Ch 8. Chebyshev Series and the FFT

Solving homework using Python

2020.12.02(전산천문학)

202050135 홍지민

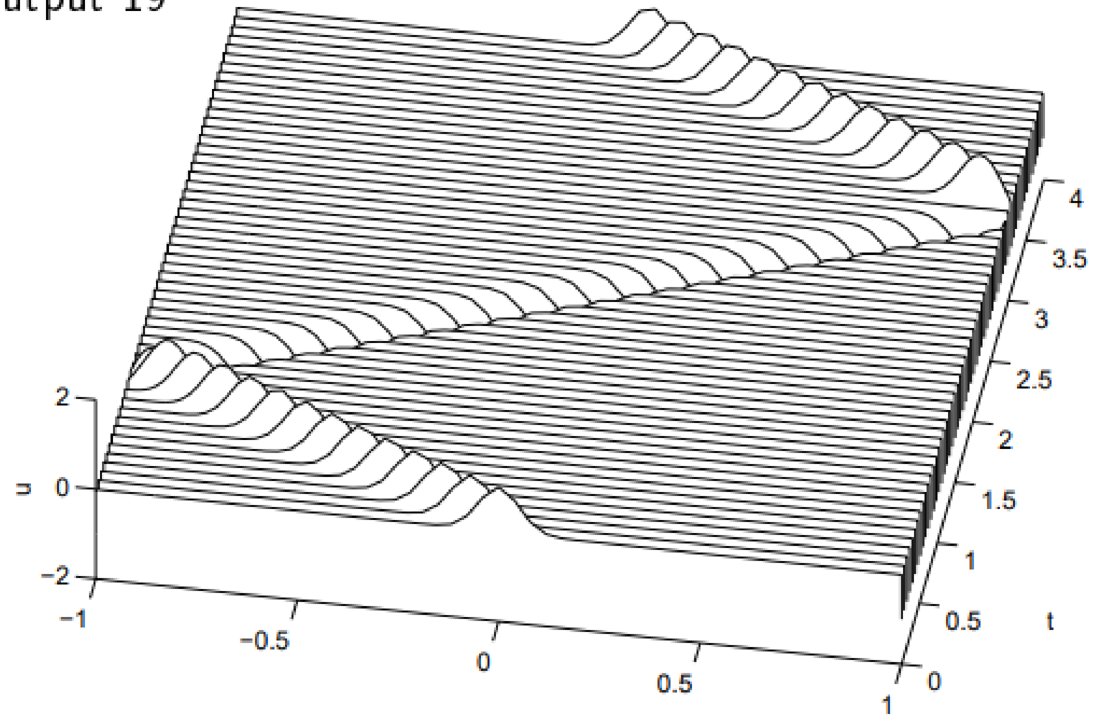
Program1. Matlab Code

1. Implement Program 19 using the Chebyshev differentiation matrix $D_N^{(2)}$ and produce a plot similar to Output 19.

Program 19

```
% p19.m - 2nd-order wave eq. on Chebyshev grid (compare p6.m)

% Time-stepping by leap frog formula:
  N = 80; x = cos(pi*(0:N)/N); dt = 8/N^2;
  v = exp(-200*x.^2); vold = exp(-200*(x-dt).^2);
  tmax = 4; tplot = .075;
  plotgap = round(tplot/dt); dt = tplot/plotgap;
  nplots = round(tmax/tplot);
  plotdata = [v; zeros(nplots,N+1)]; tdata = 0;
  clf, drawnow, h = waitbar(0,'please wait...');
  for i = 1:nplots, waitbar(i/nplots)
    for n = 1:plotgap
      w = chebfft(chebfft(v))'; w(1) = 0; w(N+1) = 0;
      vnew = 2*v - vold + dt^2*w; vold = v; v = vnew;
    end
    plotdata(i+1,:) = v; tdata = [tdata; dt*i*plotgap];
  end

% Plot results:
  clf, drawnow, waterfall(x,tdata,plotdata)
  axis([-1 1 0 tmax -2 2]), view(10,70), grid off
  colormap([0 0 0]), ylabel t, zlabel u, close(h)
```

Output 19



Output 19: Solution of second-order wave equation (8.8).

Program1. Python Code

```
N = 80 ; D,x = cheb(N) ; dt = 8/N**2 ; x = x[1:N]
D2 = np.delete(np.delete((D **2),[0,N],0),[0,N],1)
v = np.e**(-200*(x**2)) ; vold = np.e ** (-200*((x-dt)**2))
tmax = 4 ; tplot = 0.075 ; t = 0
plotgap =round(tplot/dt) ; dt = tplot/plotgap
nplots = round(tmax/tplot)
data = []
data.append(list(zip(x,v)))
tdata = [0]

for i in np.arange(1,nplots,1):
  for n in np.arange(1,plotgap +1,1):
    vnew = (dt**2) * np.array(D2*np.reshape(v,(-1,1))).flatten() + 2*v - vold
    vold = v ; v = vnew
  data.append(list(zip(x,v))) ; tdata = np.append(tdata,dt*i*plotgap)

from matplotlib.collections import LineCollection
import mpl_toolkits.mplot3d

fig = plt.figure(figsize=(16,10))
ax = plt.axes(projection='3d')
line = LineCollection(data)
ax.add_collection3d(line,zs=tdata,zdir='y')
ax.set_xlim3d(-1,1)
ax.set_ylim3d(0,4)
ax.set_zlim3d(-2,2)
```

$$function: e^{-200x^2}$$

$$old\ function: e^{-200(x-dt)^2}$$

$$t_{max} = 4s$$

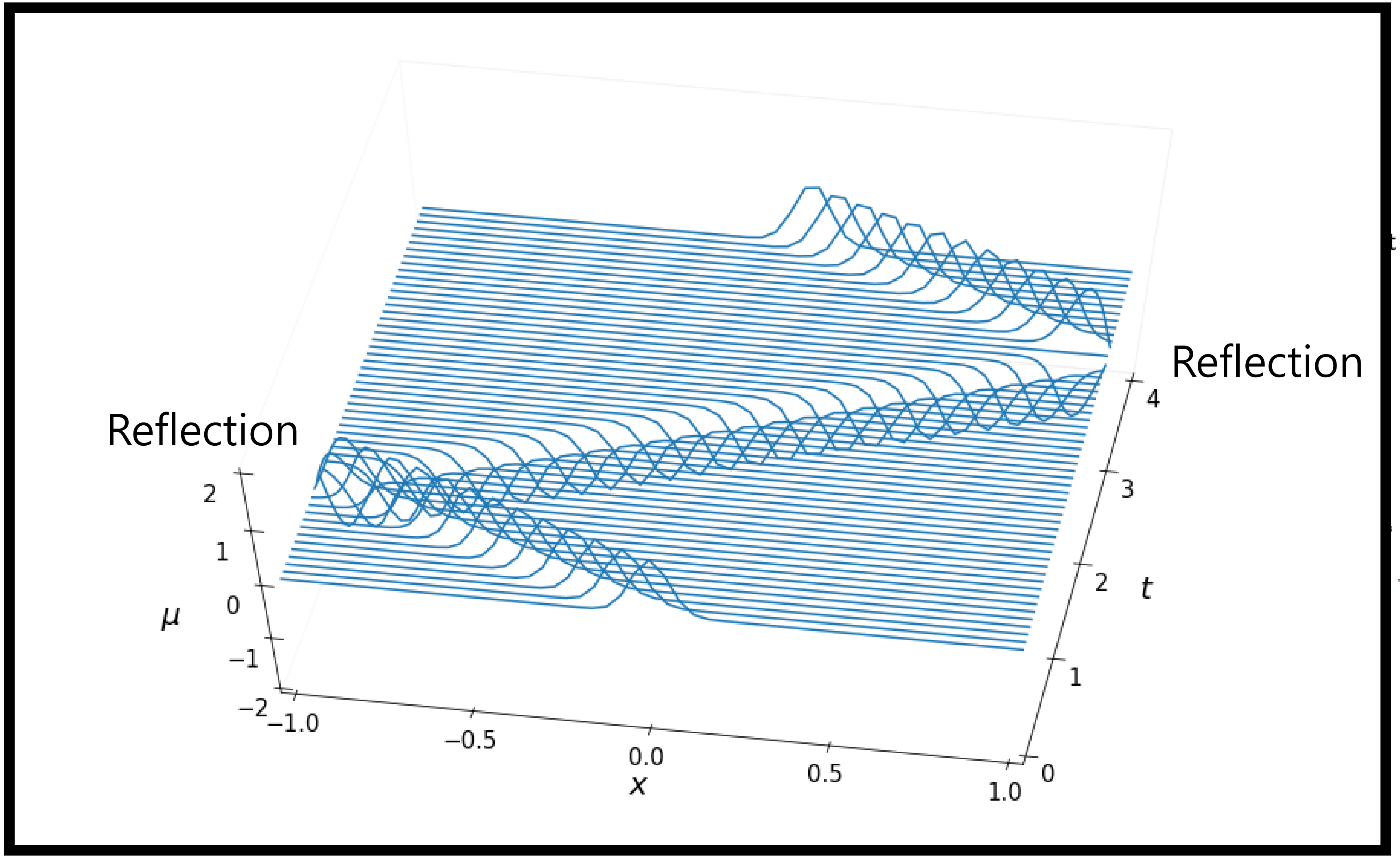
Tie the corresponding values together to make each line graph.

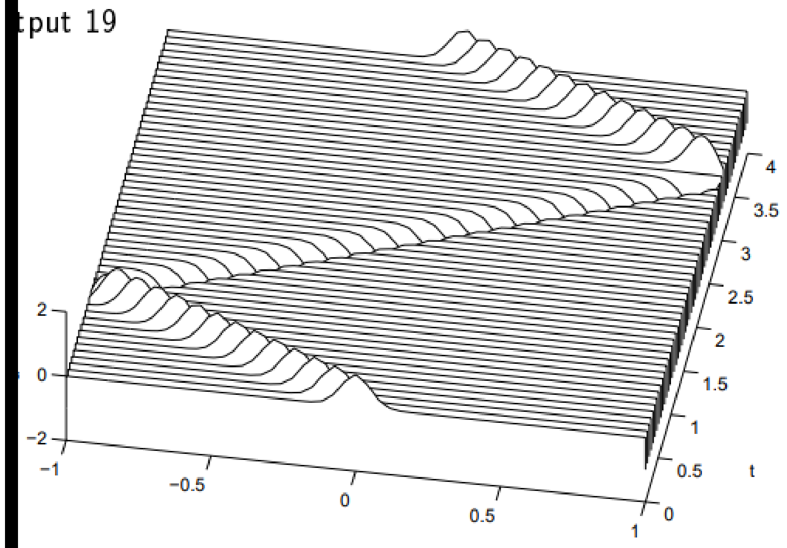
$$u_{tt} = \frac{u_{i+1} - 2u_i + u_{i-1}}{2\Delta t}$$

$$u_{tt} \approx D_N^2 v$$

plot

Change $u_{i+1}(new)$, $u_i(current)$, $u_{i-1}(old)$





Output 19: Solution of second-order wave equation (8.8).

Program2. Matlab Code

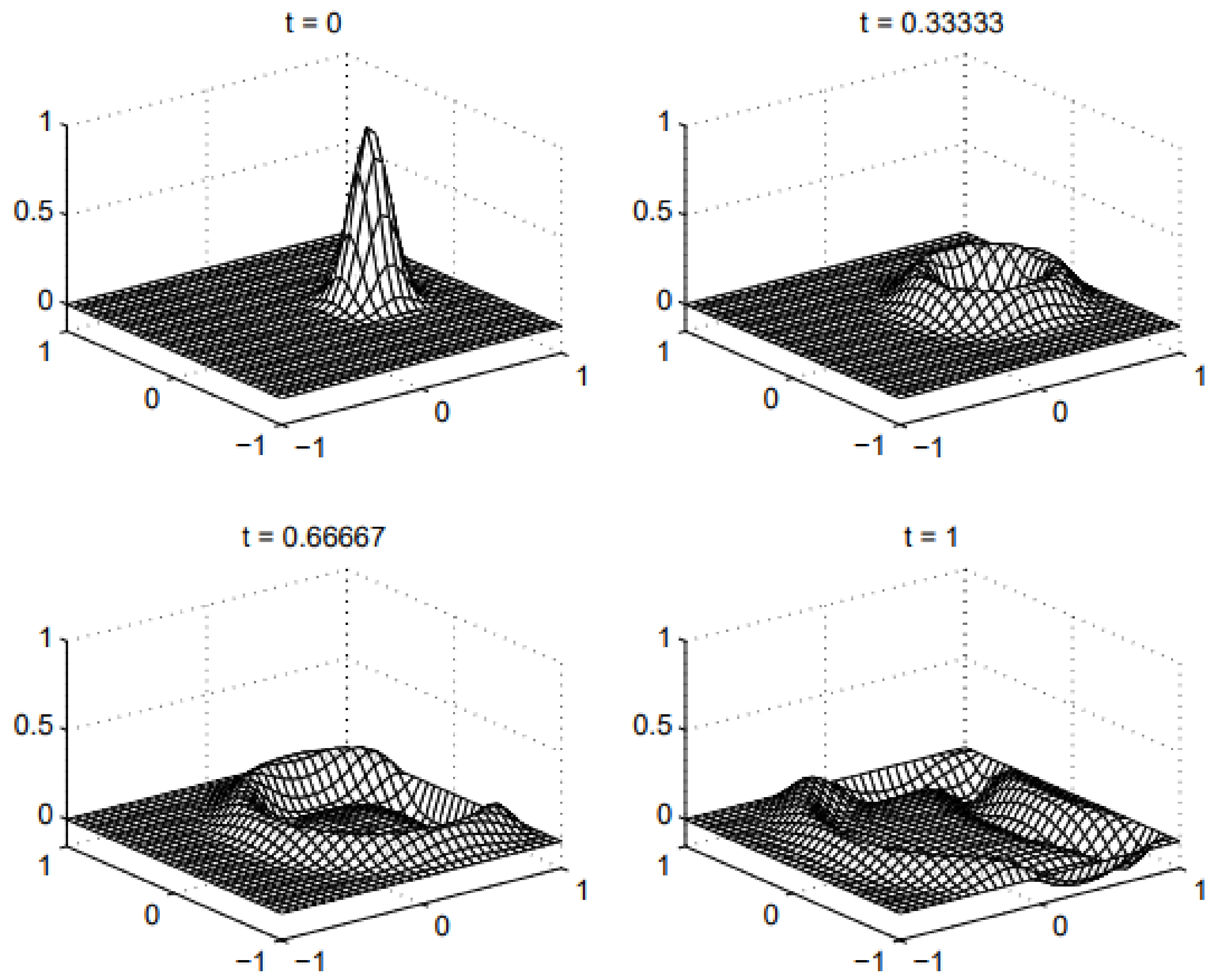
2. Implement Program 20 using the Chebyshev differentiation matrix $D_N^{(2)}$ and produce a plot similar to Output 20.

Program 20

```
% p20.m - 2nd-order wave eq. in 2D via FFT (compare p19.m)

% Grid and initial data:
  N = 24; x = cos(pi*(0:N)/N); y = x';
  dt = 6/N^2;
  [xx,yy] = meshgrid(x,y);
  plotgap = round((1/3)/dt); dt = (1/3)/plotgap;
  vv = exp(-40*((xx-.4).^2 + yy.^2));
  vvold = vv;

% Time-stepping by leap frog formula:
  [ay,ax] = meshgrid([.56 .06],[.1 .55]); clf
  for n = 0:3*plotgap
    t = n*dt;
    if rem(n+.5,plotgap)<1     % plots at multiples of t=1/3
      i = n/plotgap+1;
      subplot('position',[ax(i) ay(i) .36 .36])
      [xxx,yyy] = meshgrid(-1:1/16:1,-1:1/16:1);
      vvv = interp2(xx,yy,vv,xxx,yyy,'cubic');
      mesh(xxx,yyy,vvv), axis([-1 1 -1 1 -0.15 1])
      colormap([0 0 0]), title(['t = ' num2str(t)]), drawnow
    end
```

```
    uxx = zeros(N+1,N+1); uyy = zeros(N+1,N+1);
    ii = 2:N;
    for i = 2:N                      % 2nd derivs wrt x in each row
      v = vv(i,:); V = [v fliplr(v(ii))];
      U = real(fft(V));
      W1 = real(ifft(1i*[0:N-1 0 1-N:-1].*U)); % diff wrt theta
      W2 = real(ifft(-[0:N 1-N:-1].^2.*U));    % diff^2 wrt theta
      uxx(i,ii) = W2(ii)./(1-x(ii).^2) - x(ii).* ...
                  W1(ii)./(1-x(ii).^2).^(3/2);
    end
    for j = 2:N                      % 2nd derivs wrt y in each column
      v = vv(:,j); V = [v; flipud(v(ii))];
      U = real(fft(V));
      W1 = real(ifft(1i*[0:N-1 0 1-N:-1]'.*U));% diff wrt theta
      W2 = real(ifft(-[0:N 1-N:-1]'.^2.*U));   % diff^2 wrt theta
      uyy(ii,j) = W2(ii)./(1-y(ii).^2) - y(ii).* ...
                  W1(ii)./(1-y(ii).^2).^(3/2);
    end
    vvnew = 2*vv - vvold + dt^2*(uxx+uyy);
    vvold = vv; vv = vvnew;
  end
```

Output 20



Output 20: Solution of the second-order wave equation (8.9) on a square.

Program2. Python Code

```
N = 24 ; D,x = cheb(N) ; y = x
dt = 6/N**2

xx,yy = np.meshgrid(x[1:N],y[1:N])
xx = xx.flatten() ; yy = yy.flatten()
plotgap = round((1/3)/dt) ; dt = (1/3)/plotgap

vv = np.e**(-40*((xx - 0.4)**2 + yy**2))
vv = np.reshape(vv,(N-1,-1))
uu = np.zeros((N+1,N+1)) ; uu[1:N,1:N] = vv

vvold = vv

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

fig = plt.figure(figsize=(16,10))
ax = plt.axes(projection='3d')

for n in np.arange(1,3*plotgap +1,1):
  L = D2 * vv + vv * D2.T
  vvnew = (dt**2) * L + 2*vv - vvold
  vvold = vv ; vv = vvnew

uu = np.zeros((N+1,N+1)) ; uu[1:N,1:N] = vvnew
```

$function: e^{-40((x-0.4)^2+y^2)}$

Set Array formula to Matrix formula

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

But for the convenience of the calculation.

$$using : L_N = D_N^2 v + v(D_N^2)^T$$

x component	y component

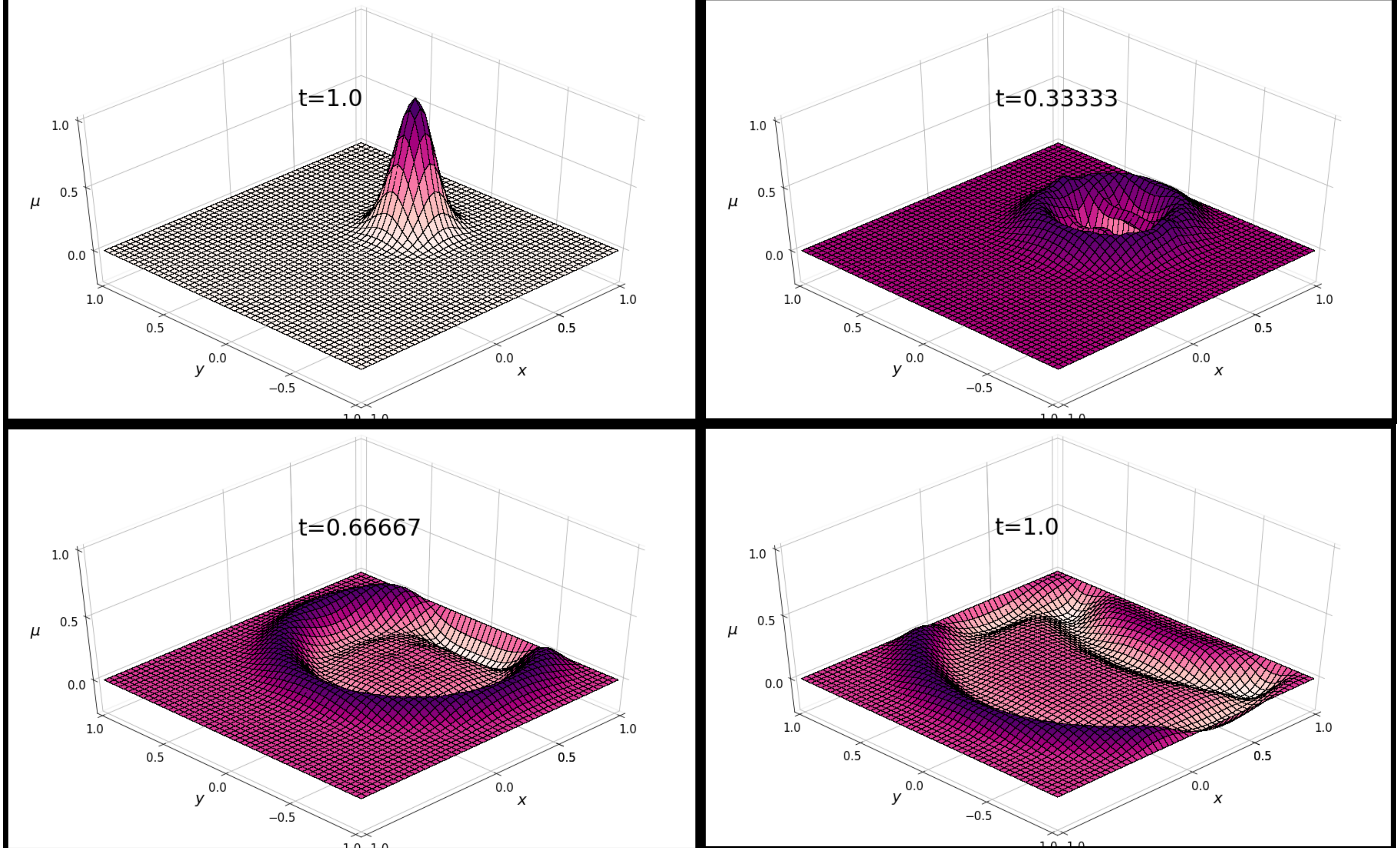
$u_{tt} \approx L_N$

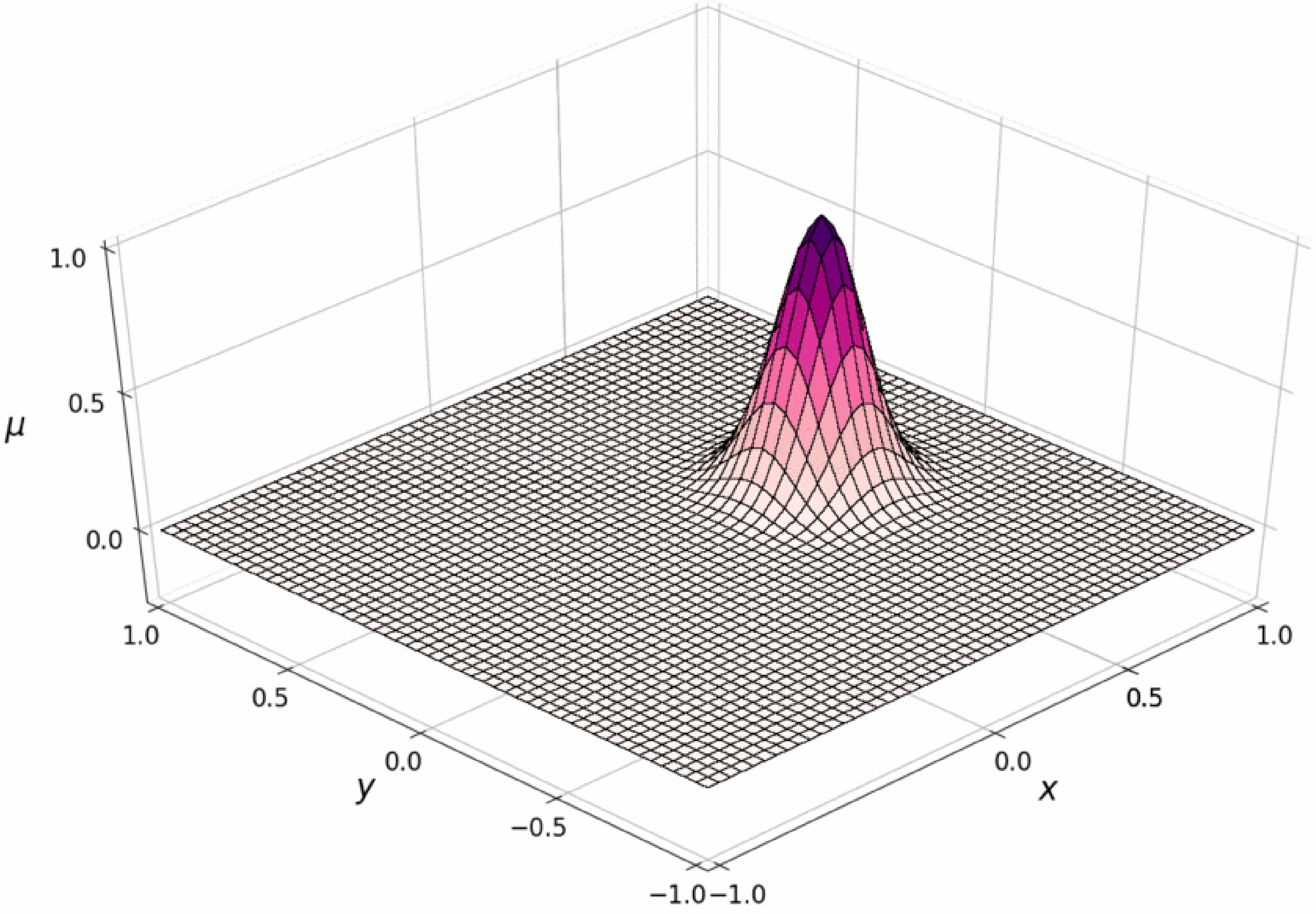
$$u_{tt} = \frac{u_{i+1} - 2u_i + u_{i-1}}{2\Delta t}$$

Program2. Python Code

```
vvold = vv

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

fig = plt.figure(figsize=(16,10))
ax = plt.axes(projection='3d')

for n in np.arange(1,3*plotgap +1,1):
  L = D2 * vv + vv * D2.T
  vvnew = (dt**2) * L + 2*vv - vvold
  vvold = vv ; vv = vvnew

uu = np.zeros((N+1,N+1)) ; uu[1:N,1:N] = vvnew

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='quintic')
uuu = interp_spline(xxx,yyy)
ax.plot_surface(yi,xi,uuu,rstride=1,cstride=1,cmap=cm.RdPu,linewidth=0.005,edgeco
ax.set_xlim(1,-1) ; ax.set_ylim(-1,1) ; ax.set_zlim(-0.25,1)
```

Calculate interpolation 2D Graph


t=1.0
t=0.33333
t=0.66667
t=1.0
μ
x
y



In some cases, small fluctuations were observed due to different interpolation library calculations.