Ch 7. Boundary Value Problems

Solving homework using Python

2020.11.18(전산천문학)

202050135 홍지민

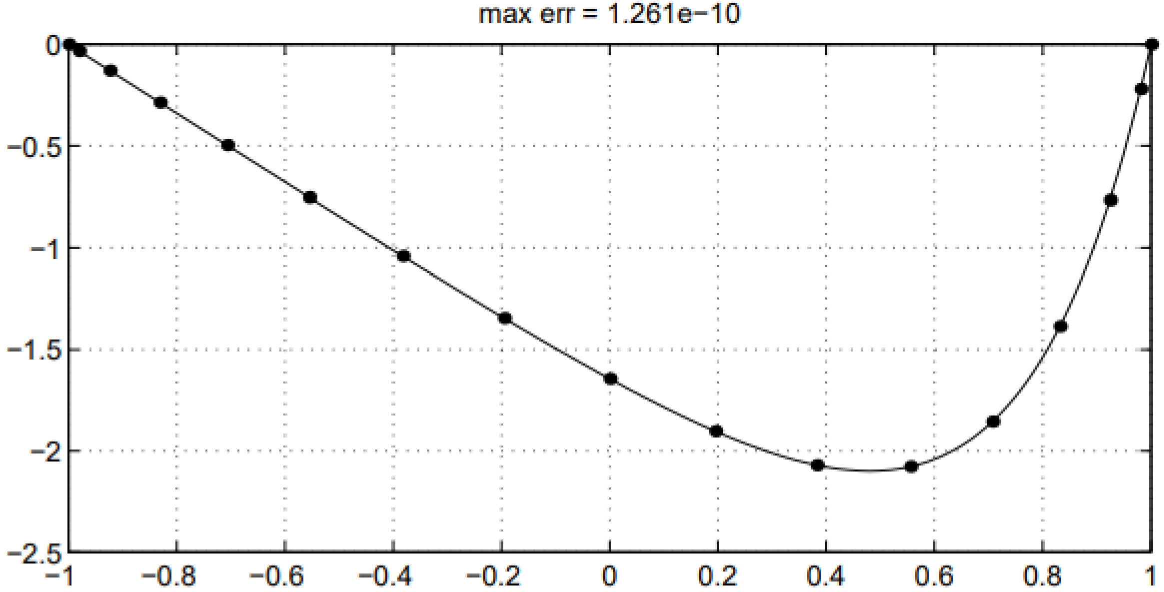
Program1. Matlab Code

1. Implement Program 13 and produce a plot similar to Output 13. Modify the program to explicitly specify the Dirichlet boundary conditions, $u(\pm 1) = 0$. Confirm that both results are identical (within machine precision).

Program 13

```
% p13.m - solve linear BVP u_xx = exp(4x), u(-1)=u(1)=0

  N = 16;
  [D,x] = cheb(N);
  D2 = D^2;
  D2 = D2(2:N,2:N);                    % boundary conditio
  f = exp(4*x(2:N));
  u = D2\f;                            % Poisson eq. solve
  u = [0;u;0];
  clf, subplot('position',[.1 .4 .8 .5])
  plot(x,u,'.','markersize',16)
  xx = -1:.01:1;
  uu = polyval(polyfit(x,u,N),xx);     % interpolate grid
  line(xx,uu,'linewidth',.8)
  grid on
  exact = ( exp(4*xx) - sinh(4)*xx - cosh(4) )/16;
  title(['max err = ' num2str(norm(uu-exact,inf))],'fontsize',12)
```

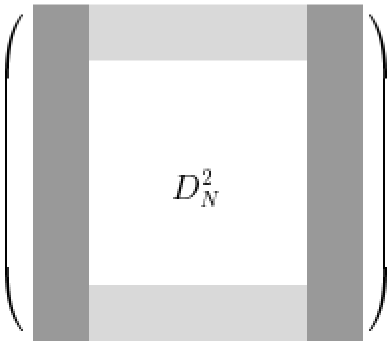
Output 13

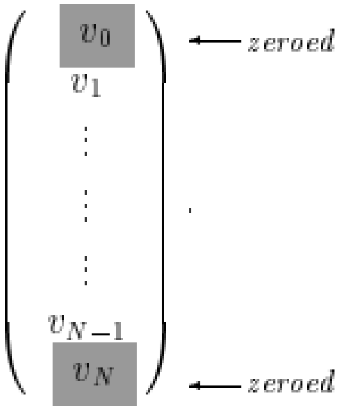


Program1. Python Code

```python
N = 16
D ,x = cheb(N)
D2 = np.delete(np.delete((D **2),[0,N],0),[0,N],1)
f = np.e **(4 *x[1:N])
u = np.linalg.solve(D2,f)
u = np.append(np.append([0],u),0)

xx = np.arange(-1,1 +0.01,0.01)
uu = polyval(polyfit(x,u,N),xx)

exact = (np.power(np.e,(4*xx)) - np.sinh(4)*xx - np.cosh(4))/16

plt.figure(figsize=(16,8))
plt.plot(xx,uu,linewidth=3,color='black',zorder=0)
plt.scatter(x,u,marker='o',s=100,color='red')
plt.title(r'max error = '+str(round(max(uu-exact),15)),fontsize=30)
plt.grid(linestyle=':')
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
```

$$\widetilde{D}_N^2 v = f.$$

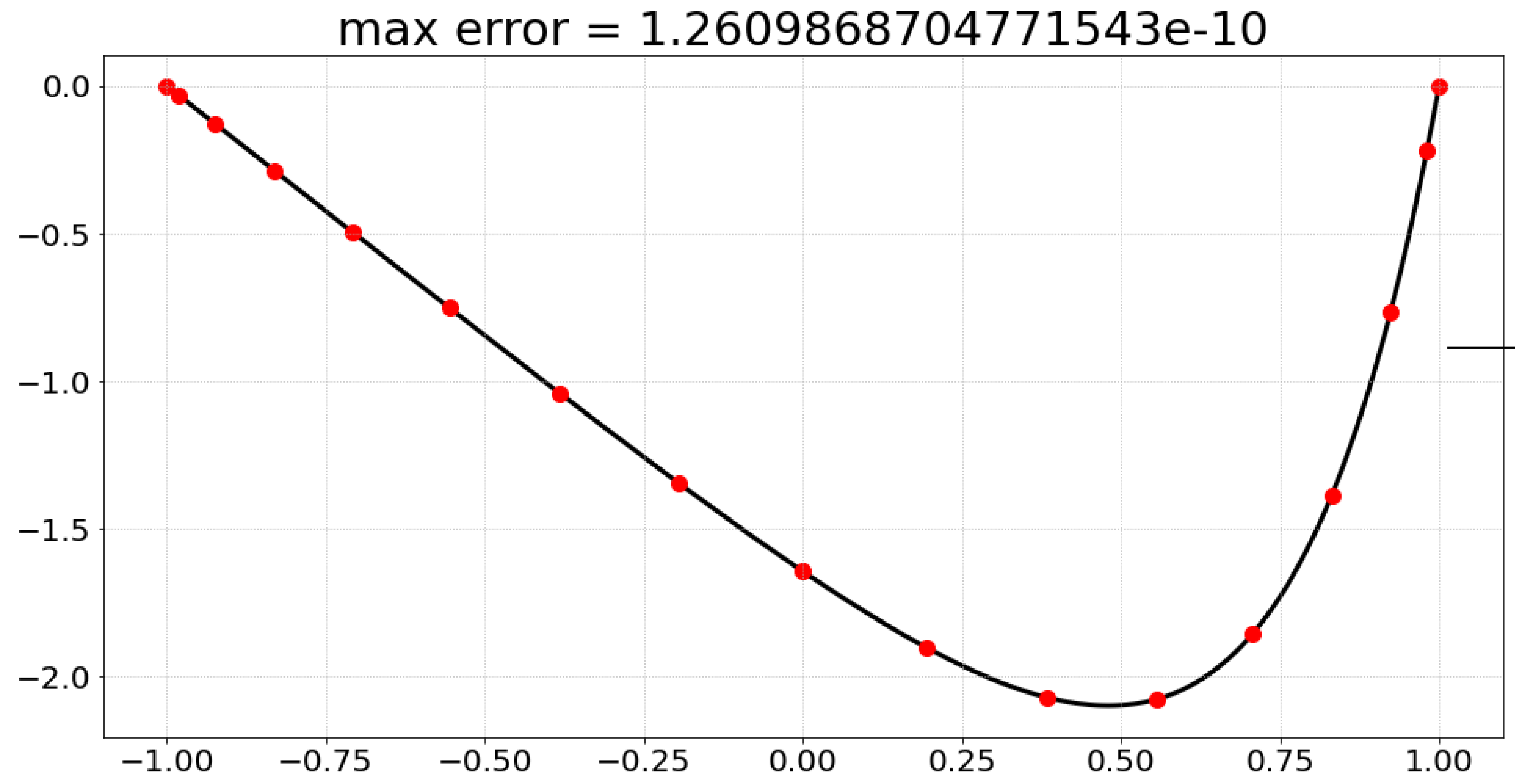




$$u(x) = [e^{4x} - x\sinh(4) - \cosh(4)]/16.$$

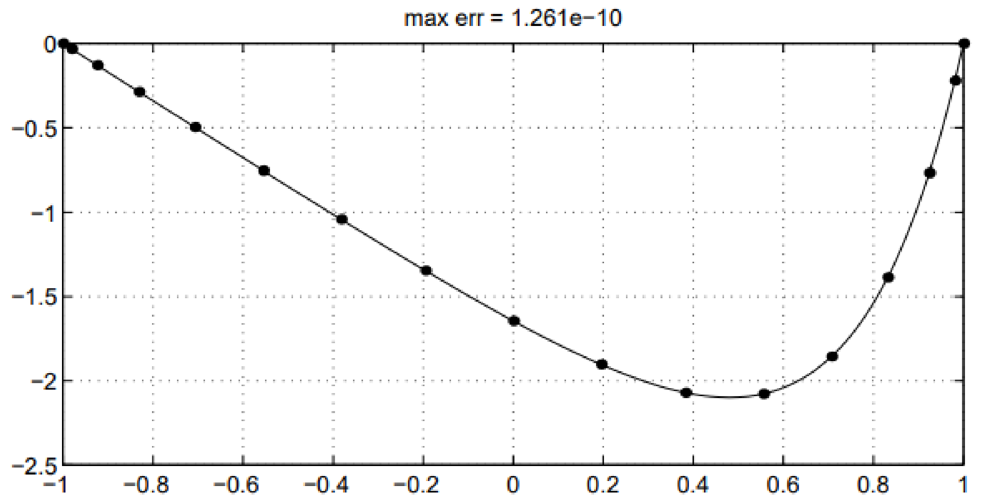
$$u_x = \frac{[4e^{4x} - \sinh(4)]}{16}$$

$$u_{xx} = \frac{[16e^{4x}]}{16} = e^{4x} = f$$

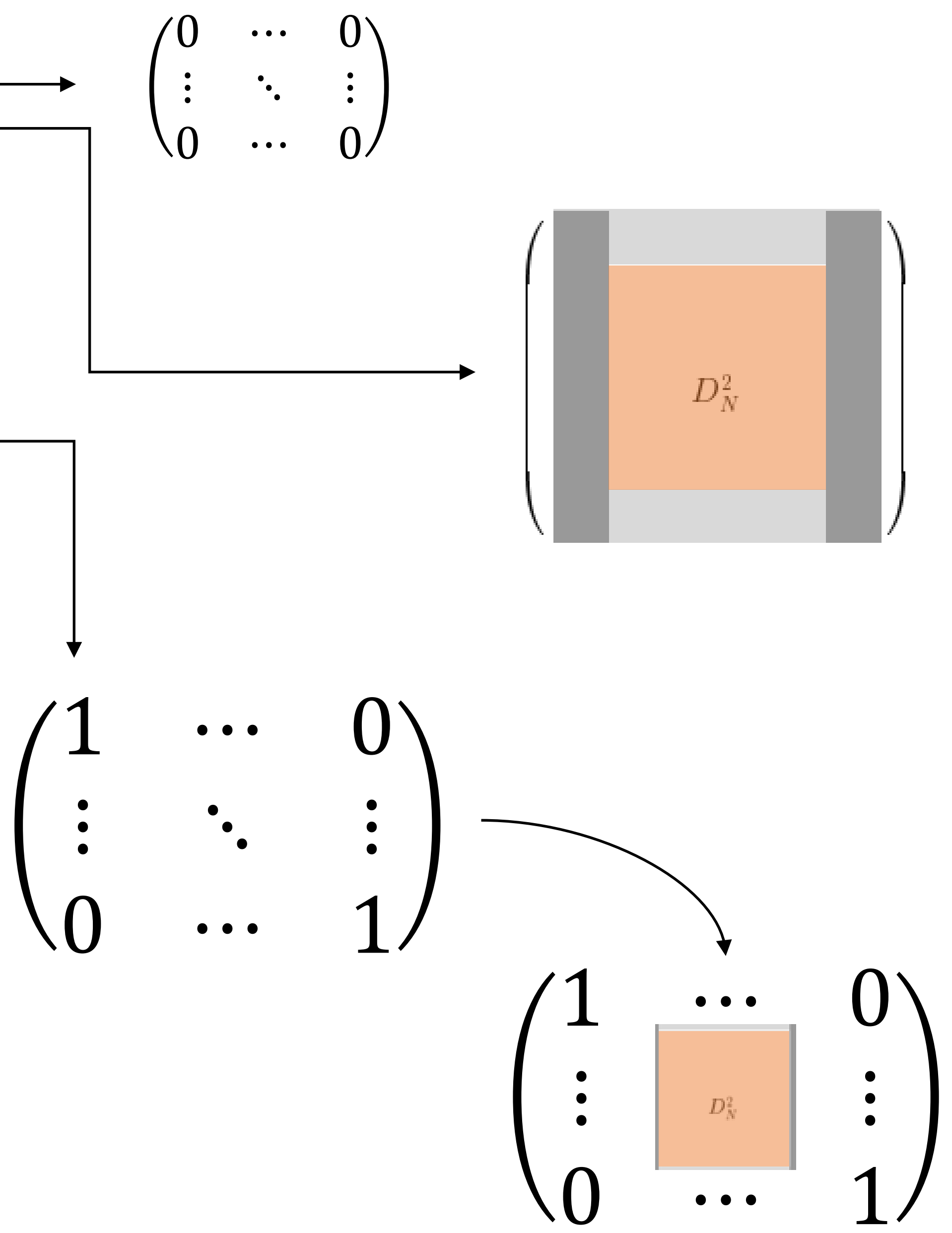


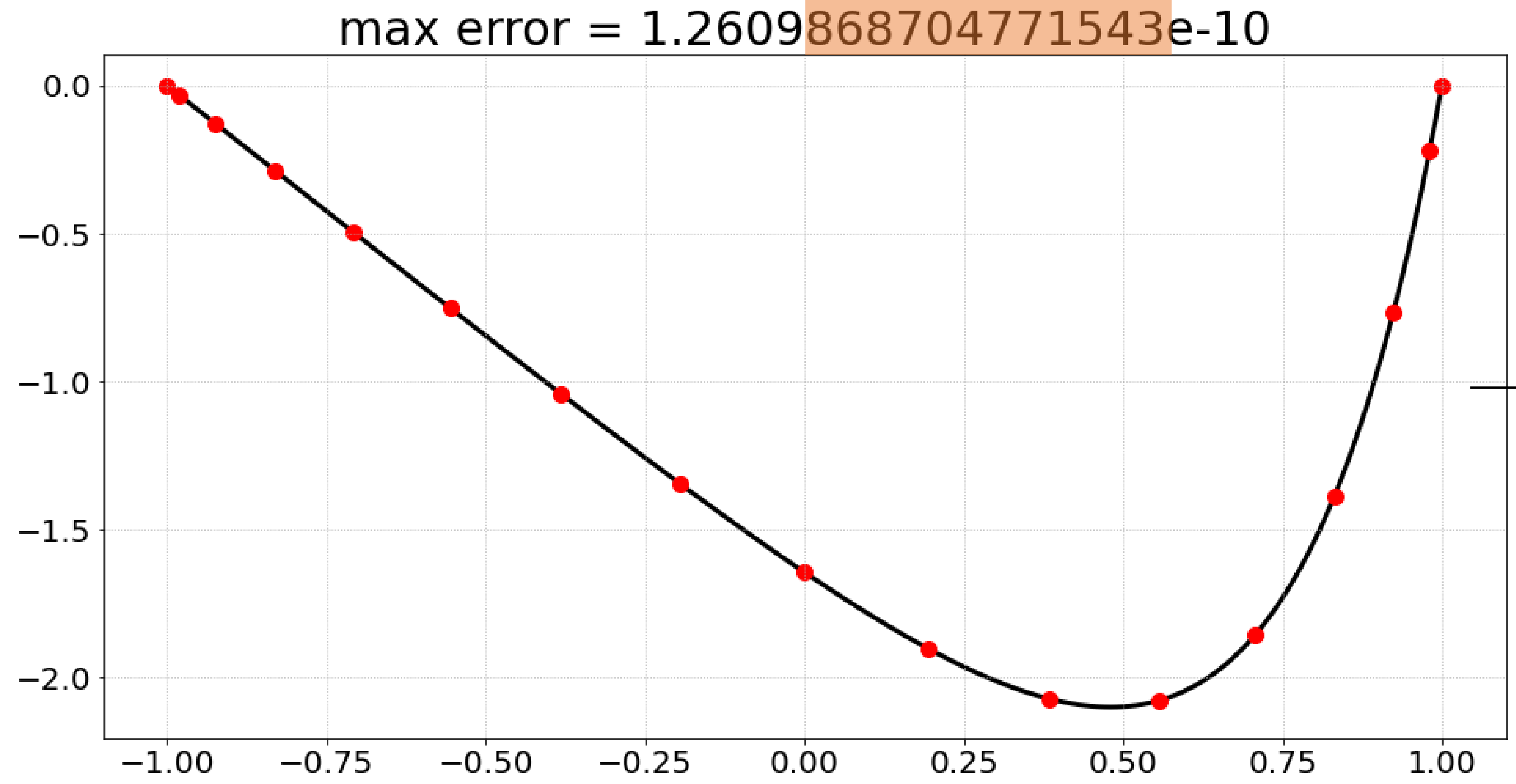
In Python

In Matlab

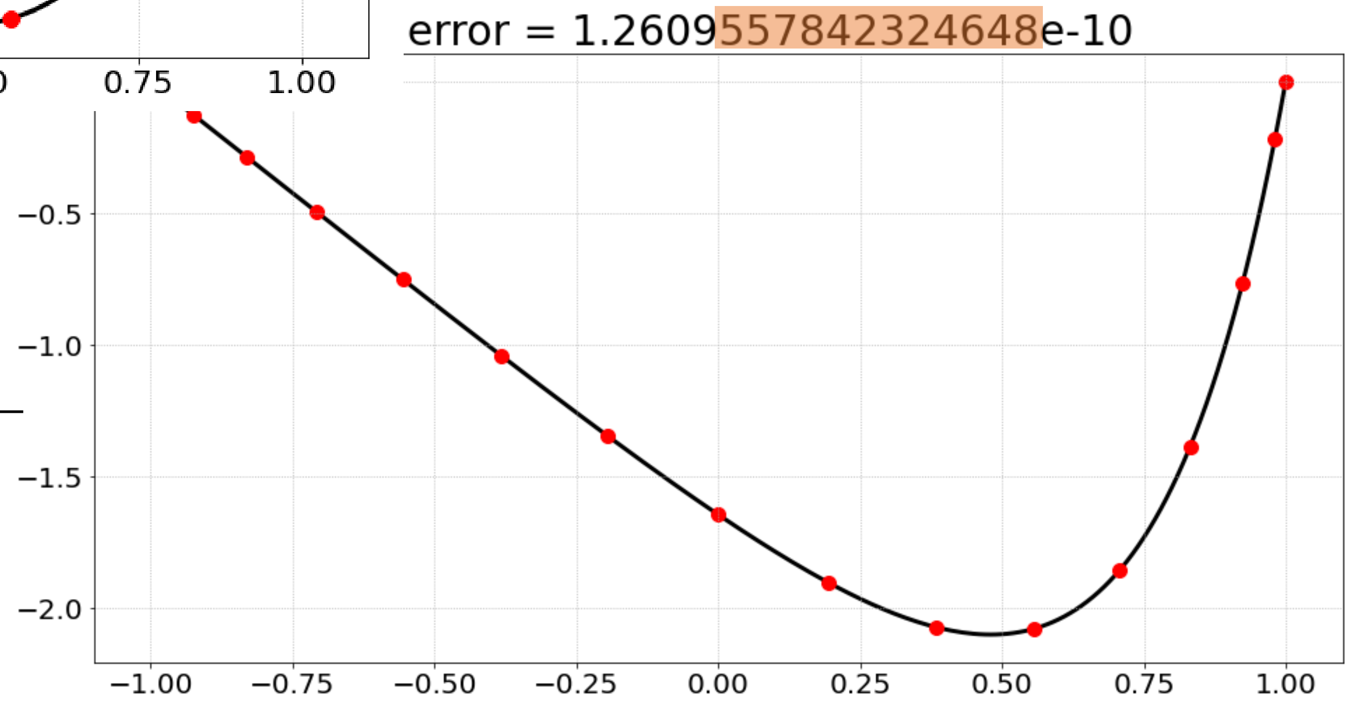


```
N = 16
D ,x = cheb(N)
matrix = np.zeros_like(D**2)
input = np.delete(np.delete((D **2),[0,N],0),[0,N],1)
matrix[1:N,1:N] = np.reshape(input,(N-1,-1)) ; matrix[0,0] = 1 ; matrix[N,N] = 1
f = np.e **(4 *x)
f = np.append(np.append([0],f[1:N]),0)
u = np.linalg.inv(matrix) * np.matrix(f).T

xx = np.arange(-1,1 +0.01,0.01)
uu = polyval(polyfit(x,u,N),xx)

exact = (np.power(np.e,(4*xx)) - np.sinh(4)*xx - np.cosh(4))/16

plt.figure(figsize=(16,8))
plt.plot(xx,uu,linewidth=3,color='black',zorder=0)
plt.scatter(x,np.array(u.flatten()),marker='o',s=100,color='red')
plt.title(r'max error = '+str(max(abs(uu-exact))),fontsize=30)
plt.grid(linestyle=':')
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
```





Original Method in Book



Full Matrix Method

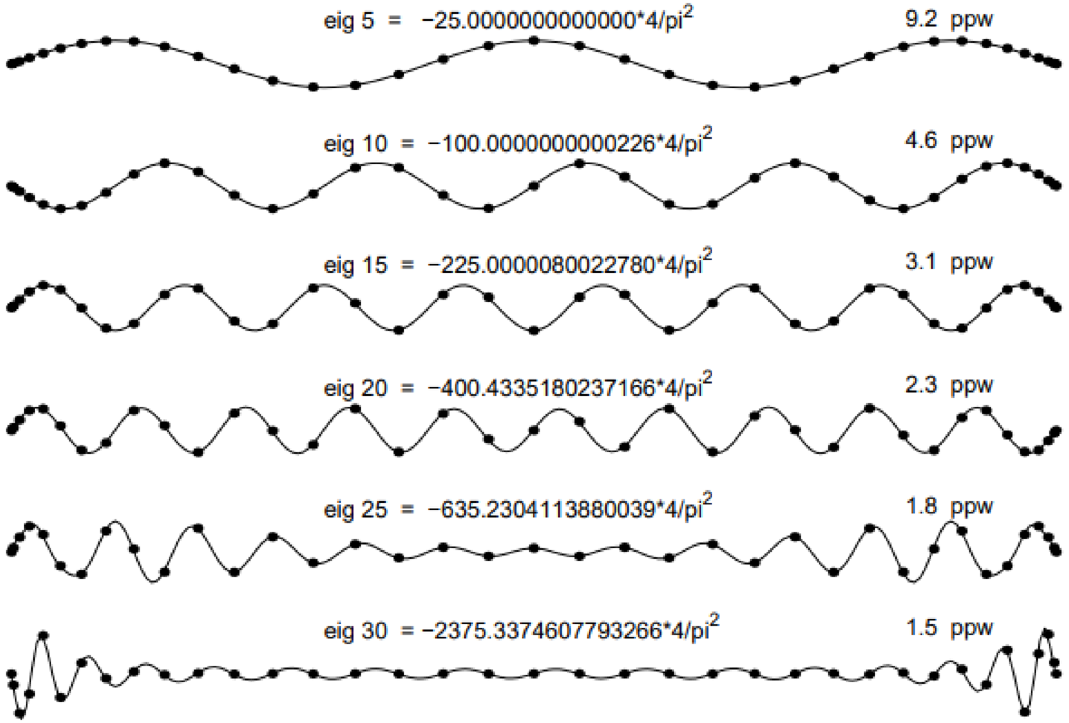
Program3. Matlab Code

3. Implement Program 15 and produce a plot similar to Output 15. Modify the program to use the Dirichlet boundary conditions, $u(\pm 1) = 0$, explicitly. (See the modified Matlab code below for reference.)

Program 15

```
% p15.m - solve eigenvalue BVP u_xx = lambda*u, u(-1)=u(1)=0

  N = 36; [D,x] = cheb(N); D2 = D^2; D2 = D2(2:N,2:N);
  [V,Lam] = eig(D2); lam = diag(Lam);
  [foo,ii] = sort(-lam);          % sort eigenvalues and -vectors
  lam = lam(ii); V = V(:,ii); clf
  for j = 5:5:30                  % plot 6 eigenvectors
    u = [0;V(:,j);0]; subplot(7,1,j/5)
    plot(x,u,'.','markersize',12), grid on
    xx = -1:.01:1; uu = polyval(polyfit(x,u,N),xx);
    line(xx,uu,'linewidth',.7), axis off
    text(-.4,.5,sprintf('eig %d =%20.13f*4/pi^2',j,lam(j)*4/pi^2))
    text(.7,.5,sprintf('%4.1f  ppw', 4*N/(pi*j)))
  end
```

Output 15



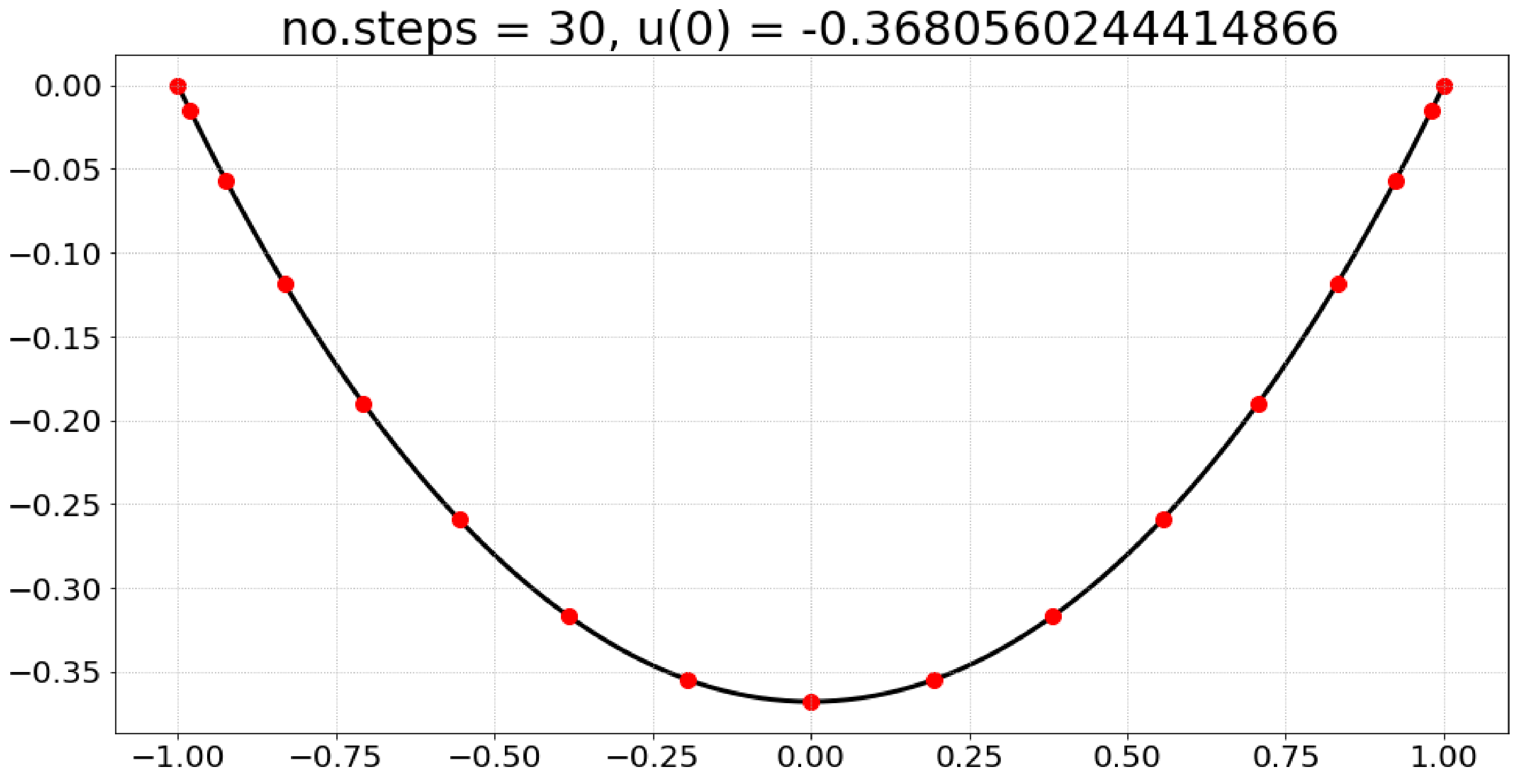
Program2. Python Code

```
N = 16
D, x = cheb(N)
D2 = np.delete(np.delete((D **2),[0,N],0),[0,N],1)
u = np.zeros(N-1)
change = 1 ; it = 0

while change > 1 * 10**(-15):
  unew = np.linalg.solve(D2,np.e **u)
  change = max(abs(unew-u))
  it += 1
  u = unew

u = np.append(np.append([0],u),0)

xx = np.arange(-1,1 +0.01,0.01)
uu = polyval(polyfit(x,u,N),xx)

plt.figure(figsize=(16,8))
plt.plot(xx,uu,linewidth=3,color='black',zorder=0)
plt.scatter(x,u,marker='o',s=100,color='red')
plt.title("no.steps = "+ str(it) + ", u(0) = "+ str(min(u)),fontsize=30)
plt.grid(linestyle=':')
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
```

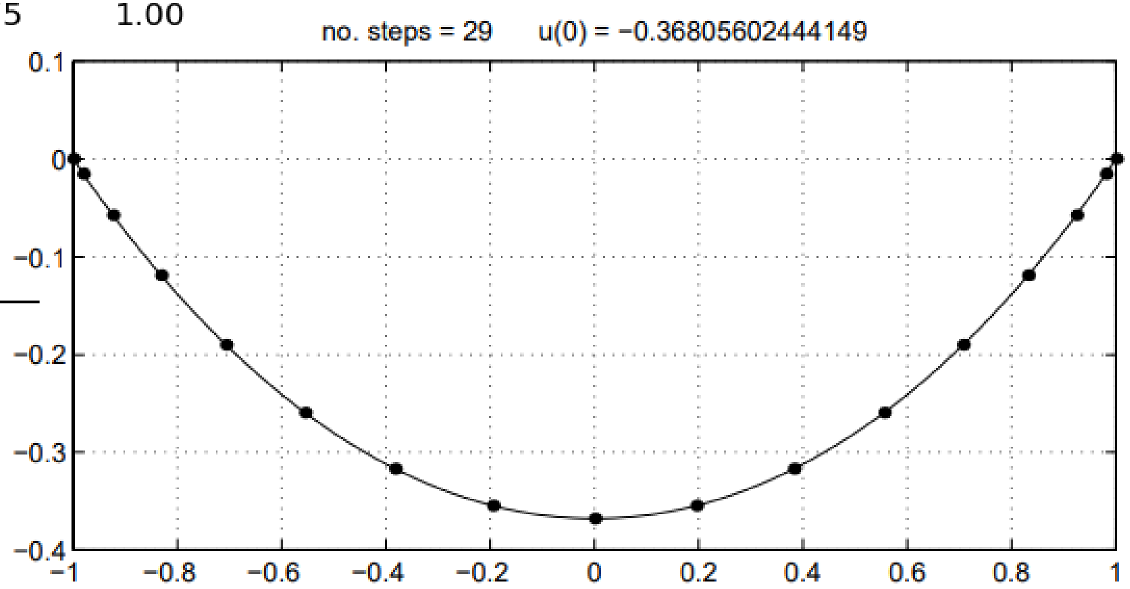
$$\widetilde{D}_N^2 v_{\text{new}} = \exp(v_{\text{old}}),$$

$$u_{xx} = e^u, \qquad -1 < x < 1, \quad u(\pm 1) = 0.$$

$$\begin{array}{cl} \begin{pmatrix} \boxed{v_0} \\ v_1 \\ \vdots \\ \vdots \\ \vdots \\ v_{N-1} \\ \boxed{v_N} \end{pmatrix} & \begin{array}{l} \longleftarrow zeroed \\ \\ \\ . \\ \\ \\ \longleftarrow zeroed \end{array} \end{array}$$



In Python



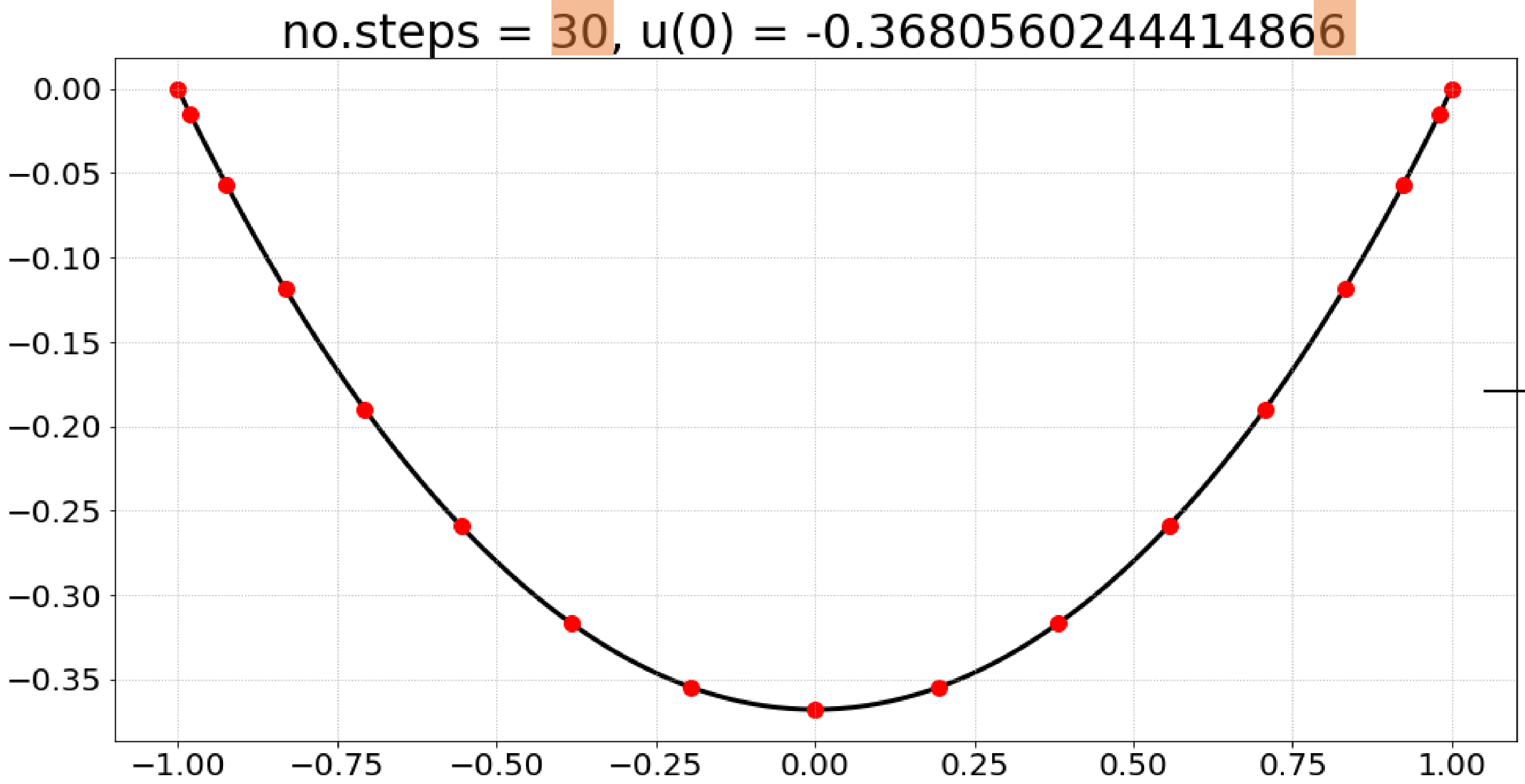
In Matlab

```
N = 16
D ,x = cheb(N)
matrix = np.zeros_like(D**2)
input = np.delete(np.delete((D **2),[0,N],0),[0,N],1)
matrix[1:N,1:N] = np.reshape(input,(N-1,-1)) ; matrix[0,0] = 1 ; matrix[N,N] = 1
#f = np.e **(4 *x)
#f = np.append(np.append([0],f[1:N]),0)
#u = np.linalg.inv(matrix) * np.matrix(f).T

u = np.matrix(np.zeros(N+1)).T
change = 1 ; it = 0


while change > 1 * 10**(-15):
  unew = np.linalg.inv(matrix) * np.power(np.e,u)
  unew[0,0] = 0 ; unew[N,0] = 0
  change = abs(unew.flatten() - u.flatten()).max()
  it += 1
  u = unew

xx = np.arange(-1,1 +0.01,0.01)
uu = polyval(polyfit(x,u,N),xx)

plt.figure(figsize=(16,8))
plt.plot(xx,uu,linewidth=3,color='black',zorder=0)
plt.scatter(x,np.array(u.flatten()),marker='o',s=100,color='red')
plt.title("no.steps = "+ str(it) + ", u(0) = "+ str((u.flatten()).min()),fontsize=30)
plt.grid(linestyle=':')
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
```

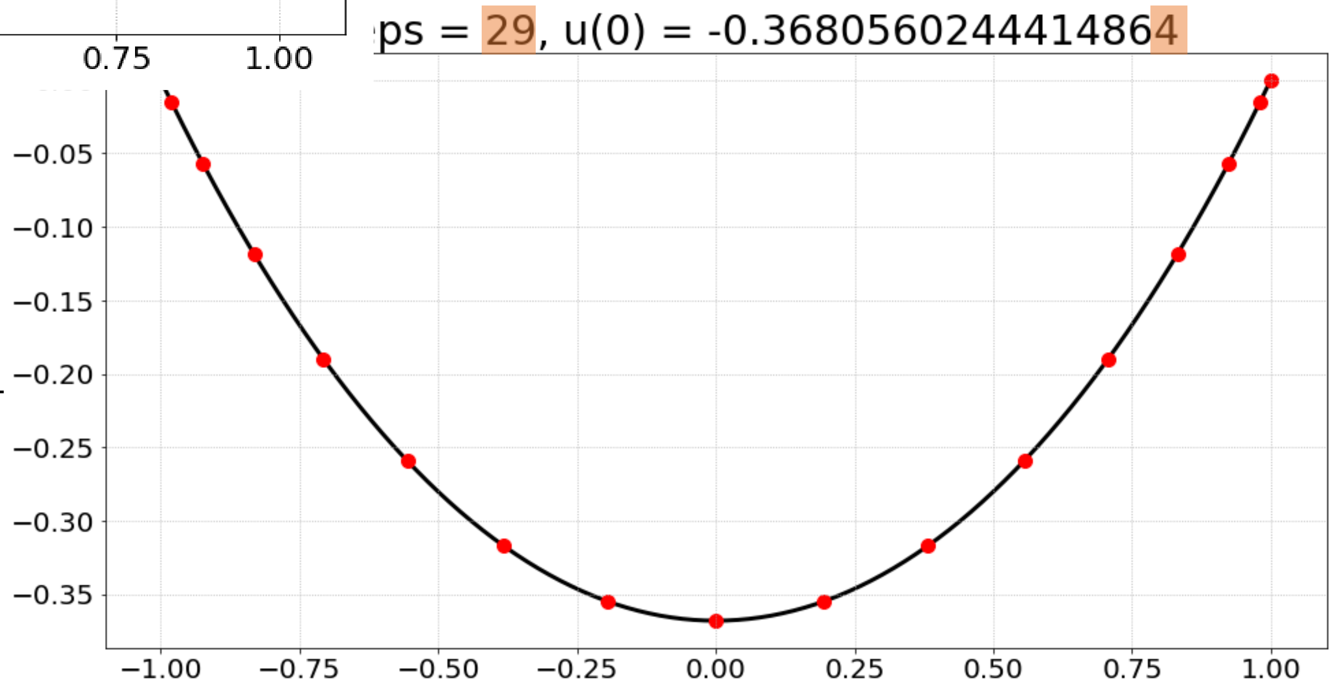
$$\begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \boxed{D_N^2} & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$$

$$\begin{array}{cl} \begin{pmatrix} \boxed{v_0} \\ v_1 \\ \vdots \\ \vdots \\ \vdots \\ v_{N-1} \\ \boxed{v_N} \end{pmatrix} & \begin{array}{l} \leftarrow zeroed \\ \\ \\ \\ \\ \\ \leftarrow zeroed \end{array} \end{array}$$

$$\widetilde{D}_N^2 v_{\text{new}} = \exp(v_{\text{old}}),$$



Original Method in Book



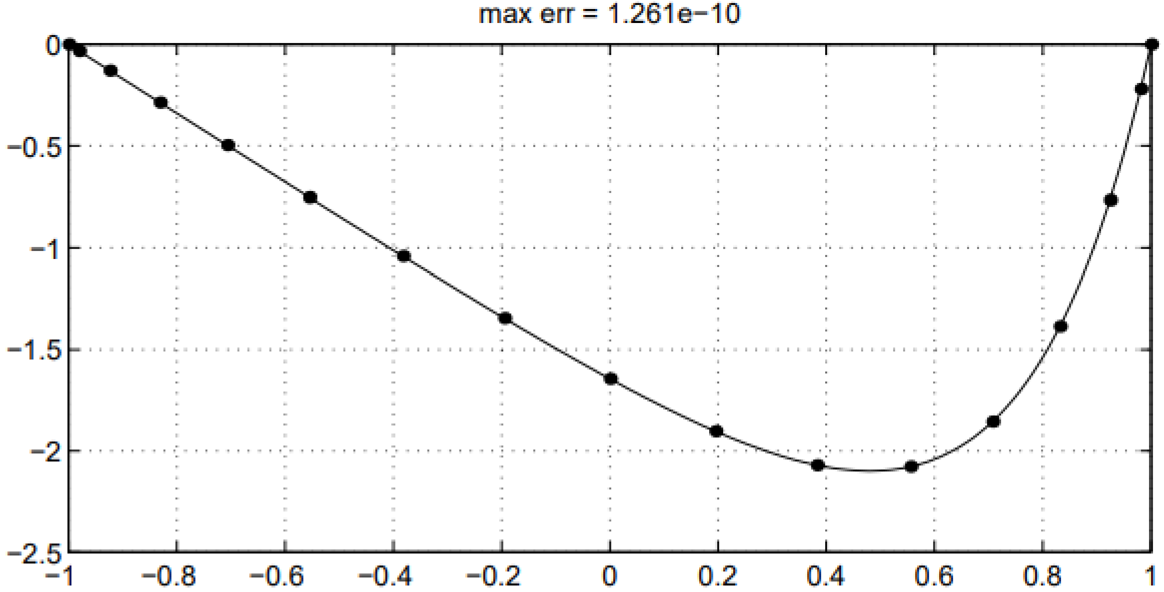
Full Matrix Method

Program3. Matlab Code

Program 13

```
% p13.m - solve linear BVP u_xx = exp(4x), u(-1)=u(1)=0

  N = 16;
  [D,x] = cheb(N);
  D2 = D^2;
  D2 = D2(2:N,2:N);                    % boundary conditio
  f = exp(4*x(2:N));
  u = D2\f;                            % Poisson eq. solve
  u = [0;u;0];
  clf, subplot('position',[.1 .4 .8 .5])
  plot(x,u,'.','markersize',16)
  xx = -1:.01:1;
  uu = polyval(polyfit(x,u,N),xx);     % interpolate grid
  line(xx,uu,'linewidth',.8)
  grid on
  exact = ( exp(4*xx) - sinh(4)*xx - cosh(4) )/16;
  title(['max err = ' num2str(norm(uu-exact,inf))],'fontsize',12)
```

Output 13



Program3. Python Code

```
from numpy.linalg import eig

N = 36
D,x = cheb(N)
D2 = np.delete(np.delete((D **2),[0,N],0),[0,N],1)

w,v = eig(D2)
lam = -(np.sort(abs(w)))
index = np.argsort(abs(w))
v = v[:,index]

xx = np.arange(-1,1 +0.01,0.01)

fig, ax = plt.subplots(6,1,sharex=True,sharey=True,figsize=(16,16))
for j in np.arange(5,30 +5,5):
  ax_index = int((j/5)-1)
  new_v = v[:,j-1]
  u = np.append(np.append([0],new_v),0)
  uu = polyval(polyfit(x,u,N),xx)
  ax[ax_index].scatter(x,u,marker='o',s=50,color='red')
  ax[ax_index].plot(xx,uu,linewidth=3,color='black',zorder=0)
  ax[ax_index].set_title("eig " + str(j) + " = " + str(round((lam[j-1]*4/np.pi**2),4)) +",
```

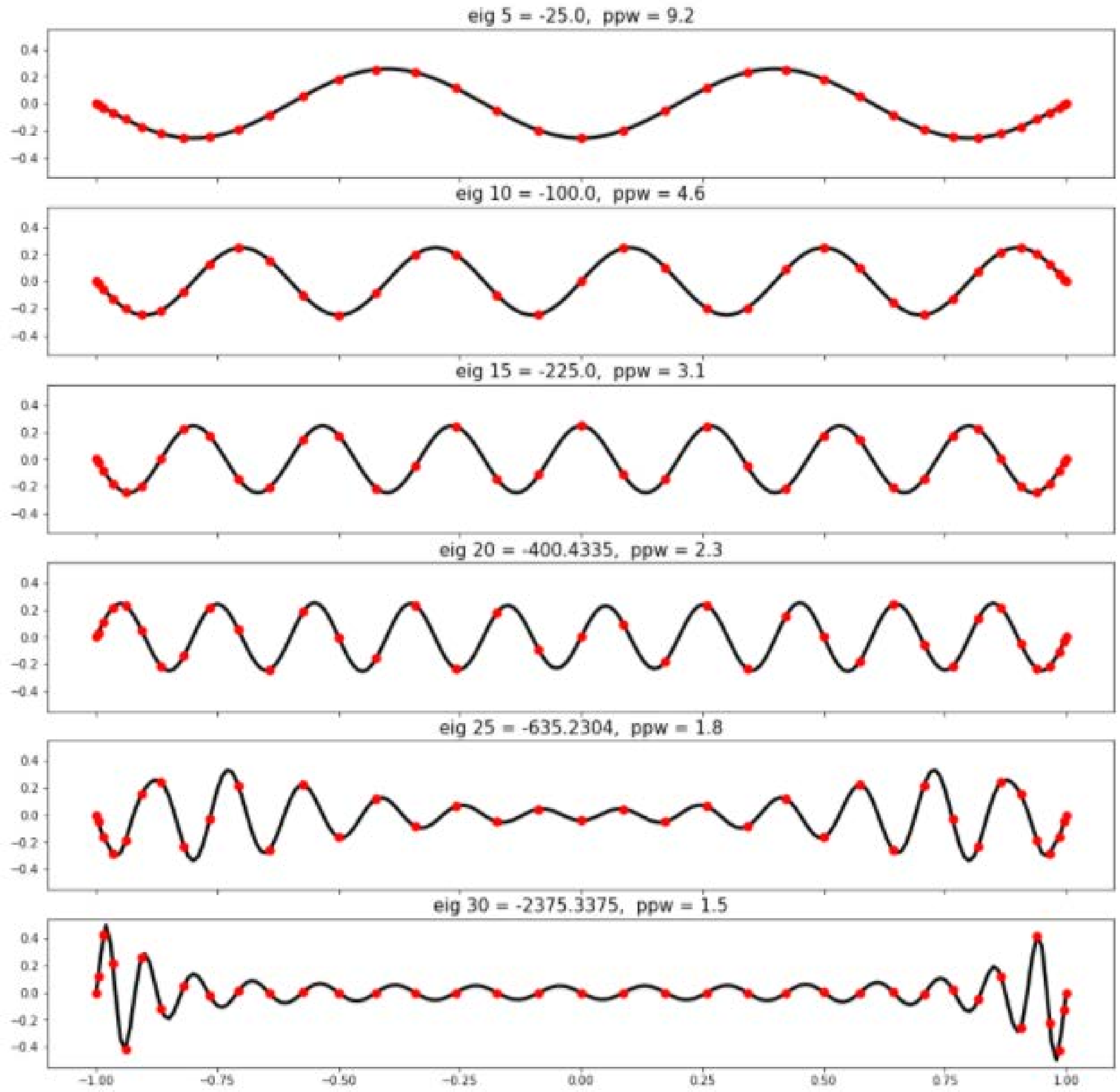
Calculate Eigen Value(w), Eigen Vectors(v)

Set -λ and Set index

Select Eigen Vector on corresponding index

Select Eigen Value and calculate

The eigenvalues of this problem are $\lambda = -\pi^2 n^2/4$, $n = 1, 2, \ldots$, with corresponding eigenfunctions $\sin(n\pi(x+1)/2)$. Program 15 calculates the eigen-



What is ppw??

factor of 3. The crucial quantity that explains this behavior is the number of points per wavelength ("ppw") in the central, coarsest part of the grid near $x = 0$. With at least 2 points per wavelength, the grid is fine enough everywhere to resolve the wave. With less than 2 points per wavelength, the

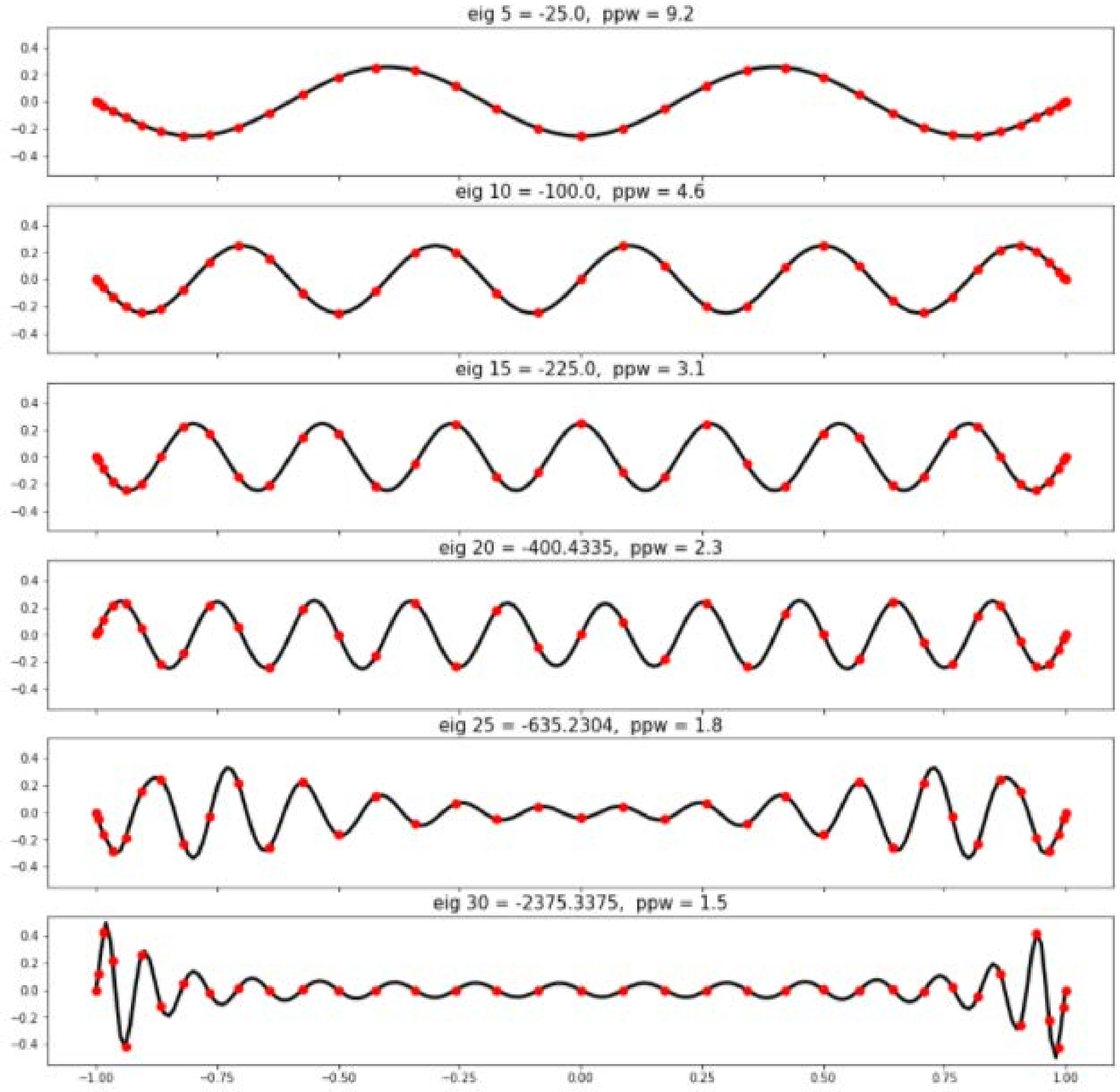
The shape is similar to that of the matlab.

However,
several graphs were drawn upside down.

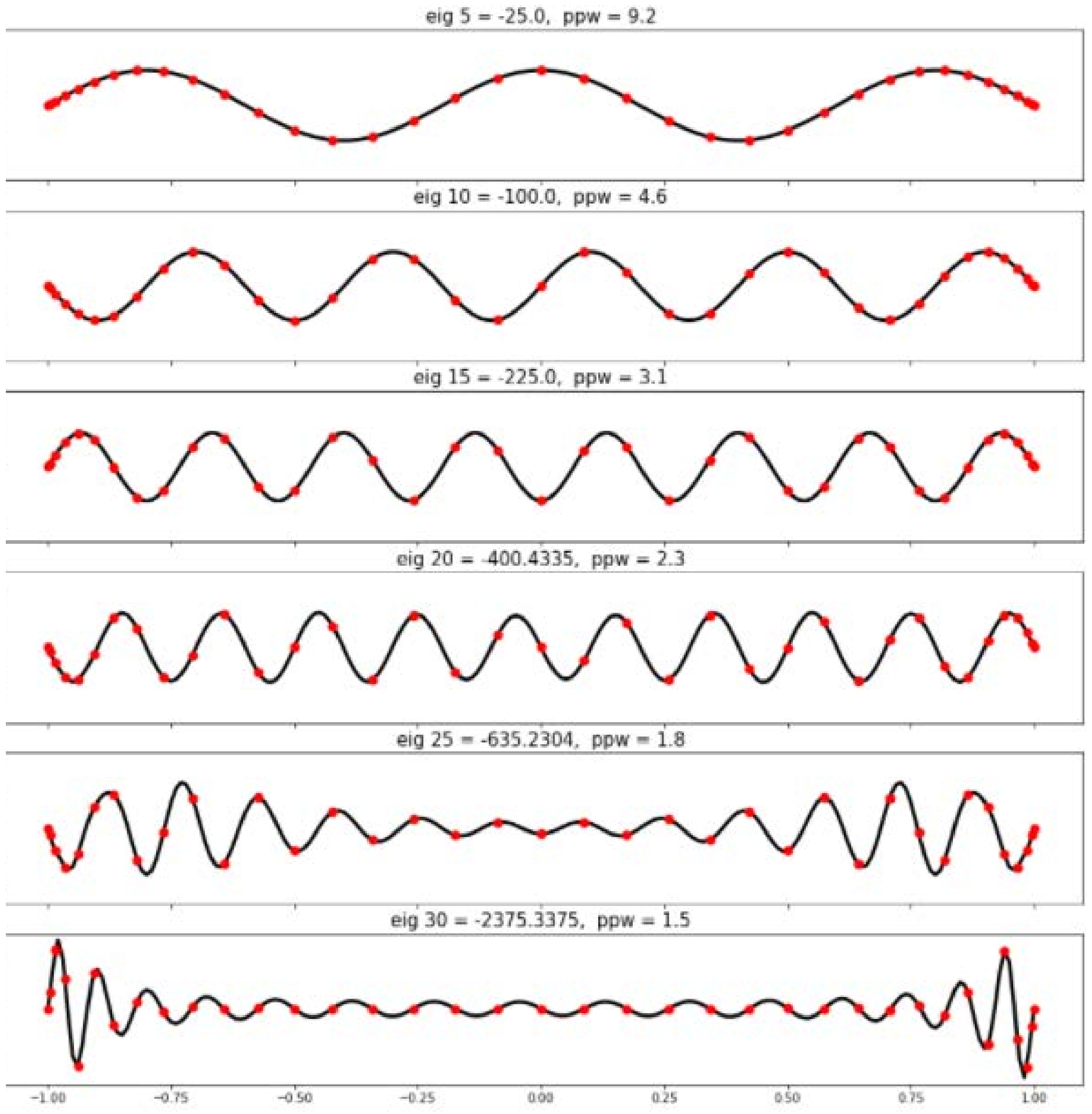
There have been a few revisions,
but they have not been resolved.

I executed the same code on the matlab (octave) and was able to get the upside-down graph.

Original Method in Book



Full Matrix Method

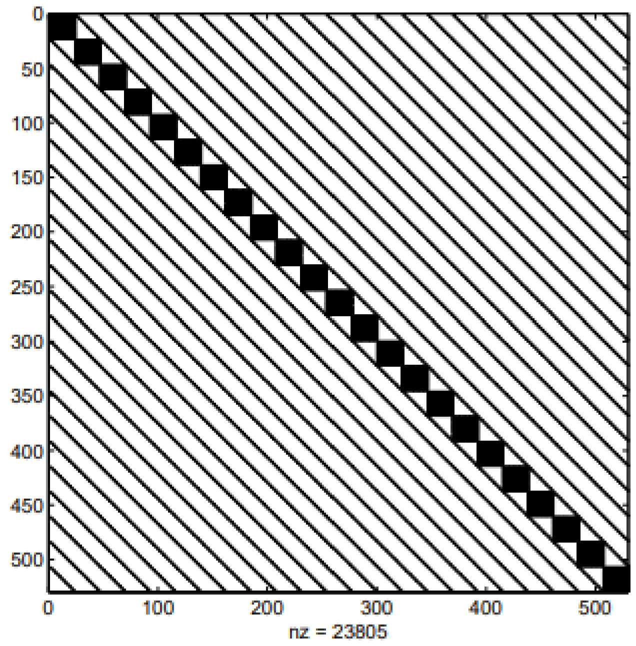


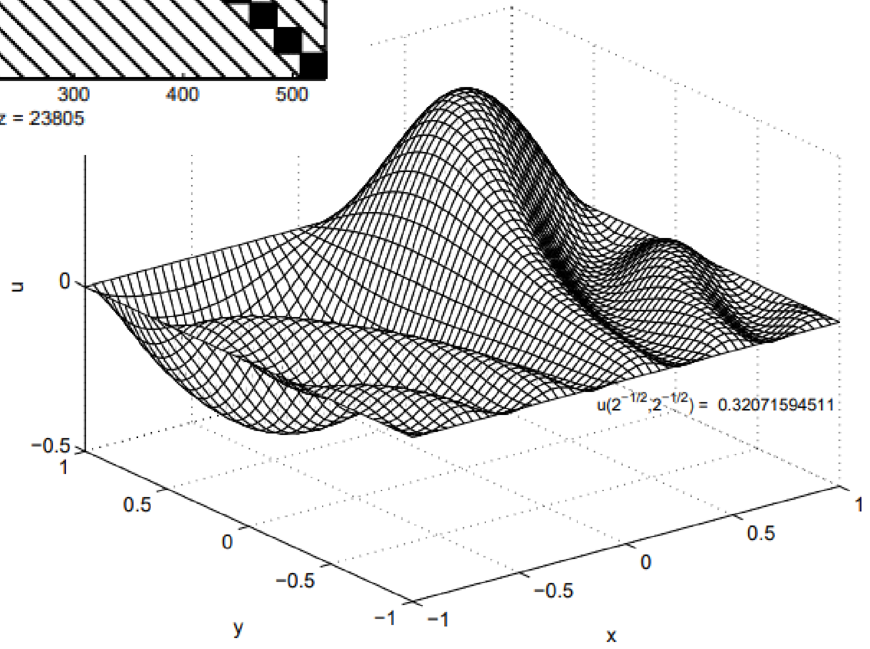
Program4. Matlab Code

4. Implement Program 16 and produce a plot similar to Output 16. Modify the program to explicitly specify the Dirichlet boundary conditions, $u(\pm 1,y) = u(x,\pm 1) = 0$. Confirm that both results are identical (within machine precision).

Program 16

```
% p16.m - Poisson eq. on [-1,1]x[-1,1] with u=0 on boundary

% Set up grids and tensor product Laplacian and solve for u:
  N = 24; [D,x] = cheb(N); y = x;
  [xx,yy] = meshgrid(x(2:N),y(2:N));
  xx = xx(:); yy = yy(:);         % stretch 2D grids to 1D vectors
  f = 10*sin(8*xx.*(yy-1));
  D2 = D^2; D2 = D2(2:N,2:N); I = eye(N-1);
  L = kron(I,D2) + kron(D2,I);                        % Laplacian
  figure(1), clf, spy(L), drawnow
  tic, u = L\f; toc          % solve problem and watch the clock

% Reshape long 1D results onto 2D grid:
  uu = zeros(N+1,N+1); uu(2:N,2:N) = reshape(u,N-1,N-1);
  [xx,yy] = meshgrid(x,y);
  value = uu(N/4+1,N/4+1);

% Interpolate to finer grid and plot:
  [xxx,yyy] = meshgrid(-1:.04:1,-1:.04:1);
  uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');
  figure(2), clf, mesh(xxx,yyy,uuu), colormap([0 0 0])
  xlabel x, ylabel y, zlabel u
  text(.4,-.3,-.3,sprintf('u(2^{-1/2},2^{-1/2}) = %14.11f',value))
```





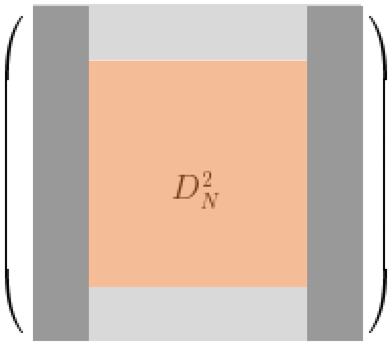
Program4. Python Code

```python
N = 24

D,x = cheb(N)
y = x
xx, yy = np.meshgrid(x[1:N],y[1:N])
xx = xx.flatten() ; yy = yy.flatten()
f = 10 * np.sin(8*xx*(yy-1))
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)

plt.figure(figsize=(8,8))

plt.spy(L)
plt.xticks(fontsize=15) ; plt.yticks(fontsize = 15)
```

$$u_{xx} + u_{yy} = 10\sin(8x(y-1)), \quad -1 < x, y < 1, \quad u = 0 \text{ on the boundary.}$$

Flatten Function

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix}$$

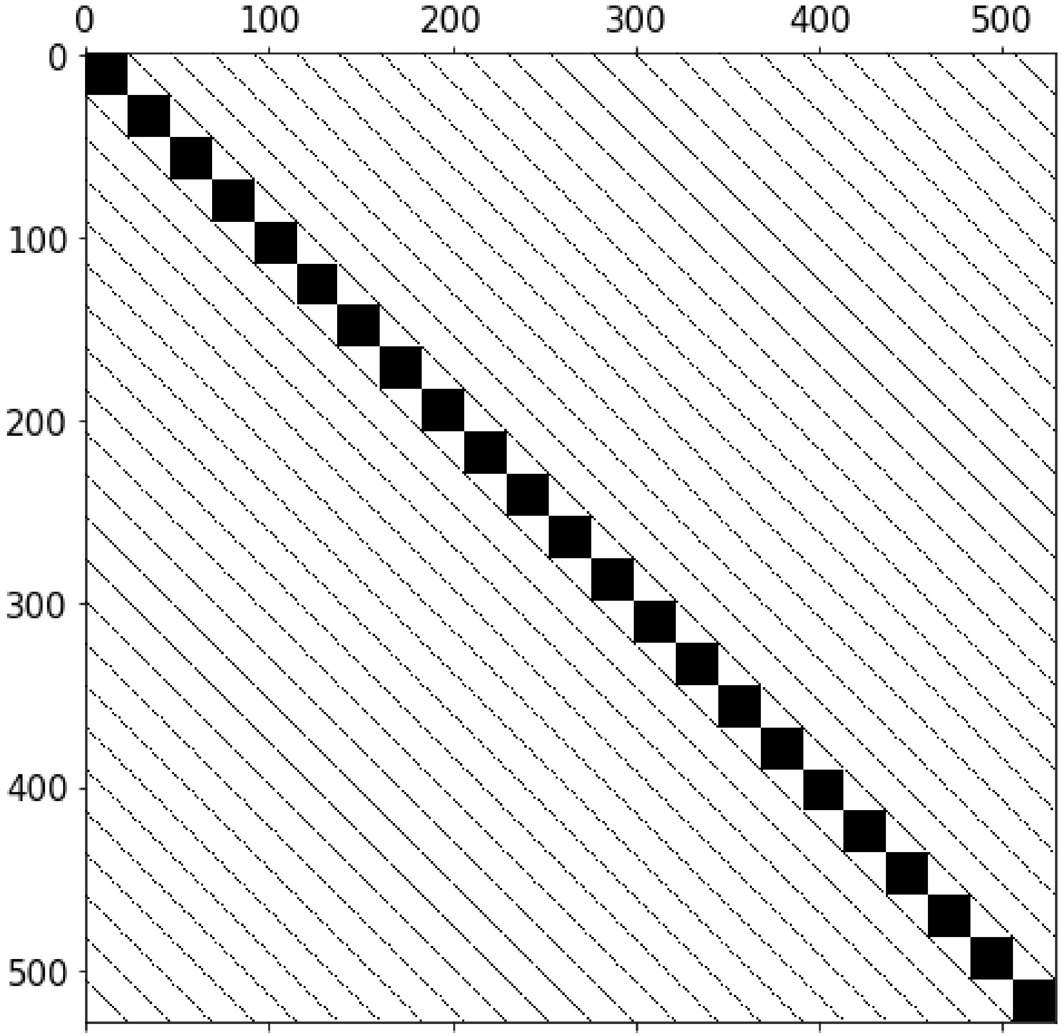


Examples

```python
>>> a = np.array([[1,2], [3,4]])
>>> a.flatten()
array([1, 2, 3, 4])
```

$$I \otimes \tilde{D}_N^2 = \left(\begin{array}{ccc|ccc|ccc} -14 & 6 & -2 & & & & & & \\ 4 & -6 & 4 & & & & & & \\ -2 & 6 & -14 & & & & & & \\ \hline & & & -14 & 6 & -2 & & & \\ & & & 4 & -6 & 4 & & & \\ & & & -2 & 6 & -14 & & & \\ \hline & & & & & & -14 & 6 & -2 \\ & & & & & & 4 & -6 & 4 \\ & & & & & & -2 & 6 & -14 \end{array}\right),$$

$$\tilde{D}_N^2 \otimes I = \left(\begin{array}{ccc|ccc|ccc} -14 & & & 6 & & & -2 & & \\ & -14 & & & 6 & & & -2 & \\ & & -14 & & & 6 & & & -2 \\ \hline 4 & & & -6 & & & 4 & & \\ & 4 & & & -6 & & & 4 & \\ & & 4 & & & -6 & & & 4 \\ \hline -2 & & & 6 & & & -14 & & \\ & -2 & & & 6 & & & -14 & \\ & & -2 & & & 6 & & & -14 \end{array}\right).$$



$$I \otimes \widetilde{D}_N^2 = \left(\begin{array}{ccc|ccc|ccc} -14 & 6 & -2 & & & & & & \\ 4 & -6 & 4 & & & & & & \\ -2 & 6 & -14 & & & & & & \\ \hline & & & -14 & 6 & -2 & & & \\ & & & 4 & -6 & 4 & & & \\ & & & -2 & 6 & -14 & & & \\ \hline & & & & & & -14 & 6 & -2 \\ & & & & & & 4 & -6 & 4 \\ & & & & & & -2 & 6 & -14 \end{array}\right).$$

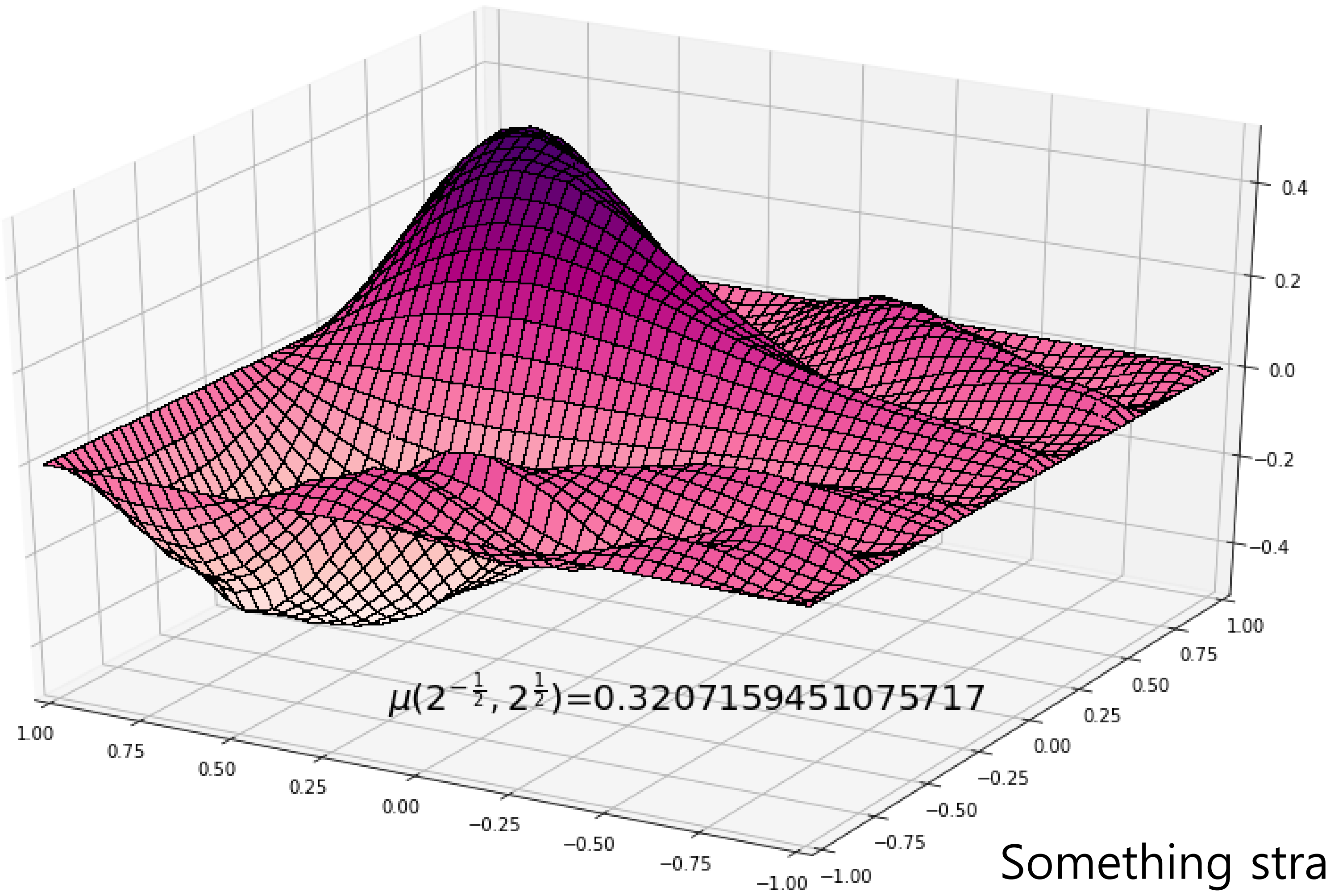
$$\widetilde{D}_N^2 \otimes I = \left(\begin{array}{ccc|ccc|ccc} -14 & & & 6 & & & -2 & & \\ & -14 & & & 6 & & & -2 & \\ & & -14 & & & 6 & & & -2 \\ \hline 4 & & & -6 & & & 4 & & \\ & 4 & & & -6 & & & 4 & \\ & & 4 & & & -6 & & & 4 \\ \hline -2 & & & 6 & & & -14 & & \\ & -2 & & & 6 & & & -14 & \\ & & -2 & & & 6 & & & -14 \end{array}\right).$$

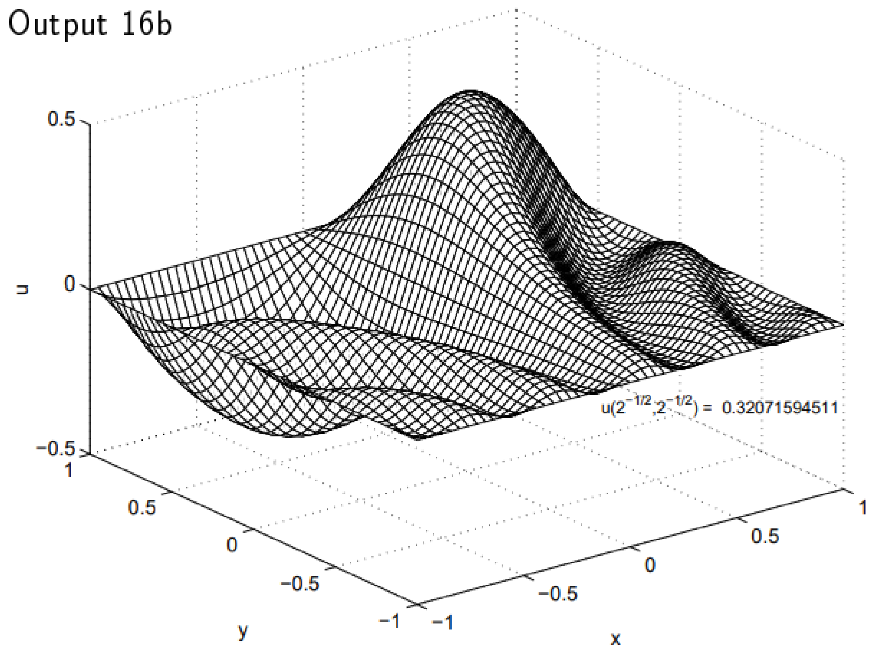
The diagonal components are all filled with non-zero components when two expressions are plus.

```python
u = np.linalg.solve(L,f)

uu = np.zeros((N+1,N+1)) ; uu[1:N,1:N] = np.reshape(u,(N-1,-1))
xx, yy = np.meshgrid(x,y)
value = uu[int(N/4)][int(N/4)]

from scipy.interpolate import interp2d
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

xxx = np.arange(-1,1 +0.04,0.04)
yyy = np.arange(-1,1 +0.04,0.04)

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

interp_spline = interp2d(xx,yy,uu)
uuu = interp_spline(xxx,yyy)
fig = plt.figure(figsize=(16,10))
ax = plt.axes(projection='3d')
ax.plot_surface(yi,xi,uuu,rstride=1,cstride=1,cmap=cm.RdPu,linewidth=0.005,edgecolors='black',antialiased=False)
ax.set_xlim(1,-1) ; ax.set_ylim(-1,1) ; ax.set_zlim(-0.5,0.5)
ax.text(0.25,-0.75,-0.5,r"$₩mu(2^{-₩,₩frac{1}{2}},2^{₩,₩frac{1}{2}}$)="+str(value),fontsize=20)
```

Set $\mu(\pm 1,(x,y)) = 0$

Interpolate 2D



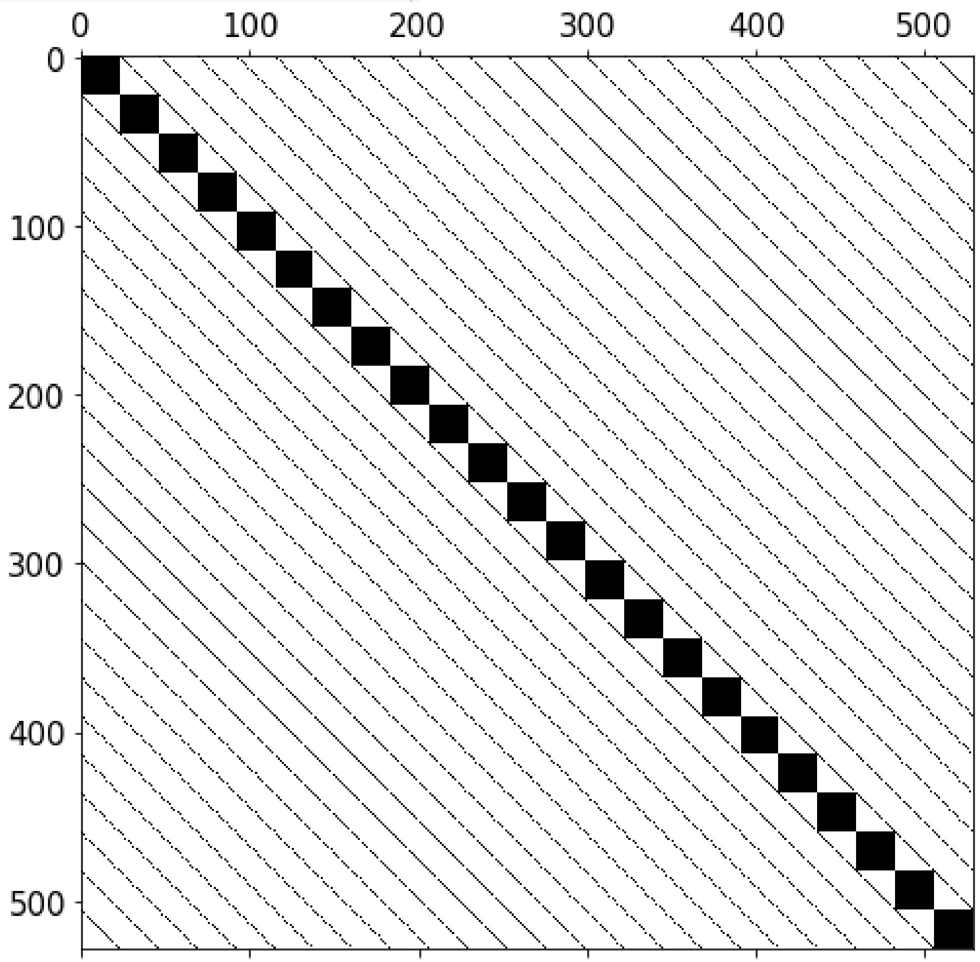


Something strange fluctuation was drawn,
but I couldn't figure out the cause.

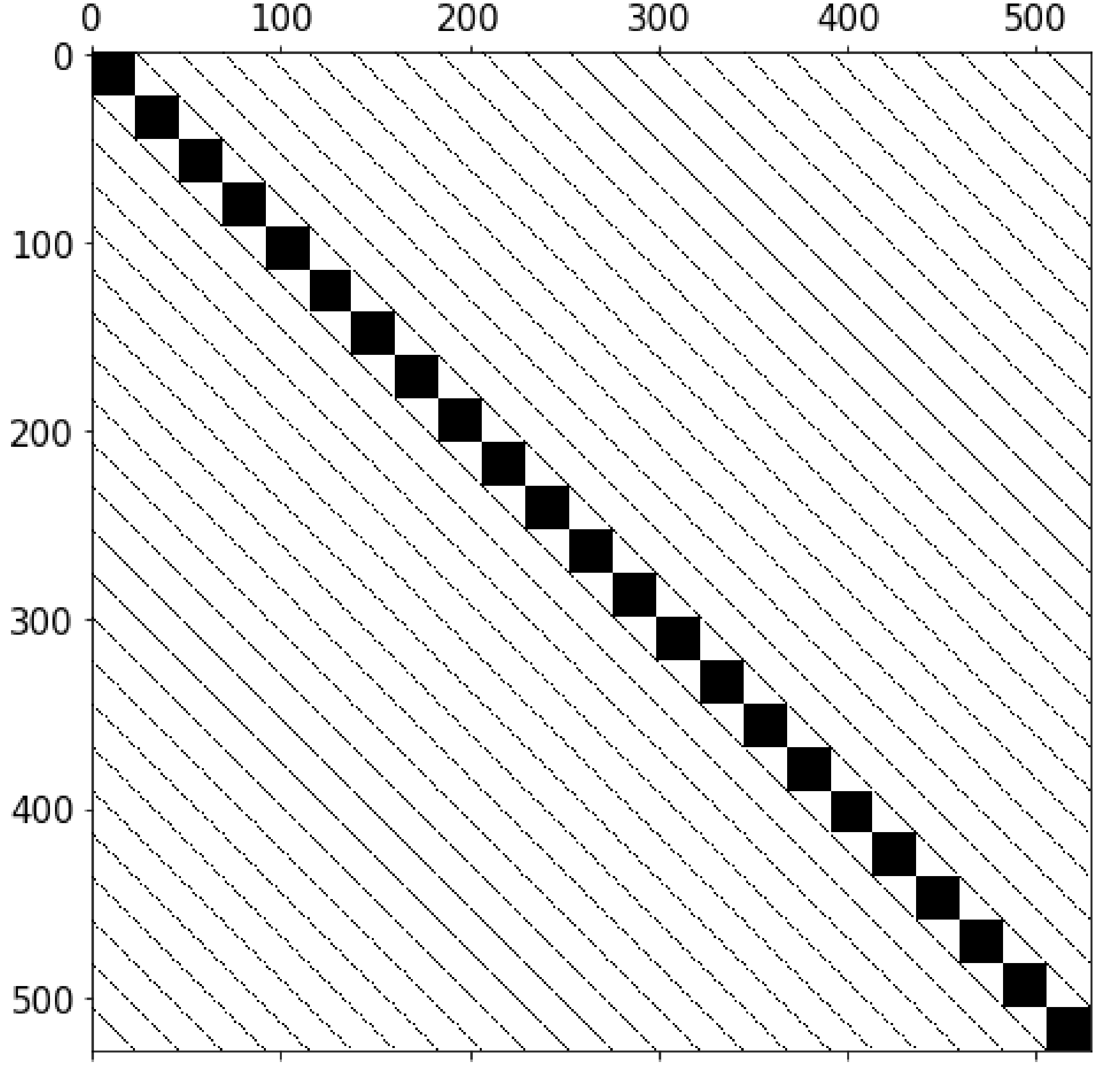
```python
matrix = np.zeros_like(D**2)
input = np.delete(np.delete((D **2),[0,N],0),[0,N],1)
matrix[1:N,1:N] = np.reshape(input,(N-1,-1)) ; matrix[0,0] = 1 ; matrix[N,N] = 1

I = np.eye(N+1)
L = np.kron(I,matrix) + np.kron(matrix,I)

plt.figure(figsize=(8,8))

plt.spy(L)
plt.xticks(fontsize=15) ; plt.yticks(fontsize = 15)
```

$$\begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \boxed{D_N^2} & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$$

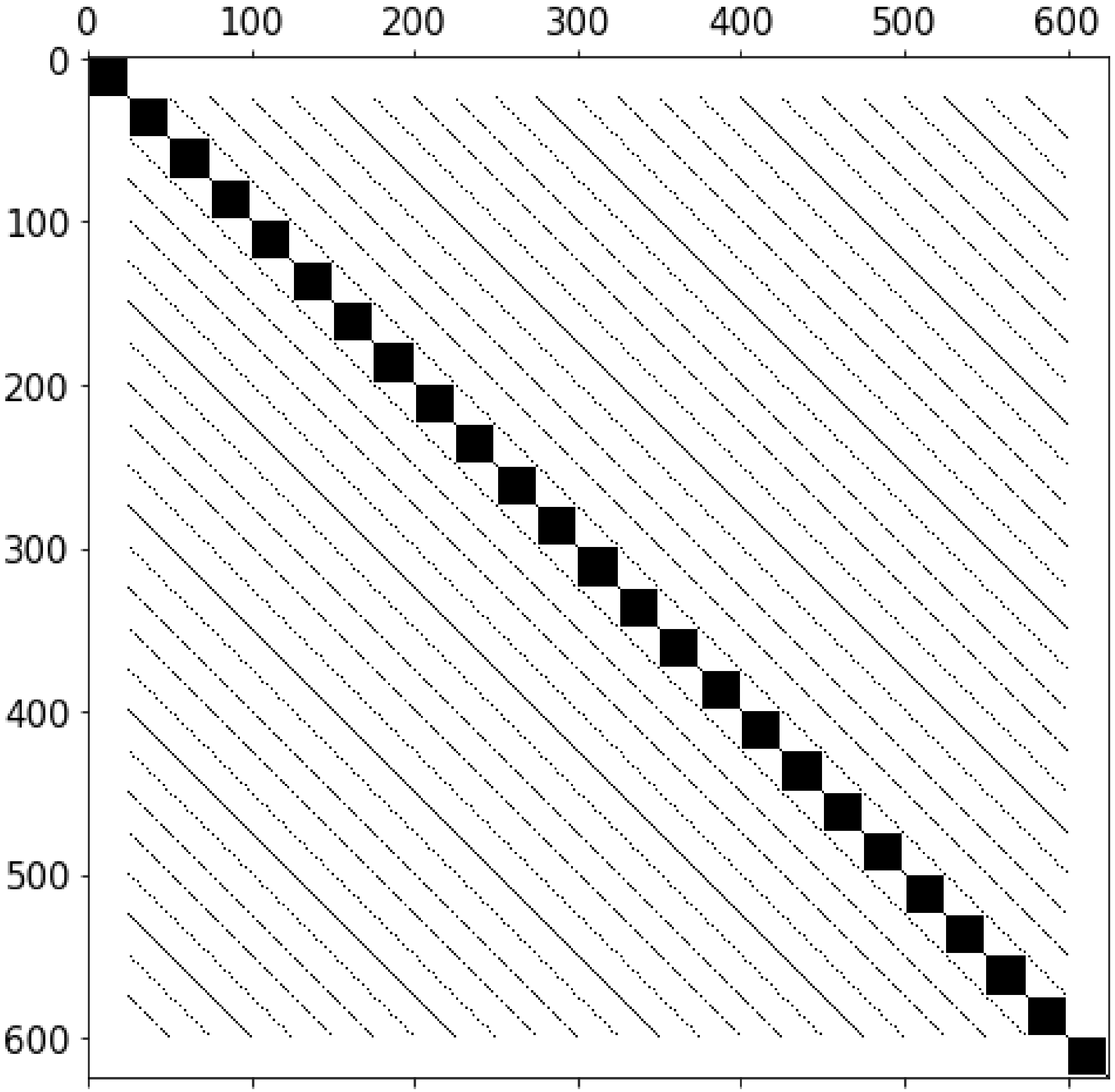
Not Tilda(?) Function



Original Method in Book

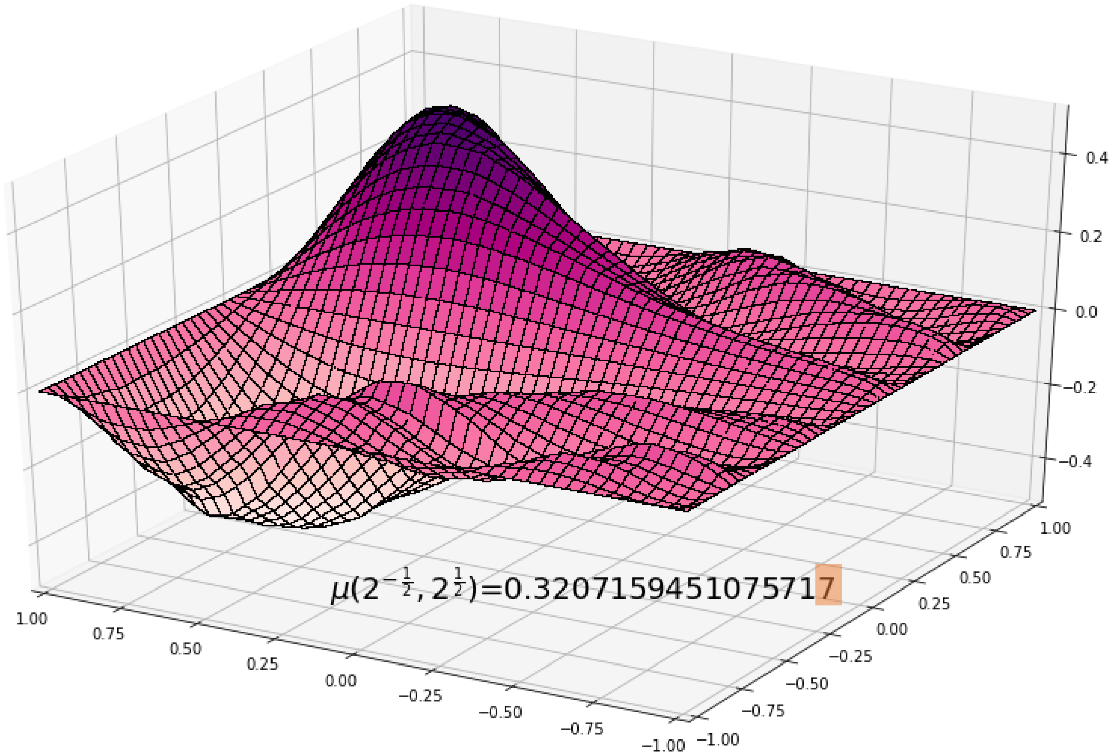


Full Matrix Method

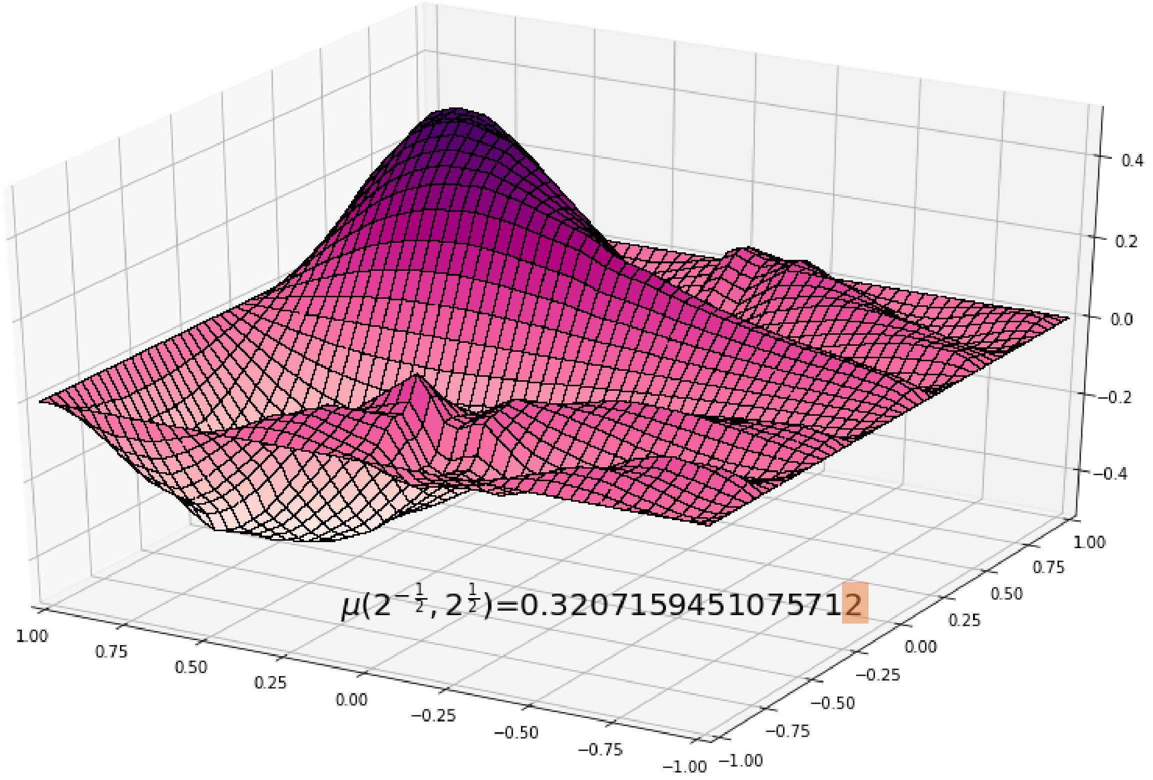


$$-1 < x, y < 1, \quad u = 0 \text{ on the boundary,}$$

Original Method in Book



Full Matrix Method



Given that the axis changes and the same fluctuation is observed, it is assumed that the matrix has an error.

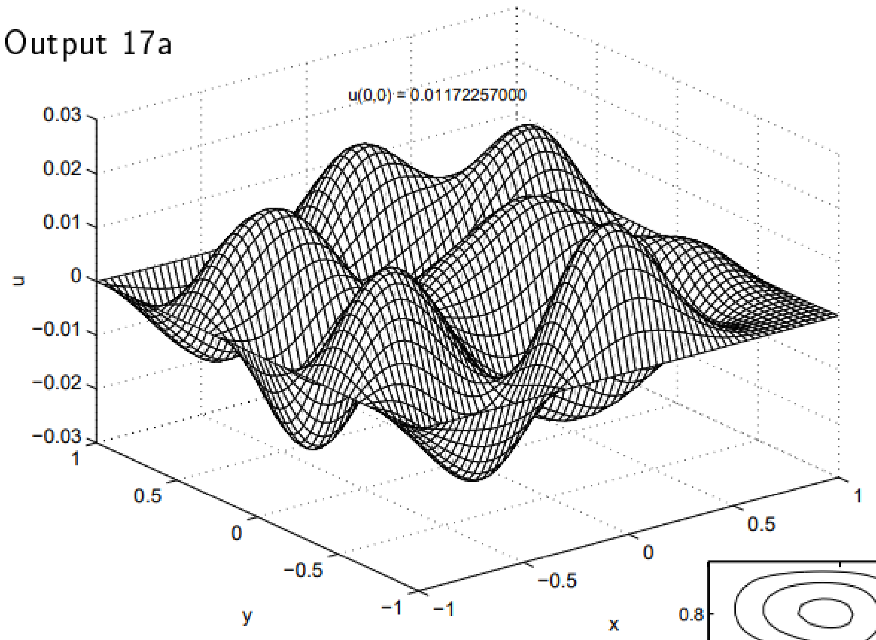
Program5. Matlab Code

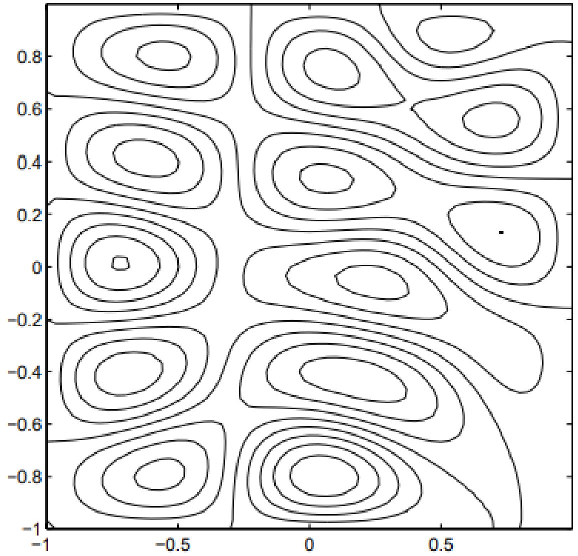
5. Implement Program 17 and produce a plot similar to Output 17. Modify the program to explicitly specify the Dirichlet boundary conditions, $u(\pm 1, y) = u(x, \pm 1) = 0$. Confirm that both results are identical (within machine precision).

Program 17

```
% p17.m - Helmholtz eq. u_xx + u_yy + (k^2)u = f
%         on [-1,1]x[-1,1]    (compare p16.m)

% Set up spectral grid and tensor product Helmholtz operator:
  N = 24; [D,x] = cheb(N); y = x;
  [xx,yy] = meshgrid(x(2:N),y(2:N));
  xx = xx(:); yy = yy(:);
  f = exp(-10*((yy-1).^2+(xx-.5).^2));
  D2 = D^2; D2 = D2(2:N,2:N); I = eye(N-1);
  k = 9;
  L = kron(I,D2) + kron(D2,I) + k^2*eye((N-1)^2);

% Solve for u, reshape to 2D grid, and plot:
  u = L\f;
  uu = zeros(N+1,N+1); uu(2:N,2:N) = reshape(u,N-1,N-1);
  [xx,yy] = meshgrid(x,y);
  [xxx,yyy] = meshgrid(-1:.0333:1,-1:.0333:1);
  uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');
  figure(1), clf, mesh(xxx,yyy,uuu), colormap([0 0 0])
  xlabel x, ylabel y, zlabel u
  text(.2,1,.022,sprintf('u(0,0) = %13.11f',uu(N/2+1,N/2+1)))
  figure(2), clf, contour(xxx,yyy,uuu)
  colormap([0 0 0]), axis square
```

Output 17a





Program5. Python Code

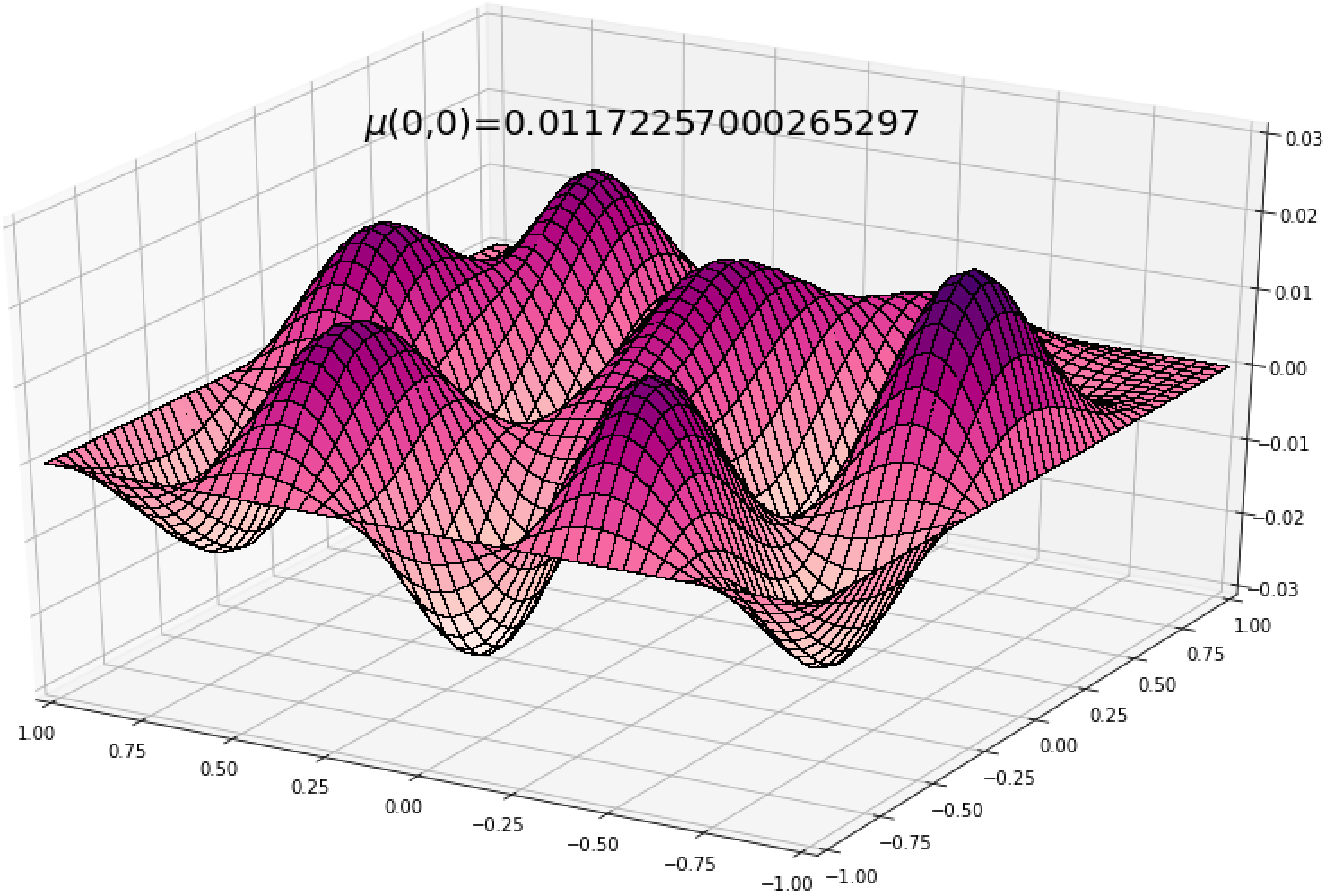
```
N = 24
D,x = cheb(N)
y = x
xx, yy = np.meshgrid(x[1:N],y[1:N])
xx = xx.flatten() ; yy = yy.flatten()
f = np.e ** (-10 * ((yy-1)**2 + (xx-0.5)**2))

D2 = np.delete(np.delete((D **2),[0,N],0),[0,N],1)
I = np.eye(N-1)
k = 9
L = np.kron(I,D2) + np.kron(D2,I) + k**2 * np.eye((N-1)**2)

u = np.linalg.solve(L,f)

uu = np.zeros((N+1,N+1)) ; uu[1:N,1:N] = np.reshape(u,(N-1,-1))
value = uu[int(N/2)][int(N/2)]

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

xxx = np.arange(-1,1 +0.04,0.04)
yyy = np.arange(-1,1 +0.04,0.04)

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

interp_spline = interp2d(xx,yy,uu,kind="cubic")
uuu = interp_spline(xxx,yyy)
```

$$k = 9, \qquad f(x, y) = \exp(-10\left[(y-1)^2 + (x-\tfrac{1}{2})^2\right]).$$

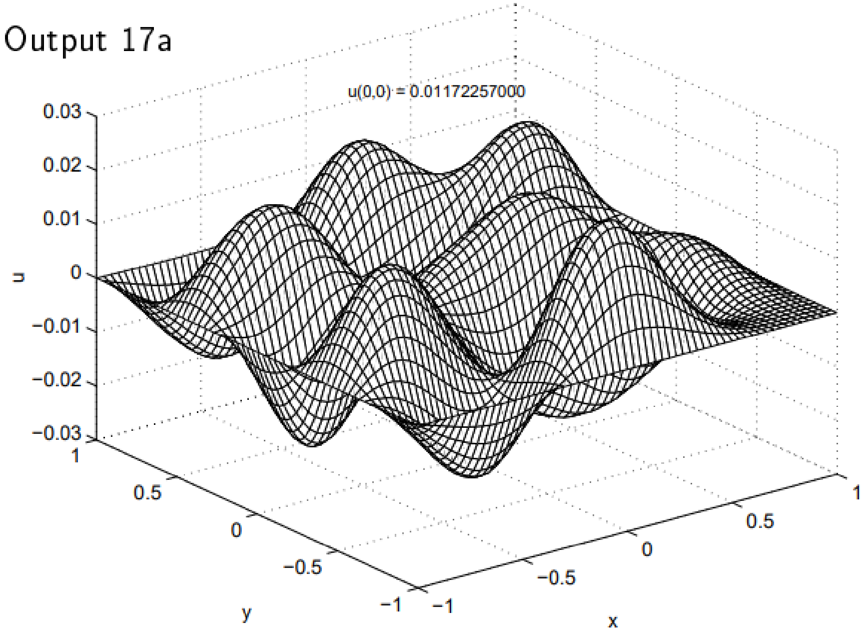
$$u_{xx} + u_{yy} + k^2 u = f(x, y), \quad -1 < x, y < 1, \quad u = 0 \text{ on the boundary},$$

$$I \otimes \widetilde{D}_N^2 = \left(\begin{array}{ccc|ccc|ccc}
-14 & 6 & -2 & & & & & & \\
4 & -6 & 4 & & & & & & \\
-2 & 6 & -14 & & & & & & \\
\hline
 & & & -14 & 6 & -2 & & & \\
 & & & 4 & -6 & 4 & & & \\
 & & & -2 & 6 & -14 & & & \\
\hline
 & & & & & & -14 & 6 & -2 \\
 & & & & & & 4 & -6 & 4 \\
 & & & & & & -2 & 6 & -14
\end{array}\right).$$

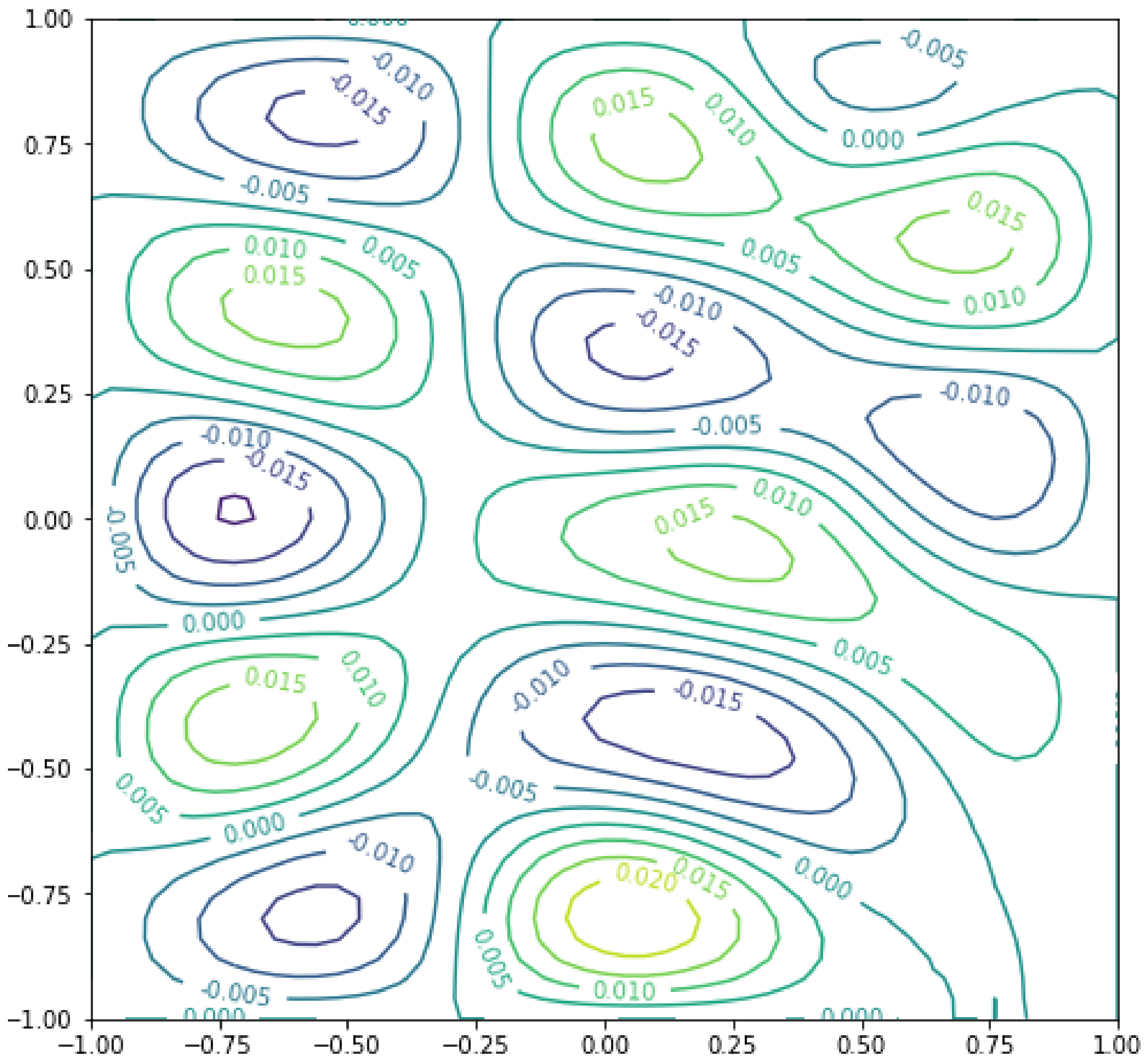
$$\widetilde{D}_N^2 \otimes I = \left(\begin{array}{ccc|ccc|ccc}
-14 & & & 6 & & & -2 & & \\
 & -14 & & & 6 & & & -2 & \\
 & & -14 & & & 6 & & & -2 \\
\hline
4 & & & -6 & & & 4 & & \\
 & 4 & & & -6 & & & 4 & \\
 & & 4 & & & -6 & & & 4 \\
\hline
-2 & & & 6 & & & -14 & & \\
 & -2 & & & 6 & & & -14 & \\
 & & -2 & & & 6 & & & -14
\end{array}\right).$$

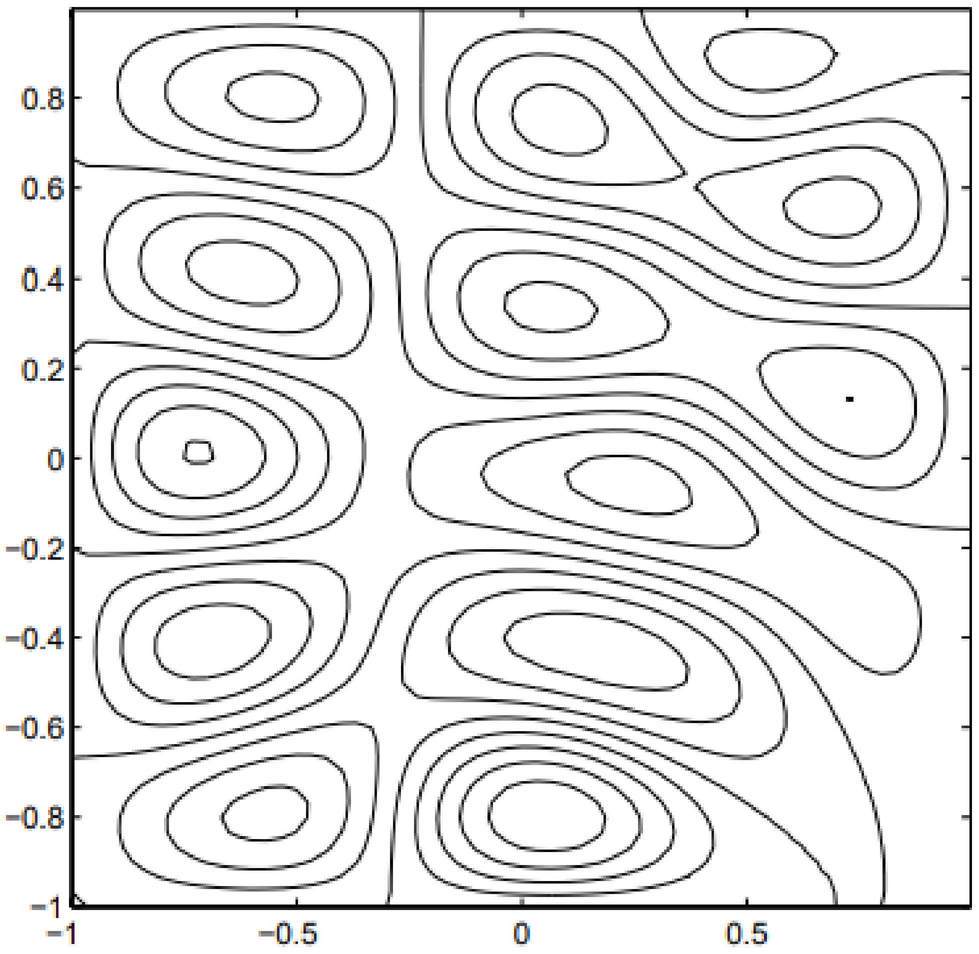


Output 17a



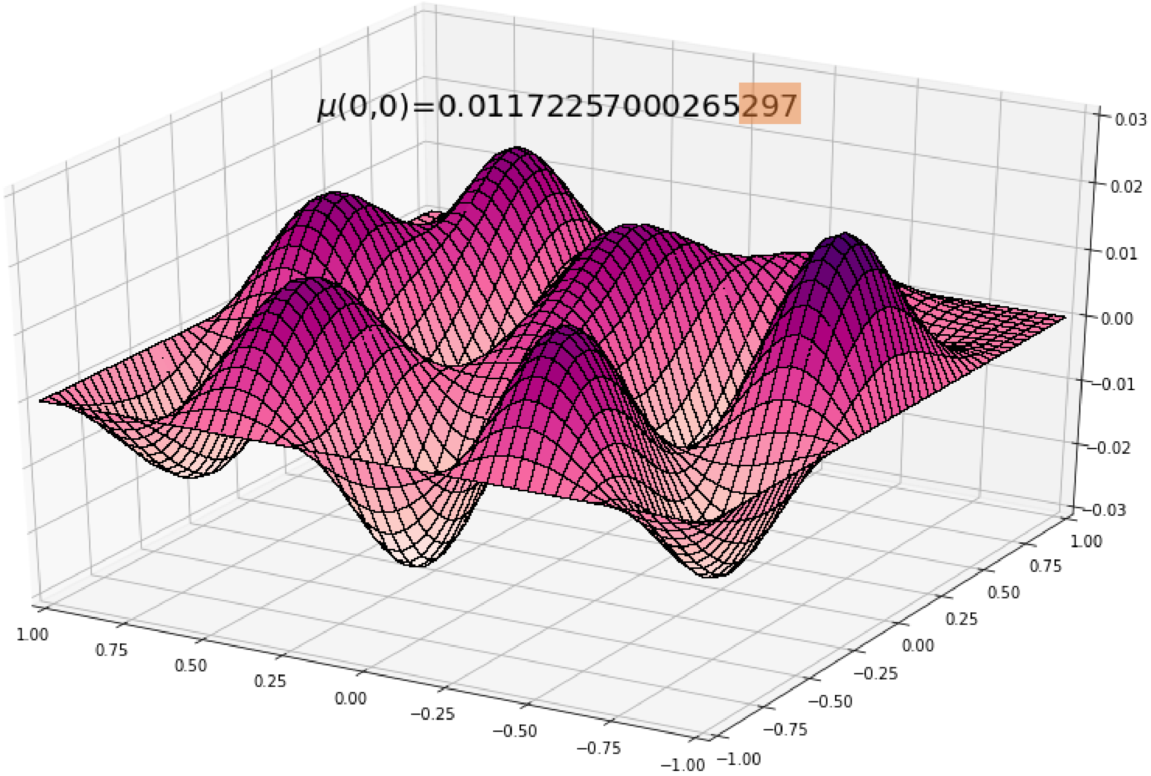
I can see that it draws well unlike No.5.



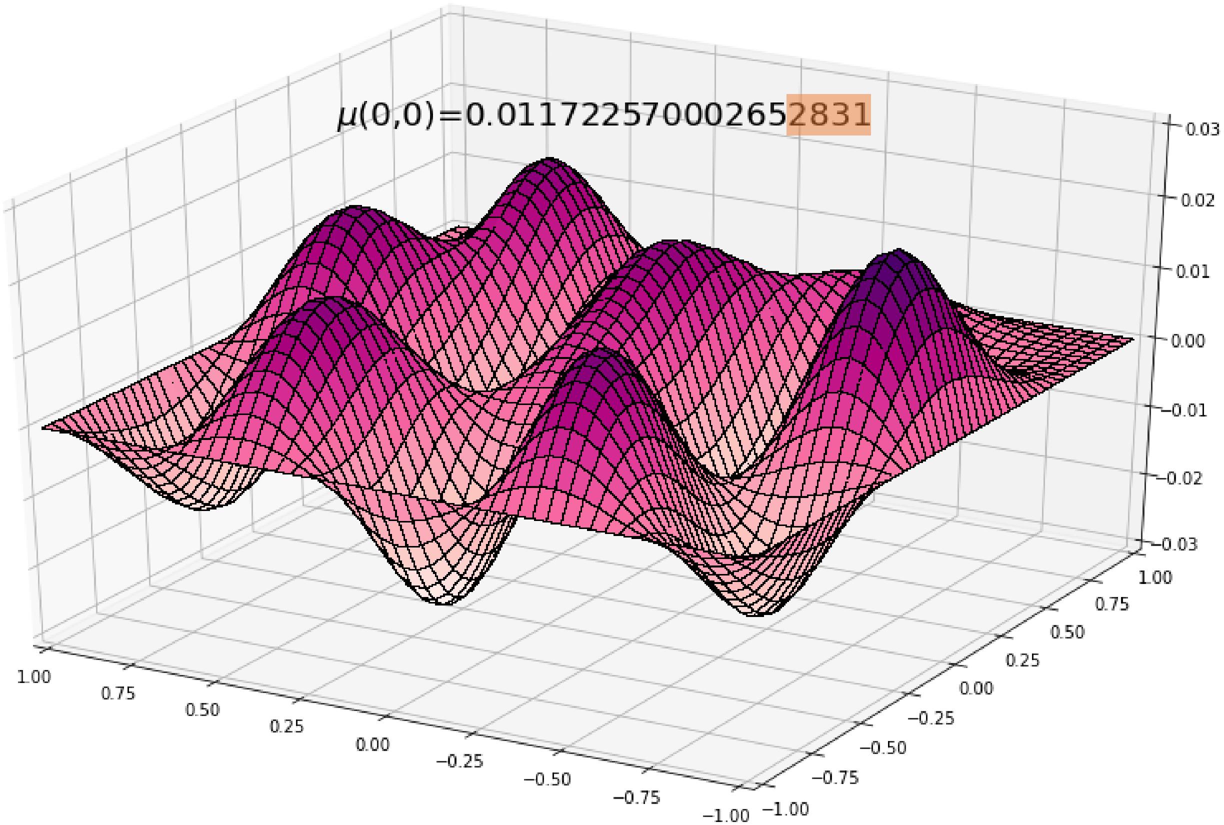


I plotted the value corresponding to u on the graph.

Original Method in Book

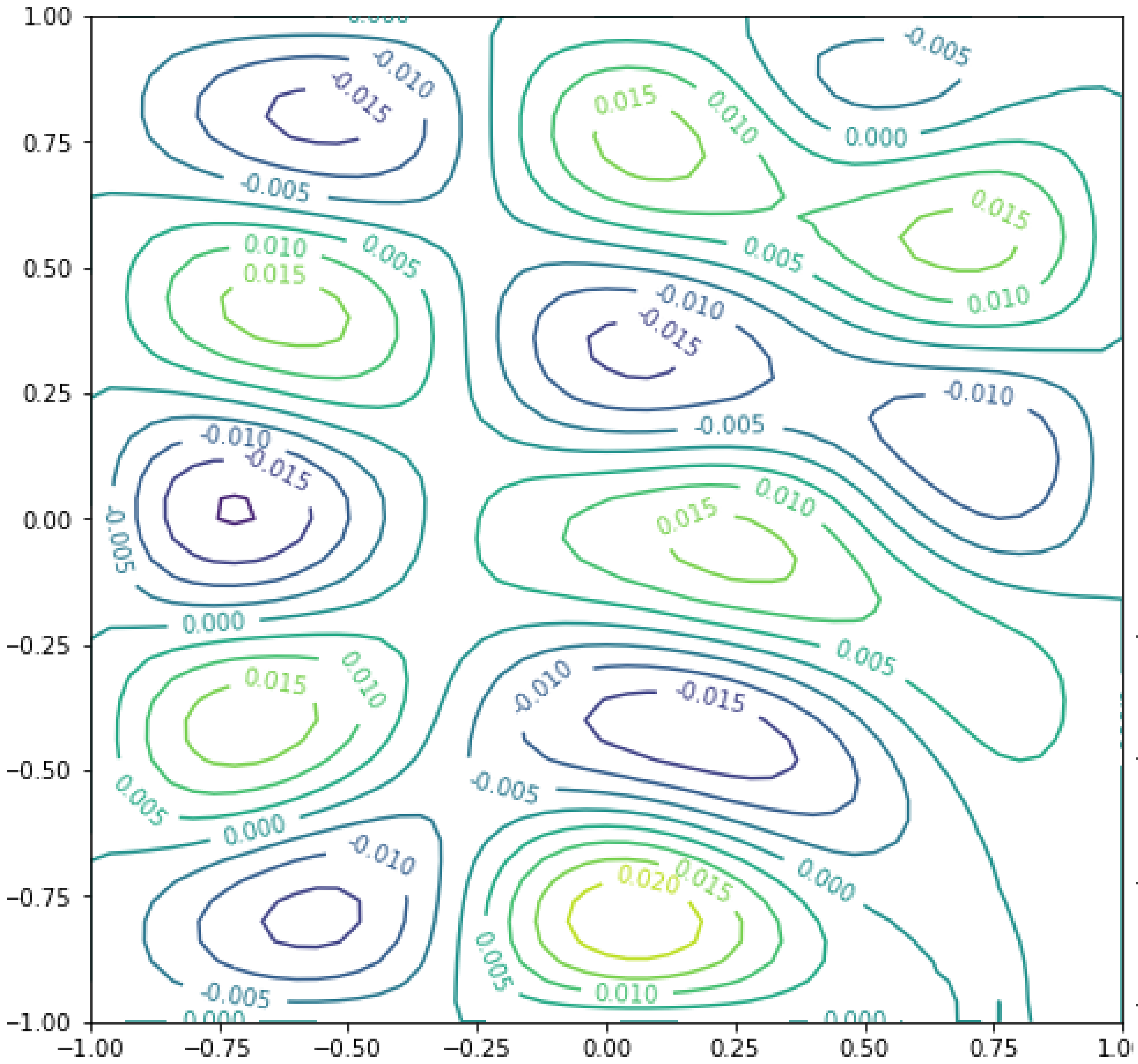


Full Matrix Method

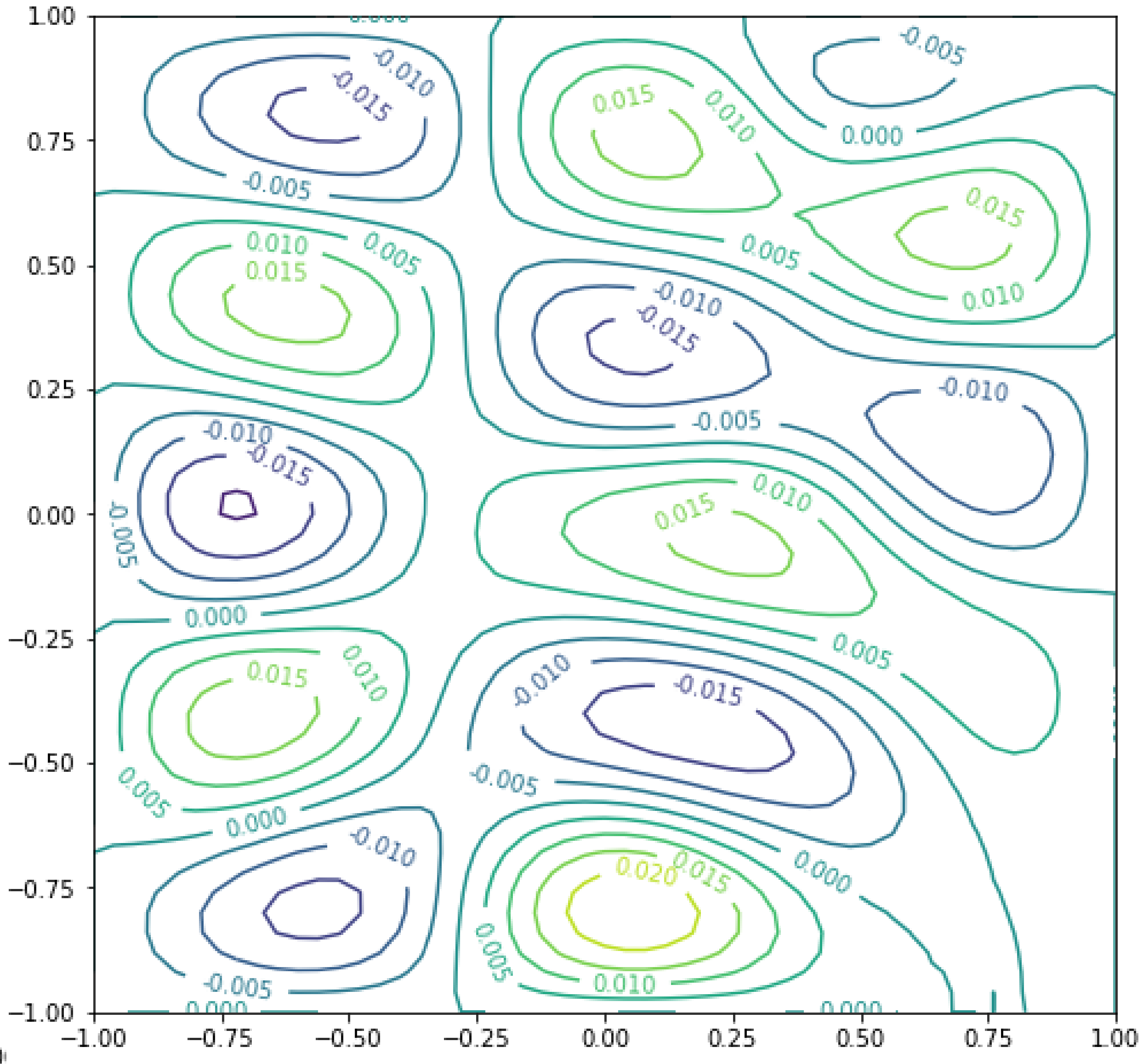


I can see that it draws well(same graph) unlike No.5.

Original Method in Book



Full Matrix Method



Program6. Python Code

6. (Exercise 7.2) Solve the boundary value problem $u_{xx} + 4u_x + e^x u = \sin(8x)$ using the spectral method on $x \in [-1, 1]$ with boundary conditions $u(\pm 1) = 0$. Using the solution, construct and plot the polynomial $p(x)$. As before, create a second version using the explicit Dirichlet boundary conditions specified. (hint: Start from Program 13.)

$$u_{xx} + 4u_x + e^x u = \sin(8x)$$

$$D_N^2 v + 4D_N^1 v + e^x v = \sin(8x)$$

$$(D_N^2 + 4D_N^1 + e^x)v = \mathcal{F}$$

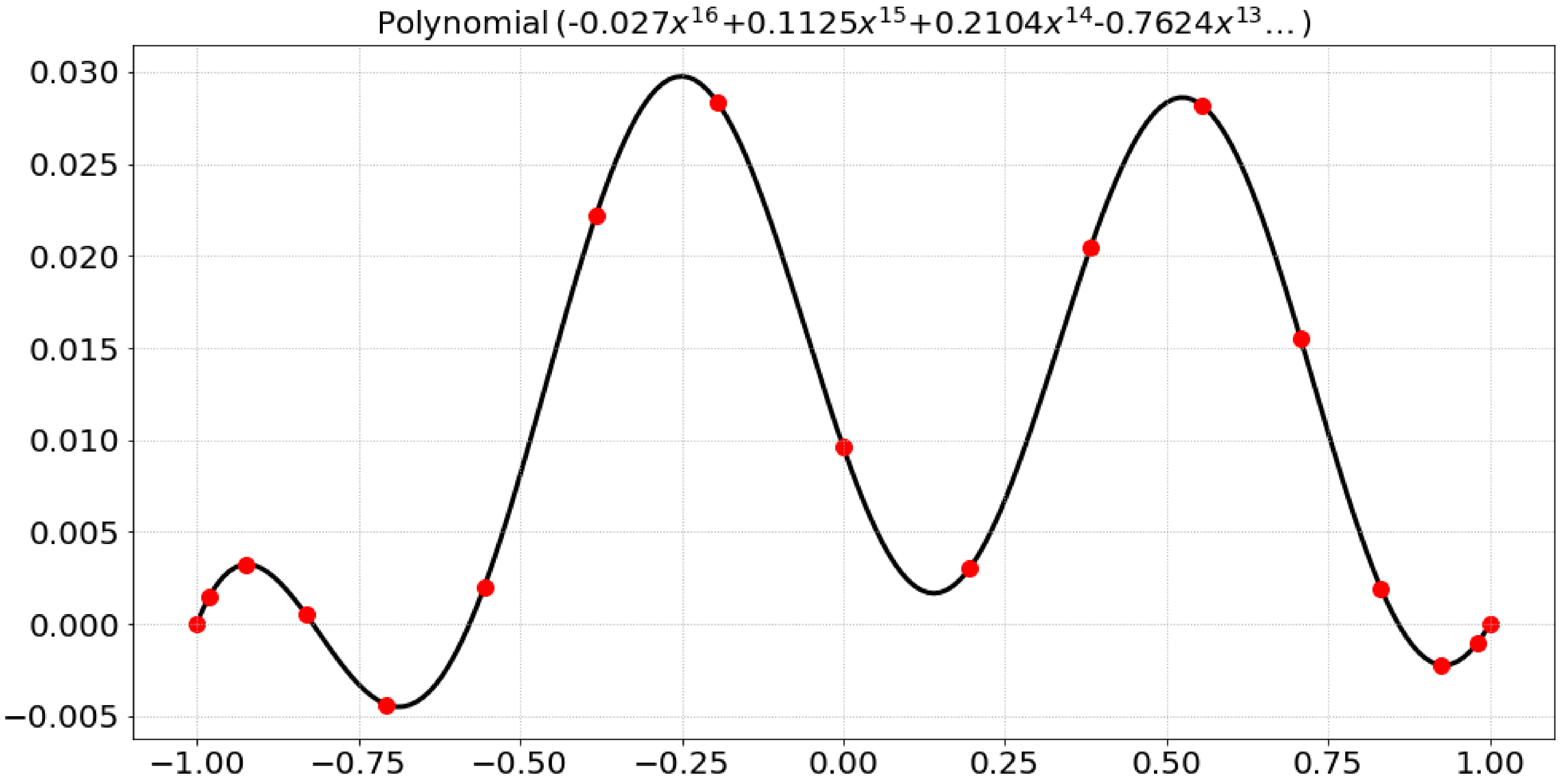
$$\mathrm{L}v = \mathcal{F}$$

$$\begin{pmatrix} e^{x_0} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & e^{x_N} \end{pmatrix}$$

```
N = 16
D ,x = cheb(N)
matrix = D**2 + 4*D + np.diagflat(np.e ** x)
D2 = np.delete(np.delete(matrix,[0,N],0),[0,N],1)
f = np.sin(8 *x[1:N])
u = np.linalg.solve(D2,f)
u = np.append(np.append([0],u),0)

xx = np.arange(-1,1 +0.01,0.01)
fit = polyfit(x,u,N)
uu = polyval(fit,xx)
```

Polyfit Value

```
[-0.02698059  0.11245209  0.210444   -0.76244019 -0.78049452  2.40618294
  1.77350127 -4.51926713 -2.58610983  5.22542545  2.29385552 -3.44441088
 -1.09202418  1.08356233  0.19821011 -0.10150461  0.00959823]
```



```
[-0.02698059  0.11245209  0.210444   -0.76244019 -0.78049452  2.40618294
  1.77350127 -4.51926713 -2.58610983  5.22542545  2.29385552 -3.44441088
 -1.09202418  1.08356233  0.19821011 -0.10150461  0.00959823]
```

Polyfit Value