# Numerical Exercise 5

## Md Shariar Imroze Khan

## Matriculation Number: 220202354

```
import numpy as np
import pandas as pd
import math
from sympy import *
import random
from numpy import linalg as la
from scipy import optimize
import matplotlib.pyplot as plt
```

```
########################### (a) ###########################

def f(x): #function
    return (exp(-x)-10**(-9))

def df(x): #derivative
    return (-exp(-x))

x0 = np.arange(0.0,0.75,0.1)

def simNewton(f, df, x0, ea=10**-3, er=10**-3, N=100):

    def g(x):
        return (x - f(x)/df(x))

    def g0(x_0):
        return (x0-f(x0)/df(x0))

    approx = [x0]
    n = 0

    while n < 100:
        approx.append(g(approx[n]))

        # Check for convergence

        #if abs(approx[n]) < ea:                              #Criteria 1
        #if abs(approx[n]) < er*abs(approx[0])+ea:            #Criteria 2
        #if abs(approx[n+1] - approx[n]) < ea :              #Criteria 3
            #return approx
        if abs(approx[n+1]-approx[n]) < er*abs(approx[1])+ea: #Criteria 4
            return approx
        n = n + 1
```

```
# # Simplified Newton's method
```

```
# different starting point x0
a = []
for i in x0:
  approx = simNewton(f,df,i)
  approx_x0 = []
  for j in range (0, len(approx)):

    if approx[j] > 0:
      approx_x0.append(approx[j])
  a.append(approx)
  it_1 = np.array(pd.DataFrame (a)).T
print(it_1)




      # print(f"{j}  {approx[j]}")

    # else:
    #   print(f"{j} {approx[j]}")
```

```
[[0.0 0.1 0.2 0.30000000000000004 0.4 0.5 0.6000000000000001
  0.7000000000000001]
 [0.999999999000000 1.09999999889483 1.19999999877860 1.29999999865014
  1.39999999850818 1.49999999835128 1.59999999817788 1.69999999798625]
 [1.99999999628172 2.09999999589066 2.19999999545848 2.29999999498084
  2.39999999445298 2.49999999386959 2.59999999322485 2.69999999251230]
 [2.99999998889266 3.09999998772449 3.19999998643347 3.29999998500666
  3.39999998342980 3.49999998168710 3.59999997976111 3.69999997763257]
 [3.99999996880713 4.09999996552654 4.19999996190094 4.29999995789402
  4.39999995346570 4.49999994857164 4.59999994316288 4.69999993718526]
 [4.99999991420898 5.09999990518626 5.19999989521461 5.29999988419423
  5.39999987201483 5.49999985855452 5.59999984367857 5.69999982723810]
 [5.99999976579583 6.09999974116436 6.19999971394239 6.29999968385745
  6.39999965060845 6.49999961386262 6.59999957325220 6.69999952837075]
 [6.99999936236713 7.09999929530671 7.19999922119348 7.29999913928571
  7.39999904876362 7.49999894872124 7.59999883815733 7.69999871596531]
 [7.99999826573467 8.09999808334049 8.19999788176376 8.29999765898705
  8.39999741278075 8.49999714068073 8.59999683996375 8.69999650762015]
 [8.99999528478185 9.09999478887873 9.19999424082117 9.29999363512408
  9.39999296572550 9.49999222592594 9.59999140832133 9.69999050472890]
 [9.99998718173613 10.0999858336327 10.1999843437491 10.2999826971745
  10.3999808774298 10.4999788663030 10.5999766436666 10.6999741872766]
 [10.9999651555527 11.0999614909681 11.1999574409842 11.2999529650701
  11.3999480184325 11.4999425515678 11.5999365097666 11.6999298325664]
 [11.9999052834972 12.0998953223561 12.1998843136547 12.2998721472339
  12.3998587013521 12.4998438414677 12.5998274188937 12.6998092693114]
 [12.9997425441206 13.0997154693214 13.1996855474994 13.2996524793322
  13.3996159340397 13.4995755460810 13.5995309115044 13.6994815839151]
 [13.9993002446159 14.0992266660061 14.1991453524546 14.2990554912201
  14.3989561842510 14.4988464392491 14.5987251598028 14.6985911344921]
 [14.9980984815662 15.0978986101533 15.1976777430898 15.2974336777758
  15.3971639811849 15.4968659659001 15.5965366636892 15.6961727963732]
 [15.9948356743842 16.0942933712007 16.1936942177734 16.2930322758054
  16.3923009914371 16.4914931324810 16.5906007194881 16.6896149500910]
 [16.9859953363382 17.0845285835978 17.1829089224764 17.2811205685695
  17.3791461424690 17.4769665161685 17.5745606463602 17.6719053938922]
 [17.9621763078303 18.0582430686338 18.1539059488227 18.2491245968661
```

```
 18.3438548738830 18.4380485515968 18.5316529982189 18.6246108548146]
[18.8989534601427 18.9886452997423 19.0773213964039 19.1648892997634
 19.2512497646050 19.3362966011008 19.4199166077846 19.5019896096615]
[19.7376249208985 19.8121781494411 19.8844910364463 19.9544118033940
 20.0217873588718 20.0864652628631 20.1482960593441 20.2071359832052]
[20.3644249604722 20.4100915079568 20.4522512573441 20.4908678335021
 20.5259357030894 20.5574831041238 20.5855741839898 20.6103100956372]
[20.6659394724895 20.6789690623354 20.6896458795032 20.6982372426880
 20.7050161278627 20.7102528145961 20.7142068501225 20.7171199049675]
[20.7216536347528 20.7222990623631 20.7227069667488 20.7229552185246
 20.7231003194173 20.7231815336475 20.7232249279498 20.7232469893383]
[20.7232645380466 20.7232653697704 20.7232656808076 20.7232657887095
 20.7232658232491 20.7232658333930 20.7232658361097 20.7232658367688]]
```

The initial guess x0 has to be bounded between [-0.5,0.75] for convergence. If the initial guess for x0 is greater than the upper bound 0.75 the algorithm fails to converge.

✓ 0s completed at 6:22 PM