Physical Sciences 10 – MATLAB Cheat Sheet
Fall 2011

- What is MATLAB?
  - MATrix LABoratory
  - An easy (hopefully) tool for mathematical and technical computing; it's great for linear algebra, creating plots of functions, solving differential equations, etc.
- Where can I use MATLAB?
  - Science Center Computer Lab
  - Download it (http://downloads.fas.harvard.edu/download). You will need to figure out whether you have a 32-bit or 64-bit computer, and install either Matlab x32 or Matlab x64, accordingly.
- Where can I get help for MATLAB?
  - There is a tutorial here: http://www.mathworks.com/academia/student_center/tutorials/mltutorial_launchpad.html. Look at: Navigating the MATLAB Desktop.
  - If you know the name of the command, type '`help <command>`' in MATLAB
  - If you don't know the name of the command, http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html
  - Office hours
- Basic syntax
  - Defining a variable: `x = 2`
  - Defining a vector [1 3 5]: `y = [1 3 5]`
  - Defining a matrix $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$: `z = [1 2; 3 4]`
  - Suppressing output: `;`
  - Accessing an element in a vector: `y(2)` *(equals 3)*
  - Accessing an element in a matrix: `z(2, 1)` *(equals 3)*
  - Accessing a row/column in a matrix: `z(2, :)` *(equals* [3 4]*)*
  - Accessing elements in a vector that satisfy some condition
    `y(y>2)` *(equals* [3 5]*)*
  - Accessing indices of elements in a vector that satisfy some condition
    `find(y>2)` *(equals* [2 3]*)*
  - Transposing a vector: `y'` *(equals* $\begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$*)*
  - Mathematical operators: `+, -, *, /, ^`
  - When working with a vector: `2.[+, -, *, /, ^]y`
    Note the dot! [1 2 3].*y = [1 6 15]
  - Special numbers (π, ∞, $\sqrt{-1}$): `pi, Inf, i`
  - Exponential (e^x): `exp(x)`
  - Natural logarithm (ln x): `log(x)`
  - Trigonometric functions: `cos(x), sin(x), tan(x), etc.`

Adapted from Matlab Cheat Sheet by Zeba Wunderlich

- Absolute value ($|x|$): `abs(x)`
- Real part of a variable `real(x)`
- Imaginary part of a variable `imag(x)`

- Basic commands
  - Ways to make vectors:
    - Equally spaced intervals: `x = 1:0.5:3` *(equals* $[1 \quad 1.5 \quad 2 \quad 2.5 \quad 3]$*)*
    - All zeros: `x = zeros(1, n)` *(equals* n *zeros)*
    - All ones: `x = ones(1, n)` *(equals* n *ones)*
  - Draw *n* uniformly distributed random numbers between 0 and 1 `rand(1, n)`
  - Draw *n* normally distributed random numbers `randn(1, n)`
  - Minimum of vector `min(y)`
  - Maximum of vector `max(y)`
  - Sum of vector `sum(y)`
  - Sum of a matrix along a particular dimension (columns in this example) `sum(y, 2)`
  - Mean of vector `mean(y)`
  - Variance of vector `var(y)`
  - Size of vector `size(y)`
- Commands for plotting functions
  - Display help for plot function `help plot`
  - Plot y vs. x `plot (x, y)`
  - Plot next graph on top of prev. graph `hold on`
  - Turn grid on plot `grid on`
- Keyboard shortcuts
  - Comment out some code highlight it, then `Ctrl-r`
  - Uncomment the code highlight it, then `Ctrl-t`
  - Run a piece of code highlight it, then `F9`
  - Run a complete m-file `F5`
  - Abort a calculation From the command line, `Ctrl-c`
- More plotting commands:
  - To set the x-axis limits to $[x_1 \; x_2]$ `xlim([x1 x2])`
  - To set the y-axis limits to $[y_1 \; y_2]$ `ylim([y1 y2])`
  - To add an x-axis label `xlabel('label')`
  - To add a y-axis label `ylabel('label')`
  - To add a title to the plot `title('title')`
  - Plotting with different colors, markers, etc. `plot(x, y, '…')`
    Where the … can be one (or a combination) of the following:

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
R	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star		
y	yellow	s	square		
k	black	d	diamond		
		v	triangle (down)		

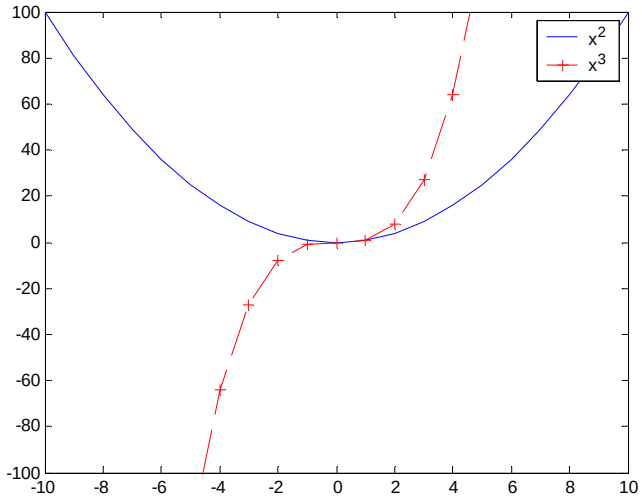
```
^     triangle (up)
<     triangle (left)
>     triangle (right)
p     pentagram
h     hexagram
```

- To add a legend: `legend('label1', 'label2'…)`
  Where label1 is the label for the first thing plotted, label2 the second, etc.

- To copy figures in MATLAB, go to Edit → Copy Figure. (Ctrl-C won't work.)

Example: Manipulating Plots

```
x = [-10:10];
y = x.^2;
y2 = x.^3;
plot(x, y);
hold on;
plot(x, y2, 'r--+');
legend('x^{2}', 'x^{3}');
ylim([-100 100]);
```



Example 1: Plot of $y = e^{-0.2x} \sin x$

```
% lines/text preceded by "%" are comments
x = 0:pi/40:4*pi;              % set range of x to (0, 4π)
y = exp(-0.2.*x).*sin(x);      % calculate y from x
plot(x, y);
x2 = x;
y2 = zeros(size(x));           % create y2 = 0
hold on;                       % overlay next plot on previous plot
plot(x2, y2, 'r');
```

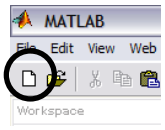
Example 2: Analyzing a set of data

```
x = rand(100, 1);          %create random data sets
x10 = 10*rand(100, 1);
mean(x)                    %compare means
mean(x10)
var(x)                     %compare variances
var(x10)
mean(x10(x10>5))           %find mean and variance of the elements of x10 > 5
hist(x);                   %create a histogram of x
title('Data Set 1');       %add labels
xlabel('x');
ylabel('Frequency');
hold on;
plot(0:0.1:1, 10*ones(11, 1), 'r'); %plot to line y = 10 on the histogram
```

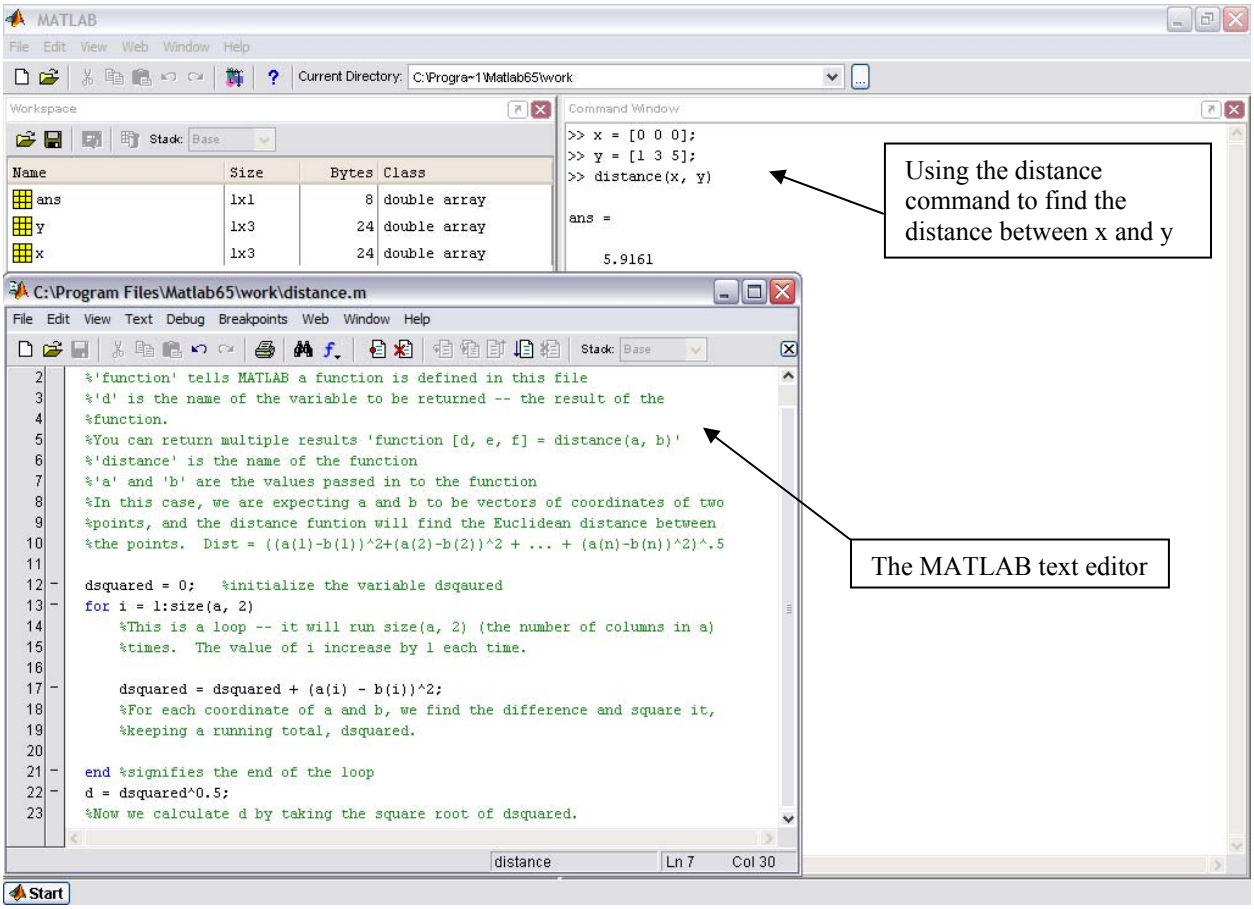
Adapted from Matlab Cheat Sheet by Zeba Wunderlich

Writing Functions

A function is a series of MATLAB commands saved in a separate text file, called an "M-file." (These files are named `<function>.m`) You can call this series of commands from MATLAB by using the function name.

To write a function, click on the "New M-File" button on the toolbar. 

This will open up the MATLAB text editor. In this new window, you will write the function, and once you are done, you will save the file with the file name `<name of function>.m`. In the example, I write a function, called distance, which calculates the distance between two points, and save it as `distance.m`.



One important note about writing functions: where you save the M-file matters! MATLAB only looks in certain folders to find M-files. So once you have saved your file, you must make sure its folder is included in the MATLAB path. To do this, go to File > Set Path… and then click on "Add Folder…" Find the folder where you M-file is saved, click "Ok" and then click "Save." You must do this each time you save a function in a new folder.

Another Example of a Function

```
function rounded = round(x)
%This function takes in a vector or matrix of numbers and rounds them
```

```
rounded = zeros(size(x));   %Create rounded, a vector/matrix same size as x
%This is the "outer loop."  It iterates over all the rows of x. For a vector,
%there is only one row, for matrices, there are many.
for i = 1:size(x, 1)
    %This is the "inner loop."  It iterates over all the columns of x.

    for j = 1:size(x, 2)
        integer = floor(x(i, j)); %floor rounds x down to the nearest integer
        remainder = x(i, j) - integer; %gives decimal part of x
        if (remainder < 0.5) %test to see if we should round down
            rounded(i, j) = integer;
        elseif (remainder == 0.5)
            %test to see if remainder is exactly 0.5.  note that to test
            %equality, we must use "=="
            rounded(i, j) = integer+1; %we round up
        else %remainder must be > 0.5
            rounded(i, j) = integer+1; %we round up
        end %ends if statement
    end %ends inner loop
end %ends outer loop
%Note that we could have eliminated the four lines starting from elseif,
%since we do the same thing whether remainder is greater than or equal
%to 0.5.  I just wanted to demonstrate the use of the "==" sign.
```

And an example of a use of round:

```
>> round([0.222 3.4; 5.5 6.89])
ans =
     0     3
     6     7
```

A Note about Loops

In the above example, we have used something called a "for loop." The basic form of a for loop (where *a* is some vector) is:

```
for n = a
      commands
end
```

This loop will be repeated a number of times equal to the size of the vector *a*. During the first iteration $n = a(1)$, and all the commands are executed. During the next iteration, $n=a(2)$, the commands are executed and so forth.

However, there is a different type of loop, a "while loop," which can also be used in MATLAB. The basic form of a while loop is this:

```
while (condition)
      commands
end
```

This loop will execute while the “condition” is true. When it is false, it will move on to the next block of code. This loop may repeat 0, 1, or any number of times (including an infinite number of times, which is something to watch out for!)

Any for loop can be rewritten as a while loop. For example:

```
a = [3 4 5];
y = 0;
for (i = a)
      y = y + i;
end
```

Instead we could write:

```
a = [3 4 5];
y = 0;
i = 1;
while (i <= size(a, 2))
      y = y + a(i);
      i = i + 1;
end
```