

METR 3250 – LABORATORY EXERCISE #5 – SPRING 2009
PLOTTING A SOUNDING USING MATLAB

Big thanks to Gretchen Mullendore at UCLA for the basis of this lab!

Before beginning, you need to copy some data into your home directory.
In the terminal window, please type the following:

```
ls /nfs
ls /nfs/share
cp /nfs/share/gso.txt .
```

If you type “ls”, and you do NOT see “gso.txt” in the list, please consult the lab instructor before going any further.

If it is there, type [more gso.txt](#) in the terminal window to see what’s in there. You should see something that looks like this:

```
9  9970  277  110  40  140  5
5  9730  479  94  47  99999  99999
5  9670  530  102  32  99999  99999
6  9579  609  99999  99999  175  12
```

The first column describes the data, the second column is the pressure (in mb*10), the third column is the height above sea level (in m*10), the fourth column is the temperature (in C*10) , the fifth column is the dewpoint (in C*10), the sixth column is the wind direction and the seventh column is the wind speed.

Now, to open Matlab, type [matlabStart](#) in the terminal window.
The box on the right is your [command window](#).
The upper-left box lists the variables currently in memory.
The lower-left box lists the commands you recently typed.

A Few Basic Commands

Type the following in your command window (or cut and paste).

```
% The words after '%' are comments and explanations
% They will not affect the commands and do not need to be typed in
```

```
A = [1 2 3]      % the variable list now shows a 1x3 matrix
```

Push the up arrow and add a semi-colon to the previous command. This tells Matlab to not echo the results to the command window. This is useful when operating on large matrices.

```
A * 4           % basic math is very simple in Matlab- just type the expression
as you would write it
```

```
B = A * 4       % assignment to a new variable
```

```

C = [0:6; 0:6] % creates a 2x7 matrix
           % a colon between two numbers will be a single count array
           % a semicolon inside the brackets indicates a new row

clear A B    % clears the variables 'A' and 'B' from the desktop
           % clear by itself will clear all variables

C = [0:.2:6; 0:.2:6] % the rows in the matrix step by .2, instead of 1
           % a negative step can also be used

C(1,1)      % element in row 1, column 1
C(:,end-2:end) % elements in all rows (1:2), columns 29 to 31

```

The `help` Command

This is the easiest way to find out more about specific Matlab commands. The command `help < name >` gives information about the Matlab command `< name >`.

```

help sin    % Information about sine.
help i      % Information about i.
help log    % Information about log (base e)

```

By itself, `help` gives a list of topics in Matlab. To explore the topic 'elementary functions', use the command `help elfun`. The `lookfor` command can also be used to search for relevant information.

Basic Scalar Arithmetic

The simplest way to start with Matlab is to use it for simple calculations. Matlab has a wide range of functions and is able to use complex numbers as well as reals. The following simple commands have obvious meanings. Type them in and see what happens. (In the examples below, all the text after the `%` is just a comment or an explanation. You do not have to type it in. It would just be ignored by Matlab.)

```

(-1+2+3)*5 - 2/3    % The four arithmetic operations

2^3                 % Means 2 to the power 3

exp( sin( pi/2 ) )  % The usual functions are provided
                   % log, log10, cos, tan, asin, ...
                   % Note Matlab always uses radians.

```

Basic Matrix Arithmetic

```

A = rand(3,4) % A 3x4 matrix filled with random numbers
x = rand(4,1)
b = A*x       % This will give you a 3x1 matrix, normal matrix multiplication.

```

Usually, working with data means we want to multiply two matrices (for example, multiplying density and temperature to find pressure), but we would like to multiply ELEMENT by ELEMENT. To do this, use the `.*` operator.

```

A = rand(3,4) % A 3x4 matrix filled with random numbers
x = rand(3,4)
b = A.*x      % This will give you a 3x4 matrix, element multiplication.

```

Roundoff Error

Like a calculator, Matlab does all calculations correct only to about 16 significant decimal digits. This is enough accuracy for most purposes. For convenience, usually only the first 5 significant digits are displayed on the screen. Some more examples.

```
pi
22/7                % pi is not the same as 22/7!
11*(15/11) - 15    % This shows there is roundoff error
                  % when Matlab uses fractions.
```

Plotting a Sine Wave

```
x = [0:.2:6]
y = sin(x)
plot(y)
```

This will plot the sine function you just saved to the variable `y`. Notice that it just plotted the values of `y` versus the index of each data point (from 1 to 31). This is a basic 2D plot. You can see a list of all the 2D plots available by typing [help graph2d](#).

```
plot(x,y)
grid on          % adds grid
```

Now we have the proper x-values along the bottom axis.

Plotting a Sine Wave AND a Cosine Wave

Now suppose you have sine AND cosine data. You could plot them on top of each other, on the same graph, or you could put them on separate graphs. For example:

```
yy = cos(x)

subplot(2,1,1)
plot(y)
ylabel('sin(x)')
subplot(2,1,2)
plot(yy)
ylabel('cos(x)')
```

So what is this 'subplot' thing? = `subplot(m,n,p)` breaks the figure window into an m-by-n matrix of small axes, selects the pth axes object for the current plot, and returns the axes handle. The axes are counted along the top row of the figure window, then the second row, etc. For example,

```
subplot(2,1,1), plot(income)
subplot(2,1,2), plot(outgo)
```

plots income on the top half of the window and outgo on the bottom half.

Download sonde data to your directory

I downloaded a Greensboro, NC sounding (12Z March 26th) from [NOAA](#). Go ahead and download this file to your working directory by right clicking on the link below and saving the file.

Open the file in an editor of your choice. Looking at information on the [format](#) of the sonde file reveals that the first 4 rows are header data. We need to delete these lines before loading the ASCII data into Matlab.

Loading ASCII data

Now that you've cropped the header information, we can load the ASCII data into Matlab.

```
load gso.txt           % loads matrix with size 110x7
p = gso(:,2);         % save pressure to variable 'p'
T = gso(:,4);         % save temperature to variable 'T'
plot(T/10,p/10);      % plot pressure (in mb) versus temperature (degrees C)
axis('ij')           % this flips the y-axis so that pressure decreases
                    upward
```

Skipping bad data

Well, that plot doesn't look very good. It's because the bad data points are filled with the value '99999'. We want to replace those values with 'NaN' (not a number). When plotting, Matlab skips values that are NaN.

```
pcorr = p;            % save pressure to the 'corrected pressure'
                    % variable
pcorr(find(p==99999)) = nan; % find all values that equal 99999 and replace
                    % with NaN
                    % notice that '==' is use for logical comparison

Tcorr = T;
Tcorr(find(T==99999)) = nan;
plot(Tcorr/10,pcorr/10);
axis('ij')

                    % But what if we don't want breaks in our plot?
nums = find( (~isnan(Tcorr)) & (~isnan(pcorr)) );
                    % this saves the indices of points that
                    % are NOT NaN (i.e. are actual numbers) in BOTH
                    % Tcorr and pcorr

plot(Tcorr(nums)/10,pcorr(nums)/10);
axis('ij');
```

Try it yourself - YOUR ASSIGNMENT FOR THE DAY!!

Load the other three variables from the sounding (dewpoint, wind direction, wind speed) and plot all four variables: temperature, dewpoint, wind speed, and wind direction, versus height. Plot these all on one plot, using 'subplots' to separate the images. You should get something that looks like this:

