Numerical Techniques Laboratory

Assignment 4.3 | Tanishq Jasoria | 16MA20047

Find the solution to ODE

$$f''' + ff'' + 1 - (f')^2 = 0$$

And the boundary conditions are

$$f_0 = f(0) = 0 = f_0' = f'(0)$$
$$f_n' = f'(10) = 1$$

To simplify our calculations we take $f' = F$

Now our equations become

$$f_0 = f(0) = 0 = F_0 = F(0)$$
$$F_n = F(10) = 1$$

And we solve for the equation with

$$X_i = \begin{bmatrix} f_i \\ F_i \end{bmatrix}$$

In [2]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline

plt.rcParams['figure.figsize'] = [20, 30]
```

In [3]:

```
def BlockTridiagonal(A, B, C, D):
    n = len(D)
    _B = np.zeros((n, 2, 2))
    _C = np.zeros(A.shape)
    _D = np.zeros((n,2,1))
    D_out = np.zeros(_D.shape)
    _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-1):
        _B[i] = B[i] - A[i-1].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-1].dot(_D[i-1]))
    _D[n-1] = np.linalg.inv(B[n-1] - A[n-2].dot(_C[n-2])).dot(D[n-1] - A[n-2].dot(_
    D_out[n-1] = np.copy(_D[n-1])
    for i in range(n-2, -1, -1):
        D_out[i] = _D[i] - _C[i].dot(D_out[i+1])

    return D_out
```

In [7]:

```
def BVP(x0, xn, h, epsilon = 0.001):
    n = int(np.ceil((xn - x0)/h)+1)
    print(n)
    A = np.zeros((n-2, 2, 2))
    B = np.zeros((n-1, 2, 2))
    C = np.zeros((n-2, 2, 2))
    D = np.zeros((n-1, 2, 1))
    f = np.zeros(n)
    F = np.zeros(n)
    F[n-1] = 1
    f[n-1] = h/2
    Solution = np.stack((f, F))
    print("Solution Shape", Solution.shape)
    count = 0
    delta_X = np.ones(Solution.shape)
    while(np.amax(np.absolute(delta_X))>epsilon):
        print("Iteration : ", count+1)
        B[0] = np.array([[1, -h/2],
                         [(Solution[1][2] - Solution[1][0])/(2*h), -2/h**2 - 2*Solu
        C[0] = np.array([[0, 0],
                         [0, 1/h**2 + Solution[0][1]/(2*h)]])
        A[n-3] = np.array([[-1, -h/2],
                           [0, 1/h**2 - Solution[0][-2]/2*h]])
        B[n-2] = np.array([[1, -h/2],
                           [(Solution[-1][-1] - Solution[1][-3])/2*h, -2/h**2 - 2
        D[0] = np.array([[-Solution[0][1] + Solution[0][0] + h*(Solution[1][1] + So
                           [Solution[1][1]**2 - 1 - Solution[0][1]*(Solution[1][2
        D[n-2] = np.array([[-Solution[0][-2] + Solution[0][-3] + h*(Solution[1][-2]
                           [Solution[1][-2]**2 - 1 - Solution[0][-2]*(Solution[1]

        for i in range(1, n-2):
            A[i-1] = np.array([[-1, -h/2],
                               [0, 1/h**2 - Solution[0][i]/2*h]])
            B[i] = np.array([[1, -h/2],
                             [(Solution[1][i+2] - Solution[1][i])/2*h, -2/h**2 - 2*
            C[i] = np.array([[0, 0],
                             [0, 1/h**2 + Solution[0][i+1]/2*h]])
            D[i] = np.array([[-Solution[0][i+1] + Solution[0][i] + h*(Solution[1][i
                             [Solution[1][i+1]**2 - 1 - Solution[0][i+1]*(Solution[
        print(delta_X.shape)
        delta_X = np.reshape(BlockTridiagonal(A, B, C, D), (n-1, 2)).T
        print(delta_X)
        b = np.array([[0],[0]])
        delta_X = np.concatenate([b, delta_X], axis=1)

        delta_X[1][-1] = 0
        delta_X[0][-1] = delta_X[0][-2] + h*(delta_X[1][-1] + delta_X[1][-2])/2
        Solution = Solution + delta_X
#         print(Solution)
        count+= 1

    print(Solution)
    return Solution[0, :]
```

In [8]:

```
def func(x0, xn, h = 0.1):
    lst = np.arange(x0, xn, h)
    lst = np.append(lst, xn)
    return lst
x0 = 0
xn = 10
x = func(x0, xn, h = 0.4)
solution = BVP(x0, xn, h=0.4, epsilon=0.05)
```

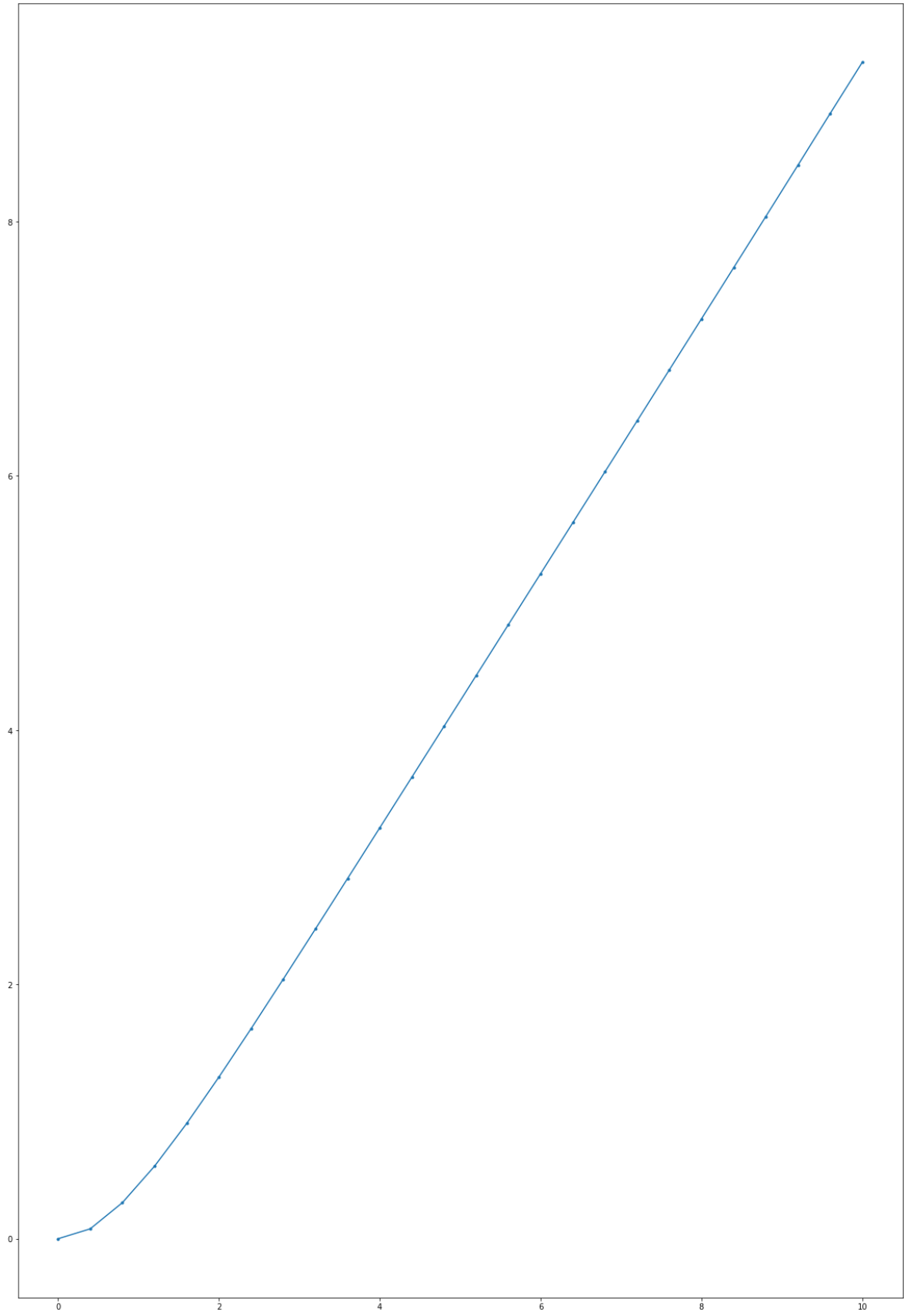
```
26
Solution Shape (2, 26)
Iteration :  1
(2, 26)
[[  0.56172776   2.21491105   4.89554987   8.53964421  13.08319407
   18.46219946  24.61266038  31.47057682  38.97194879  47.05277628
   55.6490593   64.69679785  74.13199192  83.89064152  93.90874664
  104.12230729 114.46732346 124.87979516 135.29572239 145.65110514
  155.88194341 165.92423722 175.71398654 185.1871914  192.89465375]
 [  2.80863881   5.45727763   7.94591644  10.27455526  12.44319407
   14.45183289  16.3004717   17.98911051  19.51774933  20.88638814
   22.09502696  23.14366577  24.03230458  24.7609434   25.32958221
   25.73822103  25.98685984  26.07549866  26.00413747  25.77277628
   25.3814151   24.83005391  24.11869273  23.24733154  15.28998023]]
Iteration :  2
(2, 26)
[[  -0.3199064    -1.20140925   -2.5754905    -4.44291411   -6.78033
991
    -9.55634475  -12.73761492  -16.29053055  -20.18144697  -24.37672
277
```

In [9]:

```
plt.plot(x, solution, '.-')
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x7fe46898d3c8>]
```



In [ ]:

```

```