BRAC UNIVERSITY
CSE 330: Numerical Methods (LAB)
LAB 4: MATLAB Function and Newton Raphson Method

What is a MATLAB function?

A MATLAB "function" is a MATLAB program that performs a sequence of operations specified in a text file (called an m-file because it must be saved with a file extension of *.m). A function accepts one or more MATLAB variables as inputs, operates on them in some way, and then returns one or more MATLAB variables as outputs.

How do I create a new MATLAB function?

Since an m-file is nothing more than a text file it can be created using text editor. To open a new m-file: In the MATLAB command window, go to **FILE** on the toolbar, select **NEW**, and then select **M-FILE**. This opens the MATLAB editor/debugger and gives an empty file in which you can create whatever m-file you want.

What do I have to put on the First Line of a MATLAB function?

The 1st line of a function ***must*** contain the "function definition," which has a general structure like this :

function [Out_1,Out_2,…,Out_N] = function_name(In_1,In_2,…,In_M)

where Out_1,Out_2,…,Out_N are the N output variables and In_1,In_2,…,In_M are the M input variables.

CAUTION

1. You must maintain the [] (Third bracket) for output variables and () (First bracket) for input variables.
2. The 1st word in the function definition line **function** must be typed in all lower case. A common mistake is to type **Function**.

If there is only a single output variable use:
function Out_1 = function_name(In_1,In_2,…,In_M) or
function [Out_1] = function_name(In_1,In_2,…,In_M)

If there is no output variable use:
function function_name(In_1,In_2,…,In_M) or
function [ ] =function_name(In_1,In_2,…,In_M)

What do I have to put after the 1st line?

After the first line, you just put a sequence of MATLAB commands – with one command per line – just like you are computing in MATLAB in the command line environment. This

sequence of commands is what makes up your program – you perform computations using the input variables and other variables you create within the function and in doing so, you create the output variables you desire.

When you are writing the lines that make up your function you can use the names of the input variables defined in the first line just like they are previously created variables. So if In_1 is one of the input variables you could then do something like this:
y=In_1^2;
This takes the values in In_1, squares them, and assigns the result to the variable y.

On any line in your function you can assign any result you compute to any one of the output variables specified. For example:
Out_1=cos(y);
will compute the cosine of the values in the variable y and then assigns the result to the variable
Out_1, which will then be output by the function (assuming that Out_1 was specified as an output variable name).

How do I Save a MATLAB function?

Once you have finished writing your function you have to save it as an m-file before you can use it. This is done in the same way you save a file in any other application:

- Go to **FILE**, and **SAVE**.
- Type in the name that you want to use
  - Always use the "function name" as the "file name" i.e. if the name of function is 'motion', the file name must be saved as motion.
  - You don't need to explicitly specify the file type as *.m
- Navigate to the folder where you want to save the function file
  - See below for more details on "Where to Save an M-File?"
- Click on **SAVE.**

Where to Save an M-File?

It doesn't really matter where you store it. BUT when you want to use it, it needs to be somewhere in "MATLAB's path" or should be in MATLAB's present working directory (PWD).

 the path specifies all the folders where MATLAB will look for a function's file when the function is run.

 the PWD specifies a single folder that MATLAB considers its primary folder for storing things – it is generally advisable to specify an appropriate PWD each time you start up MATLAB and specify it to be wherever you have the m-files you are working on.

Click on **FILE**, click on **SET PATH**, click on **BROWSE**, and navigate to the folder you want as PWD and click on it, and then click **OK**.

Example 01:

Say for instance that you want to write a program to compute the average (mean) of a vector x. The program should take as input the vector x and return the average of the vector.

Steps:

1. You need to create a new file, called "average.m". Open the text editor by going to the File pull-down menu; choose New, then M-file. Type the following in the empty file

```
function y=average(x)
L=length(x);
sum=0;
for i=1:L
  sum=sum+x(i);
end
y=sum/L;
```

Remarks:
y — is the output of the function "average"
x — is the input array to the function "average"

Attention:
The variables within a function are said to be **local** and are erased after the function is executed. In contrast, the variables in a script retain their existence after the script is executed.

2. From the Editor pull-down menu, go to File | Save, and enter: 'average' for the filename.
`average` — is the name of the function. It's best if it has the same name as the filename.

3. Go to the Command window to execute the program by typing:

```
>> inp=1:100;
>> out=average(inp)
out =
  50.5000
```

Remarks:
inp ------- is any given input to the function "average" i.e. value of the inp will be assigned to x. So in the function x=1:100.
out ------- is the output value which will be equal to y of the function "average".

LAB TASK 01:

Find the sum of a geometric series:

- Write a function to compute the sum of a geometric series $1+r+r^2+r^3+.....+r^n$ for a given r & n.
- The input of the function must be r & n and the output must be the sum of the series.

HOME TASK 01:

Convert temperature:

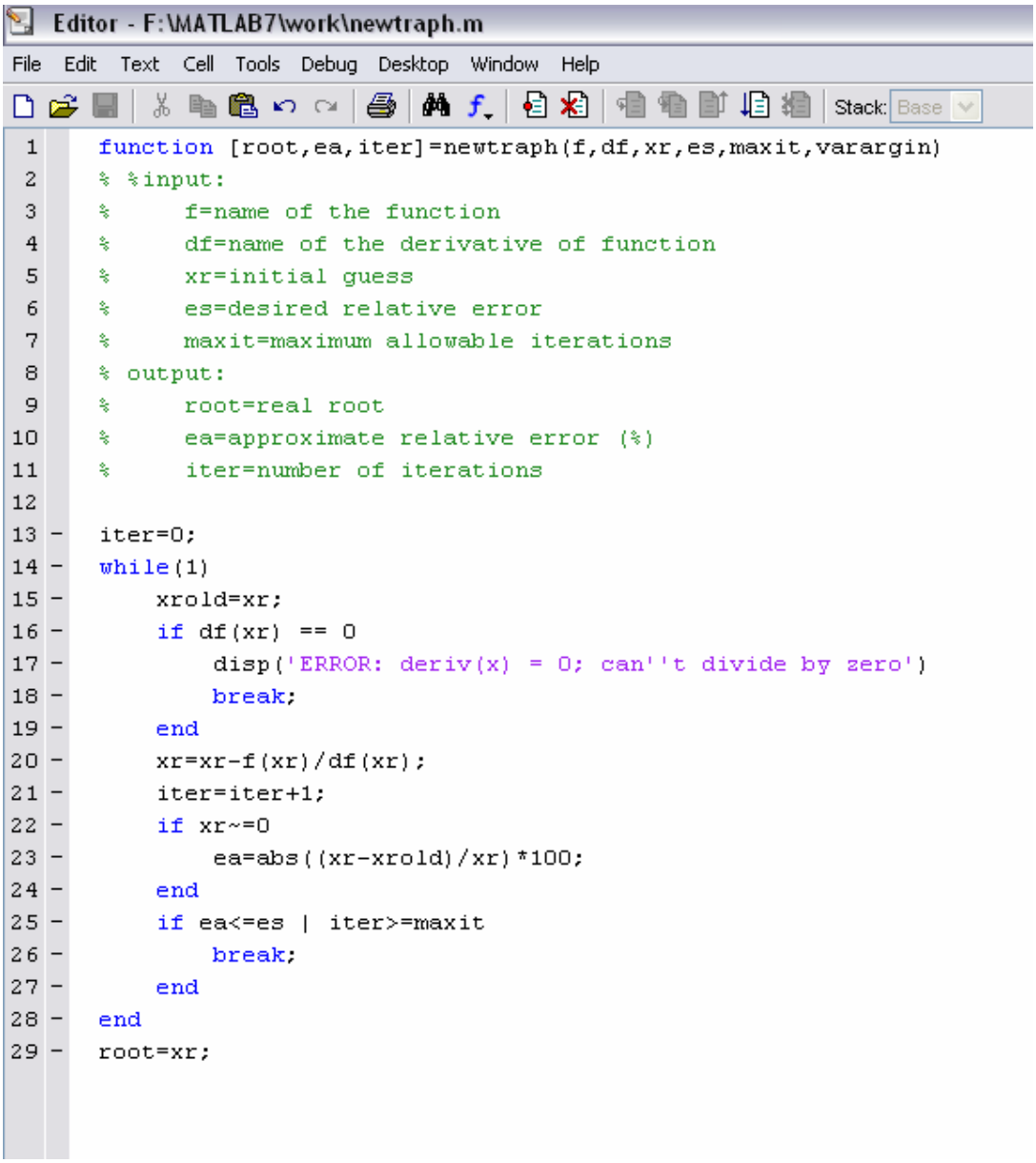
- Write a function that outputs a table showing Celsius temperature and their corresponding Fahrenheit temperature.
- The input of the function should be two variables ti & tf, specifying the lower & the upper range of the table in Celsius.
- The output should be a two column matrix; the 1st column showing the temperature in Celsius from ti to tf in the increments of $1^{\circ}C$ and the 2nd column showing and the corresponding temperatures in Fahrenheit.

Example 02:

Perhaps the most widely used of all root-locating formulas is the Newton-Raphson method. Now we would like to write a function named 'newtraph' which uses Newton-Raphson method to find the root of a function.

The Newton-Raphson method can be derived on the basis of geometrical interpretation. The Newton-Raphson formula is defined by

$$x_{i+1} = x_i - f(x_i) / f'(x_{i+1})$$



After the **M**-file is entered and saved, it can be invoked to solve for root. For example, for the simple function $x^2 - 9$, the root can be determined as in

```
>> [root,error,iter]=newtraph ( @ (x) x^2–9, @ (x) 2*x, 5, .0001, 50)

root =

     3
error =

     4.6566e-008
iter =

     5
```

HOME TASK 02:

Bisection method:

The Bisection method is another widely used root determining method.

- Write a function named 'bisect' which uses Bisection method to find the root of a function.
- The input should be the function whose root will be determined, two variables xi & xf, specifying the lower & the upper guesses, desired relative error and maximum allowable iteration number.
- The output will consist of three variables; real root, approximate relative error and number of iteration.
- The 1st line of the function can be formed this way

```
function [root, ea, iter]= bisect (func, xi, xf, es, maxit)
```