Name: Altaf Ahmad

Roll : 18MA20005

Solving differential equations using the block tridiagonal System of Equations

1. Solve the question by block tridiagonal system of equations:
   $y''' + 4y'' + y' - 6y = 1, 0 < x < 1$
   $y(0) = y'(0) = 0, y'(1) = 1$

Ans : Here, we assume $p = y'$, so the equation becomes

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

Now, Discretizing these set of equations, we get :

$$y_i - y_{i-1} - \frac{h(p_i + p_{i-1})}{2} = 0$$

$$\frac{p_{i+1} - 2p_i + p_{i-1}}{h^2} + 4 \times \frac{p_{i+1} - p_{i-1}}{2h} + p_i - 6y_i = 1$$

$$\Rightarrow \left(\frac{1}{h^2} - \frac{2}{h}\right)p_{i-1} + \left(1 - \frac{2}{h^2}\right)p_i + \left(\frac{1}{h^2} + \frac{2}{h}\right)p_{i+1} - 6y_i = 1$$

Now, let $X_i = \begin{bmatrix} y_i \\ p_i \end{bmatrix}$. So, the given system of equations become :

$$\begin{bmatrix} -1 & -h/2 \\ 0 & (1/h^2 - 2/h) \end{bmatrix} X_{i-1} + \begin{bmatrix} 1 & -h/2 \\ -6 & 1 - 2/h^2 \end{bmatrix} X_i + \begin{bmatrix} 0 & 0 \\ 0 & (1/h^2 + 2/h) \end{bmatrix} X_{i+1} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\Rightarrow A_i^* X_{i-1} + B_i^* X_i + C_i^* X_{i+1} = D_i^* \ \forall, \ i = 1, 2, \ldots, n-1$$

We can now solve this system of equation using a modification of the Thomas Algorithm.

In [1]:
```
import numpy as np
```

First of all we need to define the differential equation and in order to do that, we are using the form

$$y''' + A(x)y'' + B(x)y' + C(x)y = D(x)$$

So, we first need to define these equations for our problem

In [2]:
```
def A(x):
    return 4
def B(x):
    return 1
def C(x):
    return -6
def D(x):
    return 1
```

Now, we have to check for the value of $h \& x_0, x_n, n$

In [6]:
```
h = 0.2
x_0 = 0
x_n = 1
n = int((x_n- x_0)/h  + 0.5 )
x = np.array([])
for i in range(n+1):
    x = np.append(x, i*h)
```

Now, we define our matrices $A_i^*, B_i^*, C_i^* \& D_i^*$

In [7]:
```
A_star = np.array([ [[-1,-h/2],[0,((2 - A(x[0])*h)/(2*h*h) )]] ])
B_star = np.array([ [[ 1,-h/2],[C(x[0]),B(x[0]) - ((2)/(h*h))]] ])
C_star = np.array([ [[0 ,0],[0,((2 + A(x[0])*h)/(2*h*h) )]] ])
D_star = np.array([ [[0],[D(x[0])]] ])
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
for i in range(1,n+1):
    A_star = np.append(A_star,[ [[-1,-h/2],[0,((2 - A(x[i])*h)/(2*h*h
) )]] ], axis=0)
    B_star = np.append(B_star,[ [[ 1,-h/2],[C(x[0]),B(x[i]) - ((2)/(h
*h))]] ], axis=0)
    C_star = np.append(C_star,[ [[0 ,0],[0,((2 + A(x[i])*h)/(2*h*h)
)]] ], axis=0)
    D_star = np.append(D_star,[ [[0],[D(x[i])]] ], axis=0)
    print(i)
print("Now, after running the loop")
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
```

```
A_star =
[[[-1.  -0.1]
  [ 0.  15. ]]]
B_star =
[[[  1.   -0.1]
  [ -6.  -49. ]]]
C_star =
[[[ 0.  0.]
  [ 0. 35.]]]
D_star =
[[[0]
  [1]]]
1
2
3
4
5
Now, after running the loop
A_star =
[[[-1.  -0.1]
  [ 0.  15. ]]

 [[-1.  -0.1]
  [ 0.  15. ]]

 [[-1.  -0.1]
  [ 0.  15. ]]

 [[-1.  -0.1]
  [ 0.  15. ]]

 [[-1.  -0.1]
  [ 0.  15. ]]

 [[-1.  -0.1]
  [ 0.  15. ]]]
B_star =
[[[  1.   -0.1]
  [ -6.  -49. ]]

 [[  1.   -0.1]
  [ -6.  -49. ]]

 [[  1.   -0.1]
  [ -6.  -49. ]]

 [[  1.   -0.1]
  [ -6.  -49. ]]

 [[  1.   -0.1]
  [ -6.  -49. ]]

 [[  1.   -0.1]
  [ -6.  -49. ]]]
C_star =
[[[ 0.  0.]
  [ 0. 35.]]
```

```
 [[ 0.  0.]
  [ 0. 35.]]

 [[ 0.  0.]
  [ 0. 35.]]

 [[ 0.  0.]
  [ 0. 35.]]

 [[ 0.  0.]
  [ 0. 35.]]

 [[ 0.  0.]
  [ 0. 35.]]]
D_star =
[[[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]]
```

Here, all the values of are same because it isn't dependent on x
Now, we will define the X_i matrices that will finally store the answers.

In [8]:
```
X_ans = np.array([[[0],[0]]],dtype = float)
for i in range(n):
    X_ans = np.append(X_ans, [[[0],[0]]] ,axis = 0)
```

Here, we need to add the initial conditions, i.ie
$X_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \& X_n = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

In [9]:
```
X_ans[n] = np.array([[[0],[1]]])
```

In [10]:
```
X_ans
```

Out[10]:
```
array([[[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [1.]]])
```

Now, we need to modify the value of D_1^* because we have $A_1^*X_0 + B_1^*X_1 + C_1^*X_2 = D_1^*$ and we already know the value of X_0, so it can be shifted to the RHS
$\Rightarrow B_1^*X_1 + C_1^*X_2 = D_1^* - A_1^*X_0 \ \Rightarrow D_1^* = D_1^* - A_1^*X_0$

In [11]:
```
D_star[1] = D_star[1] - np.dot(A_star[1],X_ans[0])
```

Now, we will transform the given equations into the form :

$$X_i + C_i'X_{i+1} = D_i'$$

where, $C_1' = (B_1^*)^{-1}C_1^*$ & $D_1' = (B_1^*)^{-1}D_1^*$
and $B_i' = (B_i^* - A_i^*C_{i-1}')$
$C_i' = (B_i')^{-1}C_i$
$D_i' = (B_i')^{-1}(D_i^* - A_i^*D_{i-1}')$
for $i = 2, 3, \ldots, n-1$

In [12]:
```
cc_dash = np.dot(np.linalg.inv(B_star[1]),C_star[1] )
dd_dash = np.dot(np.linalg.inv(B_star[1]),D_star[1] )
C_dash = np.array([cc_dash])
D_dash = np.array([dd_dash])
C_dash = np.append(C_dash, [cc_dash], axis = 0)
D_dash = np.append(D_dash, [dd_dash], axis = 0)
for i in range(2,n):
    b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
    b_dashinv = np.linalg.inv(b_dash)
    cc_dash = np.dot(b_dashinv,C_star[i])
    dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[
i-1]))
    C_dash = np.append(C_dash, [cc_dash], axis = 0)
    D_dash = np.append(D_dash, [dd_dash], axis = 0)
```

In [13]:
```
print(C_dash)
print(D_dash)
```

```
[[[ 0.         -0.07056452]
  [ 0.         -0.70564516]]

 [[ 0.         -0.07056452]
  [ 0.         -0.70564516]]

 [[ 0.         -0.21171782]
  [ 0.         -0.87802707]]

 [[ 0.         -0.365797  ]
  [ 0.         -0.91559   ]]

 [[ 0.         -0.50523998]
  [ 0.         -0.90649419]]]
[[[-0.00201613]
  [-0.02016129]]

 [[-0.00201613]
  [-0.02016129]]

 [[-0.01176435]
  [-0.0320662 ]]

 [[-0.02951053]
  [-0.03639253]]

 [[-0.05259415]
  [-0.0348868 ]]]
```

Now, we have $X_{n-1} = (B^*_{n-1} - A^*_{n-1}C'_{n-2})^{-1}(D^*_{n-1} - C^*_{n-1}X_n - A^*_{n-1}D'_{n-2})$

In [14]:
```
b_dash = (B_star[n-1] - np.dot(A_star[n-1],C_dash[n-2]) )
b_dashinv = np.linalg.inv(b_dash)
final_ans = np.dot(b_dashinv, D_star[n-1] - np.dot(C_star[n-1],X_ans[
n]) - np.dot(A_star[n-1], D_dash[n-2]))
X_ans[n-1] =  np.array(final_ans)
```

After that we have, $X_i = D'_i - C'_i X_{i+1} \ \forall \, i = (n-2), (n-3), \ldots, 1$

In [15]:
```
for i in range(n-2, 0,-1):
    #D_dash[i] - np.dot(C_dash[i], X_ans[i+1])
    X_ans[i] = np.array(D_dash[i] - np.dot(C_dash[i], X_ans[i+1]))
    print("X["+str(i) +"] = ")
    print(X_ans[i])
```

```
X[3] =
[[0.28932083]
 [0.76164248]]
X[2] =
[[0.14948893]
 [0.63667652]]
X[1] =
[[0.04291064]
 [0.42910641]]
```

This was for h = 0.2, now, for a different value of h, let's say 0.1 :

In [57]:
```
h = 0.1
x_0 = 0
x_n = 1
n = int((x_n- x_0)/h  + 0.5 )
print("n = " + str(n))
x = np.array([])
for i in range(n+1):
    x = np.append(x, i*h)
A_star = np.array([ [[-1,-h/2],[0,((2 - A(x[0])*h)/(2*h*h) )]] ])
B_star = np.array([ [[ 1,-h/2],[C(x[0]),B(x[0]) - ((2)/(h*h))]] ])
C_star = np.array([ [[0 ,0],[0,((2 + A(x[0])*h)/(2*h*h) )]] ])
D_star = np.array([ [[0],[D(x[0])]] ])
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
for i in range(1,n+1):
    A_star = np.append(A_star,[ [[-1,-h/2],[0,((2 - A(x[i])*h)/(2*h*h
) )]] ], axis=0)
    B_star = np.append(B_star,[ [[ 1,-h/2],[C(x[0]),B(x[i]) - ((2)/(h
*h))]] ], axis=0)
    C_star = np.append(C_star,[ [[0 ,0],[0,((2 + A(x[i])*h)/(2*h*h)
)]] ], axis=0)
    D_star = np.append(D_star,[ [[0],[D(x[i])]] ], axis=0)
    print(i)
print("Now, after running the loop")
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
X_ans = np.array([[[0],[0]]],dtype = float)
for i in range(n):
    X_ans = np.append(X_ans, [[[0],[0]]] ,axis = 0)
X_ans[n] = np.array([[[0],[1]]])
D_star[1] = D_star[1] - np.dot(A_star[1],X_ans[0])
cc_dash = np.dot(np.linalg.inv(B_star[1]),C_star[1] )
dd_dash = np.dot(np.linalg.inv(B_star[1]),D_star[1] )
C_dash = np.array([cc_dash])
D_dash = np.array([dd_dash])
C_dash = np.append(C_dash, [cc_dash], axis = 0)
D_dash = np.append(D_dash, [dd_dash], axis = 0)
for i in range(2,n):
    b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
    b_dashinv = np.linalg.inv(b_dash)
    cc_dash = np.dot(b_dashinv,C_star[i])
    dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[
i-1]))
    C_dash = np.append(C_dash, [cc_dash], axis = 0)
```

```
    D_dash = np.append(D_dash, [dd_dash], axis = 0)
b_dash = (B_star[n-1] - np.dot(A_star[n-1],C_dash[n-2]) )
b_dashinv = np.linalg.inv(b_dash)
final_ans = np.dot(b_dashinv, D_star[n-1] - np.dot(C_star[n-1],X_ans[
n]) - np.dot(A_star[n-1], D_dash[n-2]))
X_ans[n-1] =  np.array(final_ans)
print("X[" + str(n-1)+"] = ")
print(X_ans[n-1])
for i in range(n-2, 0,-1):
    #D_dash[i] - np.dot(C_dash[i], X_ans[i+1])
    X_ans[i] = np.array(D_dash[i] - np.dot(C_dash[i], X_ans[i+1]))
    print("X["+str(i) +"] = ")
    print(X_ans[i])
```

```
n = 10
A_star =
[[[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]]
B_star =
[[[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]]
C_star =
[[[  0.   0.]
  [  0. 120.]]]
D_star =
[[[0]
  [1]]]
1
2
3
4
5
6
7
8
9
10
Now, after running the loop
A_star =
[[[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]

 [[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]

 [[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]

 [[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]

 [[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]

 [[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]

 [[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]

 [[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]

 [[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]

 [[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]

 [[-1.e+00 -5.e-02]
  [ 0.e+00  8.e+01]]]
```

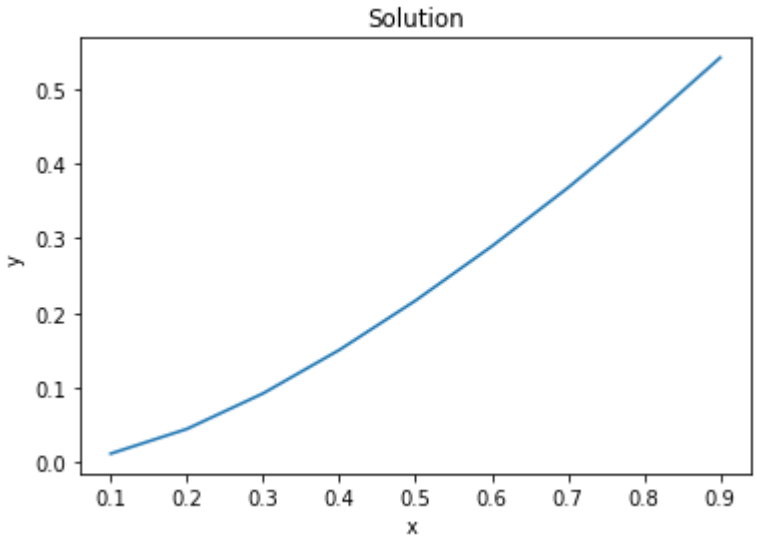
```
B_star =
[[[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]

 [[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]

 [[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]

 [[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]

 [[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]

 [[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]

 [[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]

 [[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]

 [[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]

 [[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]

 [[ 1.00e+00 -5.00e-02]
  [-6.00e+00 -1.99e+02]]]
C_star =
[[[  0.   0.]
  [  0. 120.]]

 [[  0.   0.]
  [  0. 120.]]

 [[  0.   0.]
  [  0. 120.]]

 [[  0.   0.]
  [  0. 120.]]

 [[  0.   0.]
  [  0. 120.]]

 [[  0.   0.]
  [  0. 120.]]

 [[  0.   0.]
  [  0. 120.]]

 [[  0.   0.]
  [  0. 120.]]
```

```
 [[  0.   0.]
  [  0. 120.]]

 [[  0.   0.]
  [  0. 120.]]

 [[  0.   0.]
  [  0. 120.]]]
D_star =
[[[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]]
X[9] =
[[0.54156916]
 [0.93095086]]
X[8] =
[[0.45157872]
 [0.86885795]]
X[7] =
[[0.3675745 ]
 [0.81122627]]
X[6] =
[[0.28927787]
 [0.7547065 ]]
X[5] =
[[0.2168081 ]
 [0.69468887]]
```

```
X[4] =
[[0.15083668]
 [0.6247394 ]]
X[3] =
[[0.09280878]
 [0.53581872]]
X[2] =
[[0.04525781]
 [0.41520062]]
X[1] =
[[0.01224889]
 [0.24497779]]
```

In [46]:
```
from matplotlib import pyplot as plt
```

In [58]:
```
y_axis = np.array([],dtype = float)
x_axis = np.array([],dtype = float)
for i in range(1,n):
    x_axis = np.append(x_axis, x[i])
    y_axis = np.append(y_axis, X_ans[i][0][0])
plt.title("Solution")
plt.xlabel("x ")
plt.ylabel("y ")
plt.plot(x_axis,y_axis)
plt.show()
```



for h = 0.05, we have

In [50]:
```
h = 0.05
x_0 = 0
x_n = 1
n = int((x_n- x_0)/h  + 0.5 )
print("n = " + str(n))
x = np.array([])
for i in range(n+1):
    x = np.append(x, i*h)
A_star = np.array([ [[-1,-h/2],[0,((2 - A(x[0])*h)/(2*h*h) )]] ])
B_star = np.array([ [[ 1,-h/2],[C(x[0]),B(x[0]) - ((2)/(h*h))]] ])
C_star = np.array([ [[0 ,0],[0,((2 + A(x[0])*h)/(2*h*h) )]] ])
D_star = np.array([ [[0],[D(x[0])]] ])
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
for i in range(1,n+1):
    A_star = np.append(A_star,[ [[-1,-h/2],[0,((2 - A(x[i])*h)/(2*h*h
) )]] ], axis=0)
    B_star = np.append(B_star,[ [[ 1,-h/2],[C(x[0]),B(x[i]) - ((2)/(h
*h))]] ], axis=0)
    C_star = np.append(C_star,[ [[0 ,0],[0,((2 + A(x[i])*h)/(2*h*h)
)]] ], axis=0)
    D_star = np.append(D_star,[ [[0],[D(x[i])]] ], axis=0)
    print(i)
print("Now, after running the loop")
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
X_ans = np.array([[[0],[0]]],dtype = float)
for i in range(n):
    X_ans = np.append(X_ans, [[[0],[0]]] ,axis = 0)
X_ans[n] = np.array([[[0],[1]]])
D_star[1] = D_star[1] - np.dot(A_star[1],X_ans[0])
cc_dash = np.dot(np.linalg.inv(B_star[1]),C_star[1] )
dd_dash = np.dot(np.linalg.inv(B_star[1]),D_star[1] )
C_dash = np.array([cc_dash])
D_dash = np.array([dd_dash])
C_dash = np.append(C_dash, [cc_dash], axis = 0)
D_dash = np.append(D_dash, [dd_dash], axis = 0)
for i in range(2,n):
    b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
    b_dashinv = np.linalg.inv(b_dash)
    cc_dash = np.dot(b_dashinv,C_star[i])
    dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[
i-1]))
    C_dash = np.append(C_dash, [cc_dash], axis = 0)
```

```
    D_dash = np.append(D_dash, [dd_dash], axis = 0)
b_dash = (B_star[n-1] - np.dot(A_star[n-1],C_dash[n-2]) )
b_dashinv = np.linalg.inv(b_dash)
final_ans = np.dot(b_dashinv, D_star[n-1] - np.dot(C_star[n-1],X_ans[
n]) - np.dot(A_star[n-1], D_dash[n-2]))
X_ans[n-1] =  np.array(final_ans)
print("X[" + str(n-1)+"] = ")
print(X_ans[n-1])
for i in range(n-2, 0,-1):
    #D_dash[i] - np.dot(C_dash[i], X_ans[i+1])
    X_ans[i] = np.array(D_dash[i] - np.dot(C_dash[i], X_ans[i+1]))
    print("X["+str(i) +"] = ")
    print(X_ans[i])
```

```
n = 20
A_star =
[[[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]]
B_star =
[[[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]]
C_star =
[[[  0.   0.]
  [  0. 440.]]]
D_star =
[[[0]
  [1]]]
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Now, after running the loop
A_star =
[[[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
```

```
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]

 [[-1.0e+00 -2.5e-02]
  [ 0.0e+00  3.6e+02]]]
B_star =
[[[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
```

```
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]

 [[ 1.00e+00 -2.50e-02]
  [-6.00e+00 -7.99e+02]]]
C_star =
[[[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
```

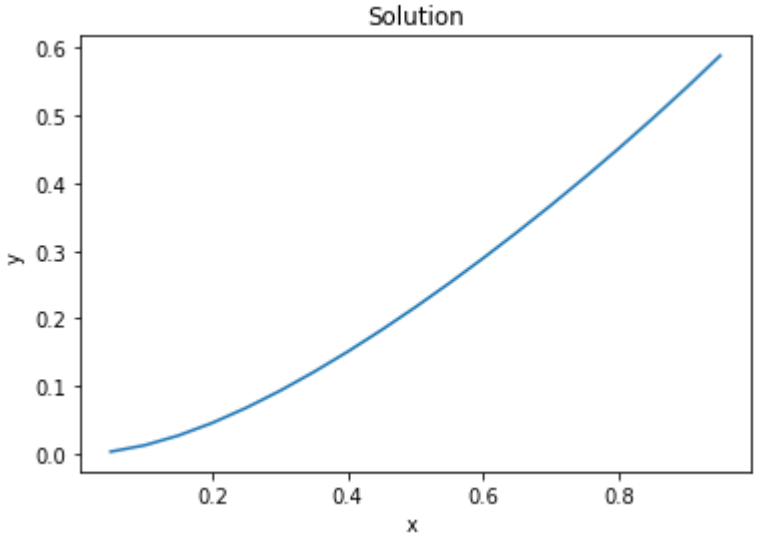
```
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]

 [[  0.   0.]
  [  0. 440.]]]
D_star =
[[[0]
  [1]]

 [[0]
```

```
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
 [1]]

[[0]
```

```
  [1]]]
X[19] =
[[0.58860637]
 [0.96433583]]
X[18] =
[[0.54123159]
 [0.93065546]]
X[17] =
[[0.49549761]
 [0.89870371]]
X[16] =
[[0.45132524]
 [0.86819125]]
X[15] =
[[0.40865079]
 [0.83878647]]
X[14] =
[[0.36742848]
 [0.81010594]]
X[13] =
[[0.32763325]
 [0.78170327]]
X[12] =
[[0.28926427]
 [0.75305581]]
X[11] =
[[0.25234915]
 [0.72354929]]
X[10] =
[[0.21694893]
 [0.6924595 ]]
X[9] =
[[0.18316417]
 [0.65893096]]
X[8] =
[[0.1511421 ]
 [0.62195178]]
X[7] =
[[0.1210852 ]
 [0.58032418]]
X[6] =
[[0.09326135]
 [0.53262985]]
X[5] =
[[0.06801587]
 [0.47718939]]
X[4] =
[[0.04578576]
 [0.41201468]]
X[3] =
[[0.02711657]
 [0.33475309]]
X[2] =
[[0.01268219]
 [0.2426221 ]]
X[1] =
```

```
[[0.00330832]
 [0.13233276]]
```

In [51]:
```
y_axis = np.array([],dtype = float)
x_axis = np.array([],dtype = float)
for i in range(1,n):
    x_axis = np.append(x_axis, x[i])
    y_axis = np.append(y_axis, X_ans[i][0][0])
plt.title("Solution")
plt.xlabel("x ")
plt.ylabel("y ")
plt.plot(x_axis,y_axis)
plt.show()
```



1. Solve the following equation :

$$y^{IV} = q/EI \, , y(0) = y''(0) = y(L) = y''(L) = 0$$

Ans: Here, we assume $y'' = p$, so the equation becomes

$$p'' = \frac{q}{EI} \; y(0) = p(0) = y(L) = p(L) = 0$$

For our purposes (to calculate the values, we are taking q/EI = 1 and L = 1)

Now, Discretizing these set of equations, we get :

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - p_i = 0$$

$$\frac{p_{i+1} - 2p_i + p_{i-1}}{h^2} = 1$$

Let, $X_i = \begin{bmatrix} y_i \\ p_i \end{bmatrix}$. So, the given system of equations become :

$$\begin{bmatrix} 1/h^2 & 0 \\ 0 & 1/h^2 \end{bmatrix} X_{i-1} + \begin{bmatrix} -2/h^2 & -1 \\ 0 & -2/h^2 \end{bmatrix} X_i + \begin{bmatrix} 1/h^2 & 0 \\ 0 & 1/h^2 \end{bmatrix} X_{i+1} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\Rightarrow A_i^* X_{i-1} + B_i^* X_i + C_i^* X_{i+1} = D_i^* \ \forall, \ i = 1, 2, \ldots, n-1$$

We can now solve this system of equation using a modification of the Thomas Algorithm.

Now, we need to add the conditions for $h \ \& \ x_0, x_n, n$

In [21]:
```
h = 0.25
x_0 = 0
x_n = 1
n = int((x_n- x_0)/h  + 0.5 )
print(n)
x = np.array([])
for i in range(n+1):
    x = np.append(x, i*h)
```

```
4
```

Now, we define our matrices $A_i^*, B_i^*, C_i^* \ \& D_i^*$

In [23]:
```
A_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
B_star = np.array([ [[-2/(h*h),-1],[0,-2/(h*h)]] ])
C_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
D_star = np.array([ [[0],[1]] ])
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
for i in range(1,n+1):
    A_star = np.append(A_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    B_star = np.append(B_star,[ [[-2/(h*h),-1],[0,-2/(h*h)]] ], axis=
0)
    C_star = np.append(C_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    D_star = np.append(D_star,[ [[0],[1]] ], axis=0)
    print(i)
print("Now, after running the loop")
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
```

```
A_star =
[[[16.  0.]
  [ 0. 16.]]]
B_star =
[[[-32.  -1.]
  [  0. -32.]]]
C_star =
[[[16.  0.]
  [ 0. 16.]]]
D_star =
[[[0]
  [1]]]
1
2
3
4
Now, after running the loop
A_star =
[[[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]]
B_star =
[[[-32.  -1.]
  [  0. -32.]]

 [[-32.  -1.]
  [  0. -32.]]

 [[-32.  -1.]
  [  0. -32.]]

 [[-32.  -1.]
  [  0. -32.]]

 [[-32.  -1.]
  [  0. -32.]]]
C_star =
[[[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]
```

```
 [[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]]
D_star =
[[[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]]
```

Here, all the values of are same because it isn't dependent on x
Now, we will define the X_i matrices that will finally store the answers.
Here, we also need to add the initial conditions, .i.e ,

$$X_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \ \& \ X_n = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In [24]:
```
X_ans = np.array([[[0],[0]]],dtype = float)
for i in range(n):
    X_ans = np.append(X_ans, [[[0],[0]]] ,axis = 0)
```

In [25]:
```
X_ans
```

Out[25]:
```
array([[[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]]])
```

Now, we need to modify the value of D_1^* because we have $A_1^* X_0 + B_1^* X_1 + C_1^* X_2 = D_1^*$ and we already know the value of X_0, so it can be shifted to the RHS
$\Rightarrow B_1^* X_1 + C_1^* X_2 = D_1^* - A_1^* X_0 \ \Rightarrow D_1^* = D_1^* - A_1^* X_0$

In [26]:
```
D_star[1] = D_star[1] - np.dot(A_star[1],X_ans[0])
```

Now, we will transform the given equations into the form :

$$X_i + C'_i X_{i+1} = D'_i$$

where, $C'_1 = (B^*_1)^{-1} C^*_1$ & $D'_1 = (B^*_1)^{-1} D^*_1$
and $B'_i = (B^*_i - A^*_i C'_{i-1})$
$C'_i = (B'_i)^{-1} C_i$
$D'_i = (B'_i)^{-1}(D^*_i - A^*_i D'_{i-1})$
for $i = 2, 3, \ldots, n-1$

In [27]:
```
cc_dash = np.dot(np.linalg.inv(B_star[1]),C_star[1] )
dd_dash = np.dot(np.linalg.inv(B_star[1]),D_star[1] )
C_dash = np.array([cc_dash])
D_dash = np.array([dd_dash])
C_dash = np.append(C_dash, [cc_dash], axis = 0)
D_dash = np.append(D_dash, [dd_dash], axis = 0)
for i in range(2,n):
    b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
    b_dashinv = np.linalg.inv(b_dash)
    cc_dash = np.dot(b_dashinv,C_star[i])
    dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[
i-1]))
    C_dash = np.append(C_dash, [cc_dash], axis = 0)
    D_dash = np.append(D_dash, [dd_dash], axis = 0)
```

In [28]:
```
print(C_dash)
print(D_dash)
```

```
[[[-0.5         0.015625  ]
  [ 0.         -0.5       ]]

 [[-0.5         0.015625  ]
  [ 0.         -0.5       ]]

 [[-0.66666667  0.03472222]
  [ 0.         -0.66666667]]

 [[-0.75        0.0546875 ]
  [ 0.         -0.75      ]]]
[[[ 0.00097656]
  [-0.03125   ]]

 [[ 0.00097656]
  [-0.03125   ]]

 [[ 0.00390625]
  [-0.0625    ]]

 [[ 0.00976562]
  [-0.09375   ]]]
```

Now, we have $X_{n-1} = (B^*_{n-1} - A^*_{n-1}C'_{n-2})^{-1}(D^*_{n-1} - C^*_{n-1}X_n - A^*_{n-1}D'_{n-2})$

In [29]:
```
b_dash = (B_star[n-1] - np.dot(A_star[n-1],C_dash[n-2]) )
b_dashinv = np.linalg.inv(b_dash)
final_ans = np.dot(b_dashinv, D_star[n-1] - np.dot(C_star[n-1],X_ans[
n]) - np.dot(A_star[n-1], D_dash[n-2]))
X_ans[n-1] =  np.array(final_ans)
```

After that we have, $X_i = D'_i - C'_i X_{i+1} \ \forall i = (n-2), (n-3), \ldots, 1$

In [30]:
```
for i in range(n-2, 0,-1):
    #D_dash[i] - np.dot(C_dash[i], X_ans[i+1])
    X_ans[i] = np.array(D_dash[i] - np.dot(C_dash[i], X_ans[i+1]))
    print("X["+str(i) +"] = ")
    print(X_ans[i])
```

```
X[2] =
[[ 0.01367187]
 [-0.125     ]]
X[1] =
[[ 0.00976562]
 [-0.09375   ]]
```

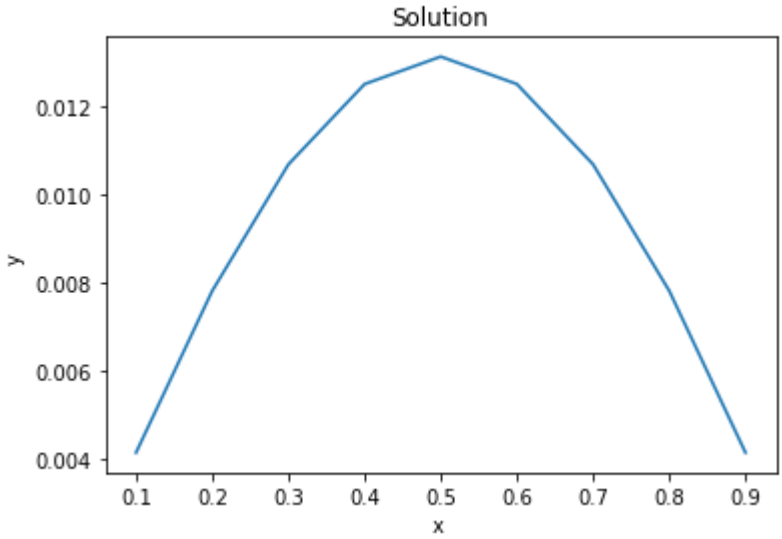
This was for h = 0.25, now, for a different value of h, let's say 0.1 :

In [52]:
```
h = 0.1
x_0 = 0
x_n = 1
n = int((x_n- x_0)/h  + 0.5 )
print(n)
x = np.array([])
for i in range(n+1):
    x = np.append(x, i*h)
A_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
B_star = np.array([ [[-2/(h*h),-1],[0,-2/(h*h)]] ])
C_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
D_star = np.array([ [[0],[1]] ])
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
for i in range(1,n+1):
    A_star = np.append(A_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    B_star = np.append(B_star,[ [[-2/(h*h),-1],[0,-2/(h*h)]] ], axis=
0)
    C_star = np.append(C_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    D_star = np.append(D_star,[ [[0],[1]] ], axis=0)
    print(i)
X_ans = np.array([[[0],[0]]],dtype = float)
for i in range(n):
    X_ans = np.append(X_ans, [[[0],[0]]] ,axis = 0)
D_star[1] = D_star[1] - np.dot(A_star[1],X_ans[0])
cc_dash = np.dot(np.linalg.inv(B_star[1]),C_star[1] )
dd_dash = np.dot(np.linalg.inv(B_star[1]),D_star[1] )
C_dash = np.array([cc_dash])
D_dash = np.array([dd_dash])
C_dash = np.append(C_dash, [cc_dash], axis = 0)
D_dash = np.append(D_dash, [dd_dash], axis = 0)
for i in range(2,n):
    b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
    b_dashinv = np.linalg.inv(b_dash)
    cc_dash = np.dot(b_dashinv,C_star[i])
    dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[
i-1]))
    C_dash = np.append(C_dash, [cc_dash], axis = 0)
    D_dash = np.append(D_dash, [dd_dash], axis = 0)
b_dash = (B_star[n-1] - np.dot(A_star[n-1],C_dash[n-2]) )
b_dashinv = np.linalg.inv(b_dash)
final_ans = np.dot(b_dashinv, D_star[n-1] - np.dot(C_star[n-1],X_ans[
n]) - np.dot(A_star[n-1], D_dash[n-2]))
X_ans[n-1] =  np.array(final_ans)
print("X[" + str(n-1)+"] = ")
print(X_ans[n-1])
for i in range(n-2, 0,-1):
    #D_dash[i] - np.dot(C_dash[i], X_ans[i+1])
    X_ans[i] = np.array(D_dash[i] - np.dot(C_dash[i], X_ans[i+1]))
```

```
    print("X["+str(i) +"] = ")
    print(X_ans[i])
```

```
10
A_star =
[[[100.   0.]
  [  0. 100.]]]
B_star =
[[[-200.   -1.]
  [   0. -200.]]]
C_star =
[[[100.   0.]
  [  0. 100.]]]
D_star =
[[[0]
  [1]]]
1
2
3
4
5
6
7
8
9
10
X[9] =
[[ 0.004125]
 [-0.045   ]]
X[8] =
[[ 0.0078]
 [-0.08  ]]
X[7] =
[[ 0.010675]
 [-0.105   ]]
X[6] =
[[ 0.0125]
 [-0.12  ]]
X[5] =
[[ 0.013125]
 [-0.125   ]]
X[4] =
[[ 0.0125]
 [-0.12  ]]
X[3] =
[[ 0.010675]
 [-0.105   ]]
X[2] =
[[ 0.0078]
 [-0.08  ]]
X[1] =
[[ 0.004125]
 [-0.045   ]]
```

In [53]:
```
y_axis = np.array([],dtype = float)
x_axis = np.array([],dtype = float)
for i in range(1,n):
    x_axis = np.append(x_axis, x[i])
    y_axis = np.append(y_axis, X_ans[i][0][0])
plt.title("Solution")
plt.xlabel("x ")
plt.ylabel("y ")
plt.plot(x_axis,y_axis)
plt.show()
```



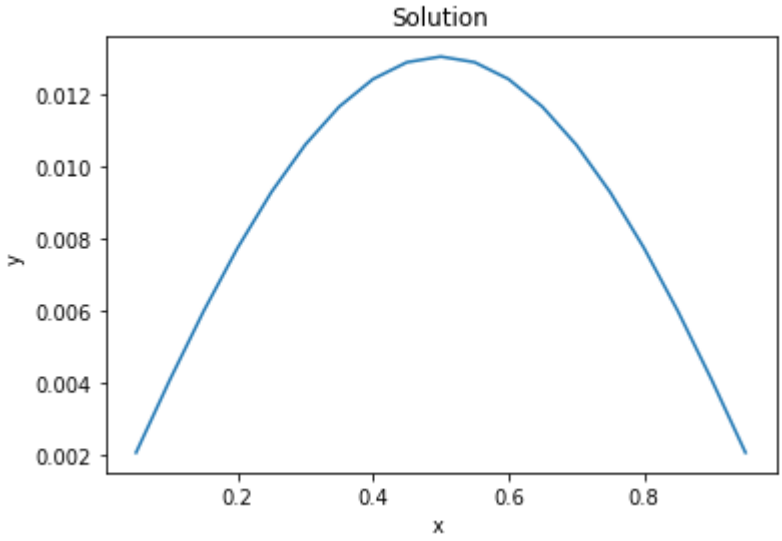
In [54]:
```
h = 0.05
x_0 = 0
x_n = 1
n = int((x_n- x_0)/h  + 0.5 )
print(n)
x = np.array([])
for i in range(n+1):
    x = np.append(x, i*h)
A_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
B_star = np.array([ [[-2/(h*h),-1],[0,-2/(h*h)]] ])
C_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
D_star = np.array([ [[0],[1]] ])
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
for i in range(1,n+1):
    A_star = np.append(A_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    B_star = np.append(B_star,[ [[-2/(h*h),-1],[0,-2/(h*h)]] ], axis=
0)
    C_star = np.append(C_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    D_star = np.append(D_star,[ [[0],[1]] ], axis=0)
    print(i)
X_ans = np.array([[[0],[0]]],dtype = float)
for i in range(n):
    X_ans = np.append(X_ans, [[[0],[0]]] ,axis = 0)
D_star[1] = D_star[1] - np.dot(A_star[1],X_ans[0])
cc_dash = np.dot(np.linalg.inv(B_star[1]),C_star[1] )
dd_dash = np.dot(np.linalg.inv(B_star[1]),D_star[1] )
C_dash = np.array([cc_dash])
D_dash = np.array([dd_dash])
C_dash = np.append(C_dash, [cc_dash], axis = 0)
D_dash = np.append(D_dash, [dd_dash], axis = 0)
for i in range(2,n):
    b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
    b_dashinv = np.linalg.inv(b_dash)
    cc_dash = np.dot(b_dashinv,C_star[i])
    dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[
i-1]))
    C_dash = np.append(C_dash, [cc_dash], axis = 0)
    D_dash = np.append(D_dash, [dd_dash], axis = 0)
b_dash = (B_star[n-1] - np.dot(A_star[n-1],C_dash[n-2]) )
b_dashinv = np.linalg.inv(b_dash)
final_ans = np.dot(b_dashinv, D_star[n-1] - np.dot(C_star[n-1],X_ans[
n]) - np.dot(A_star[n-1], D_dash[n-2]))
X_ans[n-1] =  np.array(final_ans)
print("X[" + str(n-1)+"] = ")
print(X_ans[n-1])
for i in range(n-2, 0,-1):
    #D_dash[i] - np.dot(C_dash[i], X_ans[i+1])
    X_ans[i] = np.array(D_dash[i] - np.dot(C_dash[i], X_ans[i+1]))
```

```
    print("X["+str(i) +"] = ")
    print(X_ans[i])
```

```
20
A_star =
[[[400.   0.]
  [  0. 400.]]]
B_star =
[[[-800.   -1.]
  [   0. -800.]]]
C_star =
[[[400.   0.]
  [  0. 400.]]]
D_star =
[[[0]
  [1]]]
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
X[19] =
[[ 0.00207813]
 [-0.02375   ]]
X[18] =
[[ 0.00409688]
 [-0.045     ]]
X[17] =
[[ 0.00600313]
 [-0.06375   ]]
X[16] =
[[ 0.00775]
 [-0.08   ]]
X[15] =
[[ 0.00929688]
 [-0.09375   ]]
X[14] =
[[ 0.01060938]
 [-0.105     ]]
X[13] =
[[ 0.01165938]
 [-0.11375   ]]
X[12] =
[[ 0.012425]
 [-0.12    ]]
```

```
X[11] =
[[ 0.01289063]
 [-0.12375   ]]
X[10] =
[[ 0.01304688]
 [-0.125     ]]
X[9] =
[[ 0.01289063]
 [-0.12375   ]]
X[8] =
[[ 0.012425]
 [-0.12    ]]
X[7] =
[[ 0.01165938]
 [-0.11375   ]]
X[6] =
[[ 0.01060938]
 [-0.105     ]]
X[5] =
[[ 0.00929688]
 [-0.09375   ]]
X[4] =
[[ 0.00775]
 [-0.08   ]]
X[3] =
[[ 0.00600313]
 [-0.06375   ]]
X[2] =
[[ 0.00409688]
 [-0.045     ]]
X[1] =
[[ 0.00207813]
 [-0.02375   ]]
```

In [55]:
```
y_axis = np.array([],dtype = float)
x_axis = np.array([],dtype = float)
for i in range(1,n):
    x_axis = np.append(x_axis, x[i])
    y_axis = np.append(y_axis, X_ans[i][0][0])
plt.title("Solution")
plt.xlabel("x ")
plt.ylabel("y ")
plt.plot(x_axis,y_axis)
plt.show()
```



1. Solve the following differential equation :

$$y^{IV} + 81y = 81x^2 \ , y(0) = y(1) = y''(0) = y''(1) = 0$$

Ans: Let, $y'' = p$, then the equation reduces to

$$p'' + 81y = 81x^2$$

Now, Discretizing these set of equations, we get :

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - p_i = 0$$

$$\frac{p_{i+1} - 2p_i + p_{i-1}}{h^2} + 81y_i = 81x_i^2$$

Now, let $X_i = \begin{bmatrix} y_i \\ p_i \end{bmatrix}$. So, the given system becomes :

$$\begin{bmatrix} 1/h^2 & 0 \\ 0 & 1/h^2 \end{bmatrix} X_{i-1} + \begin{bmatrix} -2/h^2 & -1 \\ 81 & -2/h^2 \end{bmatrix} X_i + \begin{bmatrix} 1/h^2 & 0 \\ 0 & 1/h^2 \end{bmatrix} X_{i+1} = \begin{bmatrix} 0 \\ 81x_i^2 \end{bmatrix}$$

$$\Rightarrow A_i^* X_{i-1} + B_i^* X_i + C_i^* X_{i+1} = D_i^* \ \forall, \ i = 1, 2, \ldots, n-1$$

We can now solve this system of equation using a modification of the Thomas Algorithm.

In [63]:
```
def D(x):
    return 81*x*x
```

Now, we need to add the conditions for h & x_0, x_n, n

In [62]:
```
h = 0.25
x_0 = 0
x_n = 1
n = int((x_n- x_0)/h  + 0.5 )
print("n = "+str(n))
x = np.array([])
for i in range(n+1):
    x = np.append(x, i*h)
```

```
n = 4
```

Now, we define our matrices A_i^*, B_i^*, C_i^* &D_i^*

In [65]:
```
A_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
B_star = np.array([ [[-2/(h*h),-1],[81,-2/(h*h)]] ])
C_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
D_star = np.array([ [[0],[D(x[0])]] ])
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
for i in range(1,n+1):
    A_star = np.append(A_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    B_star = np.append(B_star,[ [[-2/(h*h),-1],[81,-2/(h*h)]] ], axis
=0)
    C_star = np.append(C_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    D_star = np.append(D_star,[ [[0],[D(x[i])]] ], axis=0)
    print(i)
print("Now, after running the loop")
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
```

```
A_star =
[[[16.  0.]
  [ 0. 16.]]]
B_star =
[[[-32.  -1.]
  [ 81. -32.]]]
C_star =
[[[16.  0.]
  [ 0. 16.]]]
D_star =
[[[0.]
  [0.]]]
1
2
3
4
Now, after running the loop
A_star =
[[[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]]
B_star =
[[[-32.  -1.]
  [ 81. -32.]]

 [[-32.  -1.]
  [ 81. -32.]]

 [[-32.  -1.]
  [ 81. -32.]]

 [[-32.  -1.]
  [ 81. -32.]]

 [[-32.  -1.]
  [ 81. -32.]]]
C_star =
[[[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]
```

```
 [[16.  0.]
  [ 0. 16.]]

 [[16.  0.]
  [ 0. 16.]]]
D_star =
[[[ 0.    ]
  [ 0.    ]]

 [[ 0.    ]
  [ 5.0625]]

 [[ 0.    ]
  [20.25  ]]

 [[ 0.    ]
  [45.5625]]

 [[ 0.    ]
  [81.    ]]]
```

Here, we also need to add the initial conditions, .i.e ,

$X_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ & $X_n = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

In [66]:
```
X_ans = np.array([[[0],[0]]],dtype = float)
for i in range(n):
    X_ans = np.append(X_ans, [[[0],[0]]] ,axis = 0)
```

In [67]:
```
X_ans
```

Out[67]:
```
array([[[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]]])
```

Now, we need to modify the value of D_1^* because we have $A_1^* X_0 + B_1^* X_1 + C_1^* X_2 = D_1^*$ and we already know the value of X_0, so it can be shifted to the RHS

$\Rightarrow B_1^* X_1 + C_1^* X_2 = D_1^* - A_1^* X_0 \ \Rightarrow D_1^* = D_1^* - A_1^* X_0$

In [68]:
```
D_star[1] = D_star[1] - np.dot(A_star[1],X_ans[0])
```

Now, we will transform the given equations into the form :

$$X_i + C'_i X_{i+1} = D'_i$$

where, $C'_1 = (B^*_1)^{-1} C^*_1$ & $D'_1 = (B^*_1)^{-1} D^*_1$
and $B'_i = (B^*_i - A^*_i C'_{i-1})$
$C'_i = (B'_i)^{-1} C_i$
$D'_i = (B'_i)^{-1}(D^*_i - A^*_i D'_{i-1})$
for $i = 2, 3, \ldots, n-1$

In [69]:
```
cc_dash = np.dot(np.linalg.inv(B_star[1]),C_star[1] )
dd_dash = np.dot(np.linalg.inv(B_star[1]),D_star[1] )
C_dash = np.array([cc_dash])
D_dash = np.array([dd_dash])
C_dash = np.append(C_dash, [cc_dash], axis = 0)
D_dash = np.append(D_dash, [dd_dash], axis = 0)
for i in range(2,n):
    b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
    b_dashinv = np.linalg.inv(b_dash)
    cc_dash = np.dot(b_dashinv,C_star[i])
    dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[
i-1]))
    C_dash = np.append(C_dash, [cc_dash], axis = 0)
    D_dash = np.append(D_dash, [dd_dash], axis = 0)
```

In [70]:
```
print(C_dash)
print(D_dash)
```

```
[[[-0.46334842  0.01447964]
  [-1.17285068 -0.46334842]]

 [[-0.46334842  0.01447964]
  [-1.17285068 -0.46334842]]

 [[-0.54082825  0.02709317]
  [-2.19454678 -0.54082825]]

 [[-0.52500203  0.03223513]
  [-2.6110458  -0.52500203]]]
[[[ 0.00458145]
  [-0.14660633]]

 [[ 0.00458145]
  [-0.14660633]]

 [[ 0.0407396 ]
  [-0.7537204 ]]

 [[ 0.13747923]
  [-1.78435708]]]
```

Now, we have $X_{n-1} = (B^*_{n-1} - A^*_{n-1}C'_{n-2})^{-1}(D^*_{n-1} - C^*_{n-1}X_n - A^*_{n-1}D'_{n-2})$ After that we have, $X_i = D'_i - C'_i X_{i+1}\ \forall\, i = (n-2), (n-3), \ldots, 1$

In [71]:
```
b_dash = (B_star[n-1] - np.dot(A_star[n-1],C_dash[n-2]) )
b_dashinv = np.linalg.inv(b_dash)
final_ans = np.dot(b_dashinv, D_star[n-1] - np.dot(C_star[n-1],X_ans[
n]) - np.dot(A_star[n-1], D_dash[n-2]))
X_ans[n-1] =  np.array(final_ans)
print("X[" + str(n-1)+"] = ")
print(X_ans[n-1])

for i in range(n-2, 0,-1):
    #D_dash[i] - np.dot(C_dash[i], X_ans[i+1])
    X_ans[i] = np.array(D_dash[i] - np.dot(C_dash[i], X_ans[i+1]))
    print("X["+str(i) +"] = ")
    print(X_ans[i])
```

```
X[3] =
[[ 0.13747923]
 [-1.78435708]]
X[2] =
[[ 0.16343614]
 [-1.41704651]]
X[1] =
[[ 0.10082765]
 [-0.6115064 ]]
```

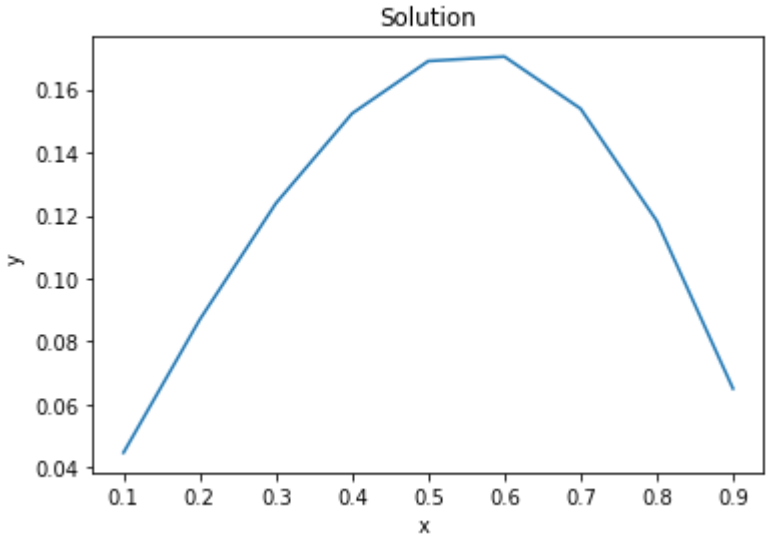
This was for h = 0.25, now, for a different value of h, let's say 0.1 :

In [73]:
```
h = 0.1
x_0 = 0
x_n = 1
n = int((x_n- x_0)/h  + 0.5 )
print("n = "+str(n))
x = np.array([])
for i in range(n+1):
    x = np.append(x, i*h)
A_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
B_star = np.array([ [[-2/(h*h),-1],[81,-2/(h*h)]] ])
C_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
D_star = np.array([ [[0],[D(x[0])]] ])
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
for i in range(1,n+1):
    A_star = np.append(A_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    B_star = np.append(B_star,[ [[-2/(h*h),-1],[81,-2/(h*h)]] ], axis
=0)
    C_star = np.append(C_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    D_star = np.append(D_star,[ [[0],[D(x[i])]] ], axis=0)
    print(i)
X_ans = np.array([[[0],[0]]],dtype = float)
for i in range(n):
    X_ans = np.append(X_ans, [[[0],[0]]] ,axis = 0)
D_star[1] = D_star[1] - np.dot(A_star[1],X_ans[0])
cc_dash = np.dot(np.linalg.inv(B_star[1]),C_star[1] )
dd_dash = np.dot(np.linalg.inv(B_star[1]),D_star[1] )
C_dash = np.array([cc_dash])
D_dash = np.array([dd_dash])
C_dash = np.append(C_dash, [cc_dash], axis = 0)
D_dash = np.append(D_dash, [dd_dash], axis = 0)
for i in range(2,n):
    b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
    b_dashinv = np.linalg.inv(b_dash)
    cc_dash = np.dot(b_dashinv,C_star[i])
    dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[
i-1]))
    C_dash = np.append(C_dash, [cc_dash], axis = 0)
    D_dash = np.append(D_dash, [dd_dash], axis = 0)
b_dash = (B_star[n-1] - np.dot(A_star[n-1],C_dash[n-2]) )
b_dashinv = np.linalg.inv(b_dash)
final_ans = np.dot(b_dashinv, D_star[n-1] - np.dot(C_star[n-1],X_ans[
n]) - np.dot(A_star[n-1], D_dash[n-2]))
X_ans[n-1] =  np.array(final_ans)
print("X[" + str(n-1)+"] = ")
print(X_ans[n-1])

for i in range(n-2, 0,-1):
    #D_dash[i] - np.dot(C_dash[i], X_ans[i+1])
    X_ans[i] = np.array(D_dash[i] - np.dot(C_dash[i], X_ans[i+1]))
```

```
    print("X["+str(i) +"] = ")
    print(X_ans[i])
```

```
n = 10
A_star =
[[[100.   0.]
  [  0. 100.]]]
B_star =
[[[-200.   -1.]
  [  81. -200.]]]
C_star =
[[[100.   0.]
  [  0. 100.]]]
D_star =
[[[0.]
  [0.]]]
1
2
3
4
5
6
7
8
9
10
X[9] =
[[ 0.0650284]
 [-1.1805192]]
X[8] =
[[ 0.1182516 ]
 [-1.75761141]]
X[7] =
[[ 0.15389869]
 [-1.91208741]]
X[6] =
[[ 0.1704249 ]
 [-1.79432135]]
X[5] =
[[ 0.16900791]
 [-1.52299946]]
X[4] =
[[ 0.15236091]
 [-1.18607397]]
X[3] =
[[ 0.12385318]
 [-0.84296083]]
X[2] =
[[ 0.08691584]
 [-0.52726876]]
X[1] =
[[ 0.04470581]
 [-0.24957853]]
```

In [74]:
```
y_axis = np.array([],dtype = float)
x_axis = np.array([],dtype = float)
for i in range(1,n):
    x_axis = np.append(x_axis, x[i])
    y_axis = np.append(y_axis, X_ans[i][0][0])
plt.title("Solution")
plt.xlabel("x ")
plt.ylabel("y ")
plt.plot(x_axis,y_axis)
plt.show()
```

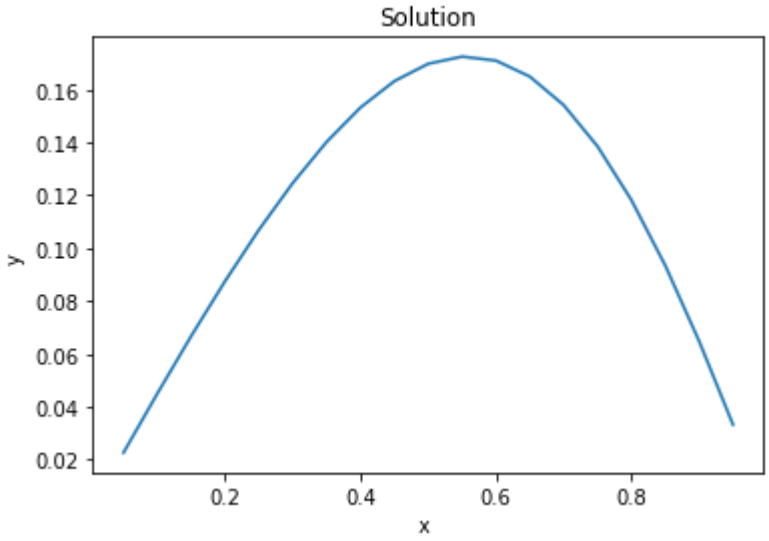


In [76]:
```
h = 0.05
x_0 = 0
x_n = 1
n = int((x_n- x_0)/h  + 0.5 )
print("n = "+str(n))
x = np.array([])
for i in range(n+1):
    x = np.append(x, i*h)
A_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
B_star = np.array([ [[-2/(h*h),-1],[81,-2/(h*h)]] ])
C_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
D_star = np.array([ [[0],[D(x[0])]] ])
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
for i in range(1,n+1):
    A_star = np.append(A_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    B_star = np.append(B_star,[ [[-2/(h*h),-1],[81,-2/(h*h)]] ], axis
=0)
    C_star = np.append(C_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    D_star = np.append(D_star,[ [[0],[D(x[i])]] ], axis=0)
    print(i)
X_ans = np.array([[[0],[0]]],dtype = float)
for i in range(n):
    X_ans = np.append(X_ans, [[[0],[0]]] ,axis = 0)
D_star[1] = D_star[1] - np.dot(A_star[1],X_ans[0])
cc_dash = np.dot(np.linalg.inv(B_star[1]),C_star[1] )
dd_dash = np.dot(np.linalg.inv(B_star[1]),D_star[1] )
C_dash = np.array([cc_dash])
D_dash = np.array([dd_dash])
C_dash = np.append(C_dash, [cc_dash], axis = 0)
D_dash = np.append(D_dash, [dd_dash], axis = 0)
for i in range(2,n):
    b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
    b_dashinv = np.linalg.inv(b_dash)
    cc_dash = np.dot(b_dashinv,C_star[i])
    dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[
i-1]))
    C_dash = np.append(C_dash, [cc_dash], axis = 0)
    D_dash = np.append(D_dash, [dd_dash], axis = 0)
b_dash = (B_star[n-1] - np.dot(A_star[n-1],C_dash[n-2]) )
b_dashinv = np.linalg.inv(b_dash)
final_ans = np.dot(b_dashinv, D_star[n-1] - np.dot(C_star[n-1],X_ans[
n]) - np.dot(A_star[n-1], D_dash[n-2]))
X_ans[n-1] =  np.array(final_ans)
print("X[" + str(n-1)+"] = ")
print(X_ans[n-1])

for i in range(n-2, 0,-1):
    #D_dash[i] - np.dot(C_dash[i], X_ans[i+1])
    X_ans[i] = np.array(D_dash[i] - np.dot(C_dash[i], X_ans[i+1]))
```

```
    print("X["+str(i) +"] = ")
    print(X_ans[i])
y_axis = np.array([],dtype = float)
x_axis = np.array([],dtype = float)
for i in range(1,n):
    x_axis = np.append(x_axis, x[i])
    y_axis = np.append(y_axis, X_ans[i][0][0])
plt.title("Solution")
plt.xlabel("x ")
plt.ylabel("y ")
plt.plot(x_axis,y_axis)
plt.show()
```

```
n = 20
A_star =
[[[400.   0.]
  [  0. 400.]]]
B_star =
[[[-800.   -1.]
  [  81. -800.]]]
C_star =
[[[400.   0.]
  [  0. 400.]]]
D_star =
[[[0.]
  [0.]]]
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
X[19] =
[[ 0.03327401]
 [-0.68166565]]
X[18] =
[[ 0.06484386]
 [-1.18731304]]
X[17] =
[[ 0.09344543]
 [-1.54206631]]
X[16] =
[[ 0.11819183]
 [-1.76943603]]
X[15] =
[[ 0.13851465]
 [-1.89113959]]
X[14] =
[[ 0.15410961]
 [-1.92698612]]
X[13] =
[[ 0.16488711]
 [-1.89481485]]
X[12] =
[[ 0.17092757]
 [-1.81047697]]
```

```
X[11] =
[[ 0.17244183]
 [-1.68785192]]
X[10] =
[[ 0.16973647]
 [-1.53889009]]
X[9] =
[[ 0.16318388]
 [-1.37367489]]
X[8] =
[[ 0.15319711]
 [-1.20049818]]
X[7] =
[[ 0.14020909]
 [-1.02594389]]
X[6] =
[[ 0.12465621]
 [-0.85497569]]
X[5] =
[[ 0.10696589]
 [-0.69102537]]
X[4] =
[[ 0.08754801]
 [-0.53607939]]
X[3] =
[[ 0.06678993]
 [-0.39076189]]
X[2] =
[[ 0.04505494]
 [-0.25441309]]
X[1] =
[[ 0.02268393]
 [-0.12516292]]
```



In [77]:
```
h = 0.01
x_0 = 0
x_n = 1
n = int((x_n- x_0)/h  + 0.5 )
print("n = "+str(n))
x = np.array([])
for i in range(n+1):
    x = np.append(x, i*h)
A_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
B_star = np.array([ [[-2/(h*h),-1],[81,-2/(h*h)]] ])
C_star = np.array([ [[1/(h*h),0],[0,1/(h*h)]] ])
D_star = np.array([ [[0],[D(x[0])]] ])
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
for i in range(1,n+1):
    A_star = np.append(A_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    B_star = np.append(B_star,[ [[-2/(h*h),-1],[81,-2/(h*h)]] ], axis
=0)
    C_star = np.append(C_star,[ [[1/(h*h),0],[0,1/(h*h)]] ], axis=0)
    D_star = np.append(D_star,[ [[0],[D(x[i])]] ], axis=0)
    print(i)
X_ans = np.array([[[0],[0]]],dtype = float)
for i in range(n):
    X_ans = np.append(X_ans, [[[0],[0]]] ,axis = 0)
D_star[1] = D_star[1] - np.dot(A_star[1],X_ans[0])
cc_dash = np.dot(np.linalg.inv(B_star[1]),C_star[1] )
dd_dash = np.dot(np.linalg.inv(B_star[1]),D_star[1] )
C_dash = np.array([cc_dash])
D_dash = np.array([dd_dash])
C_dash = np.append(C_dash, [cc_dash], axis = 0)
D_dash = np.append(D_dash, [dd_dash], axis = 0)
for i in range(2,n):
    b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
    b_dashinv = np.linalg.inv(b_dash)
    cc_dash = np.dot(b_dashinv,C_star[i])
    dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[
i-1]))
    C_dash = np.append(C_dash, [cc_dash], axis = 0)
    D_dash = np.append(D_dash, [dd_dash], axis = 0)
b_dash = (B_star[n-1] - np.dot(A_star[n-1],C_dash[n-2]) )
b_dashinv = np.linalg.inv(b_dash)
final_ans = np.dot(b_dashinv, D_star[n-1] - np.dot(C_star[n-1],X_ans[
n]) - np.dot(A_star[n-1], D_dash[n-2]))
X_ans[n-1] =  np.array(final_ans)
print("X[" + str(n-1)+"] = ")
print(X_ans[n-1])

for i in range(n-2, 0,-1):
    #D_dash[i] - np.dot(C_dash[i], X_ans[i+1])
    X_ans[i] = np.array(D_dash[i] - np.dot(C_dash[i], X_ans[i+1]))
```

```
    print("X["+str(i) +"] = ")
    print(X_ans[i])
y_axis = np.array([],dtype = float)
x_axis = np.array([],dtype = float)
for i in range(1,n):
    x_axis = np.append(x_axis, x[i])
    y_axis = np.append(y_axis, X_ans[i][0][0])
plt.title("Solution")
plt.xlabel("x ")
plt.ylabel("y ")
plt.plot(x_axis,y_axis)
plt.show()
```

```
n = 100
A_star =
[[[10000.     0.]
  [    0. 10000.]]]
B_star =
[[[-2.0e+04 -1.0e+00]
  [ 8.1e+01 -2.0e+04]]]
C_star =
[[[10000.     0.]
  [    0. 10000.]]]
D_star =
[[[0.]
  [0.]]]
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
```

```
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
X[99] =
```

```
[[ 0.00670387]
 [-0.1519056 ]]
X[98] =
[[ 0.01339255]
 [-0.29592669]]
X[97] =
[[ 0.02005164]
 [-0.43227703]]
X[96] =
[[ 0.0266675 ]
 [-0.56116849]]
X[95] =
[[ 0.03322724]
 [-0.68281099]]
X[94] =
[[ 0.0397187 ]
 [-0.79741239]]
X[93] =
[[ 0.04613042]
 [-0.90517835]]
X[92] =
[[ 0.05245163]
 [-1.00631228]]
X[91] =
[[ 0.0586722 ]
 [-1.10101522]]
X[90] =
[[ 0.06478267]
 [-1.1894858 ]]
X[89] =
[[ 0.07077419]
 [-1.27192012]]
X[88] =
[[ 0.07663852]
 [-1.3485117 ]]
X[87] =
[[ 0.082368  ]
 [-1.41945141]]
X[86] =
[[ 0.08795553]
 [-1.48492741]]
X[85] =
[[ 0.09339457]
 [-1.54512509]]
X[84] =
[[ 0.0986791 ]
 [-1.60022702]]
X[83] =
[[ 0.1038036 ]
 [-1.65041289]]
X[82] =
[[ 0.10876306]
 [-1.69585948]]
X[81] =
[[ 0.11355294]
 [-1.73674061]]
X[80] =
```

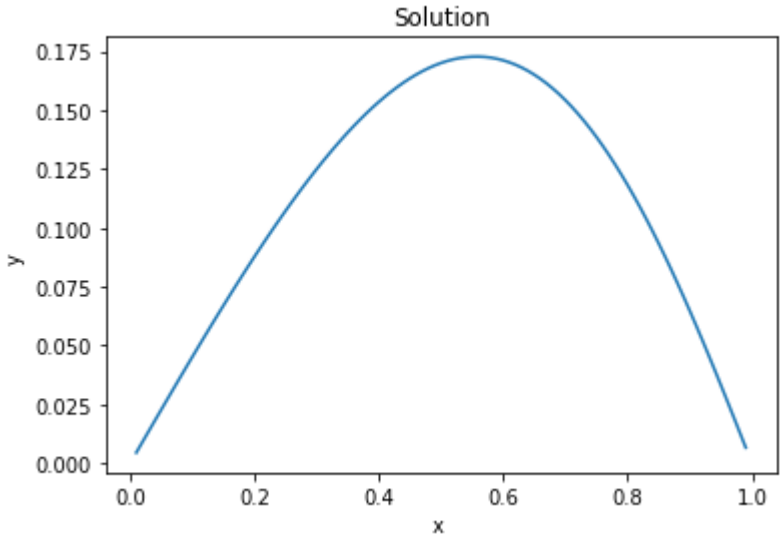
```
[[ 0.11816915]
 [-1.7732271 ]]
X[79] =
[[ 0.12260803]
 [-1.80548677]]
X[78] =
[[ 0.12686636]
 [-1.83368435]]
X[77] =
[[ 0.13094132]
 [-1.85798151]]
X[76] =
[[ 0.13483049]
 [-1.87853681]]
X[75] =
[[ 0.1385318 ]
 [-1.89550567]]
X[74] =
[[ 0.14204356]
 [-1.90904039]]
X[73] =
[[ 0.14536442]
 [-1.9192901 ]]
X[72] =
[[ 0.14849335]
 [-1.92640077]]
X[71] =
[[ 0.15142964]
 [-1.9305152 ]]
X[70] =
[[ 0.15417287]
 [-1.931773  ]]
X[69] =
[[ 0.15672293]
 [-1.93031061]]
X[68] =
[[ 0.15907996]
 [-1.92626125]]
X[67] =
[[ 0.16124436]
 [-1.91975501]]
X[66] =
[[ 0.16321679]
 [-1.91091875]]
X[65] =
[[ 0.16499813]
 [-1.89987619]]
X[64] =
[[ 0.16658947]
 [-1.88674786]]
X[63] =
[[ 0.16799215]
 [-1.87165115]]
X[62] =
[[ 0.16920765]
 [-1.85470029]]
X[61] =
```

```
[[ 0.17023769]
 [-1.83600636]]
X[60] =
[[ 0.17108413]
 [-1.81567736]]
X[59] =
[[ 0.171749  ]
 [-1.79381813]]
X[58] =
[[ 0.17223448]
 [-1.77053046]]
X[57] =
[[ 0.17254292]
 [-1.74591305]]
X[56] =
[[ 0.17267676]
 [-1.72006155]]
X[55] =
[[ 0.1726386 ]
 [-1.69306857]]
X[54] =
[[ 0.17243113]
 [-1.66502371]]
X[53] =
[[ 0.17205716]
 [-1.63601358]]
X[52] =
[[ 0.17151958]
 [-1.60612183]]
X[51] =
[[ 0.1708214 ]
 [-1.57542915]]
X[50] =
[[ 0.16996567]
 [-1.54401331]]
X[49] =
[[ 0.16895554]
 [-1.51194919]]
X[48] =
[[ 0.16779421]
 [-1.4793088 ]]
X[47] =
[[ 0.16648496]
 [-1.4461613 ]]
X[46] =
[[ 0.16503108]
 [-1.41257304]]
X[45] =
[[ 0.16343595]
 [-1.37860758]]
X[44] =
[[ 0.16170296]
 [-1.34432569]]
X[43] =
[[ 0.15983554]
 [-1.30978544]]
X[42] =
```

```
[[ 0.15783714]
 [-1.27504217]]
X[41] =
[[ 0.15571123]
 [-1.24014854]]
X[40] =
[[ 0.15346131]
 [-1.20515456]]
X[39] =
[[ 0.15109088]
 [-1.17010761]]
X[38] =
[[ 0.14860343]
 [-1.13505249]]
X[37] =
[[ 0.14600248]
 [-1.10003142]]
X[36] =
[[ 0.14329152]
 [-1.06508408]]
X[35] =
[[ 0.14047406]
 [-1.03024764]]
X[34] =
[[ 0.13755357]
 [-0.99555679]]
X[33] =
[[ 0.13453353]
 [-0.96104377]]
X[32] =
[[ 0.13141738]
 [-0.92673838]]
X[31] =
[[ 0.12820856]
 [-0.89266802]]
X[30] =
[[ 0.12491047]
 [-0.85885775]]
X[29] =
[[ 0.12152649]
 [-0.82533025]]
X[28] =
[[ 0.11805999]
 [-0.79210591]]
X[27] =
[[ 0.11451427]
 [-0.75920281]]
X[26] =
[[ 0.11089263]
 [-0.72663679]]
X[25] =
[[ 0.10719833]
 [-0.69442143]]
X[24] =
[[ 0.10343458]
 [-0.66256814]]
X[23] =
```

```
[[ 0.09960458]
 [-0.6310861 ]]
X[22] =
[[ 0.09571147]
 [-0.59998237]]
X[21] =
[[ 0.09175836]
 [-0.56926187]]
X[20] =
[[ 0.08774833]
 [-0.53892739]]
X[19] =
[[ 0.0836844 ]
 [-0.50897968]]
X[18] =
[[ 0.07956958]
 [-0.4794174 ]]
X[17] =
[[ 0.07540681]
 [-0.4502372 ]]
X[16] =
[[ 0.07119902]
 [-0.4214337 ]]
X[15] =
[[ 0.06694909]
 [-0.39299955]]
X[14] =
[[ 0.06265985]
 [-0.36492544]]
X[13] =
[[ 0.05833413]
 [-0.33720012]]
X[12] =
[[ 0.05397468]
 [-0.30981041]]
X[11] =
[[ 0.04958425]
 [-0.28274126]]
X[10] =
[[ 0.04516555]
 [-0.25597573]]
X[9] =
[[ 0.04072125]
 [-0.22949504]]
X[8] =
[[ 0.036254  ]
 [-0.20327858]]
X[7] =
[[ 0.03176643]
 [-0.17730394]]
X[6] =
[[ 0.02726112]
 [-0.15154692]]
X[5] =
[[ 0.02274066]
 [-0.12598155]]
X[4] =
```

```
[[ 0.0182076 ]
 [-0.10058013]]
X[3] =
[[ 0.01366448]
 [-0.07531324]]
X[2] =
[[ 0.00911383]
 [-0.05014973]]
X[1] =
[[ 0.00455817]
 [-0.02505681]]
```



In [ ]: