MAE3456 – MEC3456 LAB 01

Due: 11:00PM (Sharp), Wednesday 25th March 2020 (Mid of Week 3)

This lab should be completed INDIVIDUALLY. Plagiarism will result in a mark of zero. Plagiarism includes letting others copy your work and using code you did not write yourself without citing the source. Collaborating with others to discuss algorithms and details of MATLAB syntax and structures is acceptable (indeed encouraged), however you *MUST* write your own MATLAB code. All assignments will be checked using plagiarism-detecting software and similarities in submitted code will result in a human making a decision on whether the similarity constitutes plagiarism.

INSTRUCTIONS

Download **`template.zip`** from Moodle and update the M-Files named **`Lab01_Q1a.m`**, **`Lab01_Q1b.m`**, etc… with your Lab code. DO NOT rename the M-Files in the template and ONLY modify **`run_all.m (except to add in running scripts that answer any bonus questions I might ask)`**. Once you have coded, check your solutions to the questions by running **`run_all.m`** and ensuring all questions are answered as required.

SUBMITTING YOUR ASSIGNMENT

Submit your assignment online using Moodle. You must include the following attachments:

A **ZIP** file (**NOT .rar or any other format**) named in the following way:

Surname_StudentID_Lab_01.zip (e.g. Rudman_23456789_Lab_01.zip)

The zip file should contain the following:

a. All MATLAB m-files for lab tasks: **run_all.m**, **LabNN.m, `Q1a.m`**, **`Q1b.m`**, etc…

b. Any additional MATLAB function files required by your code

c. All data files needed to run the code, **including any input data provided to you**

d. Any hand calculations or written responses asked for - scanned in as a ***SINGLE*** PDF file

YOUR ZIP FILE WILL BE DOWNLOADED FROM MOODLE AND ONLY THOSE FILES INCLUDED IN YOUR SUBMISSION WILL BE MARKED

We will extract (unzip) your ZIP file and mark the lab based on the output of **run_all.m** and any hand calculations or written responses. It is your responsibility to ensure that everything needed to run your solution is included in your ZIP file.

RECOMMENDATION: After uploading to Moodle, download your Lab to a NEW folder, extract it and ensure it runs as expected. ***CHECK YOUR PDF IS INCLUDED***. You can upload your submission as many times as you like ***BEFORE*** the submission deadline. After this time, it is not possible to resubmit your lab without incurring a late penalty.

MARKING SCHEME

This lab is marked out of 20 and contributes 4% toward your total unit mark for the semester. Code will be graded using the following criteria:

1) `run_all.m` produces results **automatically** (no additional user interaction needed except where asked explicitly
2) Your code produces correct results (printed values, plots, etc…) and is well written.
3) Programming style, efficiency of algorithm and quality of output (figures, tables, written text ...) will be assessed in this lab and marks will be lost for poor style.

ASSIGNMENT HELP

1) You can ask questions in the Discussion Forum on Moodle
2) Hints and additional instructions are provided as comments in the assignment template M-Files
3) Hints may also be provided during lectures
4) The questions have been split into sub-questions. It is important to understand how each sub-question contributes to the whole, but each sub-question is effectively a stand-alone task that does part of the problem. Each can be tackled individually.
5) I recommend you break down each sub-question into smaller parts too, and figure out what needs to be done step-by-step. Then you can begin to put things together again to complete the whole.
6) To make it clear what must be provided as part of the solution, I have used bold italics and a statement that (usually) starts with a verb (e.g. ***Write a function ..., Print the value...***, etc.)

QUESTION 1 [6 MARKS TOTAL]

Q1a

In Lecture 3 we saw how to find the points at which two functions ($f(x,y) = 0$ and $g(x,y) = 0$) intersected.

Repeat this analysis to describe the system you would have to iteratively solve to find the points at which 3 functions of (x,y,z) intersect, i.e. the points at which $f(x,y,z) = 0$, $g(x,y,z) = 0$ and $h(x,y,z) = 0$ simultaneously.

(**HINT:** Using up to the first order terms in a multidimensional Taylor series, and assuming you have a guess (x_i,y_i,z_i) for an intersection point, derive the matrix equation that needs to be solved for the increments $(\delta x, \delta y, \delta z)$ that will give the new estimate $(x_{i+1}, y_{i+1}, z_{i+1}) = (x_i + dx, y_i + dy, z_i + dz)$.)

Q1b

Explicitly write out the matrix system for the increments $(\delta x, \delta y, \delta z)$ that would allow you to find the points at which the following functions intersect:

- $x = y^2+z^2$,
- $y = x^2+z^2$ and
- $z = x^2+y^2-0$.

You do NOT have to invert the matrix, just calculate its components.

QUESTION 2 [8 MARKS TOTAL]

Q2a

Modify the script template **Lab_01_Q2.m** to set up the problem specified in Q1b above.

Write a MATLAB function **NR_3D.m** that implements Multi-dimensional Newton-Raphson root finding. Call it **NR_3D.m**. It should have a function header that looks something like

```
function [root,iter]= NR_3D(r0,f,g,h, [derivative functions], tol)
```

where: the return value is a 3-vector (root) is the intersection point found by the NR_3D and iter is the number of iterations the function took to find the root. The input parameters are

- the initial guess (a 3-vector stored in r0)
- The 3 functions ($f(x,y,z)$, $g(x,y,z)$ and $h(x,y,z)$).
- The derivative functions needed inside NR_3D can be passed in as 9 separate functions OR can be put into a single array-valued function handle.
- The root-finding tolerance is given by tol.

It is possible that a root does not exist, so ***implement*** some kind of error checking inside this function that prints a warning message if the function does not appear to be converging.

Q2b

i) Solve the problem specified in Q1b using an initial guess for the root n of (0.3,0.3,0.3) and a tolerance of 10^{-6}. ***Write*** the intersection value to the screen with 6 decimal points and on a separate line write a statement on how many iterations the function took to reach the tolerance.

ii) How can you tell if the root you have found is correct? ***Print*** a few sentences to the command window describing how you can do this and demonstrate that you have found the root to the accuracy defined by tol.

iii) Repeat the root finding problem (part ***i***)with an initial guess of (0.1,0.1,0.1). Do you get the same answer? ***Print*** a short message to the screen that suggest why or why not

iv) Print a short message to the screen that describes the convergence error checking you implemented when writing the function.

QUESTION 3 [6 MARKS TOTAL]

Background

The Thomas tri-diagonal algorithm uses a direct solve to find solutions of the linear system **Ax=b**, where the matrix A has tridiagonal form (Workshop 2). The method is essentially the same as Gaussian elimination, but because most of the matrix entries are zero, the method is far more efficient.

i) Write a MATLAB function (TriDiag) that implements the Thomas tri-diagonal algorithm. The function header MUST be

```
function [x] = TriDiag(a,b,c,r)
```

where **x** is the solution vector, a,b,c are the coefficients of the tridiagonal matrix A (a stores the lower diagonal elements, b the central diagonal and c the upper diagonal) and **r** is the right hand side vector.

(**NOTE:** The equation for variable x_i is $a_i*x_{i-1} + b_i*x_i + c_i*x_{i+1} = r_i$, thus a(1) = 0 and c(n) = 0);

ii) You are provided with a MATLAB function `set_coeff(N)` that sets the coefficients **a,b,c** and the rhs vector **r** you should use when running your code. You set the vectors a,b,c,r for an NxN system by choosing an appropriate N and using `[a,b,c,r]=set_coeff(N)`.

Modify the m-file **Lab_01_Q3.m** and ***calculate*** the solution **x** for tridiagonal systems defined by set_coeff(N) with N=1000.

iii) How can you determine if you have found the correct answer for ***x***? ***Write*** a short statement to the command window stating how and then ***implement*** this in the code in **Lab_01_Q3.m** and demonstrate the answer you find is correct.

Poor Programming Practices **[-5 Marks]**

(Includes, but is not limited to, poor coding style or insufficient comments or unlabeled figures, etc.)

(END OF LAB)