Assignment - 1

ARNAB BIR - 14MA20009

Thomas Algorithm:

Thomas algorithm is an efficient way of solving tridiagonal matrix systems. It is based on LU decomposition. Thomas algorithm consists of two steps.

Step 1 | Matrix Decomposition:
The initial form of the coefficient matrix is,

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & 0 \\ 0 & 0 & a_4 & b_4 & c_4 & 0 \\ 0 & 0 & 0 & a_5 & b_5 & c_5 \\ 0 & 0 & 0 & 0 & a_6 & b_6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{pmatrix}$$

In the first stage the matrix equation is converted to the following form .

$$\begin{pmatrix} 1 & \gamma_1 & 0 & 0 & 0 & 0 \\ 0 & 1 & \gamma_2 & 0 & 0 & 0 \\ 0 & 0 & 1 & \gamma_3 & 0 & 0 \\ 0 & 0 & 0 & 1 & \gamma_4 & 0 \\ 0 & 0 & 0 & 0 & 1 & \gamma_5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \rho_4 \\ \rho_5 \\ \rho_6 \end{pmatrix}$$

Step 2 | Back Substitution:
In the next step, back substitution is done and the gamma value becomes 0 and we get the final solution.

Problem 1:

$$x^2y'' + xy' = 1$$
$$y(1) = 0, y(1.4) = 0.0566$$

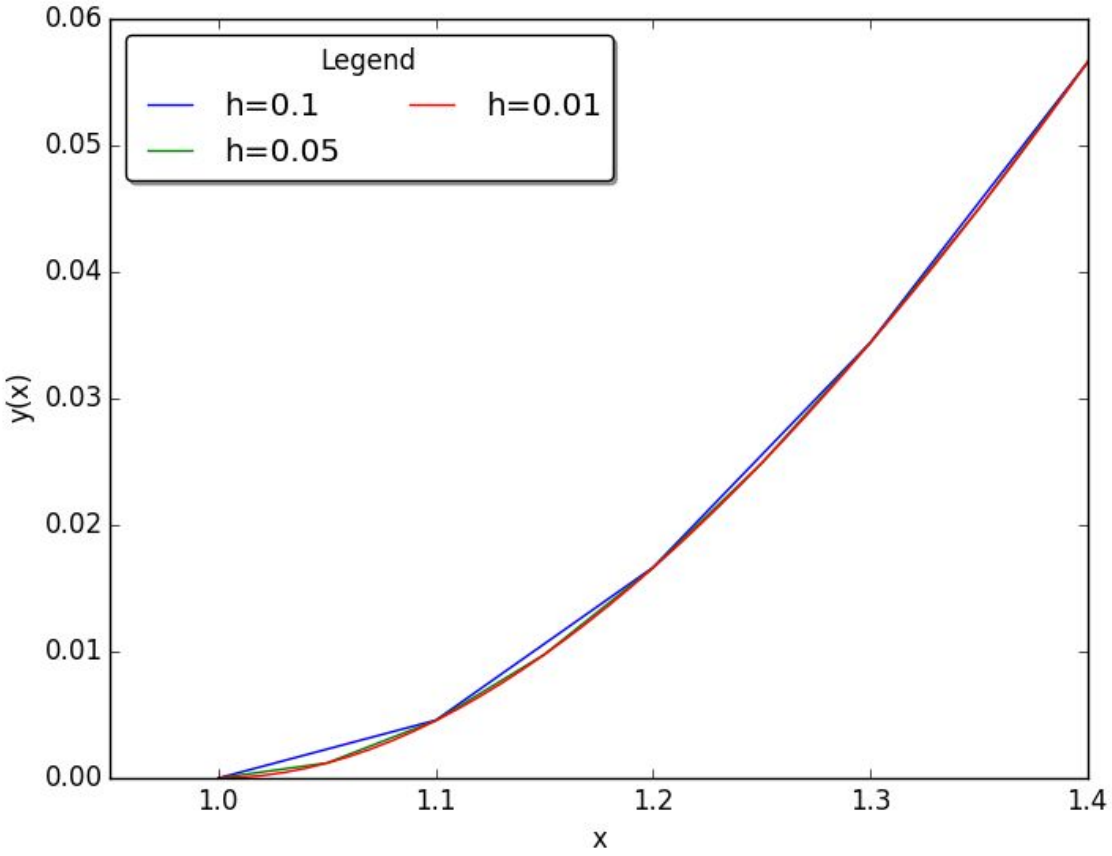
Code:

```
#Name : Arnab Bir
#Roll No. : 14MA20009

import sys
import numpy as np
import math
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

# x^2y'' + xy' = 1
#The equation can be rewritten as y'' + 1/x y' = 1/x^2
# Hence A(x) = 1/x ; B(x) = 0 ; C(x) = 1/x^2

def A(x):
     return 1 / x
def B(x):
     return 0
def C(x):
     return 1/x**2

def get_a(x, h):
     return 1/h**2 - A(x)/(2 * h)
def get_b(x, h):
     return - 2/h**2 + B(x)
def get_c(x, h):
     return 1/h**2 + A(x)/(2 * h)

def ThomasAlgorithm(a, b, c, d, n):
      c_dash = np.zeros(n-1)
      d_dash = np.zeros(n-1)
      c_dash[0] = c[0] / b[0]
      d_dash[0] = d[0] / b[0]
```

```
      for itr in xrange(1, n-1):
          c_dash[itr] = c[itr] / (b[itr] - a[itr] *
c_dash[itr-1])
          d_dash[itr] = (d[itr] - a[itr]*d_dash[itr-1]) /
(b[itr] - a[itr] * c_dash[itr-1])
      y = np.zeros(n-1)
      y[n-2] = d_dash[n-2]
      for itr in reversed(xrange(n-2)):
          y[itr] = d_dash[itr] - c_dash[itr] * y[itr+1]
      return y

def TridiagonalBVP(x0, y0, xn, yn, h, n):
     x = [(x0 + itr * h) for itr in xrange(1, n)]
     #print x
     a = [get_a(itr, h) for itr in x]
     b = [get_b(itr, h) for itr in x]
     c = [get_c(itr, h) for itr in x]
     d = np.zeros(n-1)
     d[0] = C(x[0]) - a[0]* y0
     for itr in xrange(1,n-2):
          d[itr] = C(x[itr])
     d[n-2] = C(x[n-2]) - c[n-2] * yn
     #print a, b, c, d
     return ThomasAlgorithm(a, b, c, d, n)

def main():
     # h = 0.1, 0.05, 0.01
     #h = [0.1, 0.05, 0.01]
     stepsizes = [0.1, 0.05, 0.01]
     # Boundary conditions y(1) = 0, y(1.4) = 0.0566
     x0 = 1.0
     xn = 1.4
     y0 = 0.0
     yn = 0.0566
     for step in stepsizes:
          file = open("Resut_h_" + str(step) + ".txt", 'w')
          n = int(math.ceil((xn - x0) / step))
          x = [(x0 + step * itr) for itr in xrange(n+1)]
          y = np.zeros(n+1)
```

```
          y[0] = y0
          y[1:n] = TridiagonalBVP(x0, y0, xn, yn, step, n)
          y[n] = yn
          file.write("Value of y(x) with respect to
x:\n\n\tx\ty(x)\n\n")
          for i in xrange(n+1):
               file.write("\t" + str(x[i]) + "\t" + str(y[i]) +
"\n")
          file.close()
          plt.plot(x, y,  label="h={0}".format(step))
          plt.xlabel('x')
          plt.ylabel('y(x)')
          plt.savefig('Plot_h_' + str(step) + '.png')

     plt.legend(bbox_to_anchor=(2,2),
bbox_transform=plt.gcf().transFigure)
     plt.legend(loc="upper left", bbox_to_anchor=[0, 1],
           ncol=2, shadow=True, title="Legend", fancybox=True)
    #plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
bbox_transform=plt.gcf().transFigure)
     plt.savefig('Plot_h.png')
     plt.show()
main()
```

Plot:



Output:

- **For h = 0.01;**

Value of y(x) with respect to x:

x	**y(x)**
1.0	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

- **For h = 0.05;**

Value of y(x) with respect to x:

x	y(x)
1.0	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

- **For h = 0.05;**

Value of y(x) with respect to x:

x	y(x)
1.0	0.0
1.1	0.00457418107894
1.2	0.0166557456214
1.3	0.0344374516671
1.4	0.0566

Problem 2:

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

$$y(0) - y'(1) = 0$$
$$2y(1) - y'(1) = 1$$

Code:

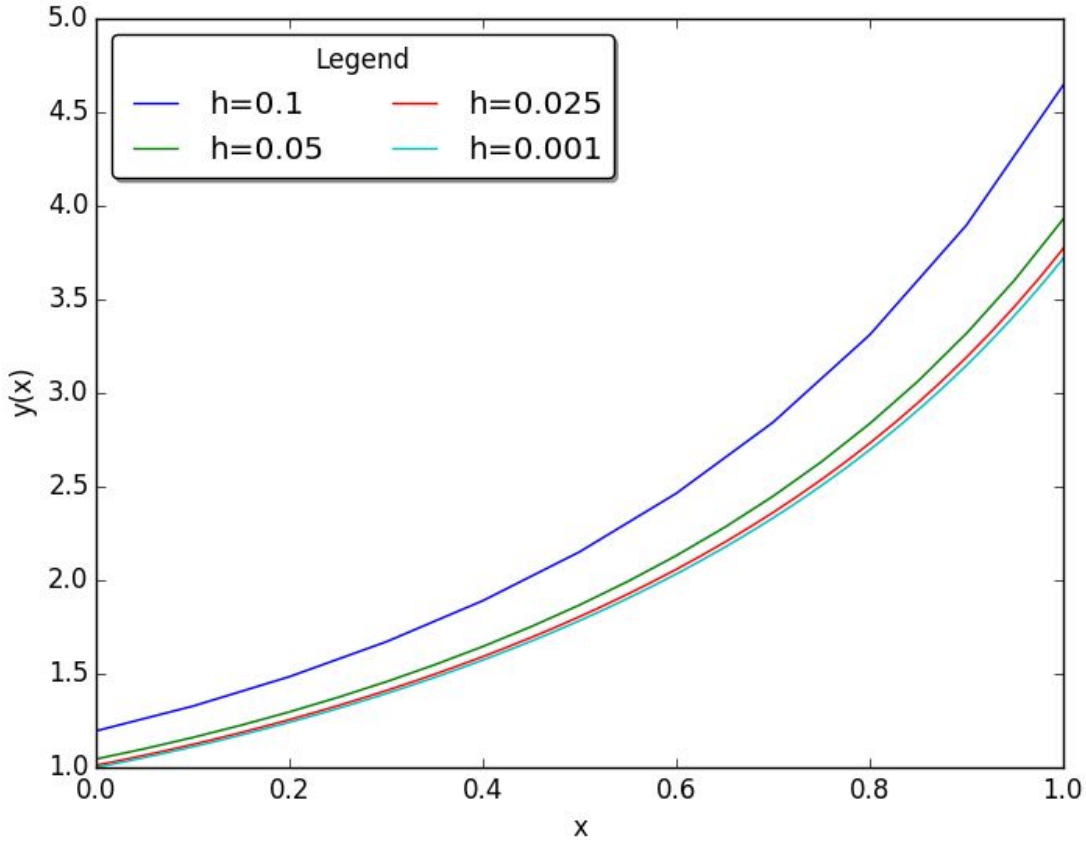
```
#Name : Arnab Bir
#Roll No. : 14MA20009

import sys
```

```
import numpy as np
import math
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

# y'' - 2xy' - 2y = -4x
# Hence A(x) =  -2x ; B(x) = - 2 ; C(x) = -4x

def A(x):
     return -2 * x
def B(x):
     return -2
def C(x):
     return -4 * x

def get_a(x, h):
     return 1/(h**2) - A(x)/(2.0 * h)
def get_b(x, h):
     return - 2/(h**2) + B(x)
def get_c(x, h):
     return 1/(h**2) + A(x)/(2.0 * h)

def ThomasAlgorithm(a, b, c, d, n):
      c_dash = np.zeros(n-1)
      d_dash = np.zeros(n-1)
      c_dash[0] = c[0] / b[0]
      d_dash[0] = d[0] / b[0]
      for itr in xrange(1, n-1):
          c_dash[itr] = c[itr] / (b[itr] - a[itr] *
c_dash[itr-1])
          d_dash[itr] = (d[itr] - a[itr]*d_dash[itr-1]) /
(b[itr] - a[itr] * c_dash[itr-1])
      y = np.zeros(n-1)
      y[n-2] = d_dash[n-2]
      for itr in reversed(xrange(n-2)):
          y[itr] = d_dash[itr] - c_dash[itr] * y[itr+1]
      return y

def TridiagonalBVP(x0, xn, h, n):
```

```
    x = [(x0 + itr * h) for itr in xrange(1, n)]
    #print x[n-1]
    a = [get_a(itr, h) for itr in x]
    b = [get_b(itr, h) for itr in x]
    c = [get_c(itr, h) for itr in x]
    d = [C(itr) for itr in x]

    b[0] += 4 / (2*h + 3) * a[0]
    c[0] += (-1) / (2*h + 3) * a[0]

    b[n-2] += 4 / (3 - 4*h) * c[n-2]
    a[n-2] += (-1) / (3 - 4*h) * c[n-2]
    d[n-2] += (2*h) / (3 - 4*h) * c[n-2]

    return ThomasAlgorithm(a, b, c, d, n)

def main():
    # h = 0.1, 0.05, 0.01
    #h = [0.1, 0.05, 0.01]
    stepsizes = [0.1, 0.05, 0.025, 0.001]
    # Boundary conditions y(1) = 0, y(1.4) = 0.0566
    x0 = 0.0
    xn = 1.0

    for step in stepsizes:
        file = open("Resut_h_" + str(step) + ".txt", 'w')
        n = int(math.ceil((xn - x0) / step))
        x = [(x0 + step * itr) for itr in xrange(n+1)]
        y = [0 for itr in xrange(n+1)]
        y[1:n] = TridiagonalBVP(x0, xn, step, n)
        y[0] = 4 / (2*step + 3) * y[1] - 1 / (2*step + 3) *
y[2]
        y[n] = (4 * y[n-1] - y[n-2] - 2*step) / (3 - 4*step)
        #print y
        file.write("Value of y(x) with respect to
x:\n\n\tx\ty(x)\n\n")
        for i in xrange(n+1):
            file.write("\t" + str(x[i]) + "\t" + str(y[i]) +
"\n")
```

```
            file.close()
            plt.plot(x, y,  label="h={0}".format(step))
            plt.xlabel('x')
            plt.ylabel('y(x)')
            plt.savefig('Plot_h_' + str(step) + '.png')
      #red_patch = mpatches.Patch(color='red', label='h = 0.01')
      #green_patch = mpatches.Patch(color='green', label='h =
0.05')
      #blue_patch = mpatches.Patch(color='blue', label='h = 0.1')
      #plt.legend(handles=[red_patch, blue_patch, green_patch])
      plt.legend(bbox_to_anchor=(2,2),
bbox_transform=plt.gcf().transFigure)
      plt.legend(loc="upper left", bbox_to_anchor=[0, 1],
            ncol=2, shadow=True, title="Legend", fancybox=True)
     #plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
bbox_transform=plt.gcf().transFigure)
      plt.savefig('Plot_h.png')
      plt.show()
main()
```

Plot:



Output:

- **For h = 0.001;**

Value of y(x) with respect to x:

x	y(x)
0.0	1.00001897407
0.001	1.00101999307
0.002	1.00202301211
0.003	1.00302803121

0.004	1.00403505041
0.005	1.00504406974
0.006	1.00605508925
0.007	1.00706810902
0.008	1.00808312913
0.009	1.00910014965
0.01	1.01011917071
0.011	1.0111401924
0.012	1.01216321485
0.013	1.01318823821
0.014	1.01421526262
0.015	1.01524428824
0.016	1.01627531526
0.017	1.01730834385
0.018	1.01834337422
0.019	1.01938040656
0.02	1.02041944112
0.021	1.02146047811
0.022	1.0225035178
0.023	1.02354856042
0.024	1.02459560626
0.025	1.0256446556
0.026	1.02669570873
0.027	1.02774876596
0.028	1.02880382761
0.029	1.029860894
…	…
…	…
……..	……..
0.98	3.59282673939
0.981	3.59895553782
0.982	3.60509964989
0.983	3.61125912629
0.984	3.61743401788
0.985	3.62362437575
0.986	3.62983025116
0.987	3.63605169557
0.988	3.64228876063
0.989	3.64854149819
0.99	3.6548099603
0.991	3.66109419921
0.992	3.66739426735
0.993	3.67371021737
0.994	3.68004210211
0.995	3.68638997461
0.996	3.69275388812
0.997	3.69913389609

0.998	3.70553005216
0.999	3.71194241018
1.0	3.71837102422

- **For h = 0.025;**

Value of y(x) with respect to x:

x	y(x)
0.0	1.01130026701
0.025	1.03721637749
0.05	1.06439969559
0.075	1.09285806541
0.1	1.12260414116
0.125	1.15365546558
0.15	1.18603457945
0.175	1.21976916291
0.2	1.25489220996
0.225	1.29144223765
0.25	1.32946353181
0.275	1.36900643161
0.3	1.41012765529
0.325	1.45289067033
0.35	1.49736611117
0.375	1.54363224845
0.4	1.59177551419
0.425	1.64189108772
0.45	1.69408354797
0.475	1.7484675984
0.5	1.80516887155
0.525	1.86432482107
0.55	1.92608571012
0.575	1.99061570597
0.6	2.05809409195
0.625	2.12871660909
0.65	2.20269694151
0.675	2.28026836109
0.7	2.36168554886
0.725	2.44722661297
0.75	2.53719532505
0.775	2.63192359978
0.8	2.7317742456
0.825	2.83714401759
0.85	2.94846700781
0.875	3.06621841269
0.9	3.19091872183

0.925	3.32313837861
0.95	3.46350296908
0.975	3.61269900309
1.0	3.77148035975

- **For h = 0.05;**

Value of y(x) with respect to x:

x	y(x)
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

- **For h = 0.1;**

Value of y(x) with respect to x:

x	y(x)
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