Differential Equations, Spring 2017
In-Class Matlab, Friday, January 27

This tutorial will walk you through some basic Matlab commands. More in-depth tutorials can be found on the Matlab website: `https://www.mathworks.com/support/learn-with-matlab-tutorials.html`.

1. Open Matlab. Entering commands into the command window will tell Matlab to do things immediately. Using a semicolon will suppress the output. Enter the following commands:

   ```
   > 4+4
   > 4+4;
   > x=5
   > y=6;
   > x+y+x*y-3*x^2
   > y=y+7
   ```

2. The basic data type in Matlab is a matrix. A matrix is a 2d array of numbers. Individual entries can be accessed by their coordinates. The first coordinate is the vertical position (row number) and the second coordinate is the horizontal position (column number).

   ```
   > A=[1 2;3 4]
   > A(1,1)
   > A(1,2)
   > A(2,1)
   > A(2,2)
   > A(1,1)=-10
   > B=[1 2 3 4]
   > C=[1; 2 ;3; 4]
   ```

   The transpose command ' turns rows into columns and vice versa. Input the following commands to see how this works.

   ```
   > A'
   > B'
   > C'
   ```

   A colon can be used to access a range of entries. The following commands give some examples.

   ```
   > A(:,1)
   ```

```
> A(:,2)
> A(1,:)
> A(2,:)
> B(1,2:4)
> C(3:4,1)
> 1:10
> 2:5
```

Matrices can easily be appended together to make bigger matrices.

```
> X=[A A]
> Y=[A;A]
> Z=[X;B]
> W=[Y C]
```

Later on we will be working with matrices consisting of a single row. These can just be thought of as a list of numbers. Entries can be accessed by a single entry number.

```
> L=[1 -1 2 -2 3 -3]
> L(1)
> L(4)
> L(3:6)
```

A very useful command is `linspace(a,b,n)`. It will create a list of numbers of length n starting at a and ending at b. Here are some examples to try:

```
> linspace(1,10,10)
> linspace(0,1,101)
```

3. Each user created function in Matlab resides in its own file of the same name (with a .m extension). We will walk through creating a new function in this problem.

   First, some basic file commands. The command pwd will print working directory, the command cd will change directory, the command ls will list files, and the command mkdir will make a directory.

   Create a directory for yourself. In this directory, create a file called "test.m" (by selecting New $\rightarrow$ Function from the menu at the top). (Alternatively, open up any text editor and create a new file called test.m in your directory. Make sure the text editor does not put a .txt extension onto the file.)

   Enter the following text into the file test.m:

```
function y=test(x)
    y=sin(2*x)+2*cos(3*x);
end
```

The first line means that a function called test is being defined. The function takes in one input (`x`), and returns one outpute (`y`). The specific variable names `x`, `y` are not important, we could have called them anything else instead, for example `xxxxx` and `y1y2blah`. Run the function using the following commands from the command window:

> `test(10)`

> `test(20)`

4. Now we are going to create a script that will contain some commands for graphing the function `test`. A script is just a text file (with a `.m` extension) that contains some Matlab commands. It can be executed by typing in the name of the file (without the extension) from the command line. To create it, select New → Script from the menu and create a file called `graphtest.m`. The basic command to plot is `plot(x,y)`, where `x` and `y` are lists of numbers, thought of as the x and y coordinates of a list of points. The `plot` command will plot these points and connect them with line segments. So we can plot a function by first generating a list of x values, and then plug these values into the function to get the corresponding y values, and then call the `plot` command. Here is some code to do that, enter it into your `graphtest.m` script. (The `%` sign indicates a comment; it is not executed by Matlab and is intended to make the code easier to read.)

```
xvals=linspace(-10,10,100); % list of x values
yvals=[]; % initialize y values to be empty
% now make a for loop to build the y values
for x=xvals
    yvals=[yvals test(x)];
end
plot(xvals,yvals);
```

A for loop is a way to execute a block of code several times (the code between `for x=xvals` and `end` in the example above), and each time the code is executed the indexing variable has a different value. In the example above, `x` is the indexing variable, and the first time the code is executed it is set to be the first entry in `xvals`, the second time it is the second entry in `xvals`, and so on. In general, a loop of the form

```
for i=data
   ...
end
```

will tell Matlab to run the code ... in the interior of the loop several times, `i` is the indexing variable. The first time the code ... is run, `i` will be set to the first entry in `data`, the second time it runs `i` will be set to the second entry in `data`, and so on.

Now run your program by calling it from the command prompt:

```
> graphtest
```

This should cause a graph to pop up (it may take a few seconds). Another way to populate the list `yvals` would be to replace the for loop in the above code with the single command `arrayfun(@test,xvals)`. You can search the Matlab documentation for complete details about how the `arrayfun` command works. Basically what it does is call the function `test` on each value of the list `xvals`. The `@` symbol in the first input tells Matlab that the name of a function is being passed in.