Chapter 1

Getting Started

How to Get Matlab

Matlab physically resides on each of the computers in the Olin Hall labs. See your instructor if you need an account on these machines.

If you are going to go to graduate school at some point, or if you think Matlab might be something you will want a personal copy, Matlab is available to download from the Mathworks website. Its a good deal- $99 for Matlab and 10 of the most used toolboxes (some of which we don't have).

Lastly, there is a clone of Matlab called **Octave** that is available for free from the internet. As is often the case with open source software, the documentation can be hard to read and help can be hard to locate. In the past, Octave for the Mac has been tricky, so use it only if you're pretty comfortable tweaking a Mac. The Linux and Windows distributions are fairly reliable.

Of course, if you have any trouble putting Octave on your own machine, there's always the Olin Hall computer labs!

1.1 Before We Begin...

When you first log in: Using a computer in the Math/CS lab, go to the upper left corner, and choose the top icon on the launch pad. You should get a search dialog, so you can type:

```
matlab
```

You will then get the Matlab icon, so select that and Matlab will start. To keep the Matlab icon on the launcher, go to the icon, right-click. In older Ubuntu versions, select "Lock to Launcher", in more recent versions, select something like "Copy to Favorites".

See Figure 1 to see the opening interface that you'll use. You might take a few minutes to just look around the interface- Try some different tabs and see what happens.

1.1.1 Matlab to Accompany Lay's Linear Algebra Text

There is a suite of functions that we can use to make Matlab a little more user friendly when we use Lay's linear algebra text. The `LayData4` toolbox is available for you to download from the course CLEo site. It comes as a zip file. After you open Matlab for the first time, you'll probably have a new matlab folder in your directory. To make LayData4 accessible to Matlab,

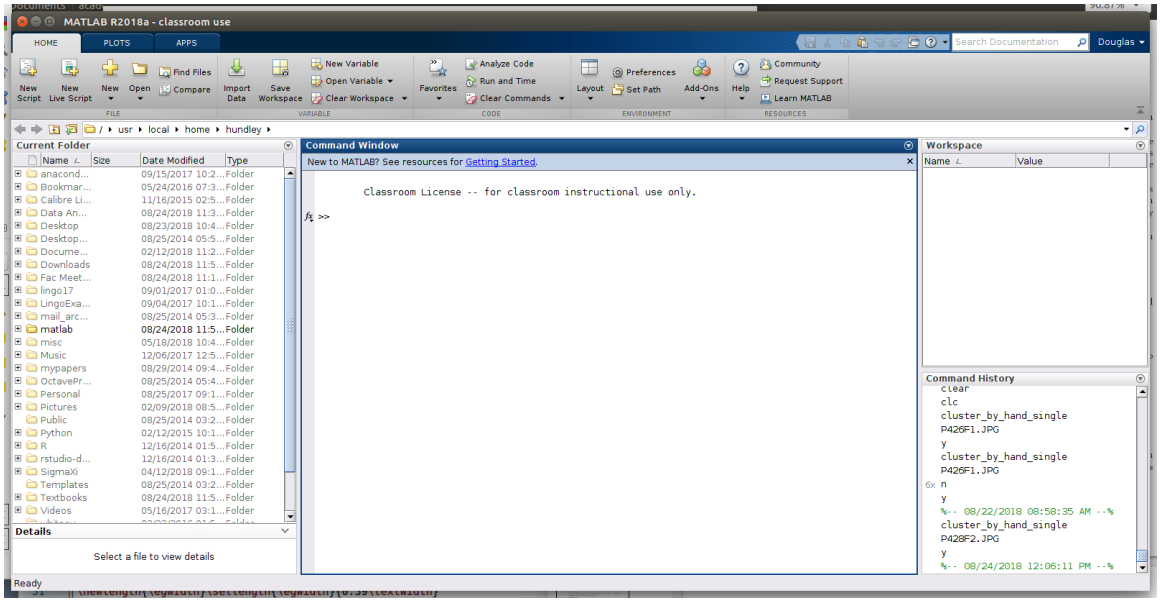


Figure 1.1: The opening interface for Matlab. The middle window is the *command window*, and is where we type Matlab commands. The other windows are there to give us information about the Workspace, the current directory, and a history of commands. Yours should be basically blank at this point.

- Put the LayData4.zip file into your Matlab folder, and unzip it. It should unzip into a LayData4 folder.
- Type `pathtool` in the Matlab command window. We're going to add the LayData4 folder to the path by pressing the "Add Folder" button.
- Once you've added this Matlab folder to Matlab's search path using `pathtool`, you're all set.

You may see a dialog box asking you to save your `pathdef.m` file. You might go ahead and save it to your local Matlab directory.

1.1.2 How does Matlab work?

You can make Matlab do computations three different ways:

- Type commands directly into the keyboard.
- Have your Matlab commands typed into a separate text file (called a **script file**), and then have Matlab read these commands in. This is very nice- it gives you documentation and allows you to run similar computations several times without having to re-type the commands. We will typically use script files to do homework problems.

  To turn in homework, you will "publish" your script, then upload the PDF to your CLEO dropbox (See our course website for more information).
- Define your own functions by typing a separate text file (called an **m-file**).

Saving your work

Matlab keeps track of your past history while you are typing on the keyboard, but for answering homework, it is best to use a *script file*, which is just a text file with Matlab commands on it. More on this later.

Matlab has a very nice text editor that you can use to type out and save Matlab functions and scripts- To access the editor, type `edit` in the Matlab command window.

1.2 Introductory Commands

On the left, we show the commands you can type in the command window, and to the right we give some commentary. Matlab uses the standard mathematical operations:

1.2.1 Numbers

```
2+3;
123.3*sin(3.2)
```

If you leave the semicolon off the end of the line, Matlab prints the result (this is OK when typing live, but usually not good for scripts- We'll see that later).

```
2^5;
exp(4);
cos(1.3);
log(3)
```

The function `exp(x)` is used for e^x. Sine and cosine assume the input is in radians. The logarithm assumes natural log.

```
i^2
(1+i)*(2-i)
pi
exp(1)
```

The number $i = \sqrt{-1}$ is defined in Matlab- But only if it hasn't been defined by you to be anything else. The constant π uses lowercase p (unlike Maple). To get the constant e, use `exp(1)`.

```
5/0
0/0
eps
```

In the first case, you'll see `Inf`, which represents "infinity". This is actually incorrect- It should be $\pm\infty$, since we don't know how the denominator is reaching zero. In the second case, Matlab gives `NaN`, which means "Not a Number". Do you know why this would be? (think limits). The last value is a very small number below which we typically would think of a quantity as "zero".

1.2.2 Helpful Administrative Commands

The following commands are useful as you begin to use Matlab more and more:

```
clear;
clc;
whos;
help
% Help for a specific command
help sin
doc
demo
```

The up arrow key is useful (try it!). Here we first clear the workspace memory of all variables, and secondly we clear the command window. Give the others a try to see what they do. Notice that we can use a percent sign to put in a *comment* (Comments are not processed by Matlab).

1.3 Assigning values to variables

To assign data to a variable name, we would use the equal sign. For example, $x = 3$ assigns the value of 3 to the variable x. The array (or matrix) is the basic data type in Matlab. Data entry is straightforward.

```
A=[1 2 3 4; 5 6 7 8];
A=[1 2 3 4
5 6 7 8];
A(2,3)
```

A is an array with two rows and 4 columns and can be entered in either manner. Inside an array, the semicolon indicates the end of a row. The $(2, 3)$ element (2d row, 3d column) of A is accessed as this.

```
x=[1;0;1]; y=[-1;2;3];
dot(x,y)
cross(x,y)
```

The dot product and cross product are available.

The dot

If we want to take an array of numbers and perform some operation to every value, we can use the dot. Here are some examples of how this works:

```
x=[1,4,7,10];
x.^2
sin(x)
y=[2 3 4 5];
x./y
```

The array x is first formed, then we take the square of each element. Similarly, the third line takes the sine of each element. If we define y as another array of the same size, we can "divide" x by y, where the first element will be $1/2 = 0.5$ and the last element is $10/5 = 2$.

Special Commands: The colon operator

- `a:b`

  Produces a vector of integers from a to b in a row.

- `a:b:c`

  Produces the numbers from a to c by adding b each time" $a, a + b, a + 2b, \ldots$, up until the last number is biggest that is less than or equal to c.

- `linspace(a,b,c)`

  Produces c numbers evenly spaced from the number a to the number b (inclusive).

```
x=2:9;
y=1:2:8
z=10:-1.3:3
w=linspace(2,5,40);
w=linspace(2,5)
```

First, x is an array with the integers from 2 to 9. Secondly, y is array with integers $1, 3, 5, 7$. Can you predict what z will be? The last line produces 40 numbers evenly spaced beginning with 2 and ending with 5. If we leave the third number off, we get 100 values as the default.

1.4 Basic Plots

Here's a quick example to get us started:

```
x=linspace(-pi,3*pi,200);
y=sin(x);
plot(x,y);
plot([1,2],[3,4]);
```

You'll see that we had to create a domain array and a range array for the function. We then plot the arrays. In the last line, Matlab will plot a line segment between the points $(1, 3)$ and $(2, 4)$. So we are drawing small line segments between data points in the plane.

1.4.1 Examples

```
x1=linspace(-2,2);
y1=sin(x1);
y2=x1.^2;
x2=linspace(-2,1);
y3=exp(x2);
plot(x1,y1,x1,y2,x2,y3);
```

These commands produce a single plot with all three functions.

```
plot(x1,y1,'*-');
```

Plots the function `y1`, and also plots the symbol `*` where the data points are.

```
plot(x1,y1,'k*-',x2,y3,'r^-');
```

Plots the function `y1` using a black (k) line with the asterisk at each data point, PLUS plots the function `y2` using a red line with red triangles at each data point.

The following lists all of the built in colors and symbols that Matlab can use in plotting: (NOTE: You can see this list anytime in Matlab by typing: `help plot`)

Code	Color	Symbol	
y	yellow	.	point
m	magenta	o	circle
c	cyan	x	x-mark
r	red	+	plus
g	green	−	solid
b	blue	*	star
w	white	:	dotted
k	black	−.	dashdot
		−−	dashed

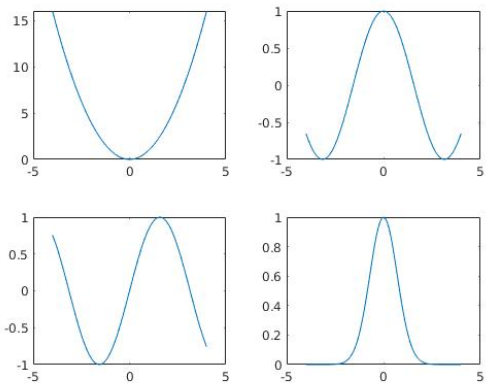
The following sequence of commands also puts on a legend, a title, and relabels the $x-$ and $y-$axes: Try it!

```
x=linspace(-2,2);
y1=sin(x);
y2=x.^2;
plot(x,y1,'g*-',x,y2,'k-.');
title('Example One');
legend('The Sine Function','A Quadratic');
xlabel('Dollars');
ylabel('Sense');
```

Multiple plots in one graph

Often its useful to put multiple plots in one graph, and that's what the subplot command is used for. Here's an example, where we insert the plots into a 2×2 array of plots. The graph this produces is shown in Figure **??**.

```
x=linspace(-4,4);
y1=x.^2;
y2=cos(x);
y3=sin(x);
y4=exp(-x.^2)
subplot(2,2,1);
plot(x,y1);
subplot(2,2,2);
plot(x,y2);
subplot(2,2,3);
plot(x,y3);
subplot(2,2,4);
plot(x,y4);
```



Other Things: If you look at the plotting window from the last example, you'll see lots of things that you can do. For example, there's a zoom in and a zoom out feature. You can also edit the colors and symbols of your plot, and the title, legend and axis labels. Try them out!