MATH 5800 Homework 1

C14061094_yuhsi (ktpss92142@gmail.com)

November 10, 2020

Source code and result of homework1:

`https://github.com/TW-yuhsi/Programming_Portfolio/tree/master/Schoolworks/Numerical_Analysis(1)/homeworks/hw1`

Question 1.

Find the rate of convergence of the following sequence as $n \to \infty$.

(a) $sin(\frac{1}{n})$, (b) $sin(\frac{1}{n^2})$, (c) $(sin(\frac{1}{n}))^2$.

Solution: (a) $sin(x) \leq x, \forall 0 \leq x \leq 1 \Longrightarrow \mid sin(\frac{1}{n}) - 0 \mid = \mid sin(\frac{1}{n}) \mid \leq \frac{1}{n} \Longrightarrow sin(\frac{1}{n}) = 0 + O(\frac{1}{n})$.

Therefore, the sequence converges to 0 with rate $O(\frac{1}{n})$.

(b) $sin(x^2) \leq x^2, \forall 0 \leq x \leq 1 \Longrightarrow \mid sin(\frac{1}{n^2}) - 0 \mid = \mid sin(\frac{1}{n^2}) \mid \leq \frac{1}{n^2} \Longrightarrow sin(\frac{1}{n^2}) = 0 + O(\frac{1}{n^2})$.

Therefore, the sequence converges to 0 with rate $O(\frac{1}{n^2})$.

(c) $\mid sin(\frac{1}{n}) - 0 \mid = \mid sin(\frac{1}{n}) \mid \leq \frac{1}{n} \Longrightarrow \mid (sin(\frac{1}{n}))^2 \mid \leq \frac{1}{n^2} \Longrightarrow (sin(\frac{1}{n}))^2 = 0 + O(\frac{1}{n^2})$.

Therefore, the sequence converges to 0 with rate $O(\frac{1}{n^2})$.

■

Question 2.

(a) Find the number of multiplication and additions are required to determine a sum of the form $\sum_{i=1}^{n} \sum_{j=1}^{i} a_i b_j$.

(b) Give an algorithm to reduce the number of computations.

Solution: (a) When n=1: $\sum_{i=1}^{n} \sum_{j=1}^{i} a_i b_j = \sum_{i=1}^{1} \sum_{j=1}^{i} a_i b_j = a_1 b_1$.

When n=2: $\sum_{i=1}^{n} \sum_{j=1}^{i} a_i b_j = \sum_{i=1}^{2} \sum_{j=1}^{i} a_i b_j = a_1 b_1 + (a_2 b_1 + a_2 b_2)$.

When n=3: $\sum_{i=1}^{n} \sum_{j=1}^{i} a_i b_j = \sum_{i=1}^{3} \sum_{j=1}^{i} a_i b_j = a_1 b_1 + (a_2 b_1 + a_2 b_2) + (a_3 b_1 + a_3 b_2 + a_3 b_3)$.

Hence, if $n = N$, the number of multiplication and the number of addition are $\sum_{i=1}^{N} i$ and $(\sum_{i=1}^{N} i) - 1$, respectively.

(b) Let $p_1 = a_1 b_1$. Then rewrite double summation into recursive function as $\sum_{i=1}^{n} \sum_{j=1}^{i} a_i b_j = p_n = p_{n-1} + \sum_{i=1}^{n} a_n b_i$, for $n \geq 2$.

```
def recursiveSummation(s, n):
    if (n==1):
        s = s+a[0]*b[0]
        return s
```

```
        else:
            for i in range(1, n+1):
                s = s+a[n]*b[i]
            return recursiveSummation(s, n-1)
```

■

Question 3.

The following methods are proposed to compute $21^{\frac{1}{3}}$, rank them in order, bases on there apparent speed of convergence, with $p_0 = 1$, and show your numerical result of convergence with $\frac{|p_n - 21^{\frac{1}{3}}|}{21^{\frac{1}{3}}} < 1e-2$.

(a) $p_n = \frac{20p_{n-1} + \frac{21}{p_{n-1}^2}}{21}$, (b) $p_n = p_{n-1}\frac{p_{n-1}^3 - 21}{3p_{n-1}^2}$, (c) $p_n = p_{n-1} - \frac{p_{n-1}^4 - 21p_{n-1}}{p_{n-1}^2 - 21}$, (d) $p_n = (\frac{21}{p_{n-1}})^{\frac{1}{2}}$.

Solution: $(b) > (d) > (a) > (c)$.

```
In [1]:
runfile('D:/_First_Semester_of_Senior_Year/Numerical_Analysis(1)/homeworks/hw1/codes/hw1_3.py',
wdir='D:/_First_Semester_of_Senior_Year/Numerical_Analysis(1)/homeworks/hw1/codes')

tolerate value = 0.001

------------------------(a)------------------------
01 :  1   0.6375398756657026
02 :  1.9523809523809523   0.292339757252086
03 :  2.1217542737896267   0.23094868211538497
04 :  2.242849692023859   0.1870564217658913
05 :  2.334839672527244   0.15371372199512492
06 :  2.4070933802042798   0.12752463412689247
07 :  2.465059287507692   0.10651430415854733
08 :  2.512243462947621   0.08941192206197938
09 :  2.551057096383374   0.07534352766099053
10 :  2.583237767202747   0.0636793177146394
11 :  2.610081444615138   0.05394955506215438
12 :  2.632580300988204   0.04579461678379341
13 :  2.651509504414566   0.03893335356325044
14 :  2.667484487618559   0.03314324095796749 4
15 :  2.6810002018890304   0.02824433348302552
16 :  2.6924588873718474   0.024091016918216086
17 :  2.702190249310728   0.020563786259907384
18 :  2.710466452525252   0.017563992613755006
19 :  2.717513483076536   0.015009725043949105
20 :  2.7235199021356973   0.012832637644961595
21 :  2.7286436885337966   0.010975469390044143
22 :  2.733017654464709   0.009390081154891679
23 :  2.7367537784807086   0.008035885179524091
24 :  2.739946704897569   0.0068785766734786
25 :  2.742676593051374   0.00588910107383166
26 :  2.7450114536058443   0.00504280722695515
27 :  2.747009075988443   0.004318748769785488
28 :  2.748718626937001   0.003699104720415233
29 :  2.750181982351313   0.003168696734998448
```

Figure 1: Q3-1

```
30 :  2.7514348413360428   0.0027145853116200156
31 :  2.7525076612388197   0.0023257308762711674
32 :  2.753426444758921   0.001992708487338976
33 :  2.7542134042130884   0.001707467065735409
34 :  2.754887523357019   0.0014631257570101267
35 :  2.755465033456749   0.0012538013744577367
36 :  2.7559598173516564   0.0010744619423909837
37 :  2.756383752878738   0.0009208022185352455

------------------------(b)------------------------
01 :  1   0.6375398756657026
02 :  7.666666666666667   1.7788609532296136
03 :  5.2302037387103555   0.8957402974266627
04 :  3.7426969186952115   0.3565783904958582
05 :  2.9948535682764033   0.08551499672047944
06 :  2.777022225866664   0.006559821266738485
07 :  2.7590418664232454   4.265794730158707e-05

------------------------(c)------------------------
01 :  1   0.6375398756657026
02 :  0.0   1.0
03 :  0.0   1.0
04 :  0.0   1.0
```

file:///D:/_First_Semester_of_Senior_Year/Numerical_Analysis(1)/homeworks/hw1/result/hw1_3.html

10/8/2020 hw1_3.html

```
05 :  0.0   1.0
06 :  0.0   1.0
07 :  0.0   1.0
08 :  0.0   1.0
09 :  0.0   1.0
10 :  0.0   1.0
11 :  0.0   1.0
12 :  0.0   1.0
13 :  0.0   1.0
14 :  0.0   1.0
15 :  0.0   1.0
16 :  0.0   1.0
17 :  0.0   1.0
18 :  0.0   1.0
19 :  0.0   1.0
20 :  0.0   1.0
```

Figure 2: Q3-2

```
20 :  0.0   1.0
21 :  0.0   1.0
22 :  0.0   1.0
23 :  0.0   1.0
24 :  0.0   1.0
25 :  0.0   1.0
26 :  0.0   1.0
27 :  0.0   1.0
28 :  0.0   1.0
29 :  0.0   1.0
30 :  0.0   1.0
31 :  0.0   1.0
32 :  0.0   1.0
33 :  0.0   1.0
34 :  0.0   1.0
35 :  0.0   1.0
36 :  0.0   1.0
37 :  0.0   1.0
38 :  0.0   1.0
39 :  0.0   1.0
40 :  0.0   1.0
41 :  0.0   1.0

------------------------(d)------------------------
01 :  1   0.6375398756657026
02 :  4.58257569495584   0.6610009561650232
03 :  2.1406951429280725   0.22408337233246423
04 :  3.132075594919797   0.1352525095590482
05 :  2.5893665274210145   0.06145788652391112
06 :  2.847822274447193   0.03222201567811121
07 :  2.715521252632272   0.015731829138479647
08 :  2.7808850947101225   0.007959957188025472
09 :  2.748008838379827   0.00395637476909969
10 :  2.764398093397935   0.0019840766425103067
11 :  2.7561912839020932   0.0009905645477404227

In [2]:
```

Figure 3: Q3-3

■

Question 4.

(a) Let p be the root of $f(x) = 0$, with f is at least C^2, please show that the Newton method for finding the root of f is at least quadratic.

(b) Show that the rate of convergence of secant method is $\frac{1+\sqrt{5}}{2}$.

Solution: (a) According to the Taylor's theorem, any function $f(x)$ which has a continuous second derivative can be represented by an expansion about a point that is close to a root of $f(x)$.

Suppose the root is α, then the expansion of $f(\alpha)$ about x_n is

$f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + R_1$,

where $R_1 = \frac{1}{2!}f''(\xi_n)(\alpha - x_n)^2$ and ξ_n is between x_n and α.

Since α is the root, $0 = f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + \frac{1}{2}f''(\xi_n)(\alpha - x_n)^2$.

$\Longrightarrow \frac{f(x_n)}{f'(x_n)} + (\alpha - x_n) = \frac{-f''(\xi_n)}{2f'(x_n)}(\alpha - x_n)^2$.

Since $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, we have $\alpha - x_{n+1} = \frac{-f''(\xi_n)}{2f'(x_n)}(\alpha - x_n)^2$

$\overset{let\ \epsilon_n = \alpha - x_n}{\Longrightarrow} \epsilon_{n+1} = \frac{-f''(\xi_n)}{2f'(x_n)} \cdot \epsilon_n^2 \Longrightarrow |\epsilon_{n+1}| = \frac{|f''(\xi_n)|}{2|f'(x_n)|} \cdot \epsilon_n^2 \sim (*)$.

Hence, if the following conditions are satisfied:

(1) $f'(x) \neq 0 \forall x \in I = [\alpha - r, \alpha + r] for some r \geq |\alpha - x_0|$;

(2) $f''(x)$ is continuous, $\forall x \in I$;

(3) x_0 sufficiently close to the root α.

By $(*)$, the rate of convergence of Newton method for finding the root is at least quadratic.

(b) Let p satisfying $f(p) = 0$, and let p_{k-1} and p_k be two approximations to p.

Denote $f(p_k)$ as f_k.

If we take next approximation to p passing through (p_{k-1}, f_{k-1}) and (p_k, f_k),

then we have: $p_{k+1} = p_k - \frac{p_k - p_{k-1}}{f_k - f_{k-1}} f_k$.

Define the error at the k^{th} step to be $e_k = p_k - p$ and using Taylor approximation

$f(p + e_k) = e_k \cdot f'(p) + e_k^2 \cdot \frac{f''(p)}{2} + O(e_k^3)$ with $f'(p) \neq 0$.

Then $e_{k+1} = p_{k+1} - p = p_k - \frac{p_k - p_{k-1}}{f_k - f_{k-1}} f_k - p = \frac{(p_{k+1} - p) \cdot f_k - (p_k - p) \cdot f_{k-1}}{f_k - f_{k-1}} = \frac{e_{k-1} \cdot f_k - e_k \cdot f_{k-1}}{f_k - f_{k-1}}$

$= \frac{e_{k-1} \cdot f(p+e_k) - e_k \cdot f(p+e_{k-1})}{f(p+e_k) - f(p+e_{k-1})} = \frac{e_{k-1} \cdot (e_k \cdot f'(p) + e_k^2 \cdot \frac{f''(p)}{2} + O(e_k^3)) - e_k \cdot (e_{k-1} \cdot f'(p) + e_{k-1}^2 \cdot \frac{f''(p)}{2} + O(e_{k-1}^3))}{e_k \cdot f'(p) + e_k^2 \cdot \frac{f''(p)}{2} + O(e_k^3) - (e_{k-1} \cdot f'(p) + e_{k-1}^2 \cdot \frac{f''(p)}{2} + O(e_{k-1}^3)}$

$= \frac{e_{k-1} \cdot e_k \cdot f''(p) \cdot \frac{(e_k - e_{k-1})}{2} + O(e_{k-1}^4)}{(e_k - e_{k-1}) \cdot (f'(p) + (e_k + e_{k-1}) \cdot \frac{f''(p)}{2} + O(e_{k-1}^2))} = \frac{e_{k-1} e_k f''(p)}{2f'(p)} + O(e_{k-1}^3)$.

Since want to find α such that $|e_{k+1}| = C|e_k|^\alpha$, we solve $\left|\frac{e_{k-1} e_k f''(p)}{2f'(p)}\right| = C|e_k|^\alpha$.

Let $|e_{k+1}|^{\alpha-1} = D|e_k|$, where $D = \left|\frac{f''(p)}{2Cf'(p)}\right|$. Then $\left|e_{k+1}^{\alpha(\alpha-1)}\right| = D^\alpha |e_k|^\alpha$.

Above forces $C = D^\alpha$ and $\alpha(\alpha - 1) = 1 \Longrightarrow \alpha = \frac{1+\sqrt{5}}{2} \approx 1.618$.

Furthermore, the asymptotic error constant must be $C = \left|\frac{f''(p)}{2f'(p)}\right|^{\alpha-1} \approx \left|\frac{f''(p)}{2f'(p)}\right|^{0.618}$.

■

Question 5.

Please use Newton method to find the root of $f(x) = x(x-1)^2$ where $x_0 = 1.5$.

Solution: The repeated root of $f(x)$ found by Newton method is 1.

```
import numpy as np

TOL = 10**(-12)

def f(x):
    return x*((x-1)**2)
def Df(x):
    return (x-1)**2 + 2*x*(x-1)
def D2f(x):
    return 2*(x-1)+2*((x-1)+x)

print('--------Newton method on f--------')
x_0 = 1.5
x = 0
xt = 0
for i in range(10**3):
    xt = x
    if (i==0):
        x = x_0-(f(x_0)/Df(x_0))
        print(x)
    else:
        x = x-(f(x)/Df(x))
        print(x)
    if (np.abs(x-xt)<TOL) or (np.abs(x-xt)==0):
        break
print('\n--------Newton method on derivative of f--------')
x_0 = 1.5
x = 0
xt = 0
for i in range(10**3):
    xt = x
    if (i==0):
        x = x_0-(Df(x_0)/D2f(x_0))
```

```
            print(x)
        else:
            x = x-(Df(x)/D2f(x))
            print(x)
        if (np.abs(x-xt)<TOL) or (np.abs(x-xt)==0):
            break

print('\nSince f(1)=0 and f\'(1)=0, f has repeated root at x=1.')
```

10/8/2020 hw1_5.html

```
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]:
runfile('D:/_First_Semester_of_Senior_Year/Numerical_Analysis(1)/homeworks/hw1/codes/hw1_5.py',
wdir='D:/_First_Semester_of_Senior_Year/Numerical_Analysis(1)/homeworks/hw1/codes')
--------Newton method on f--------
1.2857142857142858
1.1571428571428573
1.083567299752271
1.0433350533716832
1.022108353517131
1.0111724493635321
1.0056169162381714
1.0028162796567097
1.0014101143449448
1.0007055532288445
1.0003529009341905
1.00017648158539
1.0000882485770717
1.000044126235231
1.0000220636043644
1.0000110319238789
1.0000055159923649
1.000002758003789
1.000001379003796
1.0000006895023734

--------Newton method on derivative of f--------
1.15
1.0232758620689655
1.00075960710217
1.000000863536579
1.0000000000011184

Since f(1)=0 and f'(1)=0, f has repeated root at x=1.

In [2]:
```

Figure 4: Q5

■

Question 6.

A sequence $\{p_n\}$ is said to be superlinear convergent to p if $\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|} = 0$.

(a) Show that if $p_n \to p$ in order $\alpha > 1$, then $\{p_n\}$ is superlinear convergent to p.

(b) Show that $p_n = \frac{1}{n^n}$ is superlinear convergent to 0 but does not convergent to 0 of order $\alpha > 1$.

Solution: (a) Suppose that $\lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|^\alpha} = \lambda$, where $\alpha > 1$ and $\lambda \neq 0$.

Then $\lim_{n\to\infty} \left| \frac{p_{n+1}-p}{p_n-p} \right| = \lim_{n\to\infty} \frac{|p_{n+1}-p|}{|p_n-p|^\alpha} |p_n - p|^{\alpha-1} = \lambda \cdot \lim_{n\to\infty} |p_n - p|^{\alpha-1} = \lambda \cdot 0 = 0$.

(b) Check the convergence is superlinear: $\lim_{n\to\infty} \frac{\frac{1}{(n+1)^{n+1}}}{\frac{1}{n^n}} = \lim_{n\to\infty} \frac{n^n}{(n+1)^{n+1}}$

$= \lim_{n\to\infty} (\frac{n}{n+1})^n (\frac{1}{n+1}) \overset{\lim_{n\to\infty}(1+\frac{1}{n})^n = \lim_{n\to\infty}(\frac{n+1}{n})^n = e}{=} e^{-1} \cdot 0 = 0$.

Suppose that $\alpha > 1$, then $\lim_{n\to\infty} \frac{\frac{1}{(n+1)^{n+1}}}{(\frac{1}{n^n})^\alpha} = \lim_{n\to\infty} \frac{n^{\alpha n}}{(n+1)^{n+1}} = (\lim_{n\to\infty} \frac{n^n}{(n+1)^n}) \cdot (\lim_{n\to\infty} \frac{n^{(\alpha-1)n}}{n+1})$

$= e^{-1} \cdot (\lim_{n\to\infty} \frac{n^{(\alpha-1)n}}{n+1}) = e^{-1} \cdot (\lim_{n\to\infty} \frac{n}{n+1}) \cdot \lim_{n\to\infty} n^{(\alpha-1)n-1} = \frac{\lim_{n\to\infty} n^{(\alpha-1)n-1}}{e} = \infty$.

■

Question 7.

You should write functions for Newton method and secant method and paste your code here.

Use Newton and secant methods to find solution accurate within TOL $= 10^{-5}$ for the following problems.

(a) $2x\cos(2x) - (x-2)^2 = 0$ for $x \in [2, 3]$ and $[3, 4]$.

(b) $e^x - 3x^2 = 0$ for $x \in [0, 1], [3, 4], and [6, 7]$.

Solution:

```
import numpy as np


def f_a(x):
    return 2*x*np.cos(2*x) - (x-2)**2
def Df_a(x):
    return 2*(np.cos(2*x)+x*2*(-np.sin(2*x))) - 2*(x-2)
def f_b(x):
    return np.e**x - 3*x**2
def Df_b(x):
    return np.e**x - 6*x


def newtonMethod_a(p_0, TOL):
    x2 = 10**12
    x = p_0
    while 1:
```

```
        x = x-(f_a(x)/Df_a(x))
        if (np.abs(x2-x)<TOL):
            return x
        else:
            x2 = x

def newtonMethod_b(p_0, TOL):
    x2 = 10**12
    x = p_0
    while 1:
        x = x-(f_b(x)/Df_b(x))
        if (np.abs(x2-x)<TOL):
            return x
        else:
            x2 = x

def secantMethod_a(p_0, p_1, TOL):
    x0 = p_0
    x1 = p_1
    x2 = 10**10
    while 1:
        temp = x1
        x2 = x1 - f_a(x1)*((x1-x0)/(f_a(x1)-f_a(x0)))
        if (np.abs(x2-x1)<TOL):
            return x2
        x1 = x2
        x0 = temp

def secantMethod_b(p_0, p_1, TOL):
    x0 = p_0
    x1 = p_1
    x2 = 10**10
    while 1:
        temp = x1
        x2 = x1 - f_b(x1)*((x1-x0)/(f_b(x1)-f_b(x0)))
        if (np.abs(x2-x1)<TOL):
            return x2
```

```
        x1 = x2
        x0 = temp

print('-----------------------(a)-----------------------')
print('----------------Newton Method----------------')
p_0 = input('input p0 in range [2,3]: ') #2.5
TOL = input('input TOL: ') #10**(-6)
print('x = ', newtonMethod_a(float(p_0), float(TOL)))
p_0 = input('input p0 in range [3,4]: ') #3.5
TOL = input('input TOL: ') #10**(-6)
print('x = ', newtonMethod_a(float(p_0), float(TOL)))

print('\n----------------Secant Method----------------')
p_0 = input('input p0 in range [2,3]: ') #2
p_1 = input('input p1 in range [2,3]: ') #3
TOL = input('input TOL: ') #10**(-6)
print('x = ', secantMethod_a(float(p_0), float(p_1), float(TOL)))
p_0 = input('input p0 in range [3,4]: ') #3
p_1 = input('input p1 in range [3,4]: ') #4
TOL = input('input TOL: ') #10**(-6)
print('x = ', secantMethod_a(float(p_0), float(p_1), float(TOL)))

print('\n\n-----------------------(b)-----------------------')
print('----------------Newton Method----------------')
p_0 = input('input p0 in range [0,1]: ') #0.5
TOL = input('input TOL: ') #10**(-6)
print('x = ', newtonMethod_b(float(p_0), float(TOL)))
p_0 = input('input p0 in range [3,4]: ') #3.5
TOL = input('input TOL: ') #10**(-6)
print('x = ', newtonMethod_b(float(p_0), float(TOL)))
p_0 = input('input p0 in range [6,7]: ') #6.5
TOL = input('input TOL: ') #10**(-6)
print('x = ', newtonMethod_b(float(p_0), float(TOL)))

print('\n----------------Secant Method----------------')
p_0 = input('input p0 in range [0,1]: ') #0
p_1 = input('input p1 in range [0,1]: ') #1
```

```python
TOL = input('input TOL: ') #10**(-6)
print('x = ', secantMethod_b(float(p_0), float(p_1), float(TOL)))
p_0 = input('input p0 in range [3,4]: ') #3
p_1 = input('input p1 in range [3,4]: ') #4
TOL = input('input TOL: ') #10**(-6)
print('x = ', secantMethod_b(float(p_0), float(p_1), float(TOL)))
p_0 = input('input p0 in range [6,7]: ') #6
p_1 = input('input p1 in range [6,7]: ') #7
TOL = input('input TOL: ') #10**(-6)
print('x = ', secantMethod_b(float(p_0), float(p_1), float(TOL)))
```

10/7/2020 hw1_Q7_result.html

```
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/_First_Semester_of_Senior_Year/Numerical_Analysis(1)/homeworks/hw1/hw1_Q7.py',
wdir='D:/_First_Semester_of_Senior_Year/Numerical_Analysis(1)/homeworks/hw1')
------------------------(a)------------------------
----------------Newton Method----------------

input p0 in range [2,3]: 2.5

input TOL: 0.0001
x =  2.370686917662517

input p0 in range [3,4]: 3.5

input TOL: 0.0001
x =  3.722112773106613

----------------Secant Method----------------

input p0 in range [2,3]: 2

input p1 in range [2,3]: 3

input TOL: 0.0001
x =  2.370686907966889

input p0 in range [3,4]: 3

input p1 in range [3,4]: 4

input TOL: 0.0001
x =  3.722112773420417
```

Figure 5: Q7-1

```
------------------------(b)------------------------
----------------Newton Method----------------

input p0 in range [0,1]: 0.5

input TOL: 0.0001
x =  0.9100075724887138

input p0 in range [3,4]: 3.5

input TOL: 0.0001
x =  3.733079028804883

input p0 in range [6,7]: 6.5

input TOL: 0.0001
x =  3.7330790286329383

----------------Secant Method----------------

input p0 in range [0,1]: 0

input p1 in range [0,1]: 1

input TOL: 0.0001
x =  0.9100075715386231
```

file:///D:/_First_Semester_of_Senior_Year/Numerical_Analysis(1)/homeworks/hw1/hw1_Q7_result.html

10/7/2020 hw1_Q7_result.html

```
input p0 in range [3,4]: 3

input p1 in range [3,4]: 4

input TOL: 0.0001
x =  3.7330790326819776

input p0 in range [6,7]: 6

input p1 in range [6,7]: 7

input TOL: 0.0001
x =  3.7330791029591563

In [2]:
```

Figure 6: Q7-2

■