Mathematical Physics - III (Practical)
Paper: PHS-A-CC-4-8-P
Solution of PDE

1. Solution of ODE/PDE:
(b) Boundary value problems: Finite difference method with fixed step size. Application to simple physical problems.

Boundary Value Problems:

Equation: $y'' + ay' + by = 0, \quad y(0) = y_1, y(L) = y_2$

An example of boundary value problem is heat conduction along a long thin rod with length L.

Finite Difference method:

Let us consider a function u(x, t). Its spatial derivatives are found from the Taylor series expansion

$$u(x + \Delta x, t) = u(x, t) + \Delta x \frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2!} \frac{\partial^2 u}{\partial x^2} + \mathcal{O}[(\Delta x)^3$$

The first order approximation:

(i) Forward difference: $\frac{\partial u}{\partial x} = \frac{u(x+\Delta x,t)-u(x,t)}{\Delta x} + \mathcal{O}(\Delta x)$
(ii) Backward difference: $\frac{\partial u}{\partial x} = \frac{u(x,t)-u(x-\Delta x,t)}{\Delta x} + \mathcal{O}(\Delta x)$
(iii) Central difference: $\frac{\partial u}{\partial x} = \frac{u(x+\Delta x,t)-u(x-\Delta x,t)}{2\Delta x} + \mathcal{O}[(\Delta x^2)]$

The Second order approximation:

(i) Forward difference: $\frac{\partial^2 u}{\partial x^2} = \frac{u(x+2\Delta x,t)-2u(x+\Delta x,t)+u(x,t)}{(\Delta x)^2} + \mathcal{O}(\Delta x)$
(ii) Backward difference: $\frac{\partial^2 u}{\partial x^2} = \frac{u(x,t)-2u(x-\Delta x,t)+u(x-2\Delta x,t)}{(\Delta x)^2} + \mathcal{O}(\Delta x)$
(iii) Central difference: $\frac{\partial^2 u}{\partial x^2} = \frac{u(x+\Delta x,t)-2u(x,t)+u(x-\Delta x,t)}{(\Delta x)^2} + \mathcal{O}(\Delta x^2)$

As the $x - t$ plane is discretized, the coordinates (x, t) can be in general denoted by (x_i, t_i) and one increment by h (where $\Delta x = h$) in x- direction and k in t-direction.

We use the following notation

$$u_{ij} = u(x,t), u_{i+1,j} = u(x_i + h, t_i), u_{i,j+1} = u(x_i, t_i + k)$$

Thus for (i) forward difference: $\frac{\partial u}{\partial x} \approx \frac{u_{i+1,j} - u_{ij}}{h}$

(ii) Backward difference: $\frac{\partial u}{\partial x} \approx \frac{u_{i,j} - u_{i-1,j}}{h}$

(iii). Central difference: $\frac{\partial u}{\partial x} \approx \frac{u_{i+1,j} - u_{i-1,j}}{2h}$

In discrete notation, the second order derivative with central difference

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2}$$

Boundary value problem by finite difference method

$$y'' = 12x^2;\ \ y(0) = 0\ , y(1) = 0$$

Python script:

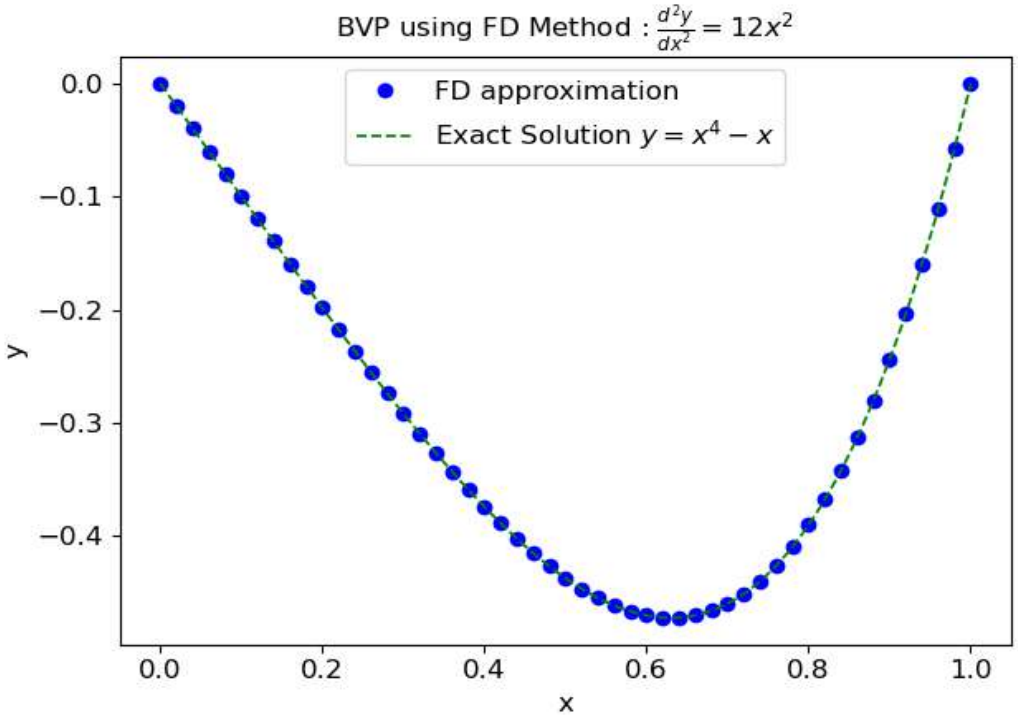
```
import numpy as np
from scipy import sparse
import matplotlib.pyplot as plt

L = 50; dx = 1.0/L

x = np.linspace(0, 1, L+1).transpose()
b = np.zeros((L+1, 1)).ravel()
b[1:L] = 12 * pow(dx,2) * (x[1:L]**2)
main_diag = -2*np.ones((L+1,1)).ravel()
off_diag =   1*np.ones((L,  1)).ravel()
a = main_diag.shape[0]
diagonals = [main_diag, off_diag, off_diag]
A = sparse.diags(diagonals, [0,-1,1], shape=(a,a)).toarray()
```

```
A[0, 0  ] = 1
A[0, 1  ] = 0
A[L, L  ] = 1
A[L, L-1] = 0


y = np.linalg.solve(A,b)
xf = np.linspace(0, 1, 10*L)
yexact = pow(xf,4) - xf
plt.plot(x,  y,      'bo', label='FD approximation', lw=1.2, ms=6)
plt.plot(xf, yexact, 'g--', label=r'Exact Solution $y=x^4-x$',   lw=1.2, ms=6)
plt.legend(loc='best', prop={'size':12})
plt.title(r'BVP using FD Method : $\frac{d^2y}{dx^2} = 12x^2$',size=12)
plt.xlabel('x', size = 12)
plt.xticks(size = 12)
plt.ylabel('y', size = 12)
plt.yticks(size = 12)
plt.tight_layout()
plt.show()
```



Classification of PDEs

Second order linear PDEs can be formally classified into three generic types: elliptic, parabolic and hyperbolic. The simplest examples are:

a) Elliptic: $U_{xx} + U_{yy} = f(x, y)$

This is Poisson's equation or Laplace's equation which may be used to model the steady state temperature distribution in a plate or incompressible potential flow.

b) Parabolic: $U_t = kU_{xx}$

This is the 1D diffusion equation and can be used to model time dependent temperature distribution along a heated 1D bar.

c) Hyperbolic: $U_{tt} = c^2 U_{xx}$

This is the wave equation and may be used to model a vibrating guitar string or 1D supersonic flow.

a) Laplace equation: $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$

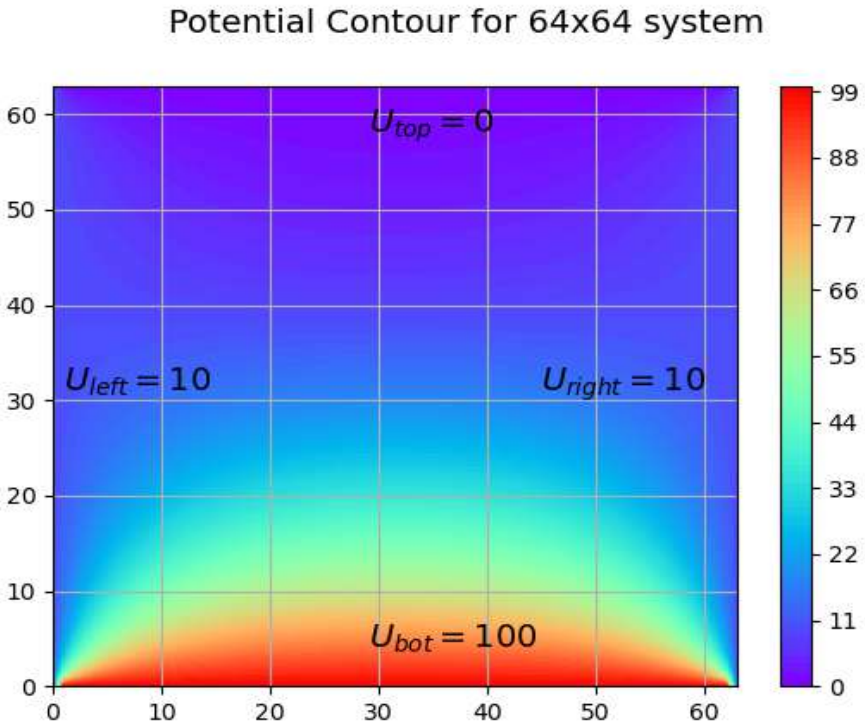
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm

maxiter = 500; L = 64; dx = 1
U = np.zeros((L,L))
Utop = 0; Ubot = 100; Uleft = 10; Uright = 10
U[(L-1):,:    ] = Utop
U[:1    ,:    ] = Ubot
U[:    ,(L-1):] = Uright
U[:    ,:1    ] = Uleft

for iteration in range(0, maxiter):
  for i in range(1, L-1, dx):
    for j in range(1, L-1, dx):
      U[i,j] = 0.25*(U[i+1][j] + U[i-1][j] + U[i][j+1] + U[i][j-1])

X,Y = np.meshgrid(np.arange(0,L), np.arange(0,L))
plt.suptitle("Potential Contour for "+str(L)+"x"+str(L)+" system", size=14)
```

```
plt.contourf(X, Y, U, 100, cmap=cm.rainbow)
plt.text(29, 4,r'$U_{bot} = $'  +str(Ubot), size = 14)
plt.text(29,58,r'$U_{top} = $'  +str(Utop), size = 14)
plt.text(1, 31,r'$U_{left} = $' +str(Uleft),size = 14)
plt.text(45,31,r'$U_{right} = $'+str(Uleft),size = 14)
plt.colorbar()
plt.grid()
plt.show()
```



b) Diffusion in one dimension: $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$

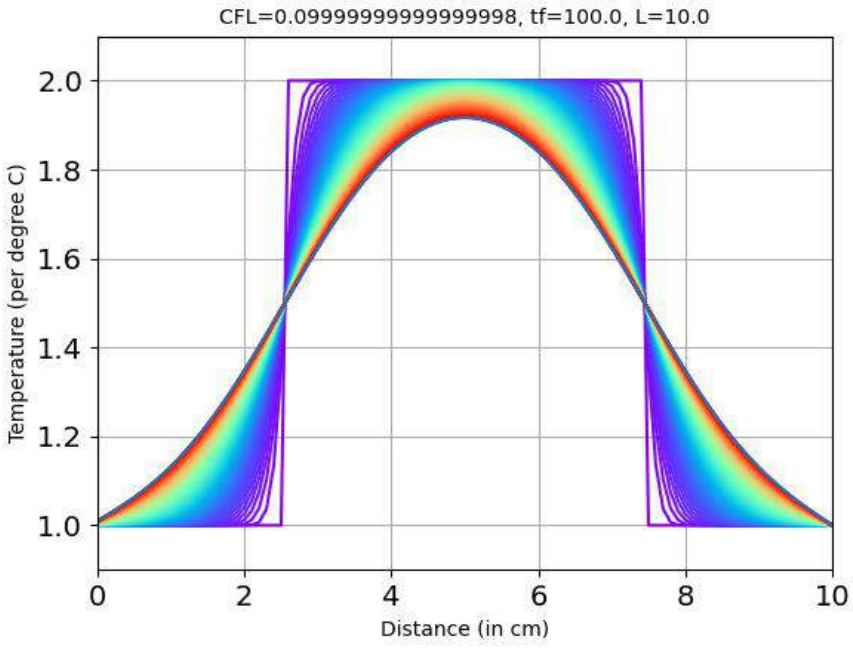
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm

tf = 100.0; tn = 1001; dt = tf/(tn-1);
L  = 10.0; xn = 101;  dx = L/(xn-1);
D = 0.01; cfl = D*dt/dx**2.0;

x = np.linspace(0, L, xn);
```

```
T = np.zeros((xn, tn))
T[0,:] = T[xn-1,:] = 1
for i in range(1,xn-1):
    if(i > (xn-1)/4 and i < 3*(xn-1)/4):
        T[i,0] = 2
    else:
        T[i,0] = 1

for t in range(0,tn-1):
    for i in range(0,xn-1):
        T[i,t+1] = T[i,t] + cfl*(T[i+1,t]-2.0*T[i,t]+T[i-1,t])
plt.figure()
colour=iter(cm.rainbow(np.linspace(0,10,tn)))
for i in range(0,tn,10):
    c = next(colour)
    plt.plot(x, T[:,i], c=c)
plt.plot(x, T[:,i])
plt.xlabel('Distance (in cm)', size=10)
plt.xticks(size = 14)
plt.ylabel('Temperature (per degree C)', size=10)
plt.yticks(size = 14)
plt.title('CFL='+str(cfl)+', tf='+str(tf)+', L='+str(L), size=10)
plt.xlim([0, L])
plt.ylim([0.9, 2.1])
plt.grid()
plt.show()
```



(c). 1D wave equation: $\frac{\partial^2 u}{\partial t^2} = \lambda \frac{\partial^2 u}{\partial x^2}$

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm

tf  = 3.5;  tn = 201; dt = tf/(tn-1)
L   = 20.0; xn = 201; dx = L/(xn-1); c = 1.0;
cfl = (c*dt/dx)**2
x = np.linspace(0, L, xn)
u = np.zeros((xn, tn))
u[:,1] = u[:,0] = np.exp(-((x-L/2)**2)/0.25)
for t in range(1,tn-1):
    for i in range(1,xn-1):
        u[i,t+1] = 2*(1-cfl)*u[i,t]-u[i,t-1]+cfl*(u[i-1,t]+u[i+1,t])

fig = plt.figure()
colour=iter(cm.rainbow(np.linspace(0,10,tn)))
for i in range(0,tn,10):
    c = next(colour)
    plt.plot(x, u[:,i], c=c)
plt.xlabel('Distance (in cm)', size=12)
plt.xticks(size = 12)
plt.ylabel('U(x,t)', size=12)
plt.yticks(size = 12)
plt.title('CFL='+str(cfl)+', tf='+str(tf)+', L='+str(L), size=12)
plt.xlim([0, L])
plt.ylim([-0.05, 1.05])
plt.grid()
plt.show()
```