

School of Engineering, Computing and Mathematical Sciences

4MM013: Computational Mathematics

4MM013-UM1 (2023-2024)

Portfolio Assignment Briefing

IMPORTANT – This is an **individual** assessment, so each student needs to work on it individually and submit it individually via Canvas.

Weightage: 30% of total module mark [100 points]

Pre-requisite: Student should have some experience of using NumPy and Matplotlib libraries, which they have learned during the lectures and workshops.

What is provided to the student?

The pseudocode/algorithm and Python Code template.

What the student is supposed to do?

Task 1 (12%) [40 points]: Implement Root Finding methods, particularly, **Bisection** and **Newton-Raphson** methods, to find all the possible roots of a given function and verify it with built-in functions `scipy.optimize.root()` in the SciPy library. Algorithm/Pseudocode for the two methods are given below.

What is expected from the student?

1. Write Python codes to plot the given function using Matplotlib library. This will give an idea where the roots of the given function are located.

Mark: 4 points

2. Write Python codes to implement Bisection method

Mark: 18 points

3. Write Python codes to Newton-Raphson Methods

Mark: 18 points

Given Functions are:

1. $y = f(x) = x^2 - x - 1$

2. $y = f(x) = x^3 - x^2 - 2x + 1$

Find all possible roots of these functions using your implemented methods and compare the obtained roots with the roots found by built-in function `scipy.optimize.root()`.

School of Engineering, Computing and Mathematical Sciences

4MM013: Computational Mathematics

Pseudocode for Bisection Method:

```

input: Function  $f$ ,
        endpoint values  $a, b$ ,
        tolerance  $TOL$ ,
        maximum iterations  $NMAX$ 
conditions:  $a < b$ ,
                $f(a)f(b) < 0$ 
output: value for which  $f(x) \approx 0$  or  $(b - a)$  less than  $TOL$ 

 $N \leftarrow 1$ 
while  $N \leq NMAX$  do # limit iterations to prevent infinite loop
     $c \leftarrow (a + b)/2$  # new midpoint
    if  $\text{abs}(f(c)) \approx 0$  or  $(b - a) \leq TOL$  then # solution found
        Output( $c$ )
        Stop
    end if
     $N \leftarrow N + 1$  # increment step counter
    if  $\text{sign}(f(c)) = \text{sign}(f(a))$  then  $a \leftarrow c$  else  $b \leftarrow c$  # new interval
end while
Print "Warning! Method exceeded maximum number of iterations".
Output( $c$ )

```

Pseudocode for Newton-Raphson Method

```

input: Function  $f$  and its gradient  $g$ 
        Initial guess  $x_0$ ,
        tolerance  $TOL$ ,
        maximum iterations  $NMAX$ 
output: value which differs from a root of  $f(x) = 0$  by less than  $TOL$ 

 $N \leftarrow 1$ 
while  $N \leq NMAX$  do # limit iterations to prevent infinite loop
    if  $\text{abs}(g(x_0)) \leq 1E-12$ 
        Print "Mathematical Error! Found root may not be correct."
        Output( $x_0$ )
        Stop
    end if
     $x_1 \leftarrow x_0 - f(x_0)/g(x_0)$  # new approximation of root
     $x_0 \leftarrow x_1$ 
    if  $\text{abs}(f(x_0)) \leq TOL$  then // solution found
        Output( $x_0$ )
        Stop
    end if
     $N \leftarrow N + 1$  # increment step counter
end while
Print "Warning! Method exceeded maximum number of iterations".
Output( $x_0$ )

```

School of Engineering, Computing and Mathematical Sciences

4MM013: Computational Mathematics

Task 2 (12%) [40 points]: Implement *Midpoint Approximation* and *Trapezium Rule* to perform numerical integration of a given function and limit, and compare the result with the Analytical result. For help and algorithms, see Week 10 Lecture and Workshop.

What is expected from the student?

1. Write Python code to plot the given function using Matplotlib library

Mark: 4 points

2. Write Python code to implement Midpoint Approximation

Mark: 18 points

3. Write Python codes to implement Trapezium Rule

Mark: 18 points

- 4.

Given Functions and Limits are:

1. $y = f(x) = \frac{x}{x^2+1}$ and Limits $[0, 5]$

2. $y = f(x) = e^x$ and Limits $[0, 5]$

Use your implemented functions to evaluate the definite integrals of the above functions over their corresponding limits. Verify your answers with the Analytical (true value obtained by definite integrals) results. You may need to change accordingly the number of partitions N to get the approximate results closer to their corresponding analytical values. The indefinite integrals of the above functions are:

1. $\int \frac{x}{x^2+1} dx = \frac{1}{2} \log |1 + x^2| + C$

2. $\int e^x dx = e^x + C,$

which are used to calculate analytical values.

Task 3(6%) [20 points]

A student needs to write a small (1-2 pages) report which should include the results obtained from their implemented methods, and comparison of the results with the reference methods (*scipy.optimize.root()* for **Task 1** and Analytical method for **Task 2**). Include the graph of the functions in both the Tasks.

School of Engineering, Computing and Mathematical Sciences

4MM013: Computational Mathematics

Report should include the following Result Table for **Task 1**

For Function 1			
Bisection Method		Newton's Method	
$[a, b]$; #iterations; Root1	$[a, b]$; #iterations; Root2	x_0 ; #iterations; Root1	x_0 ; #iterations; Root2
Your answer should be here	Your answer should be here	Your answer should be here	Your answer should be here
SciPy Method			
x_0 ; Root1	x_0 ; Root2		
Your answer should be here	Your answer should be here		

For Function 2					
Bisection Method			Newton's Method		
$[a, b]$; #iterations; Root1	$[a, b]$; #iterations; Root2	$[a, b]$; #iterations; Root3	x_0 ; #iterations; Root1	x_0 ; #iterations; Root2	x_0 ; #iterations; Root3
Your answer should be here	Your answer should be here	Your answer should be here	Your answer should be here	Your answer should be here	Your answer should be here
SciPy Method					
x_0 ; Root1	x_0 ; Root2	x_0 ; Root2			
Your answer should be here	Your answer should be here	Your answer should be here			

And should include the following Result Table for **Task 2**

N	Midpoint Approximation (M_N)	Trapezoidal Rule (T_N)	Analytical (I)	Abs Error $ M_N - I $	Abs Error $ T_N - I $
For Function 1					
10	Your answer should be here	Your answer should be here	Your answer should be here	Your answer should be here	Your answer should be here
30	Your answer should be here	Your answer should be here		Your answer should be here	Your answer should be here
50	Your answer should be here	Your answer should be here		Your answer should be here	Your answer should be here
100	Your answer should be here	Your answer should be here		Your answer should be here	Your answer should be here
500	Your answer should be here	Your answer should be here		Your answer should be here	Your answer should be here

School of Engineering, Computing and Mathematical Sciences

4MM013: Computational Mathematics

For Function 2					
10	Your answer should be here	Your answer should be here	Your answer should be here	Your answer should be here	Your answer should be here
30	Your answer should be here	Your answer should be here		Your answer should be here	Your answer should be here
50	Your answer should be here	Your answer should be here		Your answer should be here	Your answer should be here
100	Your answer should be here	Your answer should be here		Your answer should be here	Your answer should be here
500	Your answer should be here	Your answer should be here		Your answer should be here	Your answer should be here

Possibly suggest some other numerical methods to perform the above tasks stating their advantages and disadvantages.

Templates for the Python Code will be provided to you. The Python codes and the report should be submitted via Canvas.

You need submit 5 files:

1. Report (PDF format)
2. BisectionMethod.py
3. NewtonMethod.py
4. MidpointApproxMethod.py
5. TrapezoidalAprroxMethod.py

*.py files are provided to you as templates to fill the required code lines. Do not change their names.