TW324 Applied Mathematics Assignment 01

Author: 18998712

02 March 2018

Question 1

a.)

Python Source Code

```
def question1a():
    step, x = 1./10,1.0
    print "=" * 64
    for i in xrange(0,10):
        E1 = (1-cos(x)) / pow(sin(x),2)
        E2 = 1 / ( 1 + cos(x))
        print( format(x, ".14f") + "\t" + format(E1, ".14f") + "\t" + format(
        x = x * step
    print "=" * 64

question1a() #call code for question 1a
```

x	E_1	E_2
1.00000000000000	0.64922320520476	0.64922320520476
0.10000000000000	0.50125208628857	0.50125208628857
0.01000000000000	0.50001250020848	0.50001250020834
0.00100000000000	0.50000012499219	0.50000012500002
0.00010000000000	0.49999999862793	0.50000000125000
0.00001000000000	0.50000004138685	0.50000000001250
0.00000100000000	0.50004445029134	0.50000000000013
0.00000010000000	0.49960036108132	0.50000000000000
0.00000001000000	0.00000000000000	0.50000000000000
0.00000000100000	0.00000000000000	0.50000000000000

b.)

Python Source Code

```
def question_b():
    print "=" * 64
    step, x = 1./10,1.0
    for i in xrange(0,10):
        F1 = ((1-1/cos(x))/ pow(sin(x)/cos(x),2))
        F2 = -1*(cos(x) / ( 1 + cos(x)))
        print( format(x, ".14f") + "\t" + format(F1, ".14f") + "\t" + format(
        x = x * step
    print "=" * 64
question_b() #call code for question 1a
```

x	F_1	F_2
1.00000000000000	-0.35077679479524	-0.35077679479524
0.10000000000000	-0.49874791371141	-0.49874791371143
0.01000000000000	-0.49998749979096	-0.49998749979166
0.00100000000000	-0.49999987501429	-0.49999987499998
0.00010000000000	-0.49999999362793	-0.49999999875000
0.00001000000000	-0.50000004133685	-0.49999999998750
0.00000100000000	-0.50004445029084	-0.49999999999987
0.00000010000000	-0.51070259132757	-0.50000000000000
0.00000001000000	0.00000000000000	-0.50000000000000
0.00000000100000	0.00000000000000	-0.50000000000000

Question 2

Python Source Code

```
#for the square root function and absolute value
from numpy import (sqrt, abs )
def question2(debug=True):
a, b, c = 1.0, -10000.0, 1.0

    xP = (-1*b + sqrt(pow(b,2) - 4*a*c))/ 2*a
    xM = (-1*b - sqrt(pow(b,2) - 4*a*c))/ 2*a

    if debug is True:
```

```
        print("x+ = " + str(format(xP, ".20f")))
        print("x- = " + str(format(xM, ".20f")))

    xM2 = c / (a * xP)
    if debug is True:
        print format(xM2, ".20f")

question2()  #call code for question 2
```

	Quadratic	Roots
X_+	9.999999899999999E+03	9.999999899999999E+03
X_-	1.000000011117663E-04	1.000000010000000E-04

Question 3

a.)

	Approximated Values	built-in values	Absolute Error
$J_0(1)$	0.765190972222	0.765197686558	6.6.7143357444e-06
$J_1(1)$	0.440049913194	0.440050585745	6.72550489078e-07

b.)

	Approximated Values	built-in values	Absolute Error
$J_0(1)$	0.765190972222	0.765197686558	6.6.7143357444e-06
$J_1(1)$	0.440049913194	0.440050585745	6.72550489078e-07
$J_2(1)$	0.114908854167	0.114903484932	5.36923476624e-06
$J_3(1)$	0.0195855034722	0.0195633539827	2.21494895541e-05
$J_4(1)$	0.00260416666667	0.00247663896411	0.000127527702558
$J_5(1)$	0.00124782986112	0.000249757730211	0.000998072130912
$J_6(1)$	0.00987413194456	2.09383380024e-05	0.00985319360656
$J_7(1)$	0.117241753474	1.50232581744e-06	0.117240251148

c.)

	Approximated Values	built-in values	Absolute Error
$J_0(1)$	0.765190972222	0.765197686558	6.6.7143357444e-06
$J_1(1)$	0.440049913194	0.440050585745	6.72550489078e-07
$J_2(1)$	0.114908854167	0.114903484932	5.36923476624e-06
$J_3(1)$	0.0195855034722	0.0195633539827	2.21494895541e-05
$J_4(1)$	0.00260416666667	0.00247663896411	0.000127527702558
$J_5(1)$	0.00124782986112	0.000249757730211	0.000998072130912
$J_6(1)$	0.00987413194456	2.09383380024e-05	0.00985319360656
$J_7(1)$	0.117241753474	1.50232581744e-06	0.117240251148

d.)

Algorithm b is more stable compared to algorithm c because with algorithm c tends not to magnify the a

Python Source Code: 1.b)