Mech 221: Computer Pre-Lab 1

Hand in the solutions to the two questions in the pre-lab at the *beginning* of the lab.

To help you succeed in this course, there will be be pre-lab assignments for every computer lab. The goals of the pre-lab are to reinforce concepts taught in class that will be used in the labs and to introduce any new MATLAB commands that will be used. By doing the pre-labs, you will make yourself ready to do the labs in the allotted time. Four out of ten of the marks for a lab are based on the pre-lab questions.

Success in some kinds of engineering requires skill at numerical approximation. The computer labs this term build this skill. This lab in particular will build your understanding of and experience with numerical integration techniques. Some MATLAB commands you already know will be reinforced and some new commands introduced. Specifically this pre-lab will cover:

- Introduction to the new MATLAB commands: `linspace`, `trapz`, and `sum`. In addition, you will learn how to access parts of a vector.
- How to determine an adequate number of points to get sufficient accuracy from a numerical integration method.
- Algorithms for numerical integration.

Completing the pre-lab will prepare you for the lab, in which the following will be covered:

- Using basic commands and learning some syntax of MATLAB
- Evaluating some basic integrals using methods taught in class, and reaching a certain required level of accuracy.
- Writing numerical integration code in the form of an .m file.

Point of Clarification: As discussed in the lecture notes, when a problem needs to be solved using N intervals, this is done with $N+1$ function values at evenly spaced points at the ends of these intervals.

MATLAB commands to know for the lab

`linspace(a,b,N)` : This command will generate a vector of `N` evenly spaced points between the values `a` and `b`, inclusive. For example, `>> x = linspace(0,2,5)` will create the vector:

```
x =     0    0.5000    1.0000    1.5000    2.0000
```

As mentioned in the point above, these 5 points correspond to 4 subintervals of equal length.

`plot(x,y)` : This command will plot corresponding values from vectors `x` and `y` in a figure window. `x`, `y` must be the same length or an error will occur.

`trapz(x,y)` : Given two vectors `x` and `y`, this command will estimate the integral of `y` with respect to `x`, using the Trapezoidal Method. Once again, `x` and `y` must be the same length. For this command the points in `x` do not need to be equally spaced.

`sum(x)` : The command adds up all the entries in the vector `x`.

`x(1:4)` : Makes a vector with the first four entries in the vector `x`. This gives an error message if `x` has less than four entries.

Question 1: Grid Refinement

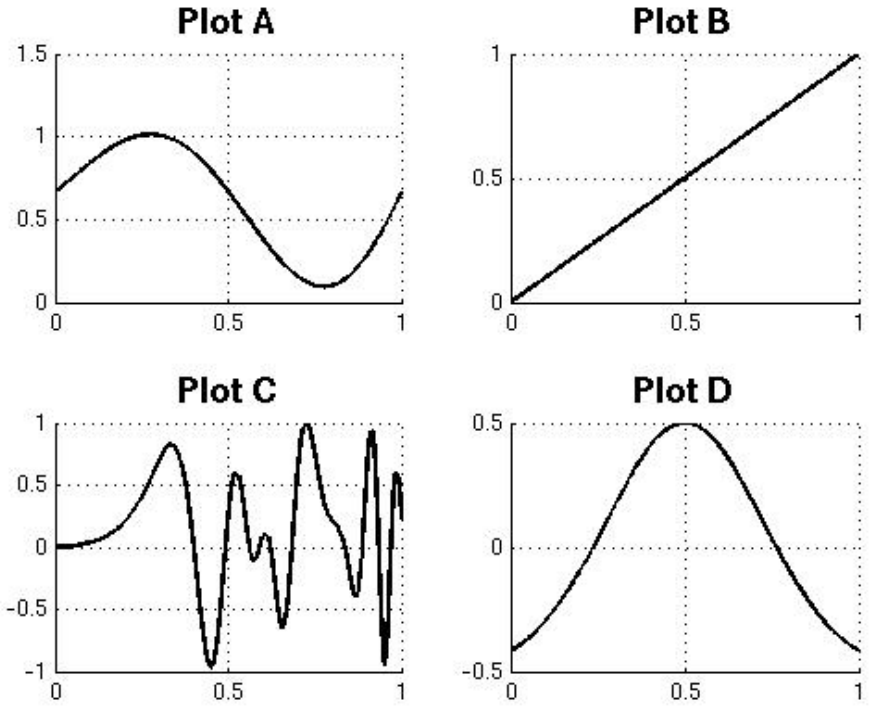
In estimating an integral using numerical methods, it is important to recognize when you are getting accurate results. One way to check is to notice when the digits to a certain decimal place cease to change as the number of intervals N is doubled. When this happens with a method that is second order or higher (Trapezoidal or Simpsons Rule) you can be fairly certain that you have an accurate answer.

When a given function has large derivatives over an interval, it becomes more difficult to estimate the integral using numerical methods. (Difficult meaning that the number of points required to evaluate the integral to a desired accuracy can be large).

Data sets a), b), c), and d) below each contain estimates of an integral for the number of intervals `N`. Each estimate was calculated using the Trapezoidal Rule. Plots A,B,C, and D each plot one of the integrands. By looking at how these estimates behave as `N` increases, match each set of numerical estimates a), b), c) and d) with Plot A,B,C or D given below.

The match of a-d with A-D is all that needs to be handed in for this question. However, a question like this could be asked on a test or exam, so make sure you understand why the numerical approximations of these integrals behave the way they do as `N` is increased.

N	a)	b)	c)	d)
2	0.6667	0.5000	0.1735	0.0410
4	0.6111	0.5000	0.3729	0.0382
8	0.5993	0.5000	0.2052	0.0442
16	0.5965	0.5000	0.1025	0.0458
32	0.5958	0.5000	0.1345	0.0462
64	0.5956	0.5000	0.1377	0.0463
128	0.5956	0.5000	0.1383	0.0463
256	0.5955	0.5000	0.1385	0.0463
512	0.5955	0.5000	0.1385	0.0463
1024	0.5955	0.5000	0.1385	0.0463



Question 2: Algorithms

A basic algorithm for the evaluation of a Left Riemann Sum, I, for a function $f(x)$ from $x =$ `a` to $x =$ `b` with `N` intervals of equal length is given below. In the sample code $f(x)$ is taken to be $\sin x$ and the integral $\int_0^1 \sin x dx$ is approximated with 4 subintervals.

```
a = 0;
b = 1;
N = 4;
x = linspace(a,b,N+1);
h = (b-a)/N;
y = sin(x);
I = h*sum(y(1:N))
```

Remember that if you leave off the `;` at the end of a MATLAB command, the result is displayed. These lines of code could be put into an .m file and used again for other integral approximations. To make this a Right Riemann Sum approximation replace the last line by

```
I = h*sum(y(2:(N+1)))
```

For a Trapezoidal Rule approximation use the MATLAB command `trapz` introduced above

```
I = trapz(x,y)
```

For this question, write out by hand the full code for evaluating the integral I of a function f over $[a, b]$, with N intervals using *Simpsons Rule*. Your Mathematics lecture notes contain details on Simpsons Rule and how to implement it in MATLAB. In your lab, you will be required to code this method in an .m file. Do this question carefully so that you can complete your lab within the allotted time.