Advanced Numerical Techniques Lab Report

Submitted By:Ramit Pahwa
14MA20029

Index

S.No	***Date***	***Assignment No.***	***Description***	***Page NO.***	***Signature***
1.	**25/1/17**	**1**	**Implementation of Thomas Algorithm for solving Second Order Boundary Value Problem**	**2-19**	

Assignment No.1

Using Thomas Algorithm Solve the Following Second Order Boundary Value Problem:

1. $x^2 y'' + xy' = 1$

   $y(1) = 0$, $y(1.4) = 0.0566$

   Solve the above problem for $h = 0.1, 0.5, 0.01$.

2. $y'' - 2xy' - 2y = -4x$

   Subject to $y(0) - y'(0) = 0$

   $2y(1) - y'(1) = 1$

   Solve the above problem for $h \leq 0.1$.

Algorithms Involved:

- ***Tridiagonal matrix algorithm***:In numerical linear algebra, the tridiagonal matrix algorithm, also known as the **Thomas algorithm** (named after Llewellyn Thomas), is a simplified form of Gaussian elimination that can be used to solve tridiagonal systems of equations.

  A tridiagonal system for n unknowns may be written as

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i,$$

where $a_i = 0$ & $c_n = 0$.

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}.$$

For such systems, the solution can be obtained $O(n)$ in operations instead of $O(n^3)$ required by **Gaussian elimination**

The forward sweep consists of modifying the coefficients as follows, denoting the new coefficients with primes:

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & ; \quad i = 1 \\ \dfrac{c_i}{b_i - a_i c'_{i-1}} & ; \quad i = 2, 3, \dots, n-1 \end{cases}$$

$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & ; \quad i = 1 \\ \dfrac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} & ; \quad i = 2, 3, \dots, n. \end{cases}$$

The solution is obtained by back substitution,

$$x_n = d'_n$$
$$x_i = d'_i - c'_i x_{i+1} \qquad ; \; i = n-1, n-2, \dots, 1.$$

- ***Forward and Backward Difference Formula Used for Discretization***

$$\begin{aligned} \text{Forward difference} \quad \frac{\delta f}{\delta x} &= \frac{f_{i+1} - f_i}{\Delta x}, \\ \text{Backward difference} \quad \frac{\delta f}{\delta x} &= \frac{f_i - f_{i-1}}{\Delta x}, \\ \text{Central difference} \quad \frac{\delta f}{\delta x} &= \frac{f_{i+1} - f_{i-1}}{2\Delta x}. \end{aligned}$$

$$f_i'' = \frac{f_{i-1} - 2f_i + f_{i+1}}{\Delta x^2} + \mathcal{O}\left(\Delta x^2\right),$$

$$\frac{df}{dx} = \frac{f_{i-2} - 4f_{i-1} + 3f_i}{2\Delta x} + \mathcal{O}\left(\Delta x^2\right),$$

- **Modules Used**
  - Numpy
    - For numerical arrays and interpolation
  - Matplotlib
    - For plotting graphs

Code has been written in python

Code For Problem :1

```
import os
import matplotlib.pyplot as plt
import numpy as np
from tabulate import tabulate

def A(x):
    return (1.0 / x)

def B(x):
    return 0.0

def C(x):
    return (1 / x **2)

def thomasAlgo(
    a,
    b,
    c,
    d,
    ):
    n = len(d)
    c1 = [0 for i in range(0, n)]
    d1 = [0 for i in range(0, n)]
    y = [0 for i in range(0, n)]

    c1[0] = c[0] / b[0]
    d1[0] = d[0] / b[0]

    for i in range(1, n, 1):
        c1[i] = c[i] / (b[i] - a[i] * c1[i - 1])
        d1[i] = (d[i] - a[i] * d1[i - 1]) / (b[i] - a[i] * c1[i - 1])

    y[n - 1] = d1[n - 1]
    for i in range(n - 2, -1, -1):
        y[i] = d1[i] - c1[i] * y[i + 1]

    return y

def solveBVP(
    a_intial,
    b_intial,
    h,
    y_a,
    y_b,
    ):
    n = int((b_intial - a_intial) / h)+1
    a = [0 for i in range(1, n)]
    b = [0 for i in range(1, n)]
```

```
    c = [0 for i in range(1, n)]
    d = [0 for i in range(1, n)]
    for i in range(1, n):
        x = a_intial + i * h
        a[i-1] = (1.0 / (h ** 2))- (A(x) / (2.0 * h))
        b[i-1] = (-2.0 / (h ** 2)) + B(x)
        c[i-1] = (1.0 / (h ** 2)) + (A(x) / (2.0 * h))
        if i == 1:
            d[i-1] = C(x) - a[i-1] * y_a
        elif i == n - 1:
            d[i-1] = C(x) - c[i-1] * y_b
        else:
            d[i-1] = C(x)

    return [y_a] + thomasAlgo(a, b, c, d) + [y_b]

def main():
    a_intial = 1
    b_intial = 1.4
    stepsize = [0.1, 0.05, 0.01]
    y_a = 0
    y_b = 0.0566
    n1 =int( (b_intial- a_intial) / stepsize[0])+1
    n2=int((b_intial - a_intial) / stepsize[1])+1
    n3=int((b_intial - a_intial) / stepsize[2])+1
    y_1 = [0 for i in range(n1 + 1)]
    y_2 = [0 for i in range(n2 + 1)]
    y_3 = [0 for i in range(n3 + 1)]
    x_1= np.linspace(a_intial, b_intial, int((b_intial - a_intial) /
stepsize[0]+1) + 1)
    x_2= np.linspace(a_intial, b_intial, int((b_intial - a_intial) /
stepsize[1]+1) + 1)
    x_3= np.linspace(a_intial, b_intial, int((b_intial - a_intial) /
stepsize[2]+1) + 1)
    y_1 = solveBVP(a_intial, b_intial, stepsize[0], y_a, y_b)
    y_2 = solveBVP(a_intial, b_intial, stepsize[1], y_a, y_b)
    y_3 = solveBVP(a_intial, b_intial, stepsize[2], y_a, y_b)
    cwd=os.getcwd()
    f=open(str(cwd)+"/Result-h="+str(stepsize[0])+".txt",'w+')
    f.write("\t\tResult\n\n")
    f.write("\tValue of X\t\tValue of Y\n\n")
    for i in range(len(x_1)):
        f.write("\t"+str(x_1[i])+"\t\t"+str(y_1[i])+"\n")
    f=open(str(cwd)+"/Result-h="+str(stepsize[1])+".txt",'w+')
    f.write("\t\tResult\n\n")
    f.write("\tValue of X\t\tValue of Y\n\n")
    for i in range(len(x_2)):
        f.write("\t"+str(x_2[i])+"\t\t"+str(y_2[i])+"\n")
    f=open(str(cwd)+"/Result-h="+str(stepsize[2])+".txt",'w+')
    f.write("\t\tResult\n\n")
    f.write("\tValue of X\t\tValue of Y\n\n")
```

```
    for i in range(len(x_3)):
        f.write("\t"+str(x_3[i])+"\t\t"+str(y_3[i])+"\n")

    plt.ylabel("Y")
    plt.xlabel ("X")

p1,p2,p3=plt.plot(x_3,np.interp(x_3,x_1,y_1),'r--',x_3,np.interp(x_3,x_2,y_2),'gs',x_3,y_3,'b')
    plt.legend([p1, p2, p3], ["h = 0.1", "h =0.05", "h = 0.01"], loc =4)
    # p11=plt.plot(x_1,y_1,'r')
    # plt.legend(p11,["h=0.1",],loc=4)
    plt.show()

main()
```

Output

Result for h=0.1

Value of X	***Value of Y***
1.0	*0*
1.1	*0.00457418107894*
1.2	*0.0166557456214*
1.3	*0.0344374516671*
1.4	*0.0566*

Result for h=0.01

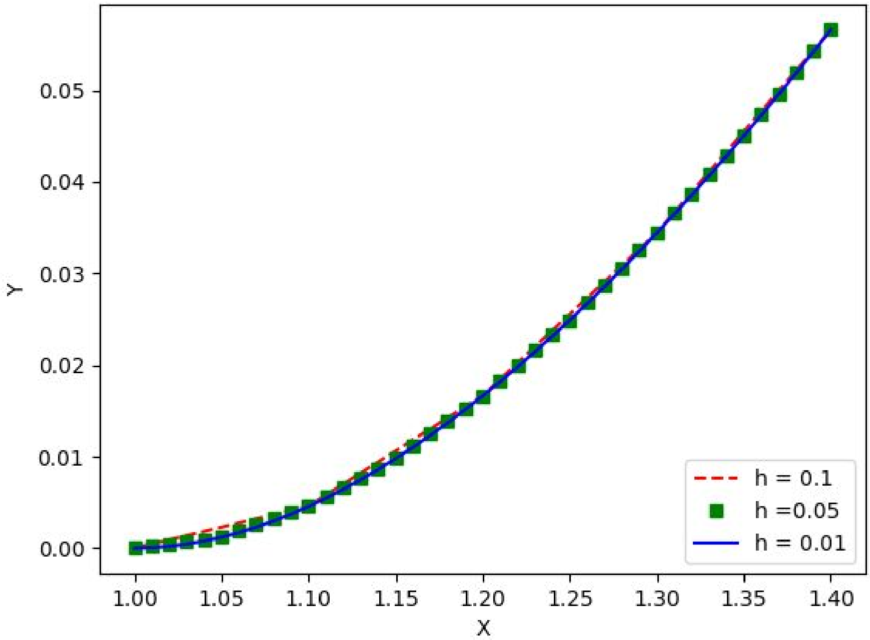
Value of X	***Value of Y***
1.0	*0*
1.01	*4.93560595464e-05*
1.02	*0.000195772552979*
1.03	*0.000436408608121*
1.04	*0.000768523907794*
1.05	*0.00118947432054*
1.06	*0.00169670775803*
1.07	*0.00228776024553*
1.08	*0.00296025219262*
1.09	*0.00371188485226*
1.1	*0.00454043695727*
1.11	*0.00544376152368*
1.12	*0.00641978281155*
1.13	*0.00746649343401*
1.14	*0.00858195160624*
1.15	*0.00976427852653*
1.16	*0.0110116558819*
1.17	*0.0123223234716*
1.18	*0.0136945769418*

1.19	*0.0151267656256*
1.2	*0.0166172904822*
1.21	*0.0181646021311*
1.22	*0.0197671989748*
1.23	*0.021423625406*
1.24	*0.0231324700952*
1.25	*0.0248923643542*
1.26	*0.0267019805714*
1.27	*0.0285600307151*
1.28	*0.0304652649029*
1.29	*0.0324164700307*
1.3	*0.0344124684625*
1.31	*0.0364521167737*
1.32	*0.0385343045486*
1.33	*0.0406579532277*
1.34	*0.0428220150027*
1.35	*0.0450254717576*
1.36	*0.0472673340523*
1.37	*0.0495466401484*
1.38	*0.0518624550734*
1.39	*0.0542138697233*
1.4	*0.0566*

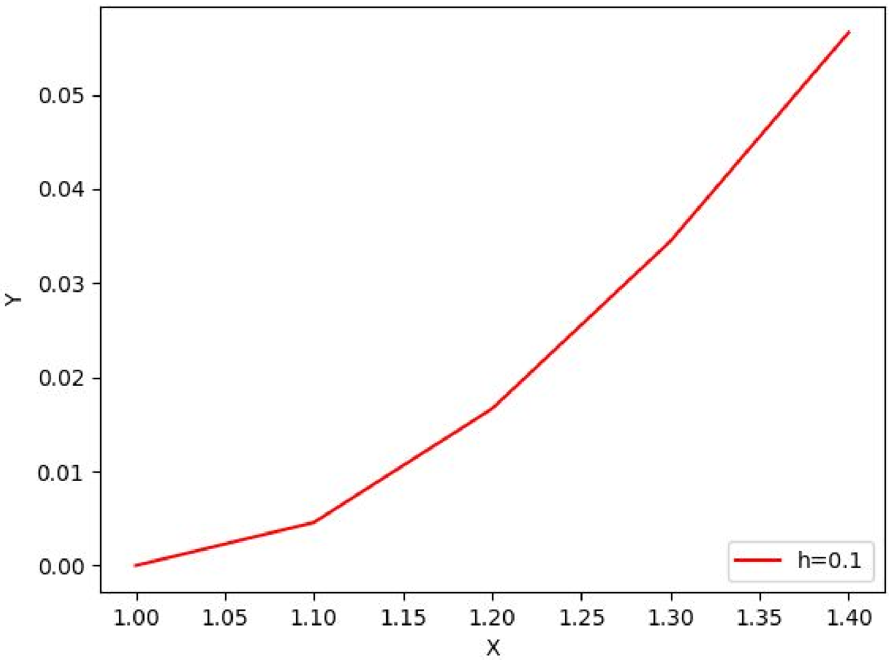
Result for h=0.05

Value of X	***Value of Y***
1.0	*0*
1.05	*0.0011946931352*
1.1	*0.00454865857198*
1.15	*0.00977376089843*
1.2	*0.0166266572054*
1.25	*0.0249005237312*
1.3	*0.034418552354*
1.35	*0.0450287888953*
1.4	*0.0566*

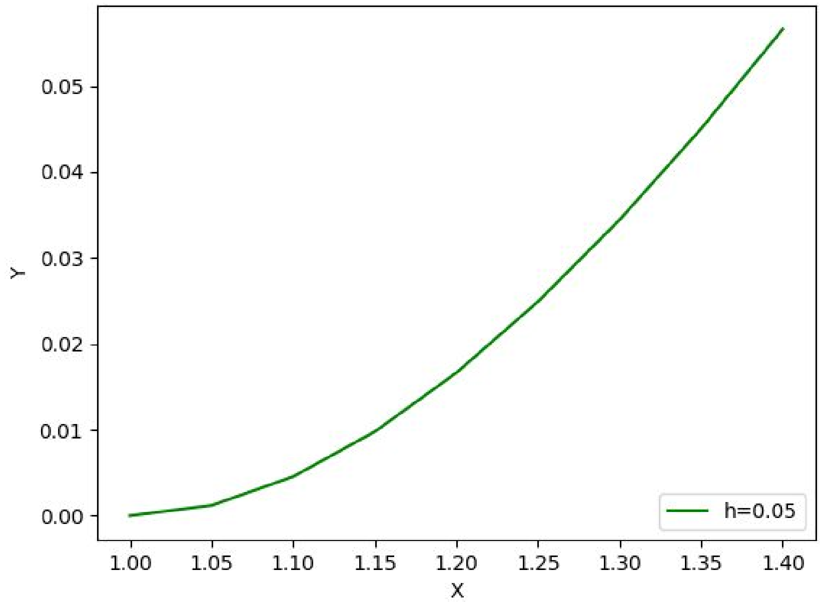
Plot



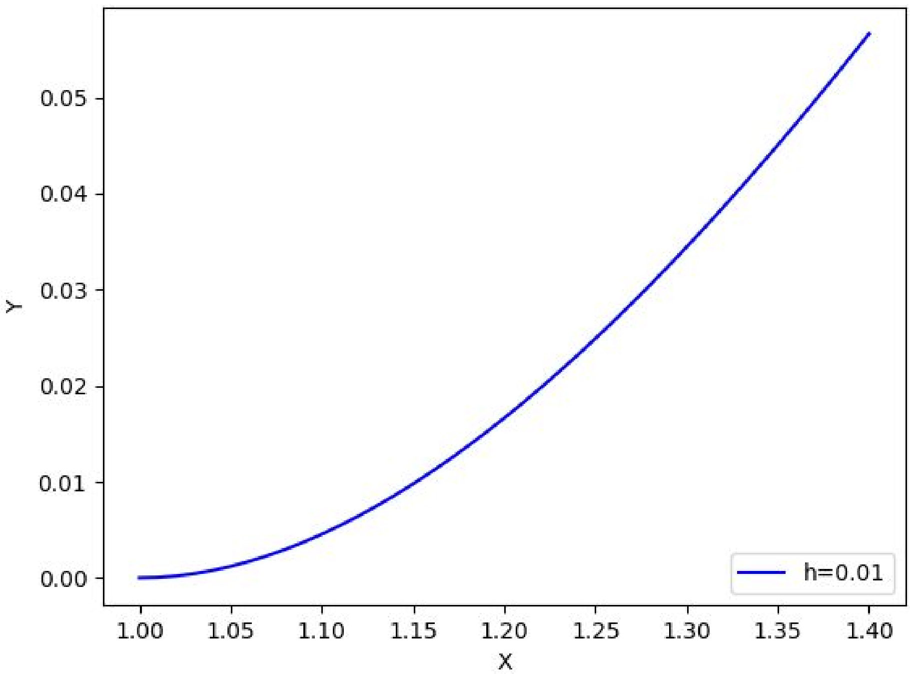
0.05
0.04
0.03
0.02
0.01
0.00
Y
1.00
1.05
1.10
1.15
1.20
1.25
1.30
1.35
1.40
X
h = 0.1
h =0.05
h = 0.01



0.05
0.04
0.03
0.02
0.01
0.00
Y
1.00
1.05
1.10
1.15
1.20
1.25
1.30
1.35
1.40
X
h=0.1



0.05
0.04
0.03
0.02
0.01
0.00
Y
1.00
1.05
1.10
1.15
1.20
1.25
1.30
1.35
1.40
X
h=0.05



0.05
0.04
0.03
0.02
0.01
0.00
Y
1.00
1.05
1.10
1.15
1.20
1.25
1.30
1.35
1.40
X
h=0.01

Code for Problem 2:

```
import os
import matplotlib.pyplot as plt
import numpy as np
from tabulate import tabulate

def A(x):
    return (-2.0*x)

def B(x):
    return (-2.0)

def C(x):
    return (-4.0*x)

def thomasAlgo(
    a,
    b,
    c,
    d,
    ):
    n = len(d)
    c1 = [0 for i in range(0, n)]
    d1 = [0 for i in range(0, n)]
    y = [0 for i in range(0, n)]

    c1[0] = c[0] / b[0]
    d1[0] = d[0] / b[0]

    for i in range(1, n, 1):
        c1[i] = c[i] / (b[i] - a[i] * c1[i - 1])
        d1[i] = (d[i] - a[i] * d1[i - 1]) / (b[i] - a[i] * c1[i - 1])

    y[n - 1] = d1[n - 1]
    for i in range(n - 2, -1, -1):
        y[i] = d1[i] - c1[i] * y[i + 1]

    return y

def solveBVP(
    alpha,
    beta,
    gamma,
    h,
    a_intial,
    b_intial
    ):
    n = int((b_intial - a_intial) / (1.0*h))
    a = [0 for i in range(1, n)]
    b = [0 for i in range(1, n)]
```

```
    c = [0 for i in range(1, n)]
    d = [0 for i in range(1, n)]
    for i in range(1, n):
        x = a_intial + i * h
        a[i-1] = (1.0 / (h ** 2))- (A(x) / (2.0 * h))
        b[i-1] = (-2.0 / (h ** 2)) + B(x)
        c[i-1] = (1.0 / (h ** 2)) + (A(x) / (2.0 * h))
        d[i-1] = C(x)
        if i == 1:
            denom0 = alpha[0] - (1.5 * beta[0] / h);
            b[i-1] += a[i-1] * (-2 * beta[0] / h) / denom0;
            c[i-1] += a[i-1] * (0.5 * beta[0] /h) / denom0;
            d[i-1] -= a[i-1] * (gamma[0] / denom0);


    denom1 = alpha[1] + (1.5 * beta[1] / h);
    b[i-1] += c[i-1] * (2 * beta[1] / h) / denom1;
    a[i-1] += c[i-1] * (-0.5 * beta[1] /h) / denom1;
    d[i-1] -= c[i-1] * (gamma[1]/ denom1);
    y=thomasAlgo(a, b, c, d)
    y_a=((-2 * beta[0]/ h)*y[0] + (0.5 * beta[0] /h)*y[1]
+gamma[0])/denom0;
    y_b= ((2 * beta[1] / h)*y[-1] +(-0.5 * beta[1] /h)*y[-2] +
gamma[1])/denom1;

    return [y_a] + y + [y_b]

def createFile(input_x,output_y,h):
    cwd=os.getcwd()
    f=open(str(cwd)+"/Result-h="+str(h)+".txt",'w+')
    f.write("\t\tResult for h="+str(h)+"\n\n")
    f.write("\tValue of X\t\tValue of Y\n\n")
    for i in range(len(input_x)):
        f.write("\t"+str(input_x[i])+"\t\t"+str(output_y[i])+"\n")

def plotGraph(input_x,output_y,h):
    plt.ylabel("Y")
    plt.xlabel ("X")
    p=plt.plot(input_x,output_y,'r')
    plt.legend(p,["h="+str(h),],loc=4)
    plt.show()

def main():
    a_intial = 0
    b_intial = 1
    stepsize = [0.1, 0.05, 0.005]
    alpha= [1.0,2.0]
    beta=[-1.0,-1.0]
    gamma=[0.0,1.0]
    n1 =int( (b_intial- a_intial) / stepsize[0])+1
    n2=int((b_intial - a_intial) / stepsize[1])+1
```

```
    n3=int((b_intial - a_intial) / stepsize[2])+1
    y_1 = [0 for i in range(n1 + 1)]
    y_2 = [0 for i in range(n2 + 1)]
    y_3 = [0 for i in range(n3 + 1)]
    x_1= np.linspace(a_intial, b_intial, int((b_intial - a_intial) /
(1.0*stepsize[0])+1))
    x_2= np.linspace(a_intial, b_intial, int((b_intial - a_intial) /
(1.0*stepsize[1])+1))
    x_3= np.linspace(a_intial, b_intial, int((b_intial - a_intial) /
(1.0*stepsize[2])+1))
    y_1 = solveBVP(alpha, beta  ,gamma, stepsize[0], a_intial, b_intial)
    y_2 = solveBVP(alpha, beta,gamma, stepsize[1], a_intial, b_intial)
    y_3 = solveBVP(alpha, beta,gamma, stepsize[2], a_intial, b_intial)
    createFile(x_1,y_1,stepsize[0])
    createFile(x_2,y_2,stepsize[1])
    createFile(x_3,y_3,stepsize[2])



p1,p2,p3=plt.plot(x_3,np.interp(x_3,x_1,y_1),'r',x_3,np.interp(x_3,x_2,y_2)
,'g',x_3,y_3,'b')
    plt.legend([p1, p2, p3], ["h = 0.25", "h =0.1", "h = 0.001"], loc =4)
    plt.show()

main()
```

Output

Result for h=0.1

Value of X	***Value of Y***
0.0	*1.19476896184*
0.1	*1.32696464931*
0.2	*1.48459791933*
0.3	*1.6707998518*
0.4	*1.89059777702*
0.5	*2.15143298304*
0.6	*2.46396522092*
0.7	*2.84328806833*
0.8	*3.31075173295*
0.9	*3.89670368126*
1.0	*4.64463961234*

Result for h=0.05

Value of X	***Value of Y***
0.0	*1.04462037002*

0.05	*1.09949391347*
0.1	*1.15965250684*
0.15	*1.22523808359*
0.2	*1.29647602716*
0.25	*1.3736807778*
0.3	*1.45726377924*
0.35	*1.54774404864*
0.4	*1.64576175282*
0.45	*1.7520952906*
0.5	*1.86768252214*
0.55	*1.99364695798*
0.6	*2.13132993239*
0.65	*2.2823300492*
0.7	*2.44855151778*
0.75	*2.63226341163*
0.8	*2.83617240585*
0.85	*3.06351221419*
0.9	*3.31815379255*
0.95	*3.60474145573*
1.0	*3.92886143942*

Result for h=0.005

Value of X	***Value of Y***
0.0	*1.0004693245*
0.005	*1.00549668485*
0.01	*1.01057407265*
0.015	*1.0157014994*
0.02	*1.02087898409*
0.025	*1.02610655323*
0.03	*1.03138424086*
0.035	*1.03671208853*
0.04	*1.04209014534*
0.045	*1.04751846794*
0.05	*1.05299712052*
0.055	*1.05852617489*
0.06	*1.06410571043*
0.065	*1.06973581414*
0.07	*1.07541658068*
0.075	*1.08114811235*
0.08	*1.08693051916*
0.085	*1.09276391881*
0.09	*1.09864843678*
0.095	*1.1045842063*
0.1	*1.11057136841*
0.105	*1.11661007203*
0.11	*1.12270047394*
0.115	*1.12884273883*
0.12	*1.13503703939*

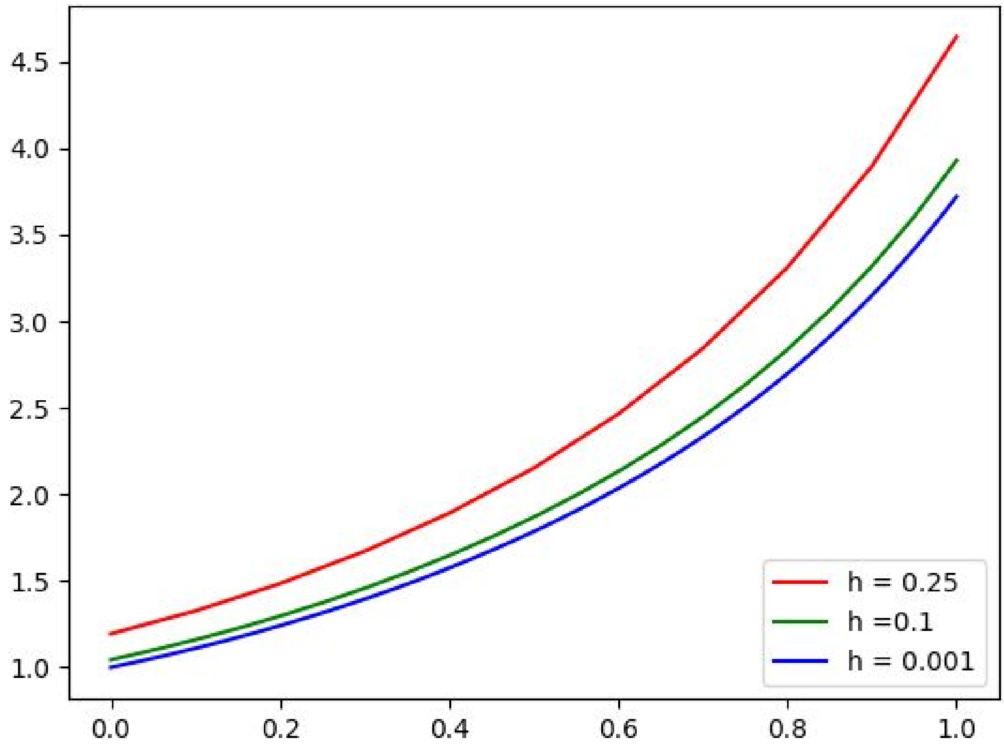
0.125	1.14128355629
0.13	1.14758247827
0.135	1.15393400216
0.14	1.16033833295
0.145	1.16679568383
0.15	1.17330627626
0.155	1.17987034
0.16	1.18648811317
0.165	1.19315984236
0.17	1.19988578261
0.175	1.20666619755
0.18	1.21350135944
0.185	1.2203915492
0.19	1.22733705657
0.195	1.23433818009
0.2	1.24139522723
0.205	1.24850851447
0.21	1.25567836735
0.215	1.26290512059
0.22	1.27018911814
0.225	1.2775307133
0.23	1.28493026878
0.235	1.29238815685
0.24	1.29990475934
0.245	1.30748046785
0.25	1.31511568376
0.255	1.3228108184
0.26	1.3305662931
0.265	1.33838253936
0.27	1.3462599989
0.275	1.35419912383
0.28	1.36220037674
0.285	1.37026423082
0.29	1.37839116998
0.295	1.38658168902
0.3	1.39483629371
0.305	1.40315550092
0.31	1.41153983882
0.315	1.41998984694
0.32	1.42850607638
0.325	1.43708908992
0.33	1.44573946216
0.335	1.45445777971
0.34	1.46324464133
0.345	1.47210065807
0.35	1.48102645347
0.355	1.4900226637
0.36	1.49908993775
0.365	1.50822893759
0.37	1.51744033836
0.375	1.52672482854

0.38	1.53608311016
0.385	1.54551589897
0.39	1.55502392465
0.395	1.56460793098
0.4	1.57426867609
0.405	1.58400693264
0.41	1.59382348803
0.415	1.60371914463
0.42	1.61369472
0.425	1.62375104709
0.43	1.63388897453
0.435	1.6441093668
0.44	1.65441310453
0.445	1.66480108469
0.45	1.67527422089
0.455	1.6858334436
0.46	1.69647970046
0.465	1.70721395648
0.47	1.71803719437
0.475	1.72895041479
0.48	1.73995463667
0.485	1.75105089743
0.49	1.76224025336
0.495	1.77352377986
0.5	1.78490257179
0.505	1.79637774376
0.51	1.80795043046
0.515	1.81962178699
0.52	1.8313929892
0.525	1.84326523402
0.53	1.85523973983
0.535	1.86731774678
0.54	1.87950051719
0.545	1.89178933592
0.55	1.90418551073
0.555	1.91669037266
0.56	1.92930527647
0.565	1.94203160097
0.57	1.95487074952
0.575	1.96782415037
0.58	1.98089325713
0.585	1.99407954922
0.59	2.00738453226
0.595	2.02080973859
0.6	2.03435672768
0.605	2.04802708666
0.61	2.06182243075
0.615	2.07574440377
0.62	2.08979467867
0.625	2.10397495803
0.63	2.11828697456

0.635	2.13273249167
0.64	2.147313304
0.645	2.16203123798
0.65	2.17688815241
0.655	2.19188593903
0.66	2.2070265231
0.665	2.22231186406
0.67	2.23774395607
0.675	2.25332482872
0.68	2.2690565476
0.685	2.28494121502
0.69	2.30098097065
0.695	2.31717799222
0.7	2.33353449618
0.705	2.35005273848
0.71	2.36673501525
0.715	2.38358366355
0.72	2.40060106215
0.725	2.41778963229
0.73	2.43515183848
0.735	2.45269018929
0.74	2.47040723821
0.745	2.48830558445
0.75	2.50638787384
0.755	2.52465679968
0.76	2.54311510365
0.765	2.56176557673
0.77	2.58061106012
0.775	2.59965444621
0.78	2.61889867955
0.785	2.63834675784
0.79	2.65800173295
0.795	2.67786671196
0.8	2.69794485824
0.805	2.71823939248
0.81	2.73875359385
0.815	2.75949080111
0.82	2.78045441374
0.825	2.80164789318
0.83	2.82307476395
0.835	2.84473861496
0.84	2.8666431007
0.845	2.88879194258
0.85	2.91118893018
0.855	2.93383792264
0.86	2.95674285001
0.865	2.97990771462
0.87	3.00333659256
0.875	3.02703363508
0.88	3.05100307012
0.885	3.07524920382

0.89	3.09977642206
0.895	3.12458919207
0.9	3.14969206404
0.905	3.17508967277
0.91	3.2007867394
0.915	3.22678807308
0.92	3.2530985728
0.925	3.27972322917
0.93	3.30666712626
0.935	3.3339354435
0.94	3.36153345762
0.945	3.38946654458
0.95	3.41774018164
0.955	3.44635994938
0.96	3.47533153383
0.965	3.50466072859
0.97	3.53435343708
0.975	3.56441567472
0.98	3.5948535713
0.985	3.62567337329
0.99	3.65688144622
0.995	3.68848427721
1.0	3.72048847738

Plot



Assignment No.2

1. *Solve the following equation using Block Tridiagonal method*

$$y''' + 4y'' + y' - 6y = 1$$

Subject to . $y(0) = 0,$

$$y'(0) = 0,$$

$$y'(1) = 1$$

Algorithms is stated below .

The Thomas algorithm can be easily extended to solve a system of equations that involves a *block tridiagonal* matrix. Consider block tridiagonal system of the form

$$\begin{bmatrix} \mathbf{B}_1 & \mathbf{C}_1 & [0] & \ldots & \ldots & [0] \\ \mathbf{A}_2 & \mathbf{B}_2 & \mathbf{C}_2 & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & [0] \\ \ldots & \ldots & \ldots & \ldots & \mathbf{B}_{n-1} & \mathbf{C}_{n-1} \\ [0] & \ldots & \ldots & [0] & \mathbf{A}_n & \mathbf{B}_n \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ . \\ . \\ \mathbf{x}^{(n)} \end{bmatrix} = \begin{bmatrix} \mathbf{d}^{(1)} \\ \mathbf{d}^{(2)} \\ . \\ . \\ \mathbf{d}^{(n)} \end{bmatrix} \text{-------(41)}$$

where $\mathbf{A}_i$, $\mathbf{B}_i$ and $\mathbf{C}_i$ are matrices and $(\mathbf{x}^{(i)}, \mathbf{d}^{(i)})$ represent vectors of appropriate dimensions.

- **Step 1:Block Triangularization**

$$\Gamma_1 = [\mathbf{B}_1]^{-1}\mathbf{C}_1$$

$$\Gamma_k = [\mathbf{B}_k - \mathbf{A}_k\Gamma_{k-1}]^{-1}\mathbf{C}_k \ \textit{for} \ k = 2,3,\ldots(n-1) \text{ -------(42)}$$

$$\beta^{(1)} = [\mathbf{B}_1]^{-1}\mathbf{d}^{(1)} \text{ -------(43)}$$

$$\beta^{(k)} = [\mathbf{B}_k - \mathbf{A}_k\Gamma_{k-1}]^{-1}(\mathbf{d}^{(k)} - \mathbf{A}_k\beta^{(k-1)}) \ \textit{for} \ k = 2,3,\ldots n$$

- **Step 2: Backward sweep**

$$\mathbf{x}^{(n)} = \beta^{(n)} \text{ -------(44)}$$

$$\mathbf{x}^{(k)} = \beta^{(k)} - \Gamma_k\mathbf{x}^{(k+1)}$$
$$k = (n-1), (n-2), \ldots\ldots, 1 \text{ -------(45, 46)}$$

Code for Problem:

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

def blockAlgo(a, b, c, d):
    n = len(d)

    b_ = np.zeros(a.shape)
    c_ = np.zeros(a.shape)
    d_ = np.zeros((n,2,1))
    w = np.zeros((n,2,1))

    c_[0] = np.linalg.inv(b[0]).dot(c[0])
    d_[0] = np.linalg.inv(b[0]).dot(d[0])

    for i in range(1,n):
        b_[i] = b[i] - a[i].dot(c_[i-1])
        c_[i] = np.linalg.inv(b_[i]).dot(c[i])
        d_[i] = np.linalg.inv(b_[i]).dot((d[i] - a[i].dot(d_[i-1])))

    w[n-1] = np.copy(d_[n-1])
    for i in range(n-2, -1, -1):
        w[i] = d_[i] - c_[i].dot(w[i+1])

    return w
def solveBVP(a_intial, b_intial, c1, c2, c3, h):
    n = int ((b_intial-    a_intial) / (1.0*h))
    a = np.zeros((n-1,2,2))
    b = np.zeros((n-1,2,2))
    c = np.zeros((n-1,2,2))
    d = np.zeros((n-1,2,1))

    for i in range(n-1):
        # As i starts from 0, we define x = l + (i+1)*h
        x = a_intial+(i+1)*h
        a[i] = np.array(([[((1.0 / (h**2))-(2.0/h)), 0],[1, 2.0/h ]]))
        b[i] = np.array(([[1.0-(2.0 / (h**2)), -6.0],[1, -2.0 / h]]))
        c[i] = np.array(([[((1.0 / (h**2))+(2.0/h)), 0],[0, 0]]))
        d[i] = np.array(([[1],[0]]))
        if i == 0:
            d[i] =  d[i] - a[i].dot(np.array(([[c1], [c2]])))

    d[n-2] = d[n-2] -  c[n-2].dot(np.array(([[c3], [0]])))
    w = blockAlgo(a, b, c, d)
    w = np.vstack(([np.array(([[c1], [c2]]))], w))
    return np.vstack((w, [np.array(([[c3], [0]]))]))

def createFile(input_x,output_y,h):
    cwd=os.getcwd()
    f=open(str(cwd)+"/Result-h="+str(h)+".txt",'w+')
```

```
    f.write("\t\tResult for h="+str(h)+"\n\n")
    f.write("\tValue of X\t\tValue of Y\n\n")
    for i in range(len(input_x)):
        f.write("\t"+str(input_x[i])+"\t\t"+str(output_y[i])+"\n")

def plotGraph(input_x,output_y,h):
    plt.ylabel("Y")
    plt.xlabel ("X")
    p=plt.plot(input_x,output_y,'r')
    plt.legend(p,["h="+str(h),],loc=4)
    plt.savefig("plot-h="+str(h)+".png")
    plt.show()

def main():
       a_intial=0
       b_intial=1
       c1=0
       c2=0
       c3=1
       step_sizes=[0.1,0.5,0.002]
       n1=int((b_intial - a_intial) / (1.0*step_sizes[0])+1)
       n2=int((b_intial - a_intial) / (1.0*step_sizes[1])+1)
       n3=int((b_intial - a_intial) / (1.0*step_sizes[2])+1)
       x_1= np.linspace(a_intial, b_intial, int((b_intial - a_intial) /
(1.0*step_sizes[0])+1))
       x_2= np.linspace(a_intial, b_intial, int((b_intial - a_intial) /
(1.0*step_sizes[1])+1))
       x_3= np.linspace(a_intial, b_intial, int((b_intial - a_intial) /
(1.0*step_sizes[2])+1))
       w_1=solveBVP(a_intial,b_intial,c1,c2,c3,step_sizes[0])
       w_2=solveBVP(a_intial,b_intial,c1,c2,c3,step_sizes[1])
       w_3=solveBVP(a_intial,b_intial,c1,c2,c3,step_sizes[2])
       y_1 = w_1[[range(w_1.shape[0])],[1],[0]]
       y_2 = w_2[[range(w_2.shape[0])],[1],[0]]
       y_3 = w_3[[range(w_3.shape[0])],[1],[0]]
       y_1=y_1[0]
       y_2=y_2[0]
       y_3=y_3[0]
       z_1 = w_1[[range(w_1.shape[0])],[0],[0]]
       z_2 = w_2[[range(w_2.shape[0])],[0],[0]]
       z_3 = w_3[[range(w_3.shape[0])],[0],[0]]
       z_1=z_1[0]
       z_2=z_2[0]
       z_3=z_3[0]

y_1[n1-1]=(step_sizes[0]/2.0)*(y_1[n1-2]*(2.0/step_sizes[0])+z_1[n1-1]+z_1[n
1-1])

y_2[n2-1]=(step_sizes[1]/2.0)*(y_2[n2-2]*(2.0/step_sizes[1])+z_2[n2-1]+z_2[n
2-1])

y_3[n3-1]=(step_sizes[2]/2.0)*(y_3[n3-2]*(2.0/step_sizes[2])+z_3[n3-1]+z_3[n
3-1])
       createFile(x_1,y_1,step_sizes[0])
```

```
        createFile(x_2,y_2,step_sizes[1])
        createFile(x_3,y_3,step_sizes[2])
        plotGraph(x_1,y_1,step_sizes[0])
        plotGraph(x_2,y_2,step_sizes[1])
        plotGraph(x_3,y_3,step_sizes[2])

p1,p2,p3=plt.plot(x_3,np.interp(x_3,x_1,y_1),'r.',x_3,np.interp(x_3,x_2,y_2)
,'g--',x_3,y_3,'b')
        plt.legend([p1, p2, p3], ["h = 0.1", "h =0.5", "h = 0.002"], loc =4)
        plt.savefig("allTogether.png")
        plt.show()

main()
```

Output

```
      Result for h=0.5

Value of X        Value of Y

0.0         0.0
0.5         0.205882352941
1.0         0.705882352941


      Result for h=0.1

Value of X        Value of Y

0.0         0.0
0.1         0.0122488896478
0.2         0.0452578101855
0.3         0.0928087770617
0.4         0.150836683241
0.5         0.216808096722
0.6         0.289277865241
0.7         0.367574503952
0.8         0.45157871506
0.9         0.54156915572
1.0         0.64156915572

      Result for h=0.002

Value of X        Value of Y

0.0         0.0
0.002       5.74938658445e-06
0.004       2.29556957878e-05
0.006       5.15355376525e-05
0.008       9.14061424647e-05
```

0.01	*0.000142485356683*
0.012	*0.000204691638895*
0.014	*0.000277944055796*
0.016	*0.000362162278199*
0.018	*0.000457266577066*
0.02	*0.000563177819567*
0.022	*0.000679817465159*
0.024	*0.0008071075617*
0.026	*0.000944970741578*
0.028	*0.00109333021787*
0.03	*0.00125210978053*
0.032	*0.00142123379259*
0.034	*0.00160062718638*
0.036	*0.00179021545981*
0.038	*0.00198992467263*
0.04	*0.00219968144274*
0.042	*0.00241941294252*
0.044	*0.00264904689517*
0.046	*0.00288851157108*
0.048	*0.00313773578428*
0.05	*0.00339664888876*
0.052	*0.00366518077501*
0.054	*0.00394326186645*
0.056	*0.00423082311589*
0.058	*0.00452779600208*
0.06	*0.00483411252626*
0.062	*0.00514970520864*
0.064	*0.00547450708507*
0.066	*0.00580845170357*
0.068	*0.00615147312099*
0.07	*0.00650350589965*
0.072	*0.00686448510397*
0.074	*0.00723434629723*
0.076	*0.00761302553819*
0.078	*0.0080004593779*
0.08	*0.0083965848564*
0.082	*0.00880133949949*
0.084	*0.00921466131555*
0.086	*0.00963648879236*
0.088	*0.0100667608939*
0.09	*0.0105054170572*
0.092	*0.0109523971892*
0.094	*0.0114076416638*
0.096	*0.0118710913185*
0.098	*0.0123426874516*
0.1	*0.0128223718191*
0.102	*0.0133100866313*
0.104	*0.0138057745504*
0.106	*0.0143093786872*
0.108	*0.0148208425979*
0.11	*0.0153401102818*

0.112	*0.0158671261777*
0.114	*0.0164018351612*
0.116	*0.0169441825421*
0.118	*0.0174941140611*
0.12	*0.0180515758872*
0.122	*0.0186165146148*
0.124	*0.019188877261*
0.126	*0.0197686112626*
0.128	*0.0203556644732*
0.13	*0.020949985161*
0.132	*0.0215515220055*
0.134	*0.0221602240948*
0.136	*0.0227760409235*
0.138	*0.0233989223891*
0.14	*0.0240288187901*
0.142	*0.0246656808229*
0.144	*0.0253094595792*
0.146	*0.0259601065437*
0.148	*0.026617573591*
0.15	*0.0272818129834*
0.152	*0.0279527773682*
0.154	*0.0286304197752*
0.156	*0.0293146936139*
0.158	*0.0300055526713*
0.16	*0.0307029511093*
0.162	*0.0314068434621*
0.164	*0.0321171846338*
0.166	*0.0328339298961*
0.168	*0.0335570348854*
0.17	*0.034286455601*
0.172	*0.0350221484021*
0.174	*0.0357640700058*
0.176	*0.0365121774846*
0.178	*0.037266428264*
0.18	*0.0380267801202*
0.182	*0.0387931911777*
0.184	*0.039565619907*
0.186	*0.0403440251224*
0.188	*0.0411283659797*
0.19	*0.0419186019735*
0.192	*0.0427146929358*
0.194	*0.0435165990329*
0.196	*0.0443242807637*
0.198	*0.0451376989573*
0.2	*0.0459568147708*
0.202	*0.0467815896869*
0.204	*0.0476119855125*
0.206	*0.0484479643754*
0.208	*0.0492894887232*
0.21	*0.0501365213205*
0.212	*0.050989025247*

0.214	*0.0518469638957*
0.216	*0.0527103009701*
0.218	*0.0535790004829*
0.22	*0.0544530267535*
0.222	*0.0553323444058*
0.224	*0.0562169183669*
0.226	*0.057106713864*
0.228	*0.0580016964235*
0.23	*0.0589018318683*
0.232	*0.0598070863159*
0.234	*0.0607174261766*
0.236	*0.0616328181516*
0.238	*0.0625532292308*
0.24	*0.0634786266911*
0.242	*0.0644089780945*
0.244	*0.0653442512858*
0.246	*0.0662844143912*
0.248	*0.0672294358163*
0.25	*0.068179284244*
0.252	*0.0691339286328*
0.254	*0.0700933382151*
0.256	*0.0710574824952*
0.258	*0.0720263312473*
0.26	*0.0729998545142*
0.262	*0.0739780226051*
0.264	*0.0749608060938*
0.266	*0.0759481758172*
0.268	*0.0769401028735*
0.27	*0.07793655862*
0.272	*0.0789375146721*
0.274	*0.0799429429009*
0.276	*0.080952815432*
0.278	*0.0819671046434*
0.28	*0.0829857831642*
0.282	*0.0840088238724*
0.284	*0.0850361998938*
0.286	*0.0860678846*
0.288	*0.0871038516068*
0.29	*0.0881440747725*
0.292	*0.0891885281967*
0.294	*0.0902371862179*
0.296	*0.0912900234128*
0.298	*0.0923470145941*
0.3	*0.0934081348088*
0.302	*0.0944733593374*
0.304	*0.0955426636914*
0.306	*0.0966160236126*
0.308	*0.0976934150707*
0.31	*0.0987748142628*
0.312	*0.0998601976107*
0.314	*0.10094954176*

0.316	0.10204282358
0.318	0.103140020159
0.32	0.104241108804
0.322	0.105346067044
0.324	0.106454872619
0.326	0.107567503487
0.328	0.10868393782
0.33	0.109804154
0.332	0.110928130622
0.334	0.112055846487
0.336	0.113187280608
0.338	0.1143224122
0.34	0.115461220686
0.342	0.116603685693
0.344	0.117749787048
0.346	0.118899504781
0.348	0.12005281912
0.35	0.121209710492
0.352	0.122370159522
0.354	0.123534147029
0.356	0.124701654026
0.358	0.125872661721
0.36	0.127047151513
0.362	0.128225104989
0.364	0.129406503929
0.366	0.130591330299
0.368	0.13177956625
0.37	0.132971194121
0.372	0.134166196435
0.374	0.135364555895
0.376	0.136566255388
0.378	0.137771277983
0.38	0.138979606923
0.382	0.140191225635
0.384	0.141406117719
0.386	0.142624266952
0.388	0.143845657284
0.39	0.145070272841
0.392	0.146298097918
0.394	0.147529116984
0.396	0.148763314675
0.398	0.150000675798
0.4	0.151241185326
0.402	0.152484828399
0.404	0.153731590323
0.406	0.154981456566
0.408	0.156234412762
0.41	0.157490444705
0.412	0.15874953835
0.414	0.160011679814
0.416	0.161276855368

0.418	0.162545051447
0.42	0.163816254637
0.422	0.165090451684
0.424	0.166367629485
0.426	0.167647775092
0.428	0.16893087571
0.43	0.170216918696
0.432	0.171505891555
0.434	0.172797781943
0.436	0.174092577665
0.438	0.175390266672
0.44	0.176690837062
0.442	0.17799427708
0.444	0.179300575113
0.446	0.180609719692
0.448	0.181921699493
0.45	0.18323650333
0.452	0.18455412016
0.454	0.18587453908
0.456	0.187197749325
0.458	0.188523740266
0.46	0.189852501415
0.462	0.191184022416
0.464	0.192518293051
0.466	0.193855303234
0.468	0.195195043014
0.47	0.196537502572
0.472	0.19788267222
0.474	0.199230542401
0.476	0.200581103687
0.478	0.20193434678
0.48	0.20329026251
0.482	0.204648841834
0.484	0.206010075834
0.486	0.20737395572
0.488	0.208740472824
0.49	0.210109618604
0.492	0.21148138464
0.494	0.212855762634
0.496	0.21423274441
0.498	0.215612321911
0.5	0.216994487202
0.502	0.218379232466
0.504	0.219766550003
0.506	0.221156432233
0.508	0.222548871689
0.51	0.223943861023
0.512	0.225341393
0.514	0.226741460501
0.516	0.228144056519
0.518	0.229549174161

0.52	0.230956806644
0.522	0.232366947298
0.524	0.233779589564
0.526	0.23519472699
0.528	0.236612353237
0.53	0.238032462072
0.532	0.239455047368
0.534	0.240880103108
0.536	0.242307623381
0.538	0.243737602378
0.54	0.245170034399
0.542	0.246604913845
0.544	0.248042235223
0.546	0.24948199314
0.548	0.250924182307
0.55	0.252368797537
0.552	0.25381583374
0.554	0.25526528593
0.556	0.256717149219
0.558	0.258171418816
0.56	0.259628090031
0.562	0.26108715827
0.564	0.262548619034
0.566	0.264012467924
0.568	0.265478700633
0.57	0.26694731295
0.572	0.268418300759
0.574	0.269891660037
0.576	0.271367386853
0.578	0.272845477371
0.58	0.274325927843
0.582	0.275808734617
0.584	0.277293894126
0.586	0.278781402898
0.588	0.280271257547
0.59	0.281763454777
0.592	0.283257991381
0.594	0.284754864239
0.596	0.286254070317
0.598	0.287755606669
0.6	0.289259470434
0.602	0.290765658836
0.604	0.292274169186
0.606	0.293784998878
0.608	0.295298145388
0.61	0.296813606277
0.612	0.298331379188
0.614	0.299851461847
0.616	0.301373852061
0.618	0.302898547717
0.62	0.304425546784

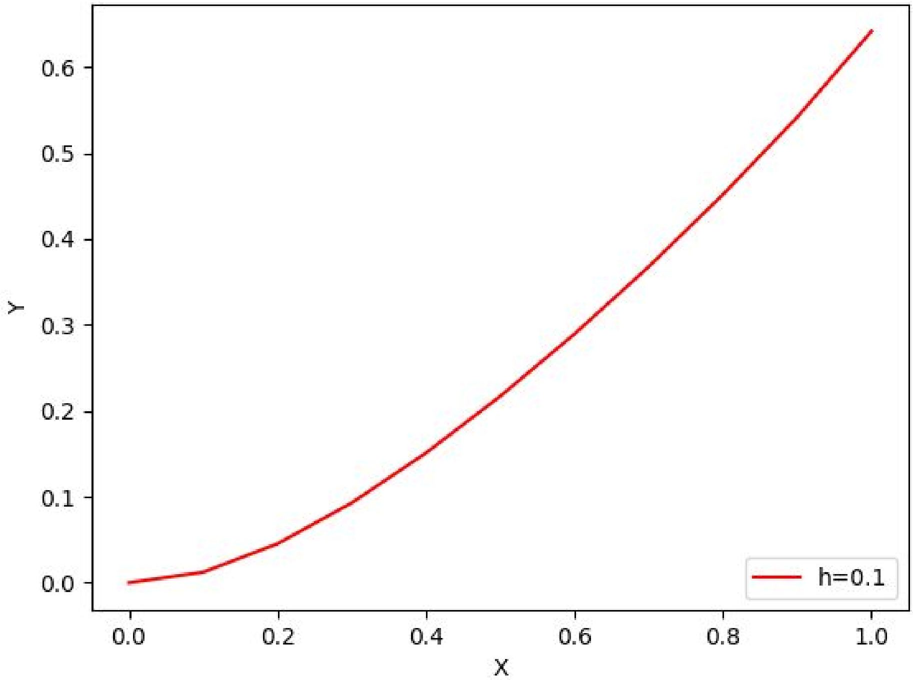
0.622	0.30595484731
0.624	0.307486447423
0.626	0.30902034533
0.628	0.310556539315
0.63	0.312095027742
0.632	0.31363580905
0.634	0.315178881758
0.636	0.316724244459
0.638	0.318271895822
0.64	0.319821834593
0.642	0.321374059592
0.644	0.322928569712
0.646	0.324485363923
0.648	0.326044441266
0.65	0.327605800856
0.652	0.32916944188
0.654	0.330735363598
0.656	0.332303565342
0.658	0.333874046513
0.66	0.335446806585
0.662	0.3370218451
0.664	0.338599161673
0.666	0.340178755985
0.668	0.341760627787
0.67	0.3433447769
0.672	0.344931203212
0.674	0.346519906677
0.676	0.348110887319
0.678	0.349704145228
0.68	0.351299680558
0.682	0.352897493531
0.684	0.354497584436
0.686	0.356099953624
0.688	0.357704601511
0.69	0.35931152858
0.692	0.360920735375
0.694	0.362532222504
0.696	0.36414599064
0.698	0.365762040516
0.7	0.36738037293
0.702	0.369000988738
0.704	0.370623888862
0.706	0.372249074282
0.708	0.37387654604
0.71	0.375506305237
0.712	0.377138353037
0.714	0.378772690659
0.716	0.380409319386
0.718	0.382048240557
0.72	0.38368945557
0.722	0.385332965881

0.724	*0.386978773004*
0.726	*0.388626878511*
0.728	*0.390277284031*
0.73	*0.391929991248*
0.732	*0.393585001904*
0.734	*0.395242317797*
0.736	*0.39690194078*
0.738	*0.398563872762*
0.74	*0.400228115706*
0.742	*0.401894671631*
0.744	*0.403563542609*
0.746	*0.405234730766*
0.748	*0.406908238284*
0.75	*0.408584067395*
0.752	*0.410262220386*
0.754	*0.411942699597*
0.756	*0.413625507419*
0.758	*0.415310646296*
0.76	*0.416998118723*
0.762	*0.418687927247*
0.764	*0.420380074467*
0.766	*0.422074563031*
0.768	*0.423771395639*
0.77	*0.42547057504*
0.772	*0.427172104033*
0.774	*0.428875985469*
0.776	*0.430582222246*
0.778	*0.43229081731*
0.78	*0.434001773659*
0.782	*0.435715094336*
0.784	*0.437430782436*
0.786	*0.439148841099*
0.788	*0.440869273512*
0.79	*0.442592082913*
0.792	*0.444317272583*
0.794	*0.446044845853*
0.796	*0.447774806098*
0.798	*0.449507156741*
0.8	*0.45124190125*
0.802	*0.45297904314*
0.804	*0.454718585968*
0.806	*0.456460533341*
0.808	*0.458204888908*
0.81	*0.459951656363*
0.812	*0.461700839444*
0.814	*0.463452441935*
0.816	*0.465206467662*
0.818	*0.466962920495*
0.82	*0.468721804349*
0.822	*0.470483123179*
0.824	*0.472246880987*

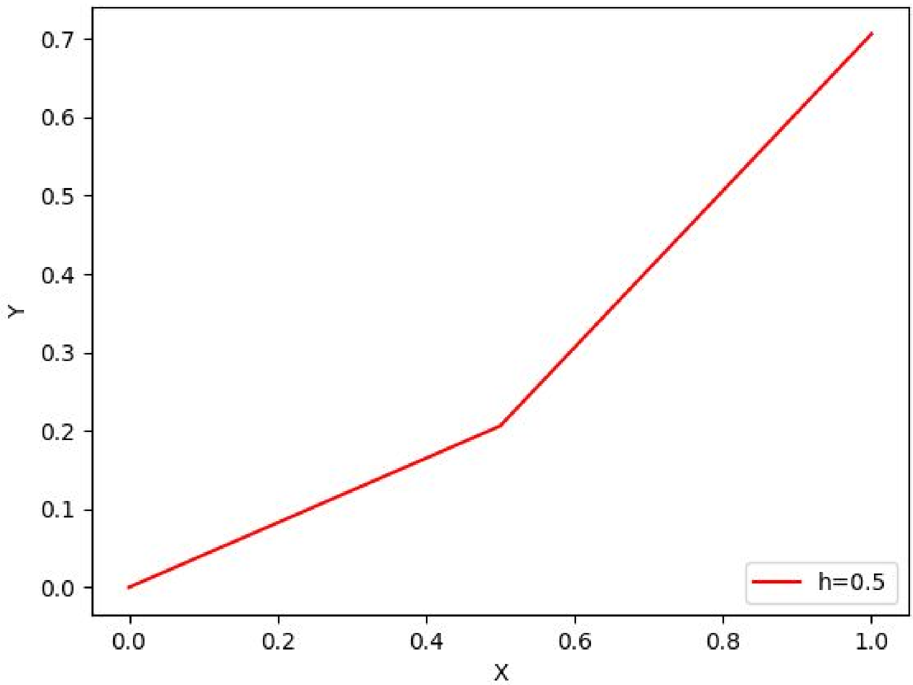
0.826	0.474013081814
0.828	0.475781729745
0.83	0.477552828908
0.832	0.479326383471
0.834	0.481102397646
0.836	0.482880875684
0.838	0.48466182188
0.84	0.486445240568
0.842	0.488231136124
0.844	0.490019512962
0.846	0.491810375541
0.848	0.493603728357
0.85	0.495399575945
0.852	0.497197922882
0.854	0.498998773785
0.856	0.500802133307
0.858	0.502608006143
0.86	0.504416397026
0.862	0.506227310727
0.864	0.508040752057
0.866	0.509856725863
0.868	0.511675237033
0.87	0.513496290489
0.872	0.515319891194
0.874	0.517146044147
0.876	0.518974754385
0.878	0.52080602698
0.88	0.522639867044
0.882	0.524476279723
0.884	0.5263152702
0.886	0.528156843697
0.888	0.530001005467
0.89	0.531847760804
0.892	0.533697115035
0.894	0.535549073522
0.896	0.537403641664
0.898	0.539260824895
0.9	0.541120628684
0.902	0.542983058533
0.904	0.544848119982
0.906	0.546715818602
0.908	0.548586160001
0.91	0.550459149819
0.912	0.552334793732
0.914	0.554213097449
0.916	0.556094066711
0.918	0.557977707296
0.92	0.559864025013
0.922	0.561753025703
0.924	0.563644715243
0.926	0.565539099541

0.928	0.567436184538
0.93	0.569335976207
0.932	0.571238480555
0.934	0.57314370362
0.936	0.575051651471
0.938	0.576962330212
0.94	0.578875745975
0.942	0.580791904928
0.944	0.582710813266
0.946	0.584632477218
0.948	0.586556903045
0.95	0.588484097035
0.952	0.590414065512
0.954	0.592346814828
0.956	0.594282351365
0.958	0.596220681537
0.96	0.598161811788
0.962	0.600105748592
0.964	0.602052498454
0.966	0.604002067907
0.968	0.605954463516
0.97	0.607909691874
0.972	0.609867759604
0.974	0.611828673361
0.976	0.613792439824
0.978	0.615759065707
0.98	0.617728557749
0.982	0.61970092272
0.984	0.621676167418
0.986	0.623654298669
0.988	0.625635323331
0.99	0.627619248285
0.992	0.629606080446
0.994	0.631595826753
0.996	0.633588494176
0.998	0.63558408971
1.0	0.63758408971

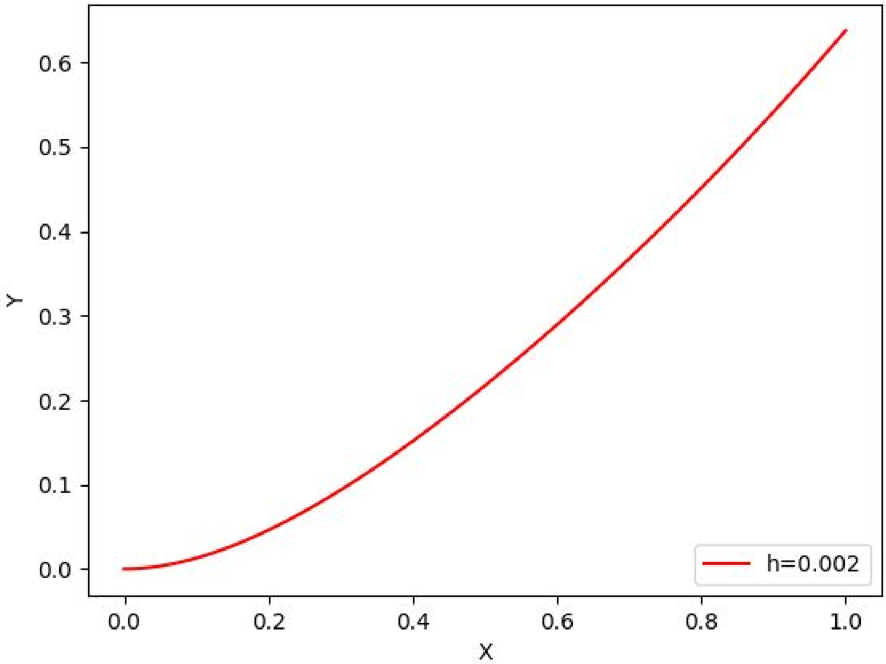
Plot

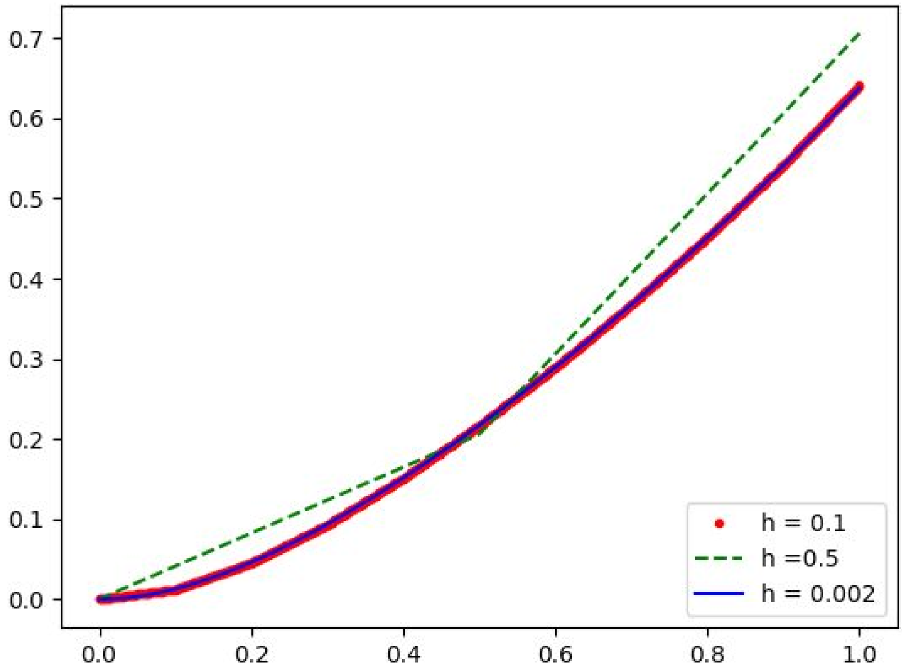


0.6
0.5
0.4
0.3
0.2
0.1
0.0
Y
0.0
0.2
0.4
0.6
0.8
1.0
X
h=0.1



0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Y
0.0
0.2
0.4
0.6
0.8
1.0
X
h=0.5


0.6
0.5
0.4
0.3
0.2
0.1
0.0
Y
h=0.002
0.0
0.2
0.4
0.6
0.8
1.0
X


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
h = 0.1
h =0.5
h = 0.002
0.0
0.2
0.4
0.6
0.8
1.0

Assignment No.3

1.Solve the following Nonlinear equations :

- $y'' = 2 + y$

  $y(0) = 0, y(1) = 0, h = 0.001$
- $y'' - (y')^2 - y^2 + y + 1 = 0$

  $y(0) = 0.5, y(\pi) = 0.5, n = 100$

*First **newton raphson** method used to convert tri-diagonal system of equation then **thomas algorithm** used to solve system for each iteration*

Code for Equation 1:

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

def intialGuess(x):
      return x*(1-x)

def thomasAlgo(
    a,
    b,
    c,
    d,
    ):
    n = len(d)
    c1 = [0 for i in range(0, n)]
    d1 = [0 for i in range(0, n)]
    y = [0 for i in range(0, n)]

    c1[0] = c[0] / b[0]
    d1[0] = d[0] / b[0]

    for i in range(1, n, 1):
        c1[i] = c[i] / (b[i] - a[i] * c1[i - 1])
        d1[i] = (d[i] - a[i] * d1[i - 1]) / (b[i] - a[i] * c1[i - 1])

    y[n - 1] = d1[n - 1]
    for i in range(n - 2, -1, -1):
        y[i] = d1[i] - c1[i] * y[i + 1]

    return y


def solver(a_intial,b_intial,y_a,y_b,h,k,y,n):
      A=np.array([1.0/h**2 for i in range(1,n)],dtype=np.float64)
      B=np.array([-2.0/h**2 -2.0*(y[i]) for i in
range(1,n)],dtype=np.float64)
      C=np.array([1.0/h**2 for i in range(1,n)],dtype=np.float64)
      # here discretized term is set negative to solve for nan problem
      D=np.array([(y[i+1] - 2*y[i] + y[i-1]) / (-1*h**2) + 2.0 +(y[i])**2
for i in range(1,n)],dtype=np.float64)
      dy=np.array([0]+thomasAlgo(A,B,C,D)+[0],dtype=np.float64)
      return dy

def createFile(input_x,output_y,h):
    cwd=os.getcwd()
    f=open(str(cwd)+"/Result-h="+str(h)+".txt",'w+')
```

```
    f.write("\t\tResult for h="+str(h)+"\n\n")
    f.write("\tValue of X\t\tValue of Y\n\n")
    for i in range(len(input_x)):
        f.write("\t"+str(input_x[i])+"\t\t"+str(output_y[i])+"\n")

def plotGraph(input_x,output_y,h):
    plt.ylabel("Y")
    plt.xlabel ("X")
    p=plt.plot(input_x,output_y,'r')
    plt.legend(p,["h="+str(h),],loc=4)
    plt.savefig("plot-h="+str(h)+".png")
    plt.show()

def main():
      a_intial=0
      b_intial=1
      y_a=0
      y_b=0
      h=0.01
      error=0.01
      n=int ((b_intial-a_intial)/h)
      y=np.zeros(n+1)
      y_temp=np.zeros(n+1)
      # dy here may not be required
      dy=np.zeros(n+1)
      for i in range(n+1):
            y[i]=intialGuess(a_intial+i*h)
      # print y
      # for k th iteration
      k=1
      flag=0
      while(flag==0):
            # print k
            dy=solver(a_intial,b_intial,y_a,y_b,h,k,y,n)
            y_temp=np.add(y,dy)
            if np.amax(np.absolute(np.subtract(y_temp,y)))<=error :
                  y=y_temp
                  flag=1
            else:
                  y=y_temp
                  flag=0
                  k=k+1
      x=np.linspace(0,1,n+1)
      createFile(x,y,h)
      plotGraph(x,y,h)

main()
```

Output

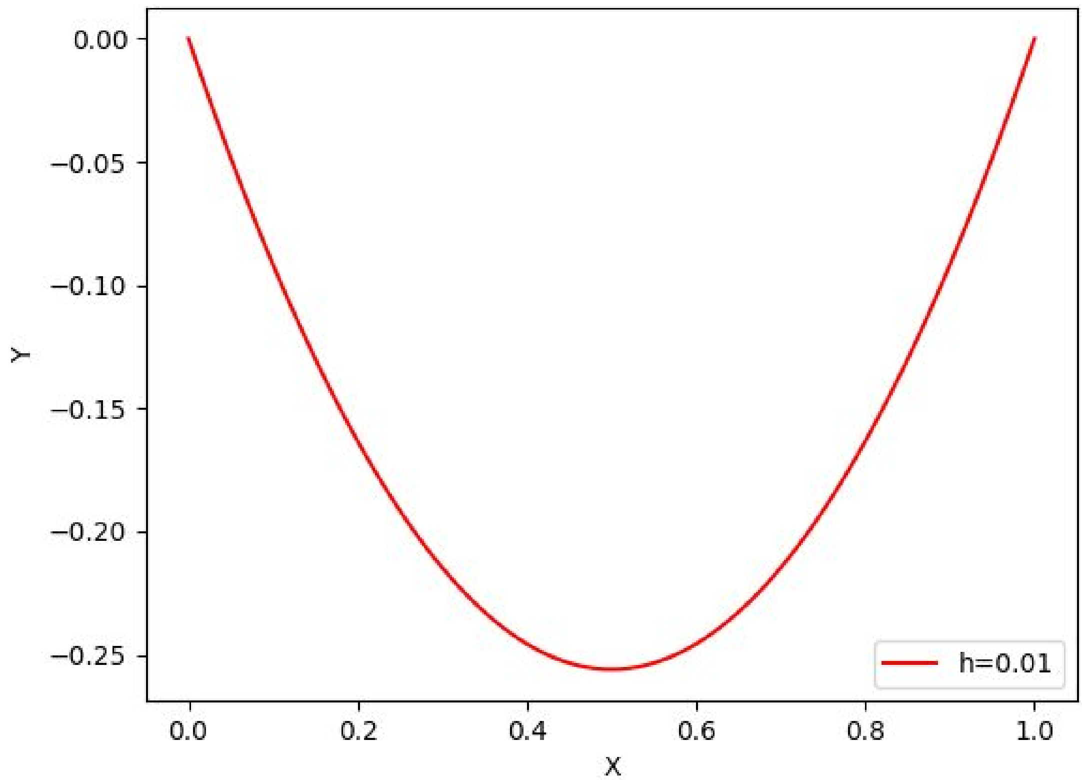
Result for h=0.01

Value of X	***Value of Y***
0.0	*0.0*
0.01	*-0.0100743699275*
0.02	*-0.0199487297057*
0.03	*-0.0296230496887*
0.04	*-0.0390972819192*
0.05	*-0.04837136129*
0.06	*-0.0574452066819*
0.07	*-0.0663187220786*
0.08	*-0.0749917976581*
0.09	*-0.0834643108606*
0.1	*-0.091736127434*
0.11	*-0.0998071024557*
0.12	*-0.107677081332*
0.13	*-0.115345900772*
0.14	*-0.122813389745*
0.15	*-0.130079370405*
0.16	*-0.137143659001*
0.17	*-0.144006066759*
0.18	*-0.150666400742*
0.19	*-0.157124464688*
0.2	*-0.163380059825*
0.21	*-0.169432985658*
0.22	*-0.175283040736*
0.23	*-0.180930023401*
0.24	*-0.186373732498*
0.25	*-0.191613968079*
0.26	*-0.196650532068*
0.27	*-0.201483228915*
0.28	*-0.206111866212*
0.29	*-0.210536255299*
0.3	*-0.214756211835*
0.31	*-0.218771556348*
0.32	*-0.222582114762*
0.33	*-0.226187718896*
0.34	*-0.229588206941*
0.35	*-0.232783423913*
0.36	*-0.235773222072*
0.37	*-0.238557461331*
0.38	*-0.241136009623*
0.39	*-0.243508743258*
0.4	*-0.245675547242*
0.41	*-0.247636315579*

0.42	-0.249390951542
0.43	-0.25093936792
0.44	-0.252281487242
0.45	-0.253417241969
0.46	-0.254346574666
0.47	-0.255069438146
0.48	-0.255585795584
0.49	-0.255895620612
0.5	-0.255998897384
0.51	-0.255895620612
0.52	-0.255585795584
0.53	-0.255069438146
0.54	-0.254346574666
0.55	-0.253417241969
0.56	-0.252281487242
0.57	-0.25093936792
0.58	-0.249390951542
0.59	-0.247636315579
0.6	-0.245675547242
0.61	-0.243508743258
0.62	-0.241136009623
0.63	-0.238557461331
0.64	-0.235773222072
0.65	-0.232783423913
0.66	-0.229588206941
0.67	-0.226187718896
0.68	-0.222582114762
0.69	-0.218771556348
0.7	-0.214756211835
0.71	-0.210536255299
0.72	-0.206111866212
0.73	-0.201483228915
0.74	-0.196650532068
0.75	-0.191613968079
0.76	-0.186373732498
0.77	-0.180930023401
0.78	-0.175283040736
0.79	-0.169432985658
0.8	-0.163380059825
0.81	-0.157124464688
0.82	-0.150666400742
0.83	-0.144006066759
0.84	-0.137143659001
0.85	-0.130079370405
0.86	-0.122813389745
0.87	-0.115345900772
0.88	-0.107677081332
0.89	-0.0998071024557
0.9	-0.091736127434
0.91	-0.0834643108606
0.92	-0.0749917976581

0.93	*-0.0663187220786*
0.94	*-0.0574452066819*
0.95	*-0.04837136129*
0.96	*-0.0390972819192*
0.97	*-0.0296230496887*
0.98	*-0.0199487297057*
0.99	*-0.0100743699275*
1.0	*0.0*

Plot



Code for Equation -2:

```
import os
import numpy as np
import matplotlib.pyplot as plt
import math

def intialGuess(x):
      return x*(math.pi-x)+0.5

def thomasAlgo(
    a,
    b,
    c,
    d,
    ):
    n = len(d)
    c1 = [0 for i in range(0, n)]
    d1 = [0 for i in range(0, n)]
    y = [0 for i in range(0, n)]

    c1[0] = c[0] / b[0]
    d1[0] = d[0] / b[0]

    for i in range(1, n, 1):
        c1[i] = c[i] / (b[i] - a[i] * c1[i - 1])
        d1[i] = (d[i] - a[i] * d1[i - 1]) / (b[i] - a[i] * c1[i - 1])

    y[n - 1] = d1[n - 1]
    for i in range(n - 2, -1, -1):
        y[i] = d1[i] - c1[i] * y[i + 1]

    return y


def solver(a_intial,b_intial,y_a,y_b,h,k,y,n):
      A=np.array([1.0/h**2 -((1.0/4*h**2)*(2*y[i-1]-2*y[i+1])) for i in
range(1,n)],dtype=np.float64)
      B=np.array([-2.0/h**2 -2.0*(y[i]) + 1 for i in
range(1,n)],dtype=np.float64)
      C=np.array([1.0/h**2 -((1.0/4*h**2)*(2*y[i+1]-2*y[i-1])) for i in
range(1,n)],dtype=np.float64)
      D=np.array([-1.0 - y[i]+y[i]**2 +((y[i+1]-y[i-1])/2*h)**2
-((1.0/h**2)*(y[i+1]-2*y[i]+y[i-1])) for i in range(1,n)],dtype=np.float64)
      dy=np.array([0]+thomasAlgo(A,B,C,D)+[0],dtype=np.float64)
      return dy

def createFile(input_x,output_y,h):
    cwd=os.getcwd()
```

```
    f=open(str(cwd)+"/Result-h="+str(h)+".txt",'w+')
    f.write("\t\tResult for h="+str(h)+"\n\n")
    f.write("\tValue of X\t\tValue of Y\n\n")
    for i in range(len(input_x)):
        f.write("\t"+str(input_x[i])+"\t\t"+str(output_y[i])+"\n")

def plotGraph(input_x,output_y,h):
    plt.ylabel("Y")
    plt.xlabel ("X")
    p=plt.plot(input_x,output_y,'r')
    plt.legend(p,["h="+str(h),],loc=4)
    plt.savefig("plot-h="+str(h)+".png")
    plt.show()

def main():
      a_intial=0
      b_intial=math.pi
      y_a=0.5
      y_b=0.5
      error=0.01
      n=100
      h=math.pi/n
      y=np.zeros(n+1)
      y_temp=np.zeros(n+1)
      # dy here may not be required
      dy=np.zeros(n+1)
      for i in range(n+1):
            y[i]=intialGuess(a_intial+i*h)
      # print y
      # for k th iteration
      k=1
      flag=0
      while(flag==0):
            # print k
            dy=solver(a_intial,b_intial,y_a,y_b,h,k,y,n)
            y_temp=np.add(y,dy)
            if np.amax(np.absolute(np.subtract(y_temp,y)))<=error :
                  y=y_temp
                  flag=1
            else:
                  y=y_temp
                  flag=0
                  k=k+1
      x=np.linspace(0,1,n+1)
      createFile(x,y,h)
      plotGraph(x,y,h)

main()
```

Output

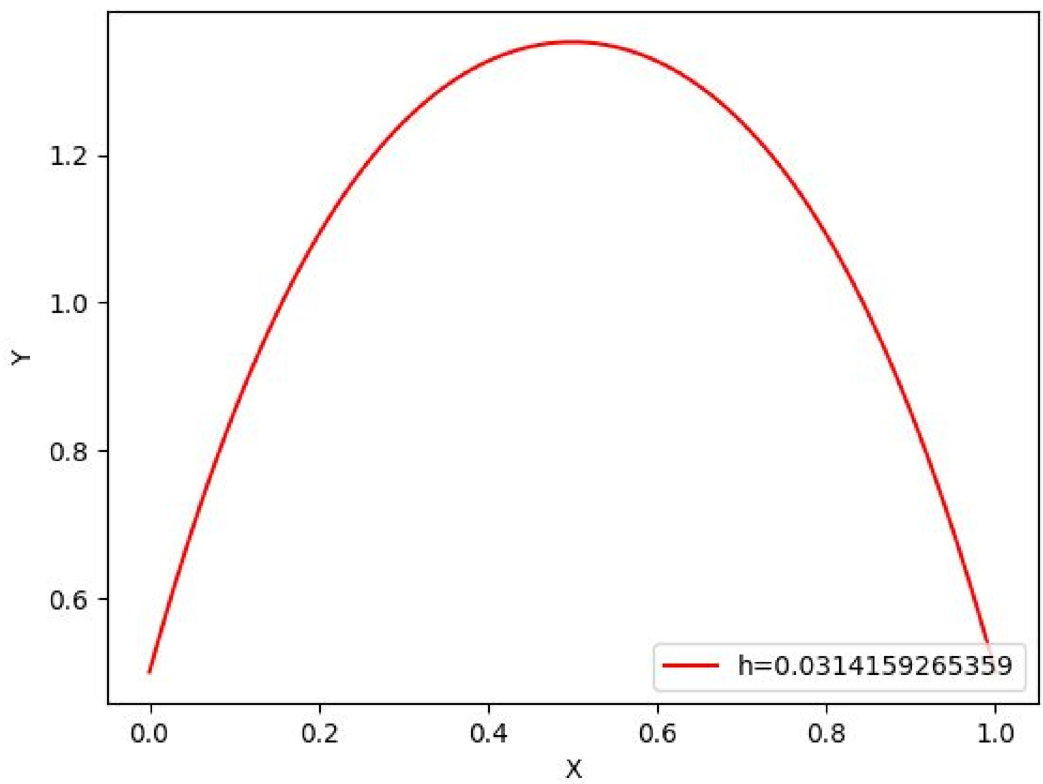
Result for h=0.0314159265359

Value of X	*Value of Y*
0.0	*0.5*
0.01	*0.540572619992*
0.02	*0.57991316566*
0.03	*0.61802631508*
0.04	*0.654919513883*
0.05	*0.690602700517*
0.06	*0.72508804346*
0.07	*0.758389690939*
0.08	*0.790523533532*
0.09	*0.821506979852*
0.1	*0.851358745369*
0.11	*0.880098654325*
0.12	*0.90774745455*
0.13	*0.934326644935*
0.14	*0.959858315215*
0.15	*0.98436499767*
0.16	*1.00786953029*
0.17	*1.03039493091*
0.18	*1.05196428184*
0.19	*1.07260062439*
0.2	*1.09232686282*
0.21	*1.11116567705*
0.22	*1.12913944379*
0.23	*1.14627016524*
0.24	*1.16257940514*
0.25	*1.17808823146*
0.26	*1.19281716521*
0.27	*1.20678613503*
0.28	*1.22001443696*
0.29	*1.23252069899*
0.3	*1.24432285001*
0.31	*1.25543809263*
0.32	*1.26588287965*
0.33	*1.27567289371*
0.34	*1.28482302985*
0.35	*1.2933473806*
0.36	*1.30125922338*
0.37	*1.30857100994*
0.38	*1.3152943575*
0.39	*1.32144004149*
0.4	*1.32701798957*
0.41	*1.33203727683*
0.42	*1.33650612196*

0.43	1.34043188416
0.44	1.34382106084
0.45	1.34667928578
0.46	1.3490113278
0.47	1.35082108974
0.48	1.35211160778
0.49	1.35288505092
0.5	1.35314272068
0.51	1.35288505092
0.52	1.35211160778
0.53	1.35082108974
0.54	1.3490113278
0.55	1.34667928578
0.56	1.34382106084
0.57	1.34043188416
0.58	1.33650612196
0.59	1.33203727683
0.6	1.32701798957
0.61	1.32144004149
0.62	1.3152943575
0.63	1.30857100994
0.64	1.30125922338
0.65	1.2933473806
0.66	1.28482302985
0.67	1.27567289371
0.68	1.26588287965
0.69	1.25543809263
0.7	1.24432285001
0.71	1.23252069899
0.72	1.22001443696
0.73	1.20678613503
0.74	1.19281716521
0.75	1.17808823146
0.76	1.16257940514
0.77	1.14627016524
0.78	1.12913944379
0.79	1.11116567705
0.8	1.09232686282
0.81	1.07260062439
0.82	1.05196428184
0.83	1.03039493091
0.84	1.00786953029
0.85	0.98436499767
0.86	0.959858315215
0.87	0.934326644935
0.88	0.90774745455
0.89	0.880098654325
0.9	0.851358745369
0.91	0.821506979852
0.92	0.790523533532
0.93	0.758389690939

0.94	*0.72508804346*
0.95	*0.690602700517*
0.96	*0.654919513883*
0.97	*0.61802631508*
0.98	*0.57991316566*
0.99	*0.540572619992*
1.0	*0.5*

Plot



Assignment No:4

Using Crank-Nicolson Solve the following:

1. $u_t = u_{xx}$

   Initial conditions

   $$u(x,0) = 1 \quad , \; 0 < x < 1$$
   $$u_x(0,t) = \quad u \quad , \; t > 0$$
   $$u_x(1,t) = \; -u \quad , \; t > 0$$

   Assume $\triangle x = 0.2, t_{max} = 50$

2. $u_t + cu_x = vu_{xx}$

   Initial conditions

   $$u(x,0) = 1 \quad , \; 0 < x < 1$$
   $$u_x(0,t) = \quad u \quad , \; t > 0$$
   $$u_x(1,t) = \; -u \quad , \; t > 0$$

   Assume $\triangle x = 0.2, t_{max} = 50$

Code for Problem 1:

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

def triDiagonal(A, C, n):
      B = np.zeros(n)
      gamma =  np.zeros(n-2)
```

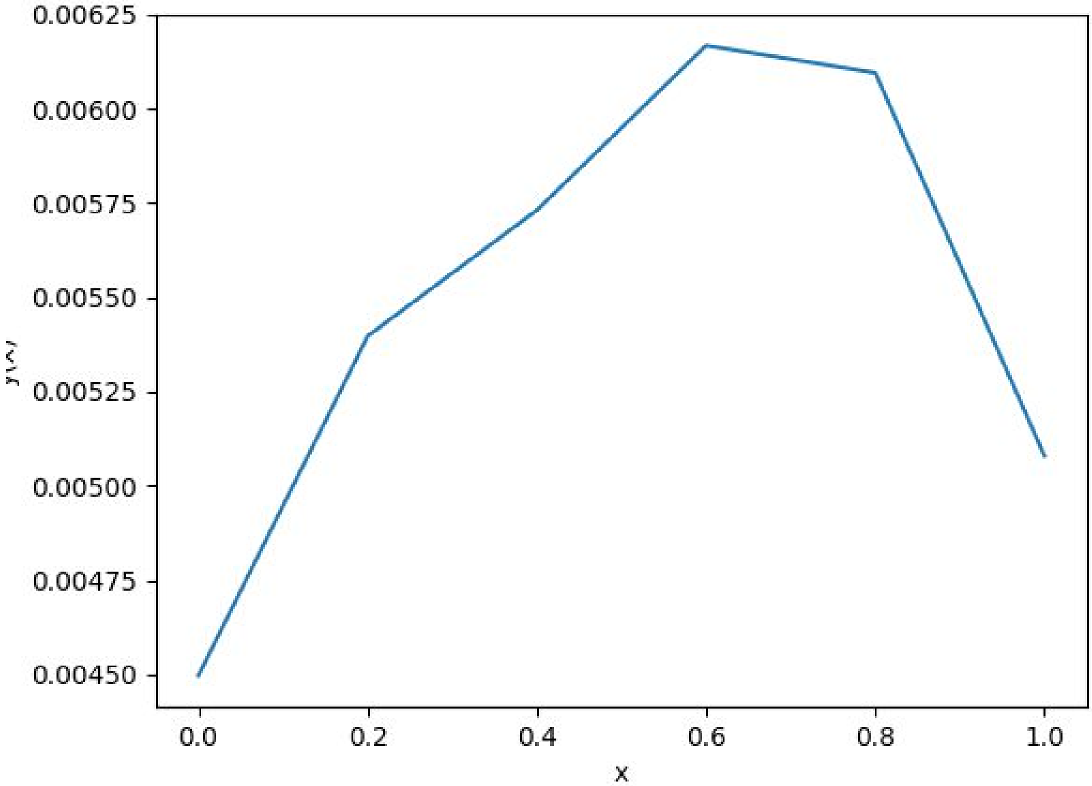
```
      rho = np.zeros(n-2)
      for i in range(n-2):
            if i==0:
                  gamma[0] = A[0][1]/A[0][0]
                  rho[0] = C[1]/A[0][0]
            elif i==n-3:
                  gamma[i] = 0
                  rho[i] =
(C[i]-A[i][i-1]*rho[i-1])/(A[i][i]-A[i][i-1]*gamma[i-1])
            else:
                  gamma[i] = (A[i][i+1])/(A[i][i]-A[i][i-1]*gamma[i-1])
                  rho[i] =
(C[i]-A[i][i-1]*rho[i-1])/(A[i][i]-A[i][i-1]*gamma[i-1])
      for i in range(n-2, 0, -1):
            if i==n-2:
                  B[i] = rho[i-1]
            else:
                  B[i] = rho[i-1] - gamma[i-1]*B[i+1]
      return B

def crank(a, b, dx, r, j_max):
    n = int((b - a)/dx) + 1
    w = np.zeros((j_max+1,n))
    A = np.zeros((n-2,n-2))
    C = np.zeros(n-2)
    for j in range(j_max+1):
      if j==0:
            for i in range(n):
                  w[j][i] = intialcondition(a+i*dx)
      else:
            for i in range(n-2):
                  if i==0:
                        A[i][i] = 2+2*r-r/(1+dx)
                        A[i][i+1] = -r
                  elif i==n-3:
                        A[i][i-1] = -r
                        A[i][i] = 2+2*r-r/(1+dx)
                  else:
                        A[i][i-1] = -r
                        A[i][i] = 2+2*r
                        A[i][i+1] = -r
                  C[i] = r*w[j-1][i-1] + (2-2*r)*w[j-1][i] + r*w[j-1][i+1]
            w[j] = triDiagonal(A,C,n)
            w[j][0] = w[j][1]/(1+dx)
            w[j][n-1] = w[j][n-2]/(1+dx)
    return w

def intialcondition(x):
      return 1

def main():
```

```
    a = 0
    b = 1
    dx = 0.2
    r = 1
    j_max = 50
    x = np.linspace(a,b,(b - a)/dx +1)
    n = int((b - a)/dx) + 1
    w = crank(a, b, dx, r, j_max)
    for j in range(j_max,j_max+1,1) :
    #for j in range(j_max+1) :
        file = open("result(j="+str(j)+").txt", 'w')
        for i in xrange(n):
            file.write(str(x[i]) + "\t" + str(w[j][i]) + "\n")
        file.close()
        plt.plot(x, w[j],  label="h={0}".format(j))
        plt.xlabel('x')
        plt.ylabel('y(x)')
        plt.savefig('Plot(j=' + str(j) + ').png')
        plt.clf()

if __name__ == '__main__':
   main()
```

Output

```
0.0      0.00449735475037
0.2      0.00539682570044
0.4      0.00573077047522
0.6      0.00616707862423
0.8      0.00609487346371
1.0      0.00507906121976
```

Plot



Code for Problem 2:

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

def triDiagonal(A, C, n):
      B = np.zeros(n)
      gamma =  np.zeros(n-2)
      rho = np.zeros(n-2)
      for i in range(n-2):
            if i==0:
                  gamma[0] = A[0][1]/A[0][0]
                  rho[0] = C[1]/A[0][0]
            elif i==n-3:
                  gamma[i] = 0
                  rho[i] =
(C[i]-A[i][i-1]*rho[i-1])/(A[i][i]-A[i][i-1]*gamma[i-1])
            else:
                  gamma[i] =
(A[i][i+1])/(A[i][i]-A[i][i-1]*gamma[i-1])
                  rho[i] =
(C[i]-A[i][i-1]*rho[i-1])/(A[i][i]-A[i][i-1]*gamma[i-1])
      for i in range(n-2, 0, -1):
            if i==n-2:
                  B[i] = rho[i-1]
            else:
                  B[i] = rho[i-1] - gamma[i-1]*B[i+1]
      return B

def crank(a, b, dx, r, c, v, j_max):
   n = int((b - a)/dx) + 1
   dt = r * dx *dx
   w = np.zeros((j_max+1,n))
   A = np.zeros((n-2,n-2))
   C = np.zeros(n-2)
   for j in range(j_max+1):
      if j==0:
            for i in range(n):
                  w[j][i] = intialcondition(a+i*dx)
      else:
            for i in range(n-2):
                  if i==0:
```

```
                        A[i][i] = 1/dt + v/(dx*dx) + 4*(-c/(4*dx)-
v/(2*dx*dx))/(3+2*dx)
                        A[i][i+1] = c/(4*dx)- v/(2*dx*dx) -
(-c/(4*dx)- v/(2*dx*dx))/(3+2*dx)
                elif i==n-3:
                        A[i][i-1] = -c/(4*dx)- v/(2*dx*dx) -
(c/(4*dx)- v/(2*dx*dx))/(3+2*dx)
                        A[i][i] = 1/dt + v/(dx*dx) + 4*(c/(4*dx)-
v/(2*dx*dx))/(3+2*dx)
                else:
                        A[i][i-1] = -c/(4*dx)- v/(2*dx*dx)
                        A[i][i] = 1/dt + v/(dx*dx)
                        A[i][i+1] = c/(4*dx)- v/(2*dx*dx)
                C[i] = (c/(4*dx)+ v/(2*dx*dx))*w[j-1][i-1] + (1/dt -
v/(dx*dx))*w[j-1][i] + (-c/(4*dx)+ v/(2*dx*dx))*w[j-1][i+1]
            w[j] = triDiagonal(A,C,n)
            w[j][0] = (4*w[j][1]-w[j][2])/(3+2*dx)
            w[j][n-1] = (4*w[j][n-2]-w[j][n-3])/(3+2*dx)
    return w

def intialcondition(x):
      return 1

def main():

    a = 0
    b = 1
    dx = 0.2
    r = 1
    j_max = 50
    v = 1
    c_vals = [0,0.05,0.1]
    x = np.linspace(a,b,(b - a)/dx +1)
    n = int((b - a)/dx) + 1
    for c in c_vals:
       w = crank(a, b, dx, r, c, v,j_max)
       for j in range(j_max,j_max+1,1):
            file = open("result(j="+str(j)+"_c="+str(c)+").txt", 'w')
            for i in xrange(n):
                        file.write(str(x[i]) + "\t" + str(w[j][i]) +
"\n")
            file.close()
            plt.plot(x, w[j],  label="h={0}".format(j))
            plt.xlabel('x')
```

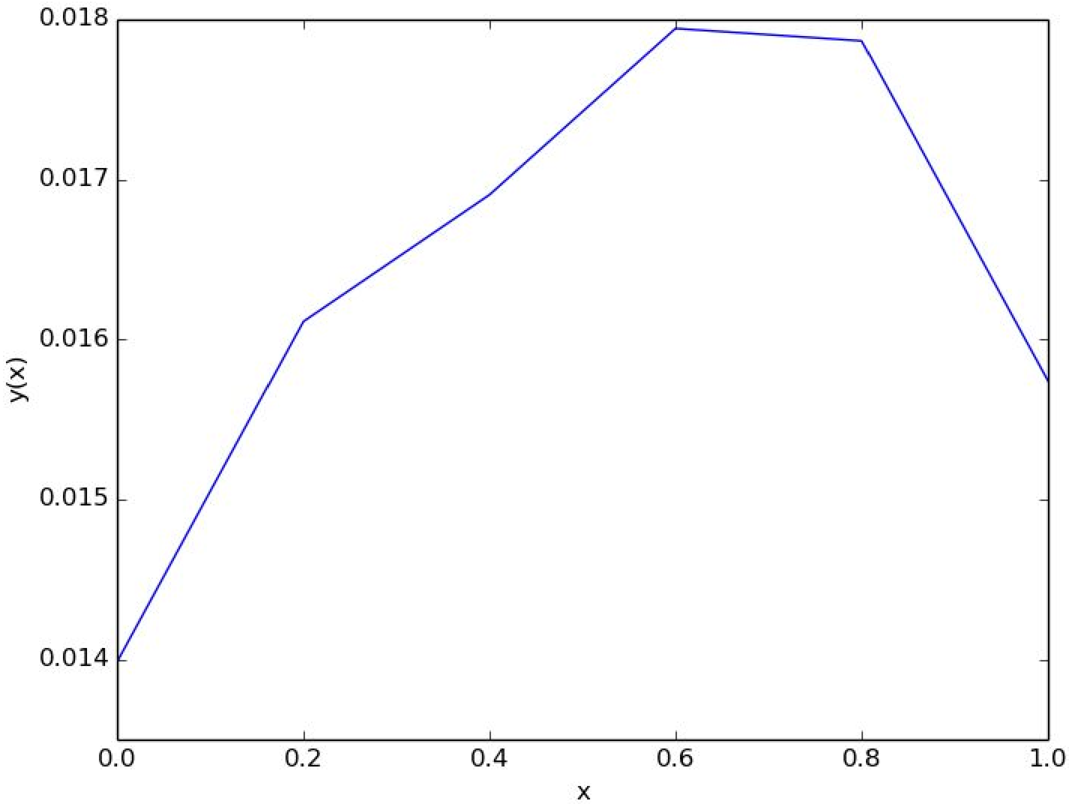
```
            plt.ylabel('y(x)')
            plt.savefig('Plot(j=' + str(j)+"_c="+str(c)+ ').png')
            plt.clf()

if __name__ == '__main__':
  main()
```

Output

C = 0

```
0.0   0.0139841666745
0.2   0.0161126909483
0.4   0.0169045970998
0.6   0.0179436974319
0.8   0.0178664291943
1.0   0.0157417703957
```

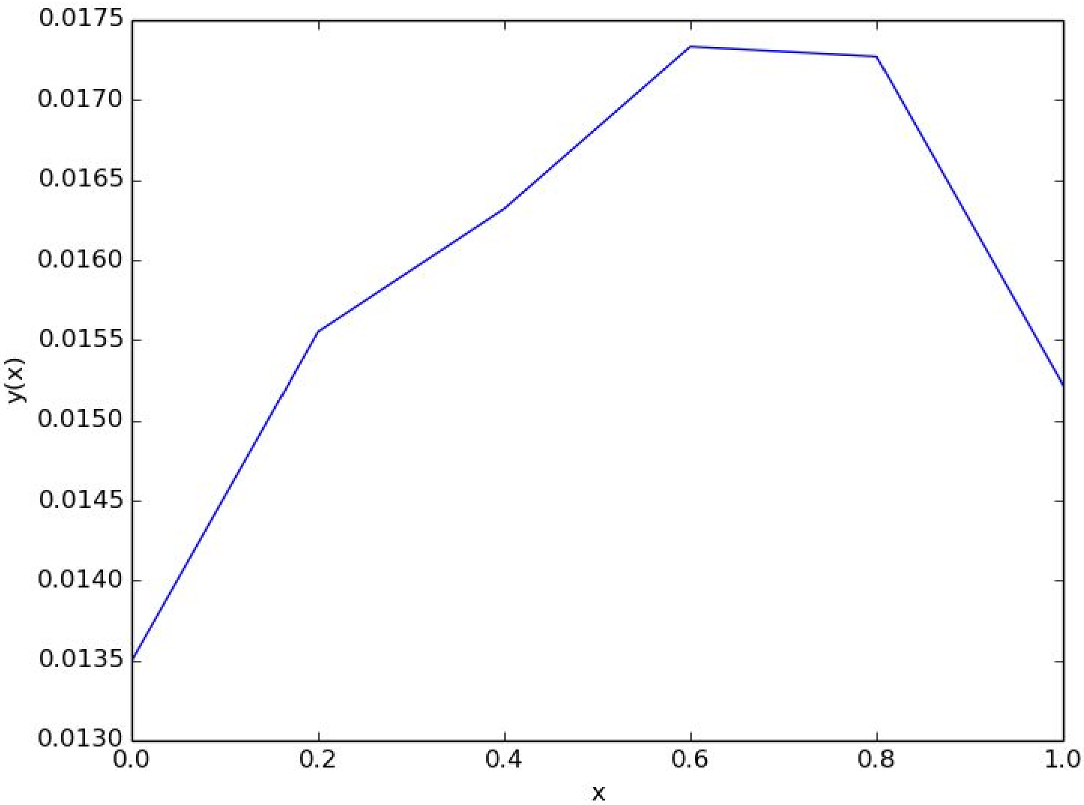
Plot



C = 0.05

0.0	0.013496336292
0.2	0.0155520260747
0.4	0.0163205609058
0.6	0.0173337981851
0.8	0.0172714670497
1.0	0.0152211970628

Plot

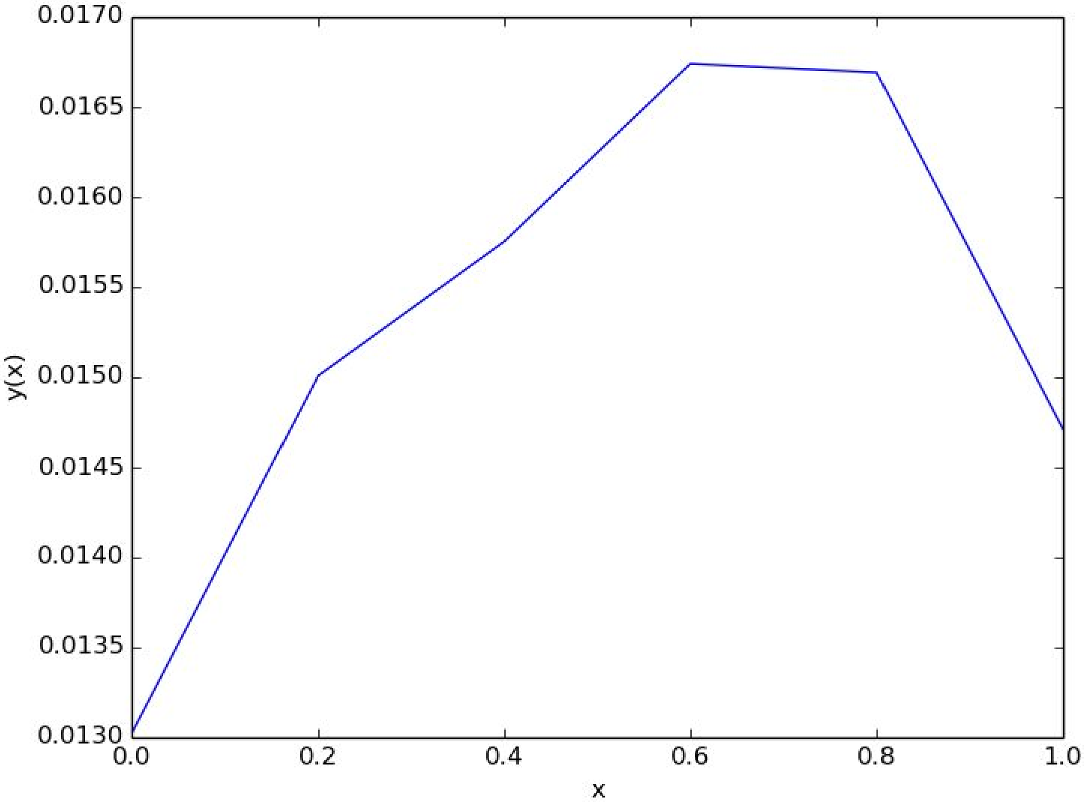


C = 0.1

0.0	0.0130220584757
0.2	0.0150068627936
0.4	0.0157524523568
0.6	0.0167400340657
0.8	0.0166916855674

```
1.0   0.014713737707
```

Plot



Index

Assignment No:5

$$C_t = C_{xx} + C_{yy}$$

Subject to

$$C(x,y,0) = cos(\tfrac{\pi x}{2})cos(\tfrac{\pi y}{2}) \quad -1 \le x,y \le 1$$
$$C(x,y,t) = 0 \text{ at boundary value}$$
$$v = 0.5$$

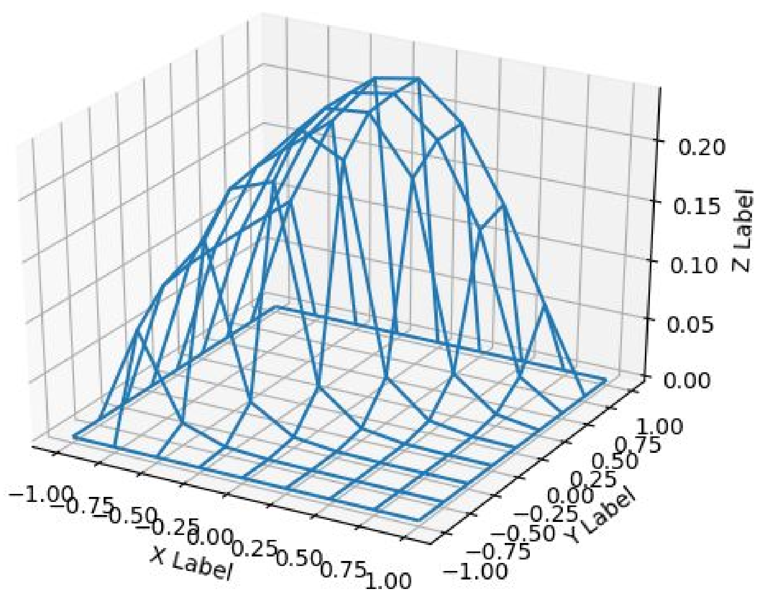
Code

```
import os
import numpy as np
from math import *
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def power(number):
	return number*number

def thomasAlgo(a,b,c,d):
	n=len(d)
	c_=np.zeros(n)
	d_=np.zeros(n)
	y=np.zeros(n)
	c_[1]=c[1]/b[1]
	d_[1]=d[1]/b[1]

	for i in range(2,n):
		c_[i]=c[i]/(b[i]-a[i]*c_[i-1])
		d_[i]=(d[i]-a[i]*d_[i-1])/(b[i]-a[i]*c_[i-1])

	y[n-1]=d_[n-1]
	for i in range(n-2,0,-1):
		y[i]=d_[i]-c_[i]*y[i+1]

	return y

def step1(C,dx,dt,dy,n,m):

	a=np.zeros(n,dtype=np.float32)
	b=np.zeros(n,dtype=np.float32)
	c=np.zeros(n,dtype=np.float32)
	d=np.zeros(n,dtype=np.float32)
```

```
	for j in range(1,m):
		print
		a=[(-1/(2*power(dx))) for i in range(n)]
		b=[(1/(power(dx))+1/dt) for i in range(n)]
		c=[(-1/(2*power(dx))) for i in range(n)]
		for i in range(n):
			d[i]=(C[i][j]/dt +
(C[i][j+1]-2*C[i][j]+C[i][j-1])/(2*(dy**2)))
		y=thomasAlgo(a,b,c,d)
		for i in range(1,len(y),1):
			C[i][j]=y[i]
		C[0][j]=0.0
		C[n][j]=0.0
	C[0][m]=0
	C[n][m]=0

def step2(C,dx,dt,dy,n,m):
	a=np.zeros(n,dtype=np.float32)
	b=np.zeros(n,dtype=np.float32)
	c=np.zeros(n,dtype=np.float32)
	d=np.zeros(n,dtype=np.float32)

	for i in range(1,n):
		a=[(-1/(2*power(dy))) for j in range(m)]
		b=[(1/(power(dy))+1/dt) for j in range(m)]
		c=[(-1/(2*power(dy))) for j in range(m)]
		for j in range(m):
			d[i]=(C[i][j]/dt +
(C[i+1][j]-2*C[i][j]+C[i-1][j])/(2*(dx**2)))
		y=thomasAlgo(a,b,c,d)
		for j in range(1,len(y),1):
			C[i][j]=y[j]
		C[i][0]=0.0
		C[i][m]=0.0
	C[n][m]=0
	C[n][0]=0



def main():
	dx=0.25
	dy=0.25
	dt=dx*dx/2
	a=-1.0
	b=1.0
	n=int((b-a)/dx)
	m=int((b-a)/dy)
	C=np.zeros((n+1,m+1),dtype=np.float32)
	for i in range(n+1):
		C[i][0]=0.0
```

```
        for j in range(1,m):
            C[i][j]=cos(pi*(-1+i*dx)/2)*cos(pi*(-1+j*dy)/2)
        C[i][m]=0.0

    step1(C,dx,dt,dy,n,m)
    step2(C,dx,dt,dy,n,m)
    x=[-1+i*dx for i in range(n+1)]
    x=np.asarray(x)
    y=[-1+j*dy for j in range(m+1)]
    y=np.asarray(y)
    X,Y=np.meshgrid(x,y)
    Z=C.reshape(X.shape)
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_wireframe(X,Y,Z)
    ax.set_xlabel('X Label')
    ax.set_ylabel('Y Label')
    ax.set_zlabel('Z Label')
    plt.show()


main()
```

Plot



Assignment No:6

$$U_{xx} + U_{yy} = 0$$

Subject to

$$U(0, y) = 0$$
$$U(4, y) = 8 + 2y$$
$$U(x, 4) = x^2$$
$$U(x, 0) = 0.5x^2$$
$$\delta t = \delta x = 0.1, 0.05, 0.01$$

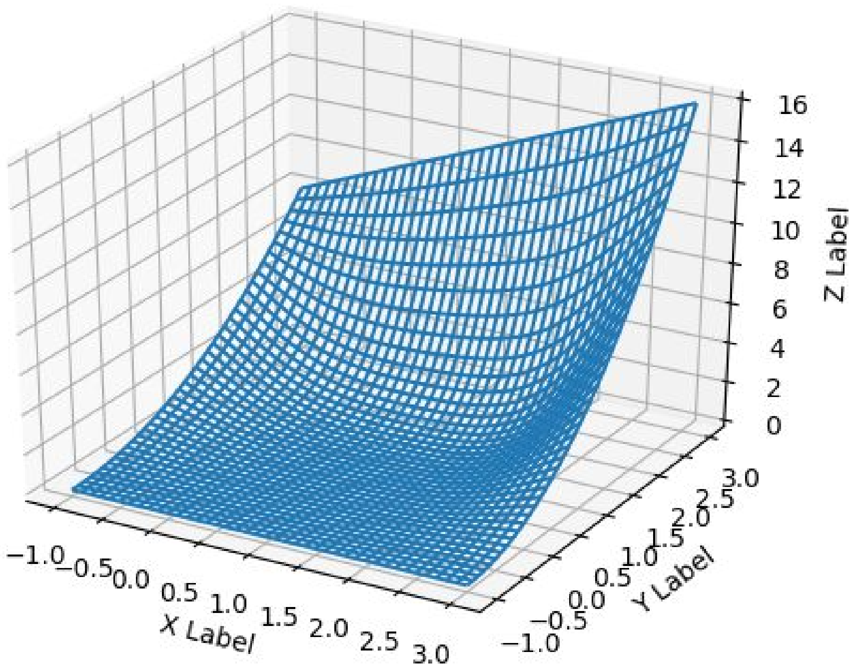
Code

```
from numpy import *
from math import *
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

deltax=[0.1,0.05,0.01]
deltay=[0.1,0.05,0.01]
for l in range(3):
	fig = plt.figure()
	ax = fig.add_subplot(111, projection='3d')

	n=int(4/deltax[l])
	m=int(4/deltay[l])

	x=[-1+i*deltax[l] for i in range(n+1)]
	x=asarray(x)
	y=[-1+j*deltay[l] for j in range(m+1)]
	y=asarray(y)

	U=[]
	for i in range(n+1):
		U.append([((i*deltax[l])**2)/2])
		if(i==n):
			for j in range(1,m+1):
				U[i].append((8+2*j*deltay[l]))
		else:
			for j in range(1,m):
				U[i].append(0)
			U[i].append((i*deltax[l])**2)
	# print U
	U=asarray(U)
	for k in range(50):
		for i in range(1,n):
			for j in range(1,m):
				U[i][j]=1/(2*((1/deltax[l]**2)+
1/deltay[l]**2))*((U[i+1][j]+U[i-1][j])/(deltax[l]**2)+(U[i][j+1]+U[i][j-1]
)/(deltay[l]**2))
```

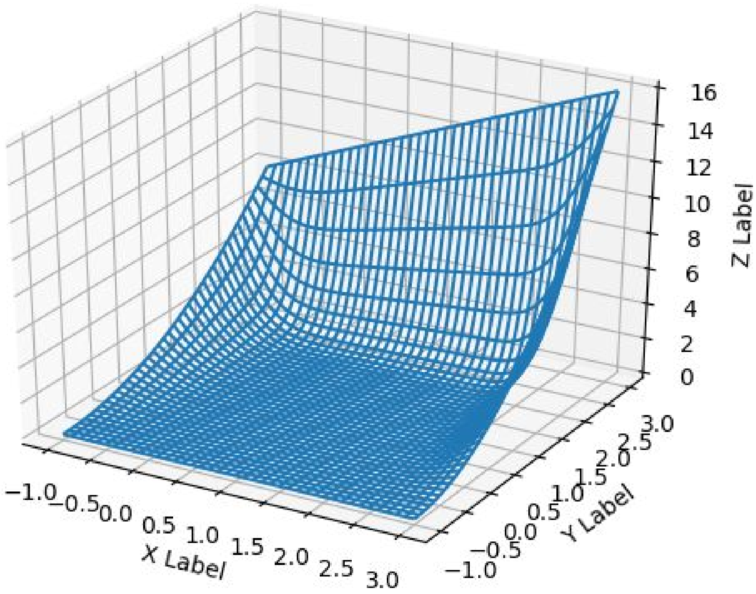
```
X, Y = meshgrid(x, y)
Z = U.reshape(X.shape)
ax.plot_wireframe(X,Y,Z)
      # ax.contour(X,Y,Z)
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
plt.show()
```

Plot

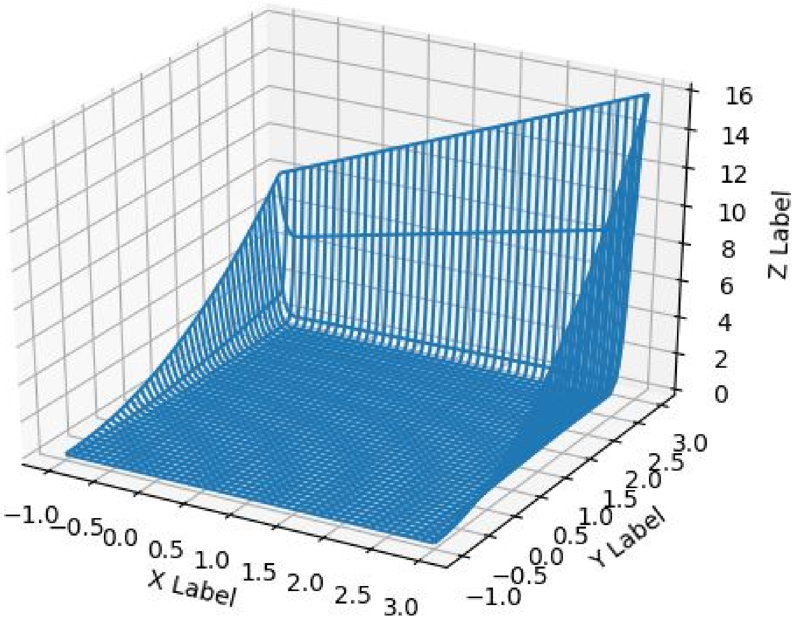
dx=dy=0.1



dx=dy=0.05



$dx=dy=0.01$



Assignment No :7

$$U_t + U_x = 0 \qquad 0 \le x \le 25$$

Subject to

$$U(x,0) = exp(-20(x-2)^2) + exp(-(x-5)^2)$$

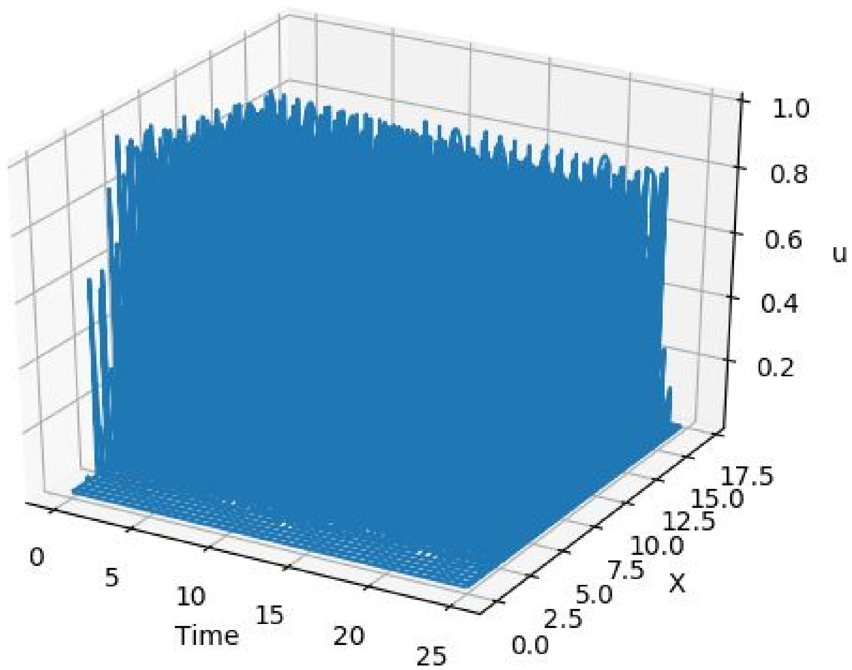
$$v = 0.8, \delta x = 0.05 \ solve\ upto\ t = 17$$

Code:

```
from numpy import *
from math import *
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

deltax=.05
mu=.8
c=1;
deltat=mu*deltax/c
n=int(25/deltax)
m=int(17/deltat)

x=[i*deltax for i in range(n+1)]
x=asarray(x)

t=[i*deltat for i in range(m+1)]
t=asarray(t)

u=[]
for i in range(n+1):
      u.append([exp(-20*(i*deltax-2)**2)+exp(-(i*deltax-5)**2)])
# print u
for i in range(m):
      u[0].append(u[0][i])
      for j in range(1,n+1):
            u[j].append((1-mu)*u[j][i]+mu*u[j-1][i])
u=asarray(u)
print shape(u)
# print u[:][m]
c=[]
for i in range(n+1):
      c.append(u[i][m-2])
c=asarray(c)
# plt.plot(x,c)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

X, T = meshgrid(x, t)
```

```
Z = u.reshape(X.shape)
ax.plot_wireframe(X,T,Z)
      # ax.contour(X,Y,Z)
ax.set_xlabel('Time')
ax.set_ylabel('X')
ax.set_zlabel('u')
# plt.show()
plt.show()
```

Plot



Assignment no:8

$$U_t + U_x = 0$$

Subject to

$$U(x,0) = x^2 \qquad 0 \le x \le 1$$
$$0 \qquad x > 1$$
$$U(0,t) = 0$$
$$\delta x = 0.25, v = 0.5$$

- *FTBS*
- *Lax*
- *Lax-Wendroff*
- *Mac-Cormack*

Code

```
import os
import matplotlib.pyplot as plt
# ut+ux=0

# Forward Time Backward Space,since c is positive

def FTBS():
	dx=0.25
	v=0.5
	dt=v*dx
	n=int(1.0/dx)
	U=[[0 for i in range(n+1) ] for j in range(100)]
	# intial contidions
	for i in range(n+1):
		x=i*dx
		U[0][i]=x*x
	for i in range(99):
		for j in range(1,n+1):
			U[i+1][j]=U[i][j]-v*(U[i][j]-U[i][j-1])

	X=[]
	for i in range(n+1):
		X.append(i*dx)

	plt.ylabel("U")
	plt.xlabel ("X")
	p=plt.plot(X,U[1],'r')
	plt.legend(p,["v="+str(v)],loc=4)
	plt.savefig("I-TimeFTBS Step v="+str(v)+".png")
	plt.show()
	f=open("ResultsFTBS.txt",'w+')
	f.write("   Time              Values\n\n")
	for i in range(99):
		f.write("   "+str(i)+"        "+str(U[i])+"\n")
```

```
def LAX():
	dx=0.25
	v=0.5
	dt=v*dx
	n=int(1.0/dx)
	U=[[0 for i in range(n+3) ] for j in range(100)]
	# intial contidions
	for i in range(n+1):
		x=i*dx
		U[0][i]=x*x
	for i in range(99):
		for j in range(1,n+2):

U[i+1][j]=0.5*((U[i][j-1]+U[i][j+1])+v*(U[i][j+1]-U[i][j-1]))

	X=[]
	for i in range(n+3):
		X.append(i*dx)

	plt.ylabel("U")
	plt.xlabel ("X")
	p=plt.plot(X,U[1],'r')
	plt.legend(p,["v="+str(v)],loc=4)
	plt.savefig("I-TimeLAX Step v="+str(v)+".png")
	plt.show()
	f=open("ResultsLAX.txt",'w+')
	f.write("   Time              Values\n\n")
	for i in range(99):
		f.write("   "+str(i)+"        "+str(U[i])+"\n")

def LW():
	# single step method
	dx=0.25
	v=0.5
	dt=v*dx
	n=int(1.0/dx)
	U=[[0 for i in range(n+3) ] for j in range(100)]
	# intial contidions
	for i in range(n+1):
		x=i*dx
		U[0][i]=x*x
	for i in range(99):
		for j in range(1,n+2):

U[i+1][j]=U[i][j]-v*0.5*(U[i][j+1]-U[i][j-1])+v*v*0.5*(U[i][j+1]+U[i][j-1]-
2*U[i][j])

	X=[]
	for i in range(n+3):
```

```
            X.append(i*dx)

      plt.ylabel("U")
      plt.xlabel ("X")
      p=plt.plot(X,U[1],'r')
      plt.legend(p,["v="+str(v)],loc=4)
      plt.savefig("I-TimeLW Step v="+str(v)+".png")
      plt.show()
      f=open("ResultsLW.txt",'w+')
      f.write("   Time              Values\n\n")
      for i in range(99):
            f.write("   "+str(i)+"        "+str(U[i])+"\n")

def MacCormack():
      dx=0.25
      v=0.5
      dt=v*dx
      n=int(1.0/dx)
      U=[[0 for i in range(n+3) ] for j in range(100)]
      # intial contidions
      for i in range(n+1):
            x=i*dx
            U[0][i]=x*x
      for i in range(99):
            # Predictor Step
            for j in range(1,n+2):
                  U[i+1][j]=U[i][j]-v*(U[i][j+1]-U[i][j])
            # Corrector Step
            for j in range(1,n+2):

U[i+1][j]=U[i][j]-(v*0.5*(U[i][j+1]-U[i][j]))-(v*0.5*(U[i+1][j]-U[i+1][j-1]
))


      X=[]
      for i in range(n+3):
            X.append(i*dx)

      plt.ylabel("U")
      plt.xlabel ("X")
      p=plt.plot(X,U[1],'r')
      plt.legend(p,["v="+str(v)],loc=4)
      plt.savefig("I-TimeMC Step v="+str(v)+".png")
      plt.show()
      f=open("ResultsMC.txt",'w+')
      f.write("   Time              Values\n\n")
      for i in range(99):
            f.write("   "+str(i)+"        "+str(U[i])+"\n")

FTBS()
LAX()
```

LW()
MacCormack()

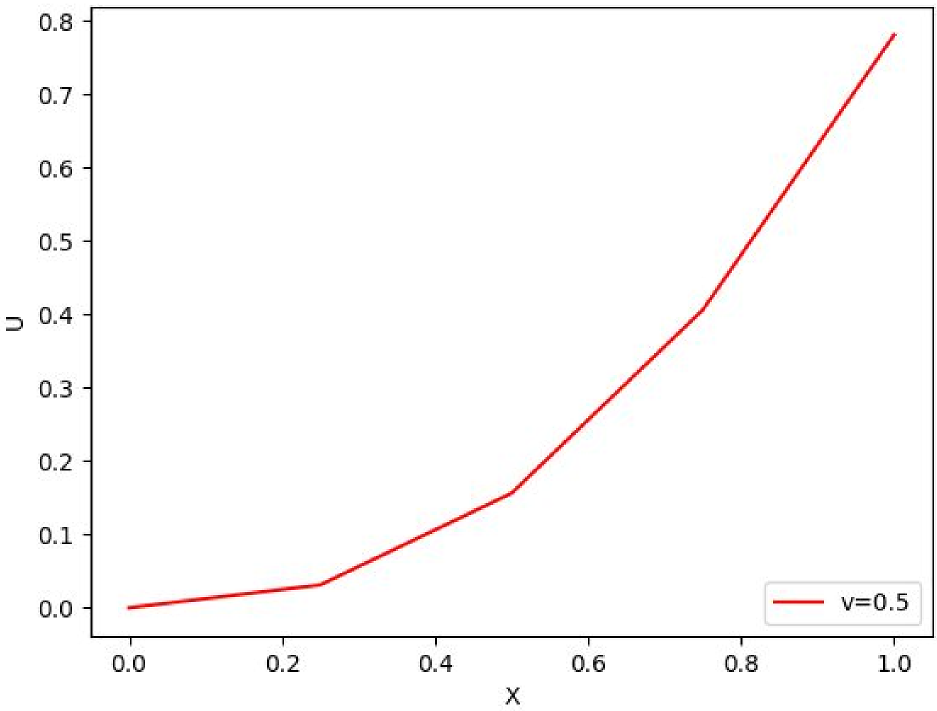
Results

- ***FTBS***

Time	*Values*
0	*[0.0, 0.0625, 0.25, 0.5625, 1.0]*
1	*[0, 0.03125, 0.15625, 0.40625, 0.78125]*
2	*[0, 0.015625, 0.09375, 0.28125, 0.59375]*
3	*[0, 0.0078125, 0.0546875, 0.1875, 0.4375]*
4	*[0, 0.00390625, 0.03125, 0.12109375, 0.3125]*
5	*[0, 0.001953125, 0.017578125, 0.076171875, 0.216796875]*
6	*[0, 0.0009765625, 0.009765625, 0.046875, 0.146484375]*
7	*[0, 0.00048828125, 0.00537109375, 0.0283203125, 0.0966796875]*
8	*[0, 0.000244140625, 0.0029296875, 0.016845703125, 0.0625]*
9	*[0, 0.0001220703125, 0.0015869140625, 0.0098876953125, 0.0396728515625]*
10	*[0, 6.103515625e-05, 0.0008544921875, 0.0057373046875, 0.0247802734375]*
11	*[0, 3.0517578125e-05, 0.000457763671875, 0.0032958984375, 0.0152587890625]*
12	*[0, 1.52587890625e-05, 0.000244140625, 0.0018768310546875, 0.00927734375]*
13	*[0, 7.62939453125e-06, 0.00012969970703125, 0.00106048583984375, 0.00557708740234375]*
14	*[0, 3.814697265625e-06, 6.866455078125e-05, 0.0005950927734375, 0.00331878662109375]*
15	*[0, 1.9073486328125e-06, 3.62396240234375e-05, 0.000331878662109375, 0.001956939697265625]*
16	*[0, 9.5367431640625e-07, 1.9073486328125e-05, 0.00018405914306640625, 0.0011444091796875]*
17	*[0, 4.76837158203125e-07, 1.0013580322265625e-05, 0.00010156631469726562, 0.0006642341613769531]*
18	*[0, 2.384185791015625e-07, 5.245208740234375e-06, 5.5789947509765625e-05, 0.0003829002380371094]*
19	*[0, 1.1920928955078125e-07, 2.7418136596679688e-06, 3.0517578125e-05, 0.0002193450927734375]*
20	*[0, 5.960464477539063e-08, 1.430511474609375e-06, 1.6629695892333984e-05, 0.00012493133544921875]*
21	*[0, 2.9802322387695312e-08, 7.450580596923828e-07, 9.03010368347168e-06, 7.078051567077637e-05]*
22	*[0, 1.4901161193847656e-08, 3.8743019104003906e-07, 4.887580871582031e-06, 3.9905309677124023e-05]*
23	*[0, 7.450580596923828e-09, 2.0116567611694336e-07, 2.637505531311035e-06, 2.2396445274353027e-05]*
24	*[0, 3.725290298461914e-09, 1.043081283569336e-07, 1.4193356037139893e-06, 1.2516975402832031e-05]*
25	*[0, 1.862645149230957e-09, 5.4016709327697754e-08, 7.618218660354614e-07, 6.96815550327301e-06]*
26	*[0, 9.313225746154785e-10, 2.7939677238464355e-08, 4.079192876815796e-07, 3.864988684654236e-06]*
27	*[0, 4.656612873077393e-10, 1.4435499906539917e-08, 2.1792948246002197e-07, 2.1364539861679077e-06]*
28	*[0, 2.3283064365386963e-10, 7.450580596923828e-09, 1.1618249118328094e-07, 1.1771917343139648e-06]*
29	*[0, 1.1641532182693481e-10, 3.841705620288849e-09, 6.181653589010239e-08, 6.466871127486229e-07]*
30	*[0, 5.820766091346741e-11, 1.979060471057892e-09, 3.282912075519562e-08, 3.5425182431936264e-07]*
31	*[0, 2.91038304733704e-11, 1.0186340659856796e-09, 1.7404090613126755e-08, 1.9354047253727913e-07]*
32	*[0, 1.4551915228366852e-11, 5.238689482212067e-10, 9.211362339556217e-09, 1.0547228157520294e-07]*
33	*[0, 7.275957614183426e-12, 2.6921043172478676e-10, 4.867615643888712e-09, 5.734182195737958e-08]*
34	*[0, 3.637978807091713e-12, 1.382431946694851e-10, 2.5684130378067493e-09, 3.1104718800634146e-08]*
35	*[0, 1.8189894035458565e-12, 7.09405867382884e-11, 1.3533281162381172e-09, 1.6836565919220448e-08]*
36	*[0, 9.094947017729282e-13, 3.637978807091713e-11, 7.121343514882028e-10, 9.094947017729282e-09]*
37	*[0, 4.547473508864641e-13, 1.864464138634503e-11, 3.7425706977955997e-10, 4.903540684608743e-09]*
38	*[0, 2.2737367544323206e-13, 9.549694368615746e-12, 1.964508555829525e-10, 2.6388988771941513e-09]*
39	*[0, 1.1368683772161603e-13, 4.888534022029489e-12, 1.0300027497578412e-10, 1.4176748663885519e-09]*
40	*[0, 5.684341886080802e-14, 2.5011104298755527e-12, 5.3944404498906806e-11, 7.60337570682168e-10]*
41	*[0, 2.842170943040401e-14, 1.2789769243681803e-12, 2.822275746439118e-11, 4.071409875905374e-10]*
42	*[0, 1.4210854715202004e-14, 6.536993168992922e-13, 1.475086719437968e-11, 2.176818725274643e-10]*
43	*[0, 7.105427357601002e-15, 3.339550858072471e-13, 7.702283255639486e-12, 1.1621636986092199e-10]*

44	[0, 3.552713678800501e-15, 1.7053025658242404e-13, 4.0181191707233666e-12, 6.195932655828074e-11]
45	[0, 1.7763568394002505e-15, 8.704148513061227e-14, 2.0943247136528953e-12, 3.298872286450205e-11]
46	[0, 8.881784197001252e-16, 4.440892098500626e-14, 1.0906830993917538e-12, 1.7541523789077473e-11]
47	[0, 4.440892098500626e-16, 2.2648549702353193e-14, 5.6754601018838e-13, 9.316103444234614e-12]
48	[0, 2.220446049250313e-16, 1.1546319456101628e-14, 2.950972799453666e-13, 4.941824727211497e-12]
49	[0, 1.1102230246251565e-16, 5.88418203051333e-15, 1.5332179970073412e-13, 2.6184610035784317e-12]
50	[0, 5.551115123125783e-17, 2.9976021664879227e-15, 7.960299086562372e-14, 1.385891401639583e-12

Plot



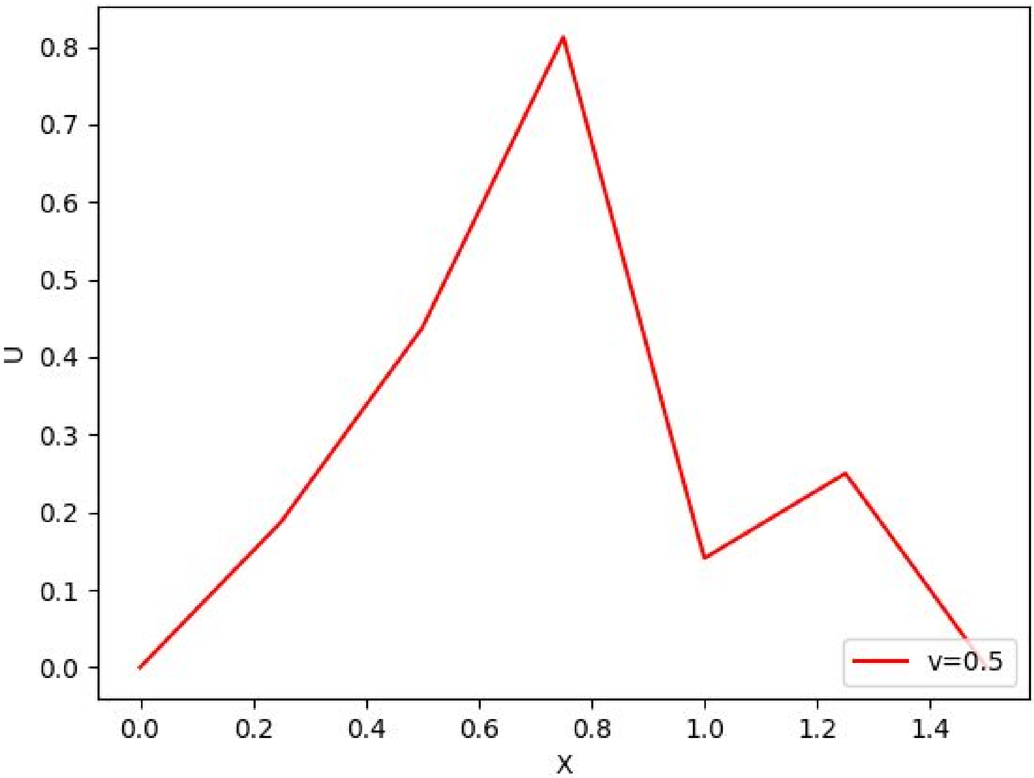
- *Lax*

Results

Time	Values
0	[0.0, 0.0625, 0.25, 0.5625, 1.0, 0, 0]
1	[0, 0.1875, 0.4375, 0.8125, 0.140625, 0.25, 0]
2	[0, 0.328125, 0.65625, 0.21484375, 0.390625, 0.03515625, 0]
3	[0, 0.4921875, 0.2431640625, 0.45703125, 0.080078125, 0.09765625, 0]
4	[0, 0.182373046875, 0.4658203125, 0.120849609375, 0.1875, 0.02001953125, 0]
5	[0, 0.349365234375, 0.13623046875, 0.257080078125, 0.04522705078125, 0.046875, 0]
6	[0, 0.1021728515625, 0.2801513671875, 0.0679779052734375, 0.09942626953125, 0.0113067626953125, 0]
7	[0, 0.210113525390625, 0.07652664184570312, 0.1446075439453125, 0.02547454833984375, 0.0248565673828125, 0]
8	[0, 0.057394981384277344, 0.16098403930664062, 0.038237571716308594, 0.0547943115234375, 0.0063686370849609375, 0]
9	[0, 0.12073802947998047, 0.04302692413330078, 0.08134174346923828, 0.014335870742797852, 0.013698577880859375, 0]
10	[0, 0.032270193099975586, 0.09119081497192383, 0.021508634090423584, 0.0306093692779541, 0.003583967685699463, 0]

11 [0, 0.06839311122894287, 0.024199023842811584, 0.04575473070144653, 0.008065134286880493, 0.007652342319488525, 0]
12 [0, 0.01814926788210869, 0.05141432583332062, 0.012098606675863266, 0.017177939414978027, 0.0020162835717201233, 0]
13 [0, 0.03856074437499046, 0.013611271977424622, 0.025737036019563675, 0.004536864347755909, 0.004294484853744507, 0]
14 [0, 0.010208453983068466, 0.028942963108420372, 0.006805466255173087, 0.009655122645199299, 0.0011342160869389772, 0]
15 [0, 0.02170722233131528, 0.007656213187146932, 0.014477082761004567, 0.0025520286289975047, 0.0024137806612998247, 0]
16 [0, 0.005742159890360199, 0.016284617653582245, 0.0038280747685348615, 0.00542960618622601, 0.0006380071572493762, 0]
17 [0, 0.012213463240186684, 0.004306596048991196, 0.008143359053065069, 0.0014355240600707475, 0.0013574015465565026, 0]
18 [0, 0.003229947036743397, 0.009160885099845473, 0.0021532920573008596, 0.003053890923183644, 0.0003588810150176869, 0]
19 [0, 0.0068706638248841045, 0.002422455802161494, 0.004580639467349101, 0.0008074837755884801, 0.000763472730795911, 0]
20 [0, 0.0018168418516211204, 0.005153145556732852, 0.0012112267822317335, 0.0017177644149342086, 0.00020187094389712001, 0]
21 [0, 0.003864859167549639, 0.0013626305495790803, 0.0025766097003838695, 0.0004542099034807734, 0.00042944110373355215, 0]
22 [0, 0.0010219729121843102, 0.002898672067175312, 0.0006813150650053501, 0.0009662332528961315, 0.00011355247587019335, 0]
23 [0, 0.002174004050381484, 0.0007664795268000901, 0.0014493429564659266, 0.00025549312315398254, 0.00024155831322403287, 0]
24 [0, 0.0005748596451000676, 0.001630508229944816, 0.00038323972406550944, 0.0005435044740345063, 6.387328078849563e-05, 0]
25 [0, 0.001222881172458612, 0.000431144704324149, 0.0008152554130120837, 0.00014371489160774908, 0.00013587611850862658, 0]
26 [0, 0.00032335852824311173, 0.0009171618528737158, 0.00021557234478684906, 0.00030572094213449086, 3.592872290193727e-05, 0]
27 [0, 0.0006878713896552868, 0.00024251889065091473, 0.0004585811698192971, 8.083962837316522e-05, 7.643023553362271e-05, 0]
28 [0, 0.00018188916798818604, 0.0005159037247782945, 0.0001212594439426026, 0.0001719679691050413, 2.0209907093291304e-05, 0]
29 [0, 0.0003869277935837209, 0.00013641687495399846, 0.0002579519080233546, 4.547229130561913e-05, 4.299199227626033e-05, 0]
30 [0, 0.00010231265621549884, 0.0002901958794134462, 6.820843721771396e-05, 9.67319712130339e-05, 1.1368072826404782e-05, 0]
31 [0, 0.00021764690956008464, 7.673449196716018e-05, 0.00014509794826313697, 2.5578163924232076e-05, 2.4182992803258474e-05, 0]
32 [0, 5.755086897537014e-05, 0.00016323518858737387, 3.8367245934964106e-05, 5.44117316682281e-05, 6.394540981058019e-06, 0]
33 [0, 0.0001224263914405304, 4.3163151695065614e-05, 8.161759589801454e-05, 1.438771721953454e-05, 1.3602932917057024e-05, 0]
34 [0, 3.237236377129921e-05, 9.181979478364352e-05, 2.158157583841731e-05, 3.0606598662296404e-05, 3.596929304883635e-06, 0]
35 [0, 6.886484608773264e-05, 2.4279272821637785e-05, 4.590989769263318e-05, 8.093090938267054e-06, 7.651649665574101e-06, 0]
36 [0, 1.8209454616228338e-05, 5.164863479140805e-05, 1.2139636409109737e-05, 1.721621167233887e-05, 2.0232727345667635e-06, 0]
37 [0, 3.8736476093556034e-05, 1.3657090960889387e-05, 2.5824317452106162e-05, 4.552363653202507e-06, 4.304052918084717e-06, 0]
38 [0, 1.024281822066704e-05, 2.905235711246863e-05, 6.8285454801242264e-06, 9.68411905159008e-06, 1.1380909133006267e-06, 0]
39 [0, 2.1789267834351473e-05, 7.682113665259929e-06, 1.4526178566809715e-05, 2.560704555006527e-06, 2.42102976289752e-06, 0]
40 [0, 5.7615852489449464e-06, 1.6341950883695153e-05, 3.841056832569877e-06, 5.447316963875568e-06, 6.401761387516317e-07, 0]
41 [0, 1.2256463162771364e-05, 4.3211889366636434e-06, 8.170975443830463e-06, 1.4403963122061928e-06, 1.361829240968892e-06, 0]
42 [0, 3.2408917024977326e-06, 9.192347373565689e-06, 2.1605944683205558e-06, 3.0641157916842847e-06, 3.600990780515482e-07, 0]
43 [0, 6.894260530174267e-06, 2.43066877686485e-06, 4.596173687154636e-06, 8.102229256188002e-07, 7.660289479210712e-07, 0]
44 [0, 1.8230015826486375e-06, 5.170695397909544e-06, 1.2153343884303127e-06, 1.7235651327294625e-06, 2.0255573140470004e-07, 0]
45 [0, 3.878021548432158e-06, 1.3672511869848939e-06, 2.5853476990244826e-06, 4.557503956611032e-07, 4.308912831823656e-07, 0]
46 [0, 1.0254383902386705e-06, 2.9085161613764014e-06, 6.836255934920509e-07, 9.695053871428948e-07, 1.139375989152758e-07, 0]
47 [0, 2.181387121032301e-06, 7.690787926787058e-07, 1.4542580807012715e-06, 2.5635959755946957e-07, 2.423763467857237e-07, 0]
48 [0, 5.768090945090294e-07, 1.6360403407840288e-06, 3.8453939633927863e-07, 5.453467802646107e-07, 6.408989938986739e-08, 0]
49 [0, 1.2270302555880217e-06, 4.3260682088171637e-07, 8.180201703944651e-07, 1.442022736272202e-07, 1.3633669506615268e-07, 0]
50 [0, 3.2445511566128725e-07, 9.202726916928543e-07, 2.1630341044084423e-07, 3.067575638982308e-07, 3.605056840680505e-08, 0]

Plot



- ***Lax -Wendroff***

Results

Time	Values
0	[0.0, 0.0625, 0.25, 0.5625, 1.0, 0, 0]
1	[0, 0.015625, 0.140625, 0.390625, 0.9609375, 0.375, 0]
2	[0, -0.005859375, 0.0625, 0.2255859375, 0.8203125, 0.6416015625, 0]
3	[0, -0.01220703125, 0.0164794921875, 0.090087890625, 0.61962890625, 0.788818359375, 0]
4	[0, -0.0112152099609375, -0.00347900390625, -0.0037078857421875, 0.39990234375, 0.823974609375, 0]
5	[0, -0.0079765319824221875, -0.006351470947265625, -0.054073333740234375, 0.1955394744873047, 0.7679443359375, 0]
6	[0, -0.005188465118408203, -0.000995635986328125, -0.06737923622131348, 0.030384063720703125, 0.6492855548858643, 0]
7	[0, -0.0037668943405151367, 0.005730003118515015, -0.054705798625946045, -0.08363986015319824, 0.49835819005966187, 0]
8	[0, -0.0035414211452007294, 0.00972314178943634, -0.028425615280866623, -0.14553934335708618, 0.34240369498729706, 0]

9 [0, -0.0038714585825800896, 0.00951752532273531, 0.0005193846300244331, -0.16261457512155175, 0.20222551748156548, 0]

10 [0, -0.004093284602276981, 0.0056214239448308945, 0.024285432358738035, -0.14704435179010034, 0.0906886724405922, 0]

11 [0, -0.0037726414448115975, -0.00035459281207295135, 0.038702652222127654, -0.11251231076312251, 0.012874872409156524, 0]

12 [0, -0.002785156982099579, -0.006518516678625019, 0.0429580557074587, -0.07148009754018858, -0.03253596222930355, 0]

13 [0, -0.001274053151746557, -0.011303078340688444, 0.038709110218633214, -0.03343380698618148, -0.05120700824954838, 0]

14 [0, 0.00045734492877613775, -0.013793717464750443, 0.02897240415948943, -0.0041585628764551075, -0.05094293380697934, 0]

15 [0, 0.0020672233796759087, -0.01379533427020796, 0.017076479429892544, 0.014113596128339623, -0.03976666143390517, 0]

16 [0, 0.0032748343185329265, -0.011705851864014072, 0.00586990970504897, 0.021959709561702567, -0.02453239752730152, 0]

17 [0, 0.003919357221901454, -0.008285064741691828, -0.0027322258654313702, 0.02173754800158298, -0.010164407059837677, 0]

18 [0, 0.003975151009137569, -0.004402511364876904, -0.007873262177405835, 0.01654912718413018, 0.0005282752057153596, 0]

19 [0, 0.0035316771774627897, -0.0008270441230553605, -0.009624529292899489, 0.009393337670856027, 0.006602129098335337, 0]

20 [0, 0.0027521383984790123, 0.001907162010869462, -0.00870270572467738, 0.0026105386310127945, 0.008474098450322513, 0]

21 [0, 0.0018257085475005764, 0.003550261623166399, -0.006138160868308586, -0.002364872979784736, 0.007334525824371683, 0]

22 [0, 0.0009254987077296325, 0.004114607031226089, -0.0029766634200709477, -0.004992280788500732, 0.004614067000859486, 0]

23 [0, 0.00017979815189396323, 0.003805100216327047, -6.548482978083604e-05, -0.00543721774900959, 0.0015884449549568398, 0]

24 [0, -0.00034078891312040846, 0.0029294350729281264, 0.002057451177413214, -0.004301025742294611, -0.0008476229396609664, 0]

25 [0, -0.0006217710689563222, 0.00181209906509929, 0.003179254753194784, -0.002348272247733382, -0.002248601858106204, 0]

26 [0, -0.0006928406848546529, 0.0007285033038164988, 0.0033575122452749943, -0.0002879084210887171, -0.0025670534864796713, 0]

27 [0, -0.0006106934266180521, -0.00013312680961749497, 0.0028273114755235224, 0.0013640174619715438, -0.0020332557727680226, 0]

28 [0, -0.00044137921876135223, -0.0006822690766353311, 0.0019000588702896384, 0.0023374118713959816, -0.001013435281336688, 0]

29 [0, -0.0002457507794915978, -0.0009147263732982102, 0.000877016765054482, 0.0025922603900726863, 0.00011645299077097716, 0]

30 [0, -6.99722879564221e-05, -0.0008878284179148171, -9.292364955053149e-06, 0.0022585199556035733, 0.0010594373893554902, 0]

31 [0, 5.849933627203558e-05, -0.0006909493758003895, -0.000622219924884793, 0.0015579756561750987, 0.0016415230253679577, 0]

32 [0, 0.00013024317417907537, -0.0004184972901376796, -0.0009205179166106279, 0.000729958892128532, 0.0018153831400916304, 0]

33 [0, 0.00014999454190151647, -0.00014996703770977794, -0.0009385697827756673, -2.4647942144040202e-05, 0.0016352719396169222, 0]

34 [0, 0.0001312417861398596, 6.109389777769364e-05, -0.0007570839834549121, -0.0005748586176010207, 0.0012172109764086766, 0]

35 [0, 9.0794602382683e-05, 0.00018967159106758158, -0.00047304544872442134, -0.0008672018290474442, 0.0006973362507061247, 0]

36 [0, 4.438700290356455e-05, 0.000235432350284745, -0.00017525701126204268, -0.0009149604463955051, 0.00019780150213680279, 0]

37 [0, 3.861208392080287e-06, 0.0002151265152101508, 7.121442870968548e-05, -0.0007766669017869952, -0.00019475904079571235, 0]

38 [0, -2.3994908107208633e-05, 0.0001538910359659325, 0.00023116662745944504, -0.0005314498854746503, -0.00043731936876690745, 0]

39 [0, -3.723256057614805e-05, 7.752435800181551e-05, 0.0002975153447661397, -0.0002572350077128324, -0.0005272832336281745, 0]

40 [0, -3.761496518233797e-05, 6.99164018953865e-06, 0.0002843625187893896, -1.5447597293800103e-05, -0.000491925553113443, 0]

41 [0, -2.908517891044581e-05, -4.440719664989646e-05, 0.0002178247038248442, 0.00015654094071485138, -0.00037473701382025726, 0]

42 [0, -1.62629846015973e-05, -7.144042755694504e-05, 0.00012714821153556554, 0.00024593209619798727, -0.00022234990759712367, 0]

43 [0, -3.2671850065798447e-06, -7.557246633525345e-05, 3.782948629307134e-05, 0.000259923389923968, -7.453789462359751e-05, 0]

44 [0, 6.9961695369718e-06, -6.263322991554145e-05, -3.2457983896412535e-05, 0.00021844583663082744, 4.156785025378987e-05, 0]

45 [0, 1.3076280892171532e-05, -4.029411087324009e-05, -7.513667871949088e-05, 0.00014646665223024214, 0.00011309307642690269, 0]

46 [0, 1.4843974528283662e-05, -1.592489298042938e-05, -8.977113214586346e-05, 6.753710009950968e-05, 0.00013974480190651783, 0]

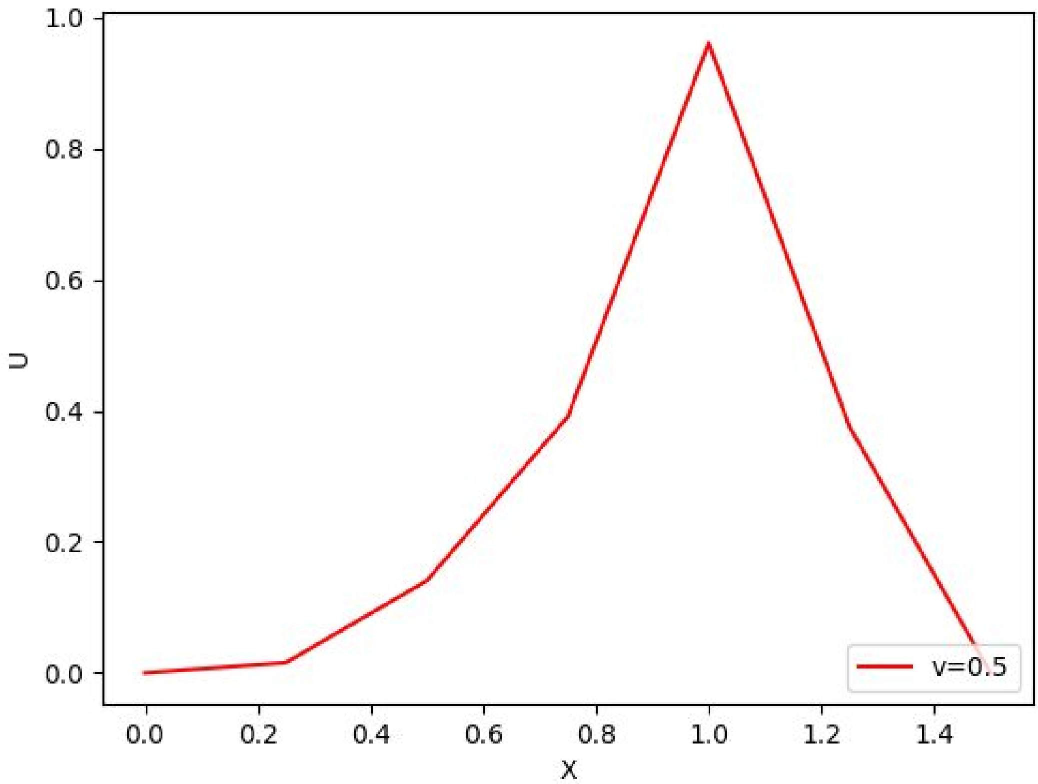
47 [0, 1.3123592518766418e-05, 4.844212231017272e-06, -8.174232148949731e-05, -4.794497183812666e-07, 0.0001301350139672045, 0]

48 [0, 9.237167860197654e-06, 1.8772296553987526e-05, -5.943023031569384e-05, -4.7279834593248006e-05, 9.74214668310104e-05, 0]

49 [0, 4.581338825899799e-06, 2.4971939152526494e-05, -3.1623082204869057e-05, -6.99238956671975e-05, 5.5336162150789805e-05, 0]

50 [0, 3.1451172535903756e-07, 2.439984169971593e-05, -5.612347513054671e-06, -7.121859784607275e-05, 1.528066073789329e-05, 0]

Plot



- ***Mac-Cormack***

Results

```
        Time                          Values

        0                  [0.0, 0.0625, 0.25, 0.5625, 1.0, 0, 0]
        1                  [0, 0.0234375, 0.154296875, 0.40576171875, 0.9764404296875, 0.244110107421875, 0]
        2                  [0, 0.001220703125, 0.0845947265625, 0.2541351318359375, 0.8874053955078125,
0.43544769287109375, 0]
        3                  [0, -0.0095062255859375, 0.03987693786621094, 0.12141180038452148, 0.7524017095565796,
0.5691171586513519, 0]
        4                  [0, -0.01330256462097168, 0.016390204429626465, 0.016282662749290466, 0.5912825167179108,
0.6457981429994106, 0]
        5                  [0, -0.013688519597053528, 0.008883966132998466, -0.05744199315086007, 0.4222869359655306,
0.6706451091158669, 0]
        6                  [0, -0.013087950414046645, 0.011681731906719506, -0.10012717802601401, 0.2606386358238524,
0.6519741294323467, 0]
        7                  [0, -0.012912173100630753, 0.01950936939647363, -0.1153137679016254, 0.11773359819142115,
0.5999107628011586, 0]
        8                  [0, -0.013736822637611112, 0.028050713550214823, -0.10860356830029616, 0.0008771609922746393,
0.5251412076990825, 0]
        9                  [0, -0.015526059001686576, 0.034238305643553346, -0.08657819097590513, -0.08651968283812128,
0.43786863602716686, 0]
```

10	[0, -0.017865089831919923, 0.03631451885211734, -0.0558623270361225, -0.1444038837457826, 0.34703408558732535, 0]
11	[0, -0.0201712684594446, 0.03371517776025683, -0.02240025624832015, -0.17533272303805547, 0.2598216441293958, 0]
12	[0, -0.021864257122046128, 0.026834748290753215, 0.009025053235165108, -0.18363757486568177, 0.18143454489680083, 0]
13	[0, -0.022485568518134512, 0.016730880970479797, 0.035034338681599636, -0.17460361144917175, 0.11510432392240781, 0]
14	[0, -0.02176623257467767, 0.004818670370300451, 0.05368516537012125, -0.15374490916579595, 0.06228005614065785, 0]
15	[0, -0.01964778729913052, -0.0074062559220348925, 0.06434106936407186, -0.1262265351966357, 0.02293841532391669, 0]
16	[0, -0.016266031896484845, -0.018589615576410725, 0.06742934869903966, -0.09645818303778593, -0.00404343235101938, 0]
17	[0, -0.011909075962372898, -0.02767185120733256, 0.0641399901895498, -0.06786048356679783, -0.020503124198841413, 0]
18	[0, -0.006961460066159716, -0.03397073359664965, 0.05611236846254339, -0.04278694048045708, -0.028636968794100506, 0]
19	[0, -0.0018449358583085451, -0.0371996719194635, 0.04514677198491673, -0.022572258824908195, -0.03070041240106499, 0]
20	[0, 0.0030356401139129607, -0.03743414939916692, 0.03296642049012393, -0.007671569799130567, -0.02878075330071451, 0]
21	[0, 0.007335453774569705, -0.035041819841894126, 0.021044109193276227, 0.0021459978866691246, -0.024646659666457918, 0]
22	[0, 0.010798749532985258, -0.03059241862757057, 0.010497241151390413, 0.007582890896990326, -0.0196701044839031, 0]
23	[0, 0.013272958169808422, -0.02476228190059595, 0.002046654170193833, 0.00960545613790288, -0.014809977388939494, 0]
24	[0, 0.014709123636156864, -0.01824554752525647, -0.003971246499632373, 0.009263209669374361, -0.010642927797978466, 0]
25	[0, 0.015151176622294316, -0.011680654116571789, -0.007552905424993069, 0.007547448079201607, -0.007425699803430755, 0]
26	[0, 0.014717361309079, -0.0055971188466064315, -0.008951502968420743, 0.005294353802625065, -0.005173898877345644, 0]
27	[0, 0.01357733100126993, -0.0003842083694105521, -0.008590411415048922, 0.003131694083202907, -0.0037442379968767115, 0]
28	[0, 0.011928190672287509, 0.003719666771718759, -0.006978155055638479, 0.0014637233085025127, -0.002910277420141494, 0]
29	[0, 0.009972208491786726, 0.006620029930155405, -0.004633843604707631 5, 0.00048608167128047714, -0.002424972324803688, 0]
30	[0, 0.00789817868904396, 0.008346301311735423, -0.0020287980350953815, 0.000221243494197033, -0.002066539910653969, 0]
31	[0, 0.005867618688946537, 0.009023518074392053, 0.00045302580111492496, 0.0005651619965328812, -0.0016669319226890024, 0]
32	[0, 0.0040062265935292135, 0.008840506738335984, 0.0025358790109929453, 0.0013368529900506325, -0.0011243521848402188, 0]
33	[0, 0.002400384927046064, 0.008018554751431383, 0.004056426198720344, 0.002324396939079417, -0.0004027089269653373, 0]
34	[0, 0.001098017467236383, 0.006783686499471513, 0.004954744931363253, 0.00332287217040597, 0.0004783477315068223, 0]
35	[0, 0.00011280447139789596, 0.005344583688466642, 0.005256188715758761, 0.004161766861606561, 0.0014589959804701099, 0]
36	[0, -0.0005693690485851714, 0.003877144875792174, 0.005048230487536139, 0.004721229128231012, 0.002456928764969099, 0]
37	[0, -0.0009828410269860468, 0.0025157626986296233, 0.004455988710222651, 0.004937956569136662, 0.003384301811632127, 0]
38	[0, -0.001174456235941494, 0.0013506797135377153, 0.0036194154786871663, 0.004802528141212355, 0.004161896120481200 5, 0]
39	[0, -0.0011964841706410215, 0.0004302967718493497, 0.002674246719162064, 0.004350536788291177, 0.004729293302493844, 0]
40	[0, -0.0011007107457920626, -0.00023294885097509266, 0.0017379115679866358, 0.003650035918939708, 0.005050640619417041, 0]
41	[0, -0.0009340032961961684, -0.0006545700146505777, 0.0009007756284581984, 0.002787645258759664, 0.005116221856679827, 0]
42	[0, -0.0007354316323403251, -0.0008692036244616116, 0.0002224221114405627, 0.001855267397189868, 0.004940510973892316, 0]
43	[0, -0.000534852225240083, -0.0009220689916440011, -0.0002678063250492414, 0.0009388435195422848, 0.004557657982041348, 0]
44	[0, -0.0003527370731295725, -0.0008615188453397389, -0.0005670656856958066, 0.00011001441042037907, 0.004015454336891274, 0]
45	[0, -0.00020095508332090855, -0.0007331845497905228, -0.0006932304137340088, -0.0005789767864270799, 0.0033687783481730947, 0]
46	[0, -8.418762918197964e-05, -0.0005759295866454513, -0.0006781869103752356, -0.0010972487092391406, 0.002673368877341672, 0]
47	[0, -1.6729772035507784e-06, -0.0004195832688187531, -0.0005611532751281269, -0.0014345520490339889, 0.001980559755415466, 0]
48	[0, 5.098405354923721e-05, -0.0002842451874380838, -0.0003827514064673834, -0.0015984908639485192, 0.001333367070001403, 0]
49	[0, 8.014169528534302e-05, -0.00018083518937856463, -0.00018030492001003671, -0.0016104266197076388, 0.0007640895313243177, 0]
50	[0, 9.272838204699572e-05, -0.00011251058019324053, 1.540887740636261e-05, -0.0015007822643081329, 0.0002933827738317448, 0]

Plot