**XJTLU Entrepreneur College (Taicang) Cover Sheet**

|  |  |  |
| --- | --- | --- |
| Module code and Title | **DTS104TC Numerical Methods** | |
| School Title | **School of Artificial Intelligence and Advanced Computing** | |
| Assignment Title | **Assignment 2** | |
| Submission Deadline | **June 2, 2021. 5pm (GMT+8)** | |
| Final Word Count | **1202** | |
| If you agree to let the university use your work anonymously for teaching and learning purposes, please type **“yes”** here. | | **yes** |

I certify that I have read and understood the University’s Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy I certify that:

* My work does not contain any instances of plagiarism and/or collusion.  
  My work does not contain any fabricated data.

**By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Scoring – For Tutor Use** | | | | | | |
| **Student ID** | | | | **1930080** | | |
|  | | | | | | |
| **Stage of Marking** | | **Marker**  **Code** | **Learning Outcomes Achieved （F/P/M/D）**  **(please modify as appropriate)** | | | **Final**  **Score** |
| **A** | **B** | **C** |
| 1st Marker – red pen | |  |  |  |  |  |
| Moderation  – green pen | | **IM**  **Initials** | The original mark has been accepted by the moderator (please circle as appropriate): | | | Y / N |
|  | Data entry and score calculation have been checked by another tutor (please circle): | | | Y |
| 2nd Marker if needed – green pen | |  |  |  |  |  |
| **For Academic Office Use** | | | **Possible Academic Infringement (please tick as appropriate)** | | | |
| **Date**  **Received** | **Days late** | **Late Penalty** | **☐ Category A** | | Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
|  |  |  | **☐ Category B** | |
| **☐ Category C** | |
| **☐ Category D** | |
| **☐ Category E** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **EXAMINER** | **DEPARTMENT** | **Email** |
| DTS104TC | Long Huang | School of Intelligent Manufacturing Ecosystem | Long.Huang@xjtlu.edu.cn |

**2nd SEMESTER 2020/21 Assignment**

**Undergraduate – Year 2**

**DTS104TC Numerical Methods**

**Submission Deadline: June 2, 2021. 5pm (GMT+8)**

**INSTRUCTIONS**

1. **The weighting of this assignment is 20% of the final mark.**
2. **The marking criteria sheet is provided as a supplementary document.**
3. **Your submission should only be in English.**
4. **Answers to questions should be typed on A4 pages as Word files. The assignment must be submitted in Word via Learning Mall Online to the correct drop box. Only electronic submissions are accepted and no hard copy submissions are permitted.**
5. **All students must download their file and check that it is viewable after submission. Documents may become corrupted during the uploading process (e.g. due to slow internet connections). However, students themselves are responsible for submitting a functional and correct file for their assessments.**

|  |  |
| --- | --- |
| **Student Name** | **Student ID** |
| Yaqi Yu | 1930080 |
| **DEPARTMENT** | **Email** |
| School of AI and Advanced Computing | Yaqi.yu19@student.xjtlu.edu.cn |

# DTS104TC Assignment2

[DTS104TC Assignment2](#dts104tc-assignment2)  
 [Q1](#q1)  
 [a)](#a)   
 [Bisection Method](#bisection-method-1)  
 [Netwon-Raphson Method](#netwon-raphson-method)  
 [Secant Method](#secant-method-1)  
 [Conclusion](#conclusion)  
 [b）](#2）)  
 [Calculations](#calculations)  
 [Table](#table-1)  
 [Other matlab source code and table](#other-matlab-source-code-and-table)  
 [Bisection Method](#bisection-method-2)  
 [Table](#table-2)  
 [Secant Method](#secant-method-2)

[Q2](#q2)  
 [a) Analytically](#a-analytically)  
 [b) Trapezoidal Rule](#b-trapezoidal-rule)  
 [c) Trapezoidal Rule(n=2)](#c-trapezoidal-rulen2)  
 [c) Trapezoidal Rule(n=4)](#c-trapezoidal-rulen4)  
 [d) Simpson's 3/8 Rule](#d-simpsons-38-rule)  
 [d) Simpson's 1/3 Rule](#d-simpsons-13-rule)

## Q1

### a)

The author chooses Newton-Raphson method to solve the problem of finding the root of this function. The following is the content of the discussion.

The three root finding methods of Bisection Method,Secant Method and Newton-Raphson are compared respectively, based on the number of iterations, iterative speed and the number of guessed values.

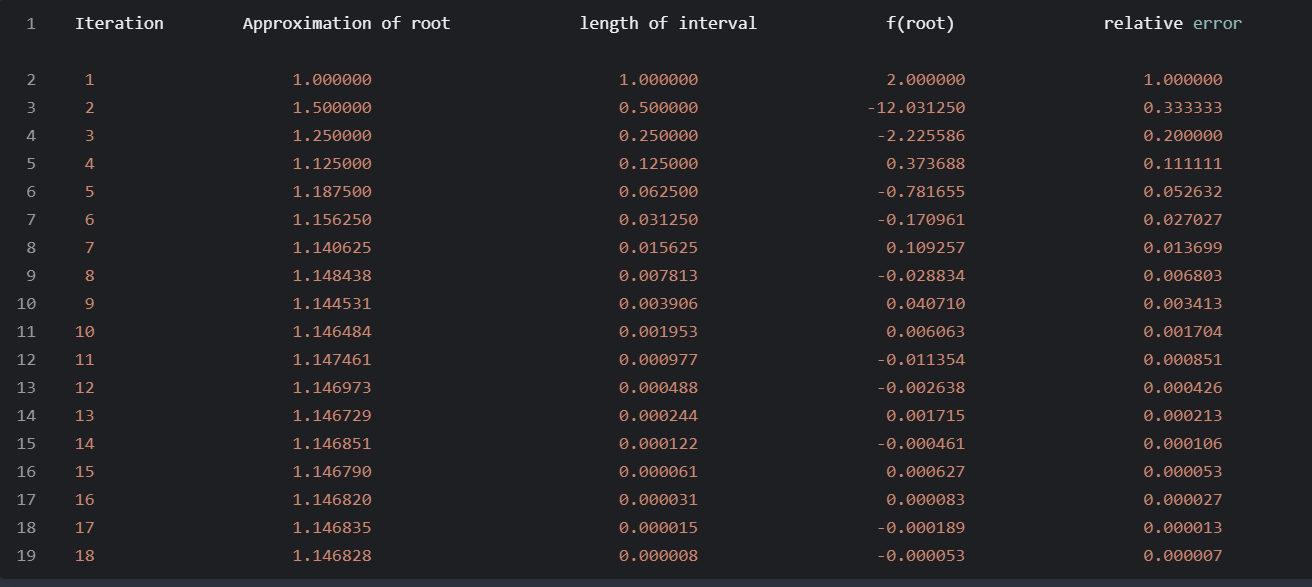
First of all, based on the following Bisection Method, Newton-Raphson and Secant Method matlab programs are used to iterate the equations in the problem.

Bisection Method and Secant Method require two guess values, which are set to 0 and 2 by the author, and the iteration stop error is set to 0.00001. The Newton-Raphson method requires a guess value, which is set to 2 by the author, and the iterative stop error is also 0.00001.

Each of the following methods will provide a call to the corresponding matlab function statement, the specific function source code in the last part of this topic

### Bisection Method

The following is result:

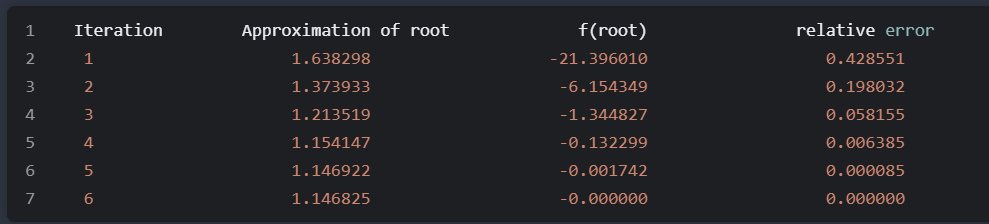


In bisection method, we use Epsilon-a instead of relative error. For more information, you can see the following matlab source code

### Netwon-Raphson Method

Newton(2, 0.00001, @cal,1.1468247)

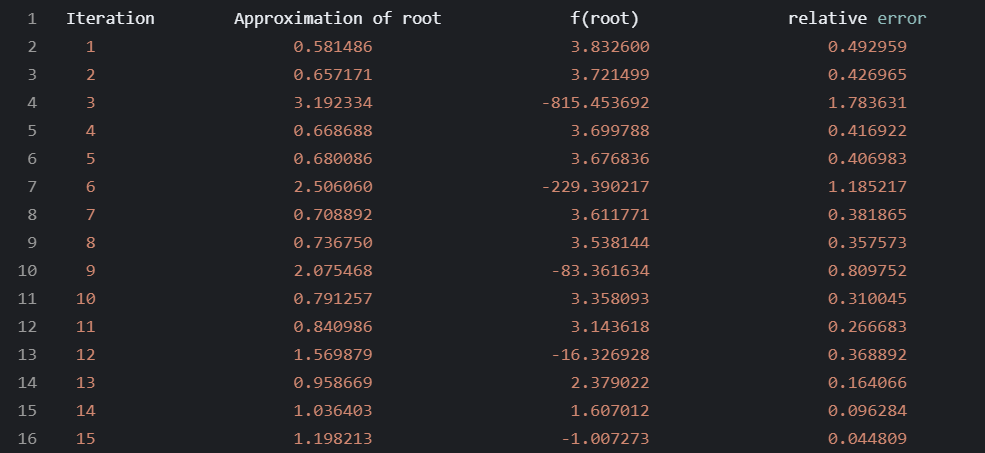
The following is result:

In Netwon method, the writter need true value(1.1468247) to compute relative error(Epsilon-t).

### Secant Method

At the beginning, we gave the initial value of this method to cause the number of iterations of this method to be too many, which is not more than shown here. We reset the initial value to 0.5. The stop error is 0.05. The result is:

secant(0.5, 2, 0.05,@cal, 1.146824)



### Conclusion

According to the above data and the number of iterations, the relative error, the conclusion is drawn by the analysis of the results：

Because the scale of the problem of Secant method is much smaller, there are some errors. The convergence speed of Newton-Raphson is the fastest, the convergence speed of Bisection method is the second, and the convergence speed of Secant method is the slowest. Second, both the Secant and Bisection methods need to give two guesses, but Newton only needs one. So this report chooses the Newton-Raphson method to solve the problem.

### b）

#### Calculations

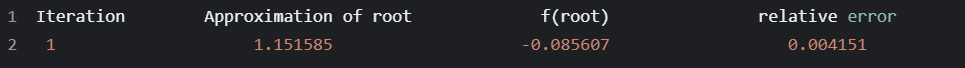
This report will use its own matlab program to calculate the iterative results of the equation, the source code is as follows, will use three functions.

Newton-Raphson



In this program, the correct result with high accuracy will be calculated by using this function in advance, that is, 1.146825. The correct result will be used to calculate the relative error. This program uses Epsilon-t to represent the relative error.

Newton(1.2, 0.05, @cal, 1.146825)



#### Table

|  |  |  |
| --- | --- | --- |
| Iteration | Approximation of root  ( 4 decimal places) | Relative Error  (% with two decimal places) |
| 1 | 1.1516 | 0.42% |

### Other matlab source code and table

In addition, there are matlab programs written in the other two ways. And verify the conditions given by the topic to prove the superiority of Newton's method in dealing with the problem.

#### Bisection Method

%assignment2-1.1 bisection method  
%compute f(x) = 0's each approximation of root, Iteration, errorlative eor  
  
%input: initial guesses: a,b;   
% maximum iteration: max; tolerance:tol;  
% Function handle: fun  
  
  
%output: ap\_root: approximation of root matrix:  
% gap: Interval length:   
% fx: each approximation of root function value: f(ap\_root)  
% count: number of iterations  
% error: relative of error  
  
  
function [ap\_root,gap,fx,count,error]=bisect(a,b,max,tol,fun)  
  
e=tol+1; %set initial errorlative error  
count=0; %set initial number of iterations  
ap\_root=[]; %ap\_root vector store approximation of root  
gap=[]; %gap vector store interval length  
error=[]; %error vector store relative error  
fx=[]; %fx vector store function value  
while(e>tol&&count<max)  
   
 count=count+1;  
 c=(a+b)/2;  
 x=c;  
   
 ap\_root=[ap\_root;x];%Store the iterative value in the ap\_root matrix  
 fc=feval(fun,x); %compute function value of ap\_root  
 fx=[fx;fc]; %store the function value in the fx matrix  
 x=a;  
   
 e=abs(a-(a+b)/2)/abs((a+b)/2); %compute relative error  
   
 %Determine which area the root is in  
 if(fc\*feval(fun,x)<0)  
 b=c;  
 else  
 a=c;  
 end  
   
 dis=abs(a-b); %compute length of interval  
 gap=[gap;dis];  
 error=[error;e];  
end  
  
%display result in matrix  
disp(' Iteration Approximation of root length of interval f(root) relative error ')  
for i=1:count  
 fprintf('%2d %20.6f %25.6f %20.6f %20.6f \n ',i,ap\_root(i),gap(i),fx(i),error(i))  
end

#### Table

|  |  |  |
| --- | --- | --- |
| Iteration | Approximation of Root | Relative Error |
| 1 | 1.2500 | 60.00% |
| 2 | 0.8750 | 42.86% |
| 3 | 1.0625 | 17.65% |
| 4 | 1.1563 | 8.11% |
| 5 | 1.1094 | 0.04% |

#### Secant Method

%1.2.2 Secant method  
  
%Input: x0,x1: Initial guess ;  
% tol: tolerance;  
% fun: function handle;  
% true\_value: true value;  
  
%output: ap\_root: approxieation of root eatrix:  
% fx: each approxieation of root function value: f(ap\_root)  
% count: nueber of iterations  
% error: relative of error  
  
  
function [ap\_root,fx,count,error]=secant(x0,x1,tol,fun,true\_value)  
  
ap\_root=[]; %ap\_root vector store approxieation of root  
fx=[]; %fx vector store function value  
error=[]; %error vector store relative error  
count=0; %set initial nueber of iterations  
x2=x0+1; %set initial x2  
e=1; %set initial error  
  
while(e>tol)  
 count=count+1; %count number of iterations  
 f0=feval(fun,x0); %get the function value  
 f1=feval(fun,x1);   
 x2=(x0\*f1-x1\*f0)/(f1-f0); %iteration each x2 using secant method  
 e=abs((x2-true\_value)/true\_value);%compute relative error  
 x0=x1;  
 x1=x2;  
 f=feval(fun,x1); %get the ap\_root function value  
   
 ap\_root=[ap\_root;x1]; %store ap\_root in vector  
 fx=[fx;f]; %store function value in vector  
 error=[error;e]; %store relative error  
   
end  
  
  
%display result in matrix  
disp(' Iteration Approximation of root f(root) relative error ')  
for i=1:count  
 fprintf('%2d %20.6f %20.6f %20.6f \n ',i,ap\_root(i),fx(i),error(i))  
end

## Q2

### a) Analytically

is a polynomial function whose highest term is 5, so this report only needs to perform symbolic operations on it to get accurate integration results:

Suppose the polynomial function is and its original function is , that is

, ;

Hence,

Here are the results of validation using matlab

f=@(x) (1-x+2\*x.^2-4\*x.^3+2\*x.^5);  
integral(f,-2,4)

The result is exactly the same as that of the matlab function.

### b) Trapezoidal Rule

Ues signle trapezoidal rule approximation

.

Trapezoidal rule estimation:

.

The error of the trapezoidal rule is:

，

Due to, , ,

The error range calculated by the trapezoid rule error formula is:

Because at present, the author has obtained the estimated value of the trapezoidal rule and the real value of the integral of this function, which can be obtained by using relative error's formula.

### c) Trapezoidal Rule(n=2)

For the compound trapezoid rule on interval , two segments mean

Multiple-application of the trapezoidal rule(n=2) estimation:

.

Relative error:

### c) Trapezoidal Rule(n=4)

For the compound trapezoid rule on interval , two segments mean

Multiple-application of the trapezoidal rule(n=4) estimation:

.

Relative error:

### d) Simpson's 3/8 Rule

Hence,

Relative error:

### d) Simpson's 1/3 Rule

Hence,

Relative error:

|  |  |  |
| --- | --- | --- |
| Method | Approximation  ( 2 decimal places) | Relative Error  (% with two decimal places) |
| Analytical | 1152.00 | N/A |
| Single application of the trapezoidal rule | 5400.00 | 368.75% |
| Multiple-application of the trapezoidal rule (n=2) | 2700.00 | 134.38% |
| Multiple-application of the trapezoidal rule (n=4) | 1569.38 | 36.23% |
| Single application of the Simpson’s 1/3 rule | 1800.00 | 56.25% |
| Single application of the Simpson’s 3/8 rule | 1440.00 | 25.00% |

Next, the author will use the matlab program to verify the correctness of the answer.





Therefore, the answers are verified to be correct. All matlab files of this report will be placed in a compressed file