3 MARCH 2022

NUMERICAL COMPUTING (CS325)

Course Instructor:
Sir Jamil Usmani

NUMERICAL COMPUTING (CS325)
PROJECT
LAB – 1

Group Members:

- Mohammad Basil Ali Khan (20K-0477)
- Ali Jodat (20K-0155)
- Abdul Ahad Shaikh (20K-0319)
- Mohammad Umer (20K-0225)

Project Title:

LAB 1: Solution of Non Linear Equation in one Variable f(x) = 0

Aim:

To understand the fundamental concepts of scientific programming using python.

Description:

We selected three methods of Lab1.

1. Bisection Method
2. Regular Falsi Method
3. Secant Method

First we have studied the algorithm of then we have written the programming of that method.

IDE and Programming Language:

We have chosen python programming language and IDE we are using is Visual Studio Code.

Library Used:

We have imported 3 libraries:

1. sympy library for to get equation solution on particular intervals and can initialize symbols.
2. tabulate library to generated table on each iteration.
3. array library to save each iteration values to use in next iteration.

Implementation and Code Snippets:

✓ Bisection Method:

Formula:

$$c = \frac{a+b}{2}\ ; where\ a\ and\ b\ are\ intervals\ [a, b]$$

Algorithm:

Step 1: Find two points, say a and b such that a < b and f(a)* f(b) < 0

Step 2: Find the midpoint of a and b, say "c"

Step 3: c is the root of the given function if f(c) = 0; else follow the next step

Step 4: Divide the interval [a, b] – If f(c)*f(a) <0, there exist a root between t and a
else if f(c) *f (b) < 0, there exist a root between t and b

Step 5: Repeat above three steps until f(c) = 0.

Code Snippets:

Bisection_method.py

```python
from sympy import *
from tabulate import tabulate
from array import *

def getFunc():
    func = input("Enter Function: ")
    return func

def getIntervals():
    print("Enter Intervals: ")
    a = input("a = ")
    b = input("b = ")
    return a, b

def f(eq, num):
    x = float(num)
    return eval(eq)

Iteration = 0
Func = getFunc()
a, b = getIntervals()
col_names = ["n", "a", "b", "c", "f(c)"]
arr = [[]]
root_check = false
c = (float(a)+float(b))/2
if(f(Func, a) * f(Func, b) < 0):
    arr.insert(Iteration, [Iteration, a, b, c, f(Func, c)])
    while(f(Func, c) != 0):
        c = (float(a)+float(b))/2
        if f(Func, a) * f(Func, c) < 0:
            b = c
        else:
            a = c
        Iteration += 1
        arr.insert(Iteration, [Iteration, a, b, c, f(Func, c)])
        ans = f(Func, c)
    print(tabulate(arr, headers=col_names, tablefmt="fancy_grid"))
    print("Root: ", format(c, ".5f"))
    print("Number of Iterations: ", Iteration)
    print("f(c) = ", ans)
else:
    print("Root is not availble in given Intervals. ")
```

Input:

$x^3 + 4x^2 - 10 = 0\,;[1,2]$

Output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\ftc\Desktop\NC Project> & C:/Python/Python310/python.exe "c:/Users/ftc/Desktop/NC Project/Bisection_method.py"
Enter Function: x**3 + 4*x**2 - 10
Enter Intervals:
a = 1
b = 2
```

n	a	b	c	f(c)
0	1	2	1.5	2.375
1	1	1.5	1.5	2.375
2	1.25	1.5	1.25	-1.79688
3	1.25	1.375	1.375	0.162109
4	1.3125	1.375	1.3125	-0.848389
5	1.34375	1.375	1.34375	-0.350983
6	1.35938	1.375	1.35938	-0.0964088
7	1.35938	1.36719	1.36719	0.0323558
8	1.36328	1.36719	1.36328	-0.03215
9	1.36328	1.36523	1.36523	7.20248e-05
10	1.36426	1.36523	1.36426	-0.0160467
11	1.36475	1.36523	1.36475	-0.00798926

35	1.36523	1.36523	1.36523	-2.3789e-11
36	1.36523	1.36523	1.36523	2.16513e-10
37	1.36523	1.36523	1.36523	9.6362e-11
38	1.36523	1.36523	1.36523	3.62874e-11
39	1.36523	1.36523	1.36523	6.24922e-12
40	1.36523	1.36523	1.36523	-8.76987e-12
41	1.36523	1.36523	1.36523	-1.26121e-12
42	1.36523	1.36523	1.36523	2.49401e-12
43	1.36523	1.36523	1.36523	6.16396e-13
44	1.36523	1.36523	1.36523	-3.21521e-13
45	1.36523	1.36523	1.36523	1.47438e-13
46	1.36523	1.36523	1.36523	-8.70415e-14
47	1.36523	1.36523	1.36523	3.01981e-14
48	1.36523	1.36523	1.36523	-2.84217e-14
49	1.36523	1.36523	1.36523	0

```
Root:  1.36523
Number of Iterations:  49
f(c) =  0.0
PS C:\Users\ftc\Desktop\NC Project>
```

✓ Secant Method:

Formula:

$$\frac{A(f(b)) - B(f(A))}{f(B) - f(A)} \text{ ; } where\ A\ and\ B\ are\ intervals\ [A, B]$$

Algorithm:

Given an equation f(c) = 0
Let the initial guesses be a and b
Do

$$c = \frac{a(f(b)) - b(f(a))}{f(b) - f(a)}$$

a = b

b = c

while (f(c) not equals 0)

Code Snippets:

Secant_method.py

```python
from sympy import *
from tabulate import tabulate
from array import *

def getFunc():
    func = input("Enter Function: ")
    return func

def getIntervals():
    print("Enter Intervals: ")
    a = input("a = ")
    b = input("b = ")
    return a, b

def f(eq, num):
    x = float(num)
    return eval(eq)

Iteration = 0
Func = getFunc()
a, b = getIntervals()
col_names = ["n", "a", "b", "c", "f(c)"]
arr = [[]]
root_check = false
if f(Func, a) * f(Func, b) < 0:
    c = float((float(a)*f(Func, b) - float(b)*f(Func, a))/((f(Func, b) - f(Func, a))))
    arr.insert(Iteration, [Iteration, a, b, c, f(Func, c)])
    while f(Func, c) != 0:
        a = b
        b = c
        c = float((float(a)*f(Func, b) - float(b)*f(Func, a))/((f(Func, b) - f(Func, a))))
        Iteration +=1
        arr.insert(Iteration, [Iteration, a, b, c, f(Func, c)])
    print(tabulate(arr, headers=col_names, tablefmt="fancy_grid"))
    print("Root: ", c)
    print("Number of Iterations: ", Iteration)
else:
    print("Root is not availble in given Intervals. ")
```

Input :

$x - \cos x$; $[0, 1]$

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\ftc\Desktop\NC Project> & C:/Python/Python310/python.exe "c:/Users/ftc/Desktop/NC Project/Secant_method.py"
Enter Function: x - cos(x)
Enter Intervals:
a = 0
b = 1
```

n	a	b	c	f(c)
0	0	1	0.685073	-0.0892993
1	1	0.685073	0.736299	-0.00466004
2	0.685073	0.736299	0.739119	5.7286e-05
3	0.736299	0.739119	0.739085	-3.52926e-08
4	0.739119	0.739085	0.739085	-2.66787e-13
5	0.739085	0.739085	0.739085	0

```
Root:  0.7390851332151607
Number of Iterations:  5
PS C:\Users\ftc\Desktop\NC Project>
```

✓ Regular Falsi Method:

Formula:

$$\frac{A(f(b)) - B(f(A))}{f(B) - f(A)}\ ; where\ A\ and\ B\ are\ intervals\ [A, B]$$

Algorithm:

Step 1: Find two points, say a and b such that a < b and f(a)* f(b) < 0

Step 2: $$c = \frac{a(f(b))-b(f(a))}{f(b)-f(a)}$$

Step 3: c is the root of the given function if f(c) = 0; else follow the next step

Step 4: Divide the interval [a, b] – If f(c)*f(a) <0, there exist a root between t and a else if f(c) *f (b) < 0, there exist a root between t and b

Step 5: Repeat above three steps until f(c) = 0.

Code Snippets:

```python
from sympy import *
from tabulate import tabulate
from array import *

def getFunc():
    func = input("Enter Function : ")
    return func

def getIntervls():
    print("Enter Intervals : ")
    a = input("a = ")
    b = input("b = ")
    return a,b

def f(eq, num):
    x = float(num)
    return eval(eq)

Iteration = 0
Func = getFunc()
a,b = getIntervls()
col_names = ["n", "a", "b", "c", "f(c)"]
arr = [[]]

c = float(float(a) * f(Func,b) - float(b) * f(Func,a))/(f(Func,b) - f(Func,a))

if(f(Func,a) * f(Func,b) < 0):
    while f(Func , c) != 0:
        arr.insert(Iteration, [Iteration, a, b, c, f(Func, c)])
        c = float(float(a) * f(Func,b) - float(b) * f(Func,a))/(f(Func,b) - f(Func,a))

        if(f(Func, a) * f(Func, c) < 0):
            b = c
        else:
            a = c

        Iteration += 1
        ans = f(Func, c)

    arr.insert(Iteration, [Iteration, a, b, c, f(Func, c)])
    print(tabulate(arr, headers=col_names, tablefmt="fancy_grid"))
    print("Root : ", c)
    print("Number of Iterations performed : ", Iteration)
    print("f(c) = ", ans)
else:
    print("Root is not available in given Intervals. ")
```

Input :

$$x^3 + 4x^2 - 10 = 0\,;[1,2]$$

Output:

```
PS C:\Users\Dell\Documents\BS(CS)NUCES\BS(CS) Course related material\Numerical_Computin
ng\Lab_1\Regular_falsi.py"
Enter Function : x**3 + 4*x**2 -10
Enter Intervals :
a = 1
b = 2
```

n	a	b	c	f(c)
0	1	2	1.26316	-1.60227
1	1.26316	2	1.26316	-1.60227
2	1.33883	2	1.33883	-0.430365
3	1.35855	2	1.35855	-0.110009
4	1.36355	2	1.36355	-0.0277621
5	1.36481	2	1.36481	-0.00698342
6	1.36512	2	1.36512	-0.00175521
7	1.3652	2	1.3652	-0.000441063
8	1.36522	2	1.36522	-0.000110828
21	1.36523	2	1.36523	-1.7657e-12
22	1.36523	2	1.36523	-4.47642e-13
23	1.36523	2	1.36523	-1.13687e-13
24	1.36523	2	1.36523	-2.84217e-14
25	1.36523	2	1.36523	-7.10543e-15
26	1.36523	2	1.36523	-3.55271e-15
27	1.36523	2	1.36523	0

```
Root :  1.3652300134140969
Number of Iterations performed :  27
f(c) =  0.0
PS C:\Users\Dell\Documents\BS(CS)NUCES\BS(CS) Course related material\Numerical_Computing\Lab_1>
```

CONCLUSION :

We can analyze the speed or the efficiency of each method by performing all the methods for a single equation.

And we have demonstrated it below :

INPUT : $x^3 + 4x^2 - 10 = 0$; $[1, 2]$

Using Bisection :

```
Root:  1.36523
Number of Iterations:  48
f(c) =  0.0
```

Using Regular Falsi :

```
Root :  1.3652300134140969
Number of Iterations performed :  27
f(c) =  0.0
```

Using Secant :

```
Root:  1.36523001
Number of Iterations:  7
f(c) :  0.0
```