Introduction to Matlab

What is Matlab

The software program called *Matlab* (short for MATrix LABoratory) is arguably the world standard for engineering- mainly because of its ability to do very quick prototyping. It was originally introduced as a front end to work with vectors and matrices, but since then has gotten quite user friendly for those with little background in higher mathematics.

Matlab versus Maple

The primary distinction between Matlab and Maple (and Mathematica) is that Maple (and Mathematica, and WolframAlpha) are designed primarily as "computer algebra systems" or CAS. These systems have the ability to manipulate expressions symbolically. Matlab is designed to work numerically.

Therefore, the basic version of Matlab does not perform symbolic differentiation or integration, or many of the other things we're accustomed to doing in Maple. Rather, we have algorithms that will (hopefully) perform these operations numerically. We don't need to worry about that for the time being, however.

Getting Started

Matlab resides on the computers in the Math Computer Lab (Olin 248). Unfortunately, our license does not permit us to allow students to put Matlab on their home computers. Therefore, to use Matlab requires us to be in the lab.

If you need an account or need help getting started with the computers in Olin, see me first. Alternatively, we can use **Octave Online**, which is a free Matlab clone. Octave may also be downloaded and run on your home computer if you like, but I think just about everything we do in class we'll be able to do using the free online version.

For more information about the Matlab user interface, Matlab has intro videos on Youtube: (also linked from our class website)

`https://www.youtube.com/watch?v=OHxR8iMHDWw&list=PL7CAABC40B2825C8B`

Basic Arithmetic and Assignment

Here are some basic arithmetic and assigment commands to get you started- Think about what you're doing as you type these into the command window, and see if the Matlab output is what you expect.

```
x=pi
y=3*sin(x/2)
w=2^3
z=y+exp(-y)
log(z)
z=(1+3*i)*(5-2*i)
help log
whos
```

Things to notice (especially as a comparison to Maple):

- Matlab uses `=` as an assignment (in Maple, this was `:=`). For example, $w = 2^3$ will assign the value 8 to the variable w, while $2^3 = w$ is not a valid command.
- Matlab uses small case π (in Maple, the constant is `Pi`).
- In both Matlab and Maple, the exponential function is `exp(x)` rather than e^x.
- Use the up-arrow key to have Matlab give you a previously typed command. For example, re-do the line `z=y+exp(-y)`, but add a semi-colon at the end. What happens?

Creating Arrays and Manipulating Arrays

Before proceeding, clear the memory and the screen:

```
clear
clc
```

Arrays: Matrices and Vectors

The way numerical data is typically stored is in an **array**, or matrix. The **size** of the array is given as the number of rows by the number of columns. A vector could be a row or a column (and they act differently).

The basic variable type that Matlab works with is the array. In the command window, here is an example of a vector and a matrix:

```
x=[1;-1;0;0];
A=[1 2 3;-1 0 2];
```

In each case, the semicolon inside the array denotes the end of a row (so that **x** is a column instead of a row). The semicolon at the end of the line suppresses printing (try leaving it off).

Alternatively, the matrix A could have been entered as the following. We can ask Matlab for the size of the array as well:

```
A=[1 2 3
-1 0 2];

[numrows, numcols]=size(A)
```

The size "function" takes in an array A and outputs the number of rows and the number of columns it has (in that order). We can assign these to variable names, as shown, or we just want the size, we could type just type: `size(A)`

We access elements of the matrix in a natural way in Matlab. For example, the $(2, 3)$ element of A is written as `A(2,3)` in Matlab (in this case, $A(2, 3)$ is 2). You can change the elements using the assignment operator `=`. For example, if we want to change the $(1, 3)$ element of A from 3 to 6, type:

```
A(1,3)=6;
```

Special Commands: The colon operator

There are special arrays that are built-in to Matlab, and some use special notation. The symbol `:` is used a lot. Here are a couple of examples, then we'll give the general case:

```
x=2:9
x=8:-2:1
x=2:3:10
```

So what does the general command `a:b:c` do?

- `a:b`

  Produces a row vector using the numbers $a, a+1, a+2$, etc., where the last element is less than or equal to b.

- `a:b:c`

  Produces the numbers from a to c by adding b each time. The last element is the largest that is less than or equal to c.

Questions:

1. What will Matlab do when you type: `a=0.5:4`? (Answer before typing it in!)
2. What will Matlab do when you type: `b=0:0.3:0.7`? (Answer before typing it in!)

Matlab commands associated with Arrays

- `linspace(a,b,c)`

  Produces c numbers evenly spaced from the number a to the number b (inclusive). For example, `x=linspace(2,3.5,40)` produces 40 numbers evenly spaced beginning with 2 and ending with 3.5.

  SHORTCUT: Leaving off the third number c will give you 100 numbers between a and b (That is, $c = 100$ is the default value.)

  Compare this with the colon operator. We would use the colon operator if we want to define the length between numbers, and use `linspace` if we want to define the endpoints.

- Random arrays (handy if you just need some quick data!)

  `A=rand(m,n)` Produces an $m \times n$ array of random numbers (uniformly distributed) between 0 and 1. If you just want a single random number between 0 and 1, just type `rand`

  `A=randn(m,n)` produces an $m \times n$ array of random numbers (with a normal distribution) with zero mean and unit variance. If you want a single random number (with a normal distribution), just type `randn`

- `A=zeros(m,n)` Produces an $m \times n$ array of zeros.

- `A=ones(m,n)` Produces an $m \times n$ array of ones.

- `A=eye(n)` Produces an $n \times n$ identity matrix (zeros everywhere except for 1's along the diagonal).

- `A=repmat(B,m,n)` Matrix A is constructed from matrix (or vector) B by replicating B m times down and n times across.

  Example: Let $B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$. Then `A=repmat(B,2,3)` creates the array:

```
A =
     1     2     1     2     1     2
     3     4     3     4     3     4
     1     2     1     2     1     2
     3     4     3     4     3     4
```

Some Operations on Arrays

- You can add arrays of equal size (each has to have the same number of rows and columns). The result is performed elementwise- For example,

```
A=[1 2 3;4 5 6;7,8,9];
B=eye(3);
C=A+B
```

The result C adds 1 to each diagonal element of A.

- To raise each element to the same power, use `.^` For example, to square every element in A, type:

```
A.^2
```

- You can multiply elementwise using `.*` For example,

```
C=A.*B
```

Important Note: In Linear Algebra, we learn how to multiply two matrices A and B. If you leave off the dot, that is what you're telling Matlab to do. Therefore, the dot is really important here- That is,

$$A * B \neq A. * B$$

- Most other built-in functions act on array element by element- Here are a couple of examples:

```
sin(A)
exp(A)
```

Side Remark: The matrix exponential, e^A, is actually defined as a special type of matrix- There is a separate Matlab command for that. We will only work with it element-wise.

- You can "flip" an array by making the first row the new first column (and so on). This is called **transposition**, and is performed in Matlab using an apostrophe. For example,

```
A=[1 2 3 4;
5 6 7 8];
A'
```

- We can take the dot product between vectors (as defined in Calc III):

```
x=[1 2 3 4];
y=[5 6 7 8];
C=dot(x,y)
```

(In this case, $C = 5 + 12 + 21 + 32 = 70$)

The Colon

The colon symbol, `:`, is very useful in Matlab. Here are some examples of how it is used:

Accessing Submatrices

Let A be an $m \times n$ array of numbers. Then:

The notation:	Yields:
`A(i,j)`	The (i,j)th element
`A(i,:)`	The entire ith row
`A(:,j)`	The entire jth column
`A(:,2:5)`	The 2d to fifth columns, all rows
`A(1:4,2:3)`	A 4×2 submatrix

Example: What kind of an array would the following command produce?

```
A([1,3,6],[2,5])
```

A 3×2 matrix consisting of the elements:

$$\begin{matrix} A(1,2) & A(1,5) \\ A(3,2) & A(3,5) \\ A(6,2) & A(6,5) \end{matrix}$$

Example: Create a 5×5 zero array, and change it to:

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Answer:

```
A=zeros(5,5);  %Create the matrix of zeros
b=[1 2 3;4 5 6; 7 8 9];
A(2:4,2:4)=b;
```

Note also the use of the `%` sign. It is used to denote *comments*; that is, Matlab would ignore everything on the same line after the `%` sign.

Adding/Deleting Columns and Rows:

Its straightforward to insert and/or delete rows and columns into a matrix. Before doing it, we define `[ ]` as "the empty array": the array with nothing in it.

In the following, let A be a 4×5 array, let b be a 1×5 row, and c be a 4×1 column.

Examples of use (each of these are independent from the previous):

The command:	Produces:
`A(1,:)=[];`	Delete the first row.
`A([1,3],:)=[];`	Delete rows 1 and 3.
`A(:,3)=[];`	Delete column 3.
`A(:,1:2:5)=[];`	Delete the odd columns.
`A(1,:)=b;`	Put **b** as row 1.
`A(:,6)=c;`	Add c as the last column.
`d=[c , A(:,1:3)];`	d is c and columns $1-3$ of A.
`A=[A(:,1), c, A(:,2:5)];`	Insert c as column 2 of A, others shift 1 over.
`A=[A(1,:); b; A(2:4,:)];`	Insert b as row 2 of A, others shifted 1 down.

Example: Matlab comes with some built-in data sets. One such set is the image of a clown. For fun, we'll load the array in, display it, then we'll remove all of the even rows and columns, then re-display it:

```
load clown
whos
image(X);
colormap(map);
X(2:2:200,:)=[];
X(:,2:2:320)=[];
image(X);
```

Once you're done, you may want to clear the memory and the screen:

```
clear
clc
```

How do I get a Plot?

Here's a quick example to get us started:

```
x=linspace(-pi,3*pi,200);
y=sin(x);
plot(x,y);
```

You'll see that we had to create a domain array and a range array for the function. We then plot the arrays. For example,

```
plot([1,2],[3,4]);
```

will plot a line segment between the points $(1, 3)$ and $(2, 4)$. So, Matlab's plotting feature is drawing small line segments between data points in the plane.

Examples

1. Matlab can also plot multiple functions on one graph. For example:

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

   produces a single plot with all three functions.

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

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

3. `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

4. 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');
```

5. 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!

M-Files: Functions and Scripts

A **script** file (or simply, a script) is a plain text file that contains Matlab commands. When you type in the name of a script file, Matlab will execute the commands as if you were typing them in.

For example, open the editor (type `edit` in Matlab), and then type and save the following script:

```
% Exercise 1:  Write a script that will convert 60
%              degrees Fahrenheit to degrees Celsius
%
%SOLUTION:
F=60
C=(9/5)*(F-32)
```

Save this file as *Sample1.m*, then to have Matlab execute these commands, in the command window type: `Sample1`

(You should see F and C). In the editor, if you go to `File -> Publish Sample1.m` and choose to publish, you'll see both the script and the answers in your web browser. We can print this to a PDF file to turn in (we'll go through this again later).

Functions

As we know, functions are rules that change input values to output values. In programming, a function can have multiple inputs and multiple outputs.

For example, suppose I want to write a program that will tell me the cost of producing a cylindrical can. The inputs might be the radius of the can, r, the height of the can, h, and the costs of the material- We might have one cost for the top and bottom, and a different cost for the sides. We'll call these C_t, C_s, respectively.

Algebraically, the cost will be:

$$C = C_t(2\pi r^2) + C_s(2\pi rh)$$

I may also want to know the surface area, so I'll have the function output that, too:

$$A = 2\pi r^2 + 2\pi rh$$

Here is how I would write this in Matlab:

```
function [C,A]=canFunction(r,h,Ct,Cs)
% function [C,A]=canFunction(r,h,Ct,Cs)
%  Computes the cost C and surface area A of a can, given
%  radius r, height h, Ct is the cost (per unit area) of the
%  top and bottom, and Cs is the cost of the side.

TopBottom=2*pi*r^2;
Sides=2*pi*r*h;

C=Ct*TopBottom+Cs*Sides;
A=TopBottom+Sides;
```

Save this file as the function name with a `.m.` suffix, or, `canFunction.m`. Some things to notice about a function:

- The first line should always begin with the word "function". This is how Matlab distinguishes between a script and a function.
- You should always include remarks that tell you how to use the function.

Now in the command window, we can type things like:

```
help canFunction
[C,A]=canFunction(3,6,10,15);
```

You should notice that when the function is called, only the output variable names are present- that is, the variables `TopBottom` and `Sides` that the function uses are only present for the function itself (these are called "local variables" in computer programming).

We will write some functions, but we will probably stick mostly with scripts.

Exercise Set

1. Give the Matlab command (using the colon operator) to create the following array:

$$4.5 \quad 4.0 \quad 3.5 \quad 3.0 \quad \ldots \quad 1.5$$

2. What does the following do?

```
x=[1 2 3 4 5];
y=2.^x;
z=x.*x;
```

3. Given the array:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 2 & 3 & 4 \\ 4 & 5 & 1 & 2 & 3 \\ 3 & 4 & 5 & 1 & 2 \\ 2 & 3 & 4 & 5 & 1 \end{bmatrix}$$

   (a) How do I multiply everything in A by 2, then subtract 3 from every entry (in Matlab)?

   (b) What is the shortest Matlab command to get a subarray using the following elements:

$$B = \begin{bmatrix} A(1,2) & A(1,3) & A(1,4) \\ A(3,1) & A(3,2) & A(3,3) \end{bmatrix}$$

   (c) By hand, what is the transpose of B (from the previous problem)?

   (d) What is the Matlab command to remove the even-numbered columns of A?

   (e) In Matlab, what does `A-eye(5)` do?

4. How do I create an array that has 4 rows, 6 columns, and is filled with random numbers between 0 and 1?

5. Watch the 4 minute video "Working in the Development Environment" on the Matlab channel in YouTube (Google it). Then answer the following questions:

   (a) What are two different ways of saving the variables in the workspace?

   (b) How do you load the variables back into Matlab?

6. Watch the 6 minute video "Writing a Matlab Program" in the same playlist.

   (a) What is a "script"? How do you run a script?

   (b) Write the script that is given in the example, `simple0.m` (Note the zero to distinguish it from the next step).

   (c) Write the function that is given in the example, and save it as `simple.m`

7. After viewing the previous video, determine what Matlab does if you type the following:

```
a=1.5; b=3.6;
for k=1:100
   h(k)=(b-a)*rand+a;
end
```

(Hint: What is the biggest and smallest possible number in the vector h?)

8. Write a script file that will construct a vector x consisting of 225 evenly spaced points in the interval $[-4, 8]$, then proceeds to plot $\sin(x)$ in red, $\sin(2x)$ in black, and $\sin(3x)$ in green, all on the same plot.

9. When we compute with numbers, we might run into some difficulty if the numbers become too large or undefined. Some examples:

   - `eps` is the distance from 1 to the next largest number. That is, Matlab will see something like `1+eps/2` as the same as 1.
   - `NaN` means "Not a Number"- Happens if you do something like 1/0, and `Inf` means that the expression is infinitely large.

   Here are some fun things to try in Matlab- Tell me what happens in each case:

   (a) `eps` (This is scientific notation), `1-(1+eps/2)`

   (b) `1/0`, `-1/0` and `0/0`

   (c) `1/Inf`

   (d) `Inf+Inf`, and `Inf-Inf`

   (e) `Inf/Inf` and `Inf/0`