ME 350 Lab Exercise 1 — Introduction to MATLAB
Winter 2017

These exercises are meant to give you practice with MATLAB in the computer lab while the instructor is available to help. It is quite likely that you will not be able to finish these exercises. The goal is to learn, and not to rush through the examples. Please complete these exercises in your private study time if you don't finish them before the end of class time. Also, please finish these exercises *before* you start the homework.

For the plotting examples, practice copying one or more plots into a MS Word document. Upload the document, along with a sample of the m-files you created to the D2L dropbox for Lab 1.

Interactive MATLAB Practice

Use MATLAB to perform the following calculations.

1. Define `r`, `s`, `t`

   ```
   r = 30
   s = 50
   t = r/s
   ```

   Now, set `s = 30`. Before checking with MATLAB, decide for yourself, what is the value of `t` *after* the value of `s` has been changed?

2. Enter the following expressions in the command window.

   ```
   x = [1 5 10 15]
   y = x/5
   z = exp(-y)
   ```

   Note that MATLAB will echo the results if you do not end each line with a semicolon.

3. Given `x`, `y` and `z` from the previous exercise, enter the following commands

   ```
   r = x/y
   s = x./y
   t = x*y
   u = x.*y
   v = exp(-0.1*x)*sin(x)
   w = exp(-0.1*x).*sin(x)    %  Notice the "."
   ```

   Discuss the results of executing these statements with a neighbor in the lab. Enter "`help ./`" and "`help .*`" in the command window. Alternatively you can enter "`doc ./` and "`doc .*`".

4. Given `x`, `y` and `z` from the previous exercises, enter the following commands. Discuss the results with a neighbor in the lab.

   ```
   p = x*y'
   q = x(:)
   r = q'*q
   disp(x)
   disp(x')
   disp(q)
   ```

5. Given the `x` and `y` values from the previous exercises, enter the following commands. Note that the syntax of the `fprintf` command is very particular about the use of `'` and `,` as well as other characters inside the first argument.

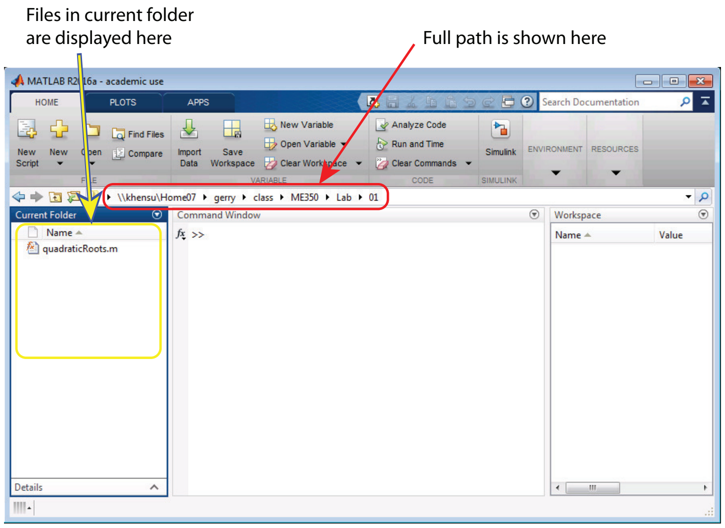
```
disp(x)
disp(x(3))
fprintf('%10.7f\n',x)
fprintf('\n\t w(4) = %12.7f\n',y(4))
fprintf('\n\t w(1) = %12.7e\n',y(1))
```

6. Use the built-in `sin` function to compute sin(30°), sin(60°), and sin(90°). What does this tell you about the way that angles are represented in MATLAB?

Downloading and Running MATLAB Functions

1. Download the `quadraticRoots.m` file from the Lab web page. *Hint*: Right click on the link and use "*Save as ...*" (or "*Save link as ...*") to put the m-file in a good location. *Hint*^{2}: A good location would be in a folder like `N:\ME350\Lab\01`.

2. Orient the MATLAB file browser so that the `quadraticRoots` function is visible.



3. Run the following commands

```
quadraticRoots(1,5,2)
quadraticRoots(-3,5,2)
r = quadraticRoots(1,5,2)
```

How do you know whether these are the correct values for the roots of the quadratic equation. Refer to the `Examples` web page that describes the `quadraticRoots` function,
`http://web.cecs.pdx.edu/~gerry/class/ME350/example/quadraticRoots.html`

Plotting Data

For each of the following exercises, create an m-file function that carries out the MATLAB calculations. Create a separate m-file for each exercise. In other words, *do not* reuse the same m-file. Store your m-files in a folder for this lab. As suggested above, a good location would be in a folder like `N:\ME350\Lab\01`.

Example: Plot $\sin(\theta)$ for $-2\pi \leq \theta \leq 2\pi$.

The following function plots $\sin(\theta)$ for $-2\pi \leq \theta \leq 2\pi$. The code is stored in `sinePlot.m`

```
function sinePlot
% sinePlot  Demonstrate plot of sin(theta) on -2*pi <= theta <= 2*pi

theta = linspace(-2*pi, 2*pi);   %  Create a row vector
y = sin(theta);                  %  y is a vector w/ same "shape" as theta
plot(theta,y);                   %  Plot with solid line connecting points
xlabel('\theta');                %  \theta draws greek letter "theta"
ylabel('sin(\theta)');

end
```

Exercises

1. Plot $\sin(\theta)$ and $\cos(\theta)$ on the same axes with solid lines connecting the $\sin(\theta)$ values and open circles at the values of $\cos(\theta)$.

   Hint: There are two primary ways to add two or more curves to the same plot. In this situation, where it is possible to evaluate and store $y = \sin(\theta)$ and $z = \cos(\theta)$ in advance, the *preferred* way is like this:

   ```
   theta = ...
   y = sin(theta);
   z = cos(theta);
   plot(theta,y,'-',theta,z,'o');
   ```

   In other situations, where it is not possible to compute both curves in advance, i.e., where the second curve is only known at a later point in a sequence of calculations, the following approach is necessary.

   ```
   theta = ...
   y = sin(theta);
   plot(theta,y,'-')

   % ... do some more calculations ...

   z = cos(theta);
   hold('on')
   plot(theta,z,'o');
   hold('off')
   ```

   Novice programmers tend to overuse the `hold('on')`/`hold('off')` method, which can sometimes cause unexpected behavior in the plots.

2. Plot $y = \exp(-1.5t)\sin(4t)$ for $0 \leq t \leq 5\pi$. *Hint:* The array operator `.*` is needed in *only one place.*

3. Execute the following statements. (*Remember*: you are saving these statements in a function m-file with a unique name.)

```
x = linspace(0,3);
y = 10*exp(-2*x);
plot(x,y);
grid('on');

figure('Name','Semilog version');   %  A simple "figure" will suffice

semilogy(x,y);
grid('on')
```

   Discuss with a neighbor in the lab:

   (a) How is using `figure` different from using `hold('on')/hold('off')`?

   (b) What other plot commands allow logarithmic scaling?

   (c) What is the advantage of using `semilogy(x,y)` instead of `plot(x,log(y))`?

4. Plot the data in the `T1` and `dT2` arrays

```
T1 = [ 32.50, 32.9, 33.07, 34.91, 36.28, 37.73 ];
dT2 =[  9.45,  9.17,  9.05,  8.72,  8.58,  8.09 ];
```

   as open circles, and on the same axes plot the function

$$T = 16.5518 - 0.2230127\ dT$$

   as a dashed red line. Add a legend with

```
legend('Data','Linear fit')
```

   Add axis labels with

```
xlabel('T_1 ({^\circ}C)');
ylabel('\Delta T_2 ( {^\circ}C)');
```

   Note that the `{^\circ}` and `\Delta` are included to demonstrate how it is possible to insert mathematical symbols in plot annotations. A simpler version of axis labels, like the following, is OK.

```
xlabel('T1 (C)');
ylabel('dT2 (C)');
```