MAE3456 – MEC3456 LAB 02

Due: 11:00PM (Sharp), Wednesday 8 April 2020 (Mid of week 4)

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. 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 **Lab02_**Q1a.m**,** **Lab02_**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 35 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 – NOTE, I have included pause commands in run_all so that intermediate answers can easily be viewed by the demonstrators – please don't remove them)
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 ...)

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 **[12 MARKS TOTAL]**

Background

In Week 2, you have learned about different methods that can be used to carry out interpolations, such as Newton's divided difference method and the Lagrange polynomial. You have also learned that the selection of data points to fit your polynomial and subsequently your interpolation, can affect your numerical prediction. Consider the following set of data points:

x	1.6	2.0	2.5	3.2	4.0	4.5	5.1	6.0
y	2	8	14	15	8	2	0.5	1.5

Q1a

Using Newton's divided difference method, by hand, construct a third order polynomial by hand using the first four data points from the table. Use the polynomial to estimate the value of $y(x = 3.3)$.

Q1b

Repeat Q1a using the set of data points that you think will yield the best accuracy for your interpolation.

Q1c

Are there any differences in the values of $y(3.3)$ obtained from Q1a and Q1b? Explain why.

Q1d

You have learned in Week 3 that Newton's divided difference method and Lagrange polynomial yield the same polynomials and the same interpolation estimate. Repeat Q1b using a third order Lagrange polynomial. Compare and comment on the value of $y(3.3)$ obtained using Newton's divided difference method and the Lagrange polynomial.

Q1e

Comment on this statement: 'More points means higher accuracy during interpolation'.

QUESTION 2 [15 MARKS TOTAL]

In this question, you will be writing Matlab functions that can perform Lagrange interpolation for arbitrary **N** points using a series of data points as input.

Q2a

Start by ***writing a MATLAB function*** that calculates the i^th Lagrange polynomial basis function,

$$L_i(x) = \prod_{(j=1,\, j\neq i)}^{n} \frac{x - x_j}{x_i - x_j}$$

The function header should be

```
function [lval] = Li(i,xp,x)
```

where the input parameters to the function are the set of x-values (xp) and the (single) value at which the function is required (x). The output is the value of $L_i(x)$.

Q2b

Using the Li function you wrote above, ***write a MATLAB function*** that calculates the Lagrange polynomial

$$y(x) = \sum_{i=1}^{n} L_i(x) y_i$$

The function header should be

```
function [yval] = LagrangeP(xi,yi,x)
```

where the input parameters to the function are the set of x-values (xi), y-values (yi) and the value (x) at which the interpolated value y(x). The function output is the value of y(x).

Q2c

Consider the following function:

$$f(x) = \frac{1}{1 + 20x^2}$$

In file Lab_02_Q2c.m, ***write*** a MATLAB code that generates 20 equidistant points from -1 to 1. For each of the 20 points, calculate the value of f(x) based on the function above.

Subsequently, use your `LagrangeP` function from Q2b to fit the Lagrange polynomial to 100 equispaced data points over [-1,1] (i.e. you are interpolating f(x) ***FROM*** 20 points ***TO*** 100 points).

Plot the original 20 data points as red dots and the interpolated function for 100 data points over the domain [-1 1] of the data as a blue line.

Print a short statement to the command window about what you observe in the figure and comment on the accuracy of the interpolation.

Question 3 [8 Marks total]

In this question you will use the 1D Lagrange polynomial interpolation function you wrote in Q2b to interpolate two-dimensional regularly gridded data.

Q3a

Write a MATLAB function that performs 2D Lagrange polynomial interpolation. The function header should be

```
function [fval] = LagrangeP_2D(m,n,xp,yp,fp,x,y)
```

where the input parameters to the function are

- m,n - the number of points in the x and y directions (i.e. the function values are an mxn matrix of values)
- xp, yp – the x and y-values where the data is defined (xp is a vector with m-elements, yp is a vector with n elements)
- fp– an mxn matrix of function values defined at the grid points (i.e. fp(i,j) = f(xp(i),yp(j))
- x,y – the point at which you want to interpolate.

and the interpolated value of the function f(x,y) is returned in fval.

`LagrangeP_2D` should use `LagrangeP` as discussed in the lecture.

Q3b

The file 'temperature.dat' contains the values of temperature (T) of a square heated plate measuring [0, 8] x [0, 8] cm^2 recorded at 81 equally-distributed points with xp = 0:1:8 and yp = 0:1:8. The temperature data is presented as a matrix. Values in each column represent the data at xp = 0, 1, 2, …, 8; while values in each row represent the data at yp = 0, 1, 2, …, 8.

In the file Lab_02_Q3b.m, the Matlab code that opens 'temperature.dat', reads in the data has been provided for you.

You must add to this file and ***write a code*** that interpolates T ***from*** the given data points(xp,yp) ***to*** a set of equally spaced points (xi, yi) defined by

- xi is 30 equally spaced points in x$\in$[0, 8]
- yi is 30 equally spaced point in y$\in$[0, 8].

Plot the temperature of the given data (i.e. the 81 points (xp,yp)) in a surface plot using the Matlab function `surf(xp,yp,Tp)`.

In a separate figure, ***plot*** the interpolated values of temperature as a surface plot using the Matlab function `surf(xi,yi,Ti)`.

HINT: you may also need to use the MATLAB function `meshgrid` to generate 2D arrays of data points (xg, yg) from the 1D vectors (xi,yi) and (xp,yp), i.e.

```
[xg yg] = meshgrid(xi,yi)
```

Print a comment to the command window about what you observe when comparing the two plots.

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

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

(END OF LAB)