue. This is equivalent to computing $\mathrm{Ai}(x)$ on the interval $[-L, L]$, where $-L = -7.944133\ldots$ is the location of the fifth zero of $\mathrm{Ai}(x)$. A rescaling back to $[-1, 1]$ then introduces a power $L^3 = 501.348\ldots$, and that is why our eigenvalue came out as L^3. Actually, what we have just said is not exactly



This equation is Airy eq.

$$\frac{d^2y}{dx^2} - xy = 0,$$



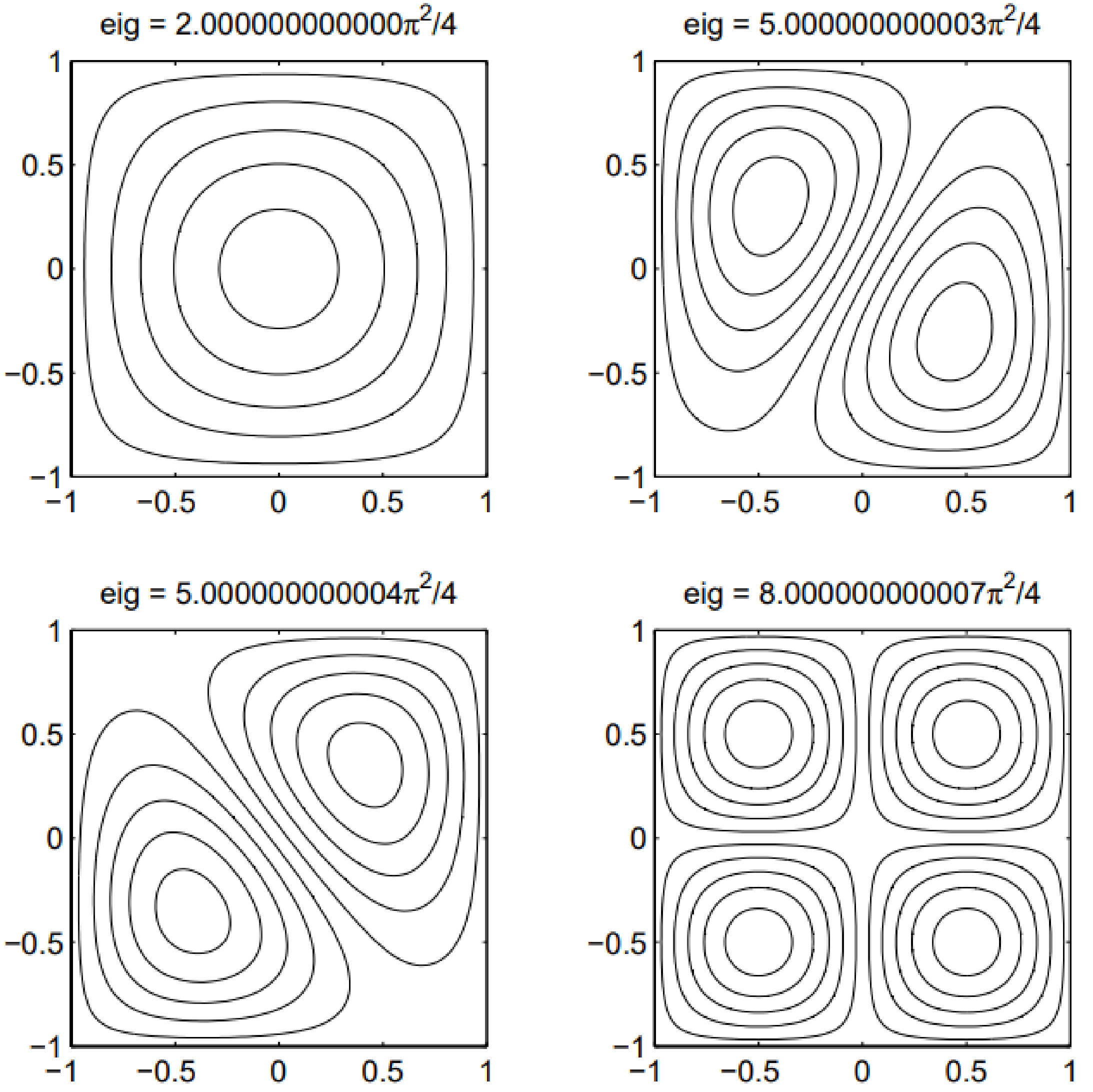
In the physical sciences, the **Airy function** (or **Airy function of the first kind**) **Ai(*x*)** is a special function named after the British astronomer George Biddell Airy (1801–1892). The function Ai(*x*) and the related function **Bi(*x*)**, are linearly independent solutions to the differential equation

Program3. Matlab Code

3. Implement Program 23 and produce a plot similar to Output 23a.

Program 23

```
% p23.m - eigenvalues of perturbed Laplacian on [-1,1]x[-1,1]
%         (compare p16.m)

% Set up tensor product Laplacian and compute 4 eigenmodes:
  N = 16; [D,x] = cheb(N); y = x;
  [xx,yy] = meshgrid(x(2:N),y(2:N)); xx = xx(:); yy = yy(:);
  D2 = D^2; D2 = D2(2:N,2:N); I = eye(N-1);
  L = -kron(I,D2) - kron(D2,I);                    % Laplacian
  L = L + diag(exp(20*(yy-xx-1)));                 % + perturbation
  [V,D] = eig(L); D = diag(D);
  [D,ii] = sort(D); ii = ii(1:4); V = V(:,ii);

% Reshape them to 2D grid, interpolate to finer grid, and plot:
  [xx,yy] = meshgrid(x,y);
  fine = -1:.02:1; [xxx,yyy] = meshgrid(fine,fine);
  uu = zeros(N+1,N+1);
  [ay,ax] = meshgrid([.56 .04],[.1 .5]); clf
  for i = 1:4
    uu(2:N,2:N) = reshape(V(:,i),N-1,N-1);
    uu = uu/norm(uu(:),inf);
    uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');
    subplot('position',[ax(i) ay(i) .38 .38])
    contour(fine,fine,uuu,-.9:.2:.9)
    colormap([0 0 0]), axis square
    title(['eig = ' num2str(D(i)/(pi^2/4),'%18.12f') '\pi^2/4'])
  end
```



Program3. Python Code

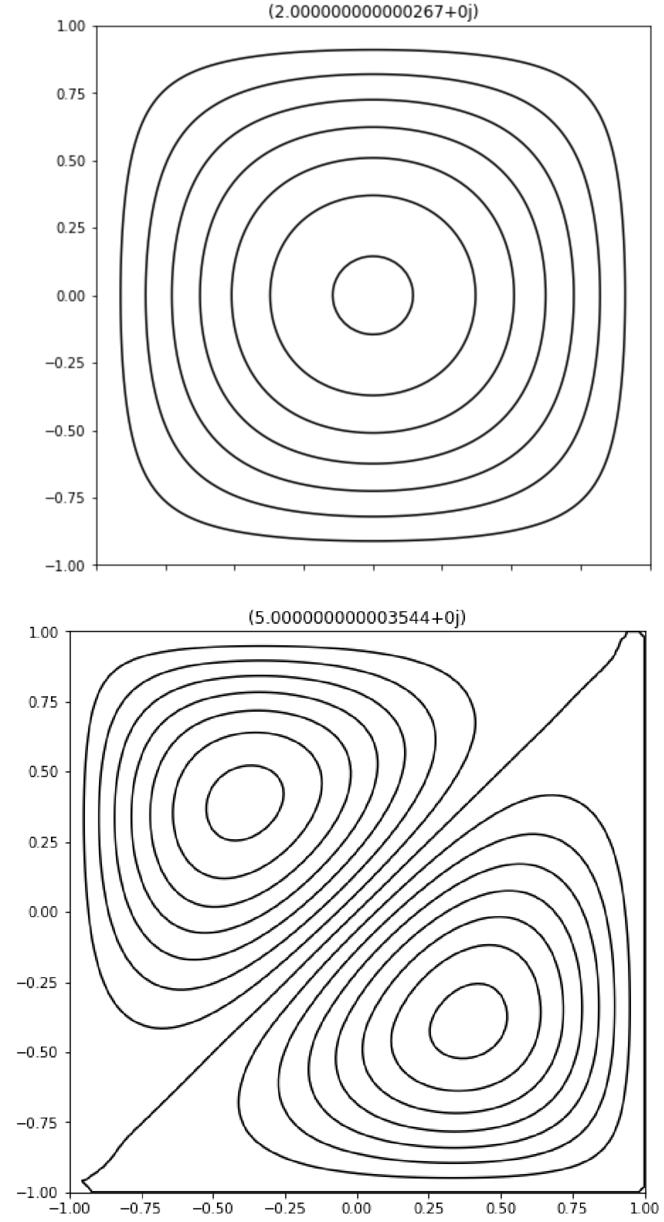
```
N = 16
D,x = cheb(N)
y = x
xx, yy = np.meshgrid(x[1:N],y[1:N])
xx = xx.flatten() ; yy = yy.flatten()
D2 = np.delete(np.delete((D **2),[0,N],0),[0,N],1)
I = np.eye(N-1)
L = -np.kron(I,D2) - np.kron(D2,I)
L = L
D,V = eig(L)
ii = np.argsort(D) ; D = sorted(D)
ii = ii[0:4] ; V = V[:,ii]

xx, yy = np.meshgrid(x,y)

xxx = np.arange(-1,1 +0.02,0.02)
yyy = np.arange(-1,1 +0.02,0.02)

xi, yi =np.meshgrid(xxx,yyy)
```

$$-\Delta u + f(x,y)u \;=\; \lambda u, \quad -1 < x,y < 1, \quad u = 0 \text{ on the boundary.}$$

Our discrete Laplacian is now

$$L_N \;=\; I \otimes \widetilde{D}_N^2 + \widetilde{D}_N^2 \otimes I.$$

Order eigenvalues and eigenvectors in order

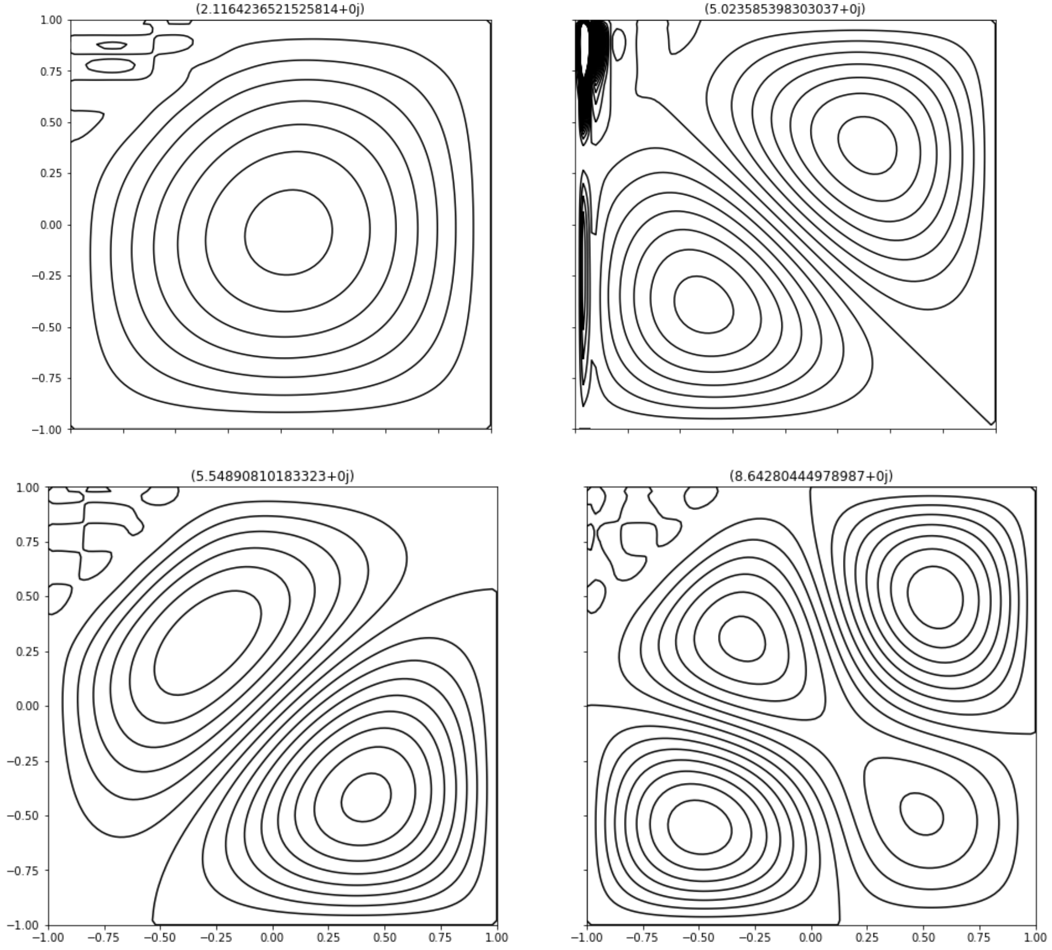


This equation is Laplacian eq.

Output 23a: First four eigenmodes of the Laplace problem (9.4) with $f(x, y) = 0$. These plots were produced by running Program 23 with the "+ perturbation" line commented out.

If a perturbation value is given,

$$f(x, y) = \exp(20(y - x - 1)).$$



Program4. Homework Code

4. (Exercise 9.5) Solve the generalized eigenvalue equation

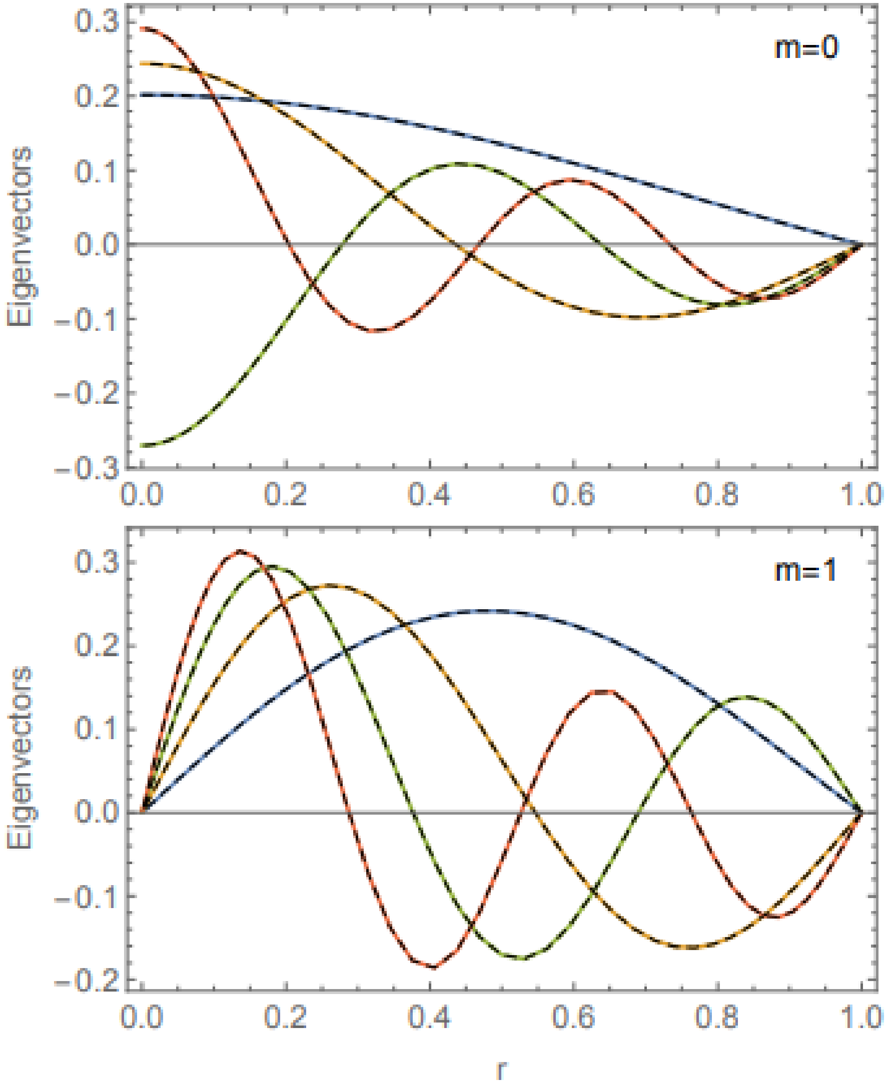
$$r^2 u_{rr} + r u_r - m^2 u = -\omega^2 r^2 u, \quad u_r(0) = u(1) = 0$$

for eigenvalues ω, assuming $m = 0, 1$ and $N = 50$.

- Note that the boundary conditions are mixed.

- Because $0 < r < 1$, be sure to make a change of variable. Don't forget to multiply appropriate scale factors to the first- and second-order differential matrices, $D_N^{(1)}$ and $D_N^{(2)}$.

- Side note: This is a form of *Bessel's equation*, and the solutions, $u(r)$, are Bessel functions,



Program4. Python Code

Calculate x to r

$$-1<x<1 \rightarrow 0<r<1$$

$$r=\frac{1}{2}(x+1),\quad \frac{d}{dr}=2\cdot\frac{d}{dx}$$

$$\left.\begin{aligned} u_{xx}&=\left(\frac{d}{dx}\right)^2 u \\ u_{rr}&=\left(\frac{d}{dr}\right)^2 u \end{aligned}\right\} \rightarrow u_{rr}=4\cdot\underbrace{\left(\frac{d}{dx}\right)^2 u}_{u_{xx}}$$

```python
N=50
D,x = cheb(N) ; r = (x+1)/2
D1 = D ; D1[0] = np.zeros(N+1) ; D1[0,0] = 1
D2 = (D**2) ; D2[0] = np.zeros(N+1) ; D2[0,0] = 1 ; D2[N] = D[N]
m = 0
r_matrix = np.diagflat(r)
m_matrix = np.diagflat(np.full_like(r,m))
L = 4*(r_matrix**2)*D2 + 2*r_matrix*D1 - m_matrix

input2 = np.insert(r[1:N+1],0,0)
B = -np.diagflat(input2**2)

w,v = eig(L,B)
ii = np.argsort(np.sqrt(w)) ; lamb = sorted(np.sqrt(w))
ii = ii[0:4] ; v = v[:,ii]
```

노이만 조건 $\underset{u(1)}{v_0}=\alpha$, $w_N=u_x(-1)=\beta$

$$1\cdot v_0+0\cdot v_1+0\cdot v_2=\alpha$$

$$\vdots$$

$$D'_{20}v_0+D'_{21}v_1+D'_{22}v_2=\beta$$

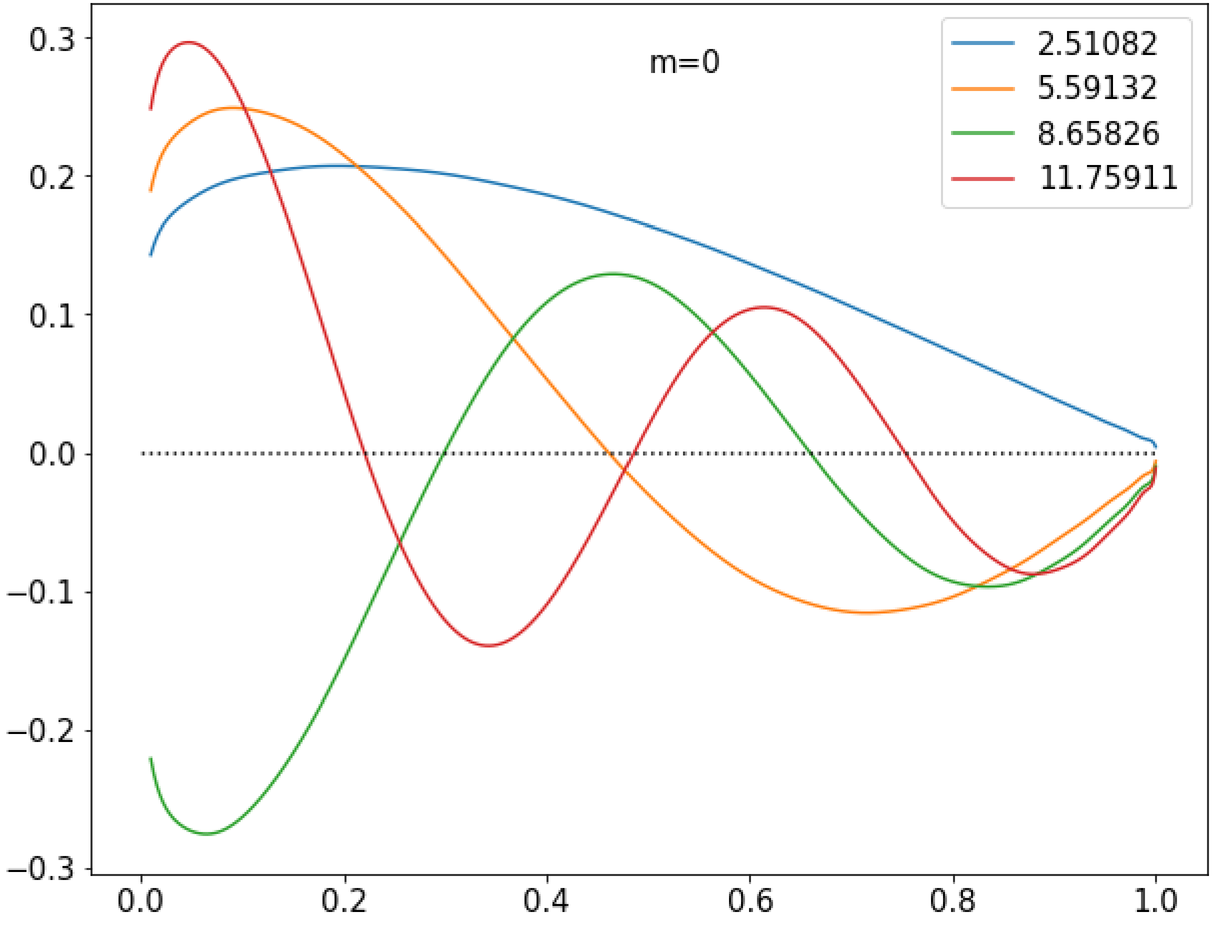
$u_x \approx D'_N v$

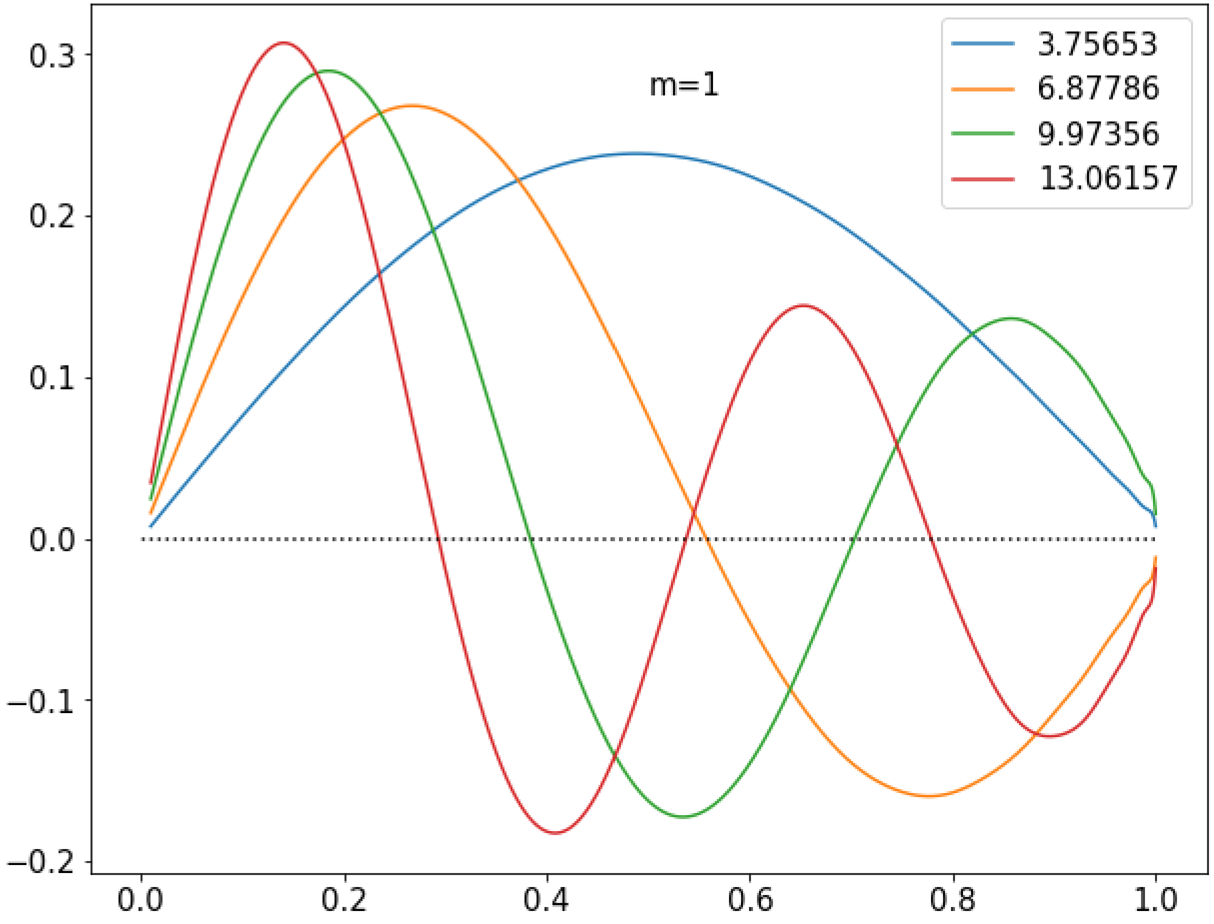
↳ $u(-1)=\beta$

$$\underbrace{\left(4r^2D_N^2+2rD_N-m^2\right)}_{L}v=\lambda Bv$$

$$\hookrightarrow \begin{pmatrix} 1 & 0 & 0 \\ D^2_{10} & D^2_{11} & D^2_{12} \\ D'_{20} & D'_{21} & D'_{22} \end{pmatrix}\begin{pmatrix} v_0 \\ v_1 \\ v_2 \end{pmatrix}=\begin{pmatrix} \alpha \\ f_1 \\ \beta \end{pmatrix}$$

Order eigenvalues and eigenvectors in order





When ‘r’ is near zero, it looks a little different from the graph of the homework(m = 0).

When r is near one, it looks a little fluctuation versus the graph of the homework.

It is assumed that there will be mistakes in calculating the matrix.