Mech 221: Computer Pre-Lab 2

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

The upcoming lab concerns the numerical approximation of differential equations using the Forward Euler method. This pre-lab will cover:

- Introduction to the new MATLAB commands: `hold`, and `plot` options. These commands will allow you to put several plots on the same graph and give you ways to have them displayed differently.
- Further MATLAB commands to generate vectors: `zeros` and the colon command.
- MATLAB commands that act on vectors: `length`, `max` and `abs`.
- You will be introduced to the MATLAB `for` loop and see some examples. This is the main new computational element needed in the code to approximate differential equations.
- A description of the Forward Euler time-stepping method you will implement.

MATLAB commands to know for the lab

Plotting commands:

`hold on` : This command will allow to you to plot multiple sets of data within the same figure, rather than plotting only the last data-set requested. The command `hold off` will turn this feature off.

`plot(x,y,options)` : To improve the appearance of plots, there are several options that MATLAB has available. You can change colour, data point markers, line style, etc. The basic options can be implemented as follows:

```
>> plot(x,y,'[colour][linestyle][marker]','linewidth',[n]
```

colour : Specifies the colour of the line. Some options are `b,r,k,` or `g`, corresponding to blue, red, black or green respectively.

linestyle : Specifies the style of the line you wish to plot. `-,--` `or (blank space)` are common examples corresponding to solid, dashed, or no line respectively. (Note: the no line option will only work if you specify a marker)

marker : Specifies the data marker at each point in your figure. (blank space) ,* or o are some examples that correspond to no markers, asterisks, and circles.

n : Specifies the thickness of the line being plotted (1 is the default).

– Note that you can choose to specify only some of these values. If unspecified, they turn to default values.

examples : `>> plot(x,y,'ro');` will plot `x` and `y` as a series of red circles, unconnected by a line.
`>> plot(x,y,'b--','linewidth',3);` will plot `x` and `y` as a thick, blue, dashed line.

Commands to generate row vectors:

`zeros(1,12)`: Creates a row vector with 12 entries, all with value zero.

`1:4` Creates a row vector with entries `1 2 3 4`. The first entry is the start value, the last the end value of the list.

`2:2:8` The middle argument of this command is an increment. This command generates the row vector with entries `2 4 6 8`.

Commands that operate on vectors:

`length(x)` : Will return the number of elements in the vector `x`.

`max(x)`: Will return the maximum value in the vector `x`.

`abs(x)`: Will return a vector with the absolute value of every entry in `x`.

If the vector `y` has the exact values of a function and `yapprox` has values computed with a numerical method, then `max(abs(y-yapprox))` will give the maximum error of the approximation.

For...end loops

A loop (a repeated list of commands) can be made with the `for` and `end` commands. The structure is as follows:

```
for i = x
```

Other commands here, possibly many lines

```
end
```

where `x` is a row vector (often of integers but this is not necessary). MATLAB will execute the commands between the `for` and `end` commands with `i` having the value of the first entry in `x`. Then, the commands will be executed again with `i` having the value of the second entry in `x` and so on. In total, the lines of code between the `for` and the `end` will be executed a number of times equal to the number of entries of `x`.

The code

```
I = 0;
for i = x
   I = I + i;
end
```

gives the same result as `sum(x)`. The code

```
N=100;
x = zeros(1,N);
h = 1/(N-1);
for i = 1:N
   x(i) = (i-1)*h;
end
```

gives the same result as `x = linspace(0,1,100)`. Note that when generating a vector using a for loop, it is a good idea to set the size of the vector before the loop starts using a `zeros` command as done above.

Question 1: A Differential Equation

Consider the following differential equation problem for $y(t)$:

$$\frac{dy}{dt} = -y + 1$$

with initial condition $y(0) = 0$. This could model a series RL circuit, with y being the current in the circuit. These models will be developed in your electrical engineering lectures.

Using techniques that you will learn in the mathematics lectures, you will know to look for a solution to the problem above in the form

$$y(t) = Ce^{-t} + a$$

where C and a are constants to be determined. Put this form into the differential equation and initial condition to determine a and C.

Hand in your analytic solution to the problem above with your working

Question 2: Programming the Forward Euler Method

The Forward Euler method is a simple numerical method used for approximating differential equations. The Forward Euler method (or simply Euler's Method) is also discussed in your text in section 2.10.

Given a differential equation, initial condition and domain of interest:

$$\frac{dy(t)}{dt} = f(y(t), t), \quad y(a) = y_0, \quad a \leq t \leq b$$

The Forward Euler method generates approximate values of y at a discrete set of t points on a grid: $a = t_0 \leq t_1 \leq .... \leq t_N = b$. As with numerical integration, the spacing between these $N + 1$ time points is $h = (b - a)/N$. Computed values y_i will approximate the exact solution $y(t_i)$. The initial condition gives y_0 exactly.

The idea of the Forward Euler method is to use known values of y and t and step forward in time, allowing us to approximate what the value of y is at time $t+h$. In other words, we use y_{i-1}, t_{i-1} to solve for y_i. We then repeat this process until we reach the end of our interval, $t_N = b$. Specifically, the scheme is

$$y_i = y_{i-1} + h \cdot f(y_{i-1}, t_{i-1})$$

with the first value y_0 known from the initial condition. Using a `for` loop, this "forward-stepping" procedure can be continued until you have reached the final time $t_N = b$.

- Consider the algorithm above with $a = 0$, $b = 1$, $N = 4$, $f(y(t), t) = -y + 1$ and $y(0) = 1$. The differential equation problem is the same as in Question #1 above.
- hand-write the MATLAB code required to use the Forward Euler method to approximate the solution to that problem on the 4 sub-intervals.

- *Hand in that hand-written code.*

Coming up in Lab #2

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

- Writing the Forward Euler method in the form of a .m file. It is important to do a careful job on pre-lab Question #2 so you are ready to write your code.
- Using this .m file to approximate solutions to the differential equation problem in the pre-lab.
- Comparing the numerical approximations to the exact solution on plots and by calculating errors.
- Modifying your code to approximate a different differential equation.