

Introduction to Matlab Modeling I Fall 2005

Doug Hundley

September 2, 2005

1 Before We Begin...

Starting the Matlab Program

Matlab physically resides on each of the computers in the Math Dept's computer lab.

NOTE: You need to have an account on the Math Lab computers. If you have any questions about logging in, please ask!

NOTE: The Math Lab was reconfigured this summer, so pay attention to the new interface!

When you first log in: Go to the *Application* button at the TOP of the screen, then choose *Accessories*, and you should find the Matlab icon. Drag and drop the icon to the bar at the TOP of the screen, and then you can start Matlab by pressing the icon.

Alternatively, open a "shell" screen, then type `matlab`

SIDE REMARK: You can use Matlab from a computer that is not in the Math lab, but you lose the ability to have a nice front-end, and you won't be able to graph:

- Send your java-enabled web browser to `math.whitman.edu` and open a secure login.
- Log into `math.whitman.edu` NOTE: Matlab is not installed on this computer, so we need an additional login:
- Log into another computer in the math lab, like `ssh avarice`
- Now type `matlab -nojvm`

1.1 If You Know Maple:

There are some significant differences between Matlab and Maple. Here are a few:

- In Maple, we save our work in Worksheets. In Matlab, we will use *script files* and *functions*. This has some implications- If you're working on a project, use Matlab's editor to type your commands into a script- Don't type them live in the command window.
- We use Maple for symbolic work (such as differentiation, integration, etc), and Matlab is meant to be used for numerical work. (Note: It is possible to call Maple from Matlab and vice-versa, but this is a more advanced topic).
- In Maple (until recently), the end of every line must have a semi-colon. In Matlab, if you put in a semi-colon, the output is suppressed. If you do not put in a semi-colon, the output is not suppressed (we will see this in the examples below).

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.
- Define your own functions by typing a separate text file (called an **m-file**).

Saving your work

If you haven't written a script file, but are doing your computations "live", you may want to begin the session by typing: `diary filename`

All subsequent keyboard input and output will then not only be on the computer monitor, but will also be saved as "filename". For example, if you're using Matlab for homework problem 3.1, you may use the command: `diary hw3_1` to save your work.

Important: The "diary" command *must* be used *prior* to typing in the commands you want to save.

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.

2 Introductory Commands

1. Arithmetic

Matlab understands all of the basic arithmetic functions, `+`, `-`, `*`, `/`, `^` are addition, subtraction, multiplication, division and exponentiation. Type them in just as you would write them. For example, 2^5 would be typed as `2^5`.

2. Trigonometric Functions

Matlab understands the basic trig functions sine, cosine and tangent as `sin`, `cos`, `tan`. So, for example, the sine of 3.1 would be typed as: `sin(3.1)`

The number π is used so frequently that Matlab has its (approximate) value built-in as the constant `pi`. For example, $\sin(\pi)$ is typed as `sin(pi)`. Note that π uses a lowercase “P”.

3. Exponential and Logarithmic Functions

Matlab does not have the number e built-in as a constant (like π). To take the number e to a power, use the functional form: $e^x = \exp(x)$. So if I want the number e , I would type `exp(1)`, and so on.

For the natural log (log base e), use the notation `log`. For example, $\ln(3)$ is written as `log(3)`. We will only use the natural log- if in the future you want a different base, look up the `log` command by typing `help log`.

4. Complex Numbers and Arithmetic

Matlab has complex arithmetic built-in. Either the letter i or j can be used to represent $\sqrt{-1}$, but a word of caution is in order here: You can only use i or j for $\sqrt{-1}$ ONLY if you have not previously defined them. If you think you’re going to use complex numbers, do not use the letter i for anything but complex arithmetic! Example: `(0.2+3*i)*(5+2*i)` will multiply the two complex numbers together (using complex arithmetic).

2.1 Helpful Administrative Commands

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

`clear` Clears the workspace of all variables

`clc` Clears the command window

`who` List all variables currently in use.

`whos` List all variables, and their sizes.

`ls` or `dir` List the contents of the current directory.

`help command` List the help file for the function *command*. For example, to get help on the sine function, type `help sin`.

`demo` Lists all the demonstration programs that Matlab came with- This is fun to look at. We don't have all of them; you can go to Matlab's website to look at more: www.mathworks.com.

3 In Class Exercises

1. Type in the following (in the Matlab command window) and write down your results:

- (a) `cos(5*pi/4)` (Note: The constant π is smallcase `pi`)
- (b) The following are examples of a "for-loop". It is how we tell a program to repeat a set of commands for a given set of iterations.

```
for k=1:6
    cos(k*pi/2)
end
```

```
for k=1:6
    if k<3
        fprintf('The number is less than 3\n');
    else
        k
    end
end
```

2. The following **script file** is an example of Newton's method applied to a function $f(x) = xe^x - \cos(x)$. Recall that Newton's Method solves for x : $f(x) = 0$ by taking an initial guess, x_0 , and refines the guess by:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

A script file is typed out just as you would if you were doing it live. Type the following using Matlab's editor. Once you are finished, save it as `newton1.m`

```
x=0.2; %Initial guess for solution to f(x)=0
for k=1:5
    y=x*exp(x)-cos(x);
    dy=(x+1)*exp(x)+sin(x);
    x=x-(y/dy)
end
```

Notes about the code:

- Note the use of `%` to make comments.

- Note that `x=x-(y/dy)` does NOT have a semicolon at the end.
- (a) Run the code after you've saved it by typing `newton1` in the command window.
 - (b) Write down Matlab's output.
 - (c) To see more significant digits, type `format long`
 - (d) Type `whos` and write down Matlab's answer. If you're continuing to the next exercise, type `clear` to clear Matlab's memory.
3. We will now change our script file into a function. Recall that *functions* have inputs and outputs. In Matlab, the important thing about a function is that the following MUST appear as the first line (type the following into the editor, and save the result as `newton2.m`:

```
function y=newton2(initialx, numits)

x=initialx;
for k=1:numits
    y=x*exp(x)-cos(x);
    dy=(x+1)*exp(x)+sin(x);
    x=x-(y/dy)
end
```

To run this as a function, in the Matlab command window, type: `y=newton2(0.2,5)`

Note that the first value input is the initial guess, and the second value is the number of iterations.

SIDE REMARK: It is also possible to input the function and the derivative. We will do these later.

4 Arrays of Data

For those of you that have had linear algebra, Matlab was originally designed as a “front end” to access LINPACK and EISPACK, which are numerical linear algebra packages written in FORTRAN. From this beginning, Matlab's basic data type is the matrix.

If you haven't had linear algebra, simply think of a matrix as an array of numbers- like on a spreadsheet. We'll use arrays to store numbers.

Arrays that have m rows and n columns are said to have size $m \times n$ (remember, rows come first!).

For example, suppose I want to enter the following 2×4 array (or matrix) in Matlab:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

I would type:

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

or as:

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

Note the use of the semicolon: Inside a matrix, the semicolon indicates the end of a row. Outside the matrix, the semicolon suppresses Matlab output. You can also separate numbers using a comma if you'd prefer that. Rows and columns are entered in a corresponding way, as either a $1 \times n$ matrix or as a $n \times 1$ matrix.

We access elements of the matrix in a natural way. For example, the $(2,3)$ element A is written as $A(2,3)$ in Matlab.

Special Commands: The colon operator

- **a:b**

Produces the integers from a to b in a row. For example, $x = 2 : 9$ puts x as a row vector whose elements are the integers from 2 to 9.

- **a:b:c**

Produces the numbers from a to c by adding b each time. For example, $1 : 2 : 7$ returns the numbers 1, 3, 5, 7. Type the following into Matlab to see what you get: $1:2:8$ and $1:0.5:6$

- **linspace(a,b,c)**

Produces c numbers evenly spaced from the number a to the number b (inclusive). For example, $x=\text{linspace}(2,3.5,40)$ 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.)

4.1 Matlab commands associated with Arrays

- **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. **A=randn(m,n)** produces an $m \times n$ array of random numbers (with a normal distribution) with zero mean and unit variance.

- **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.

- **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

Matrix Arithmetic

- Transposition is denoted by the single quote character '. That is, $A' = A^T$. (CAUTION: If A contains complex numbers, then A' is the *conjugate transpose* of A , sometimes denoted as $A^* = \overline{A^T}$)
- Matrix addition and subtraction is performed automatically and is only defined for matrices of the same size.
- Scalar addition. If we want to add a constant c to every item in an array A , type: $A+c$
- Scalar Multiplication: We can multiply every number in the array by a constant: If A is the array and c is the constant, we would write: $B=c*A$
- Matrix Multiplication: Use the regular multiplication sign for standard matrix multiplication. If A is $m \times n$ and B is $n \times p$, then $A*B$ is an $m \times p$ matrix, as we did in linear algebra.
- Elementwise Multiplication. We can multiply and divide the elements of an array A and an array B *elementwise* by $A.*B$ and $A./B$
Exponentiation is done in a similar way. To square every element of an array A , we would write: $A.^2$ This is the same as saying $A.*A$
- Functions applied to arrays: Matlab will automatically apply a given function to each element of the array. For example, $\sin(A)$ will apply the sine function to each element of the array A , and $\exp(A)$ will apply e^x to each element of the array. If you write your own functions, you should always decide ahead of time how you want the function to operate on a matrix.

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 i th row
$A(:,j)$	The entire j th 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:
<code>A(1,:)=[];</code>	Delete the first row.
<code>A([1,3],:)=[];</code>	Delete rows 1 and 3.
<code>A(:,3)=[];</code>	Delete column 3.
<code>A(:,1:2:5)=[];</code>	Delete the odd columns.
<code>A(1,:)=b;</code>	Put b as row 1.
<code>A(:,6)=c;</code>	Add c as the last column.
<code>d=[c , A(:,1:3)];</code>	d is c and columns 1 – 3 of A .
<code>A=[A(:,1) , c , A(:,2:5)];</code>	Insert c as column 2 of A , others shift 1 over.
<code>A=[A(1,:); b; A(2:4,:)];</code>	Insert b as row 2 of A , others shifted 1 down.

4.2 In Class Exercises

1. What is the Matlab command to create the array x which holds the integers: 2, 5, 8, 11, ... 89
2. (Referring to the array above) What would the Matlab command be that zeros out the even-numbered indices (That is, $x(2), x(4), x(6), \dots$)?
3. What is the difference in Matlab between typing: $x=[1 \ 2 \ 3]$ and $x=[1,2,3]$ and $x=[1;2;3]$? What happens if you type a semicolon at the end of the commands (i.e., $x=[1 \ 2 \ 3];$)?
4. (Referring to the last question) For each of those, what happens if you type $x.^2+3$? What happens if you forget the period (e.g., x^2+3)
5. What do the following commands do: $x=2:3:6;$, $x=2:3:6;$, $a=\pi:\pi:8*\pi;$
6. Describe the output for each of the following Matlab commands. Recall that typing a semicolon at the end of the line suppresses Matlab output- to see the results, leave off the semicolon.

```
A=rand(3,4);
A([1,2],3)=zeros(2,1);
B=sin(A);
C=B+6;
D=2*B';
E=A./2;
F=sum(A.*A);
```

4.3 Magic Squares

Go into Matlab's "Help" (Go to "Help", select "Matlab Help"). In "Getting Started", go to "Matrices and Arrays", then "Matrices and Magic Squares". Look over the demonstrations- Feel free to type in the commands as you go along.

4.4 An Image Example

Type in the following commands to load the image of a clown and see the image:

```
load clown
whos %NOTE: X and map are new variables now
image(X)
colormap(map)
colorbar
```

The matrix X is just a 200×320 array of integers. Each integer represents a different color. You can see what colors we're using by using the `colorbar` command.

Exercise: Remove every other column and every other row from the matrix X and display the result- Do you still see a clown?

5 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.

5.1 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 y_1 , and also plots the symbol $*$ where the data points are.

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

Plots the function y_1 using a black (k) line with the asterisk at each data point, PLUS plots the function y_2 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!
6. In our last exercise, we'll look at a sound file. (Before you begin, you might clear everything by using `clear` and `clc`. If you have multiple graphs open, you can close them all by using `close all`)

```
load handel %Loads in variables y and Fs
plot(y) %This is the sound file
soundsc(y) %Play the sound sample
y1=y; %Put the sound sample into variable y1
load laughter
```

“Laughter” is loaded into variable y now (“Handel” was put into $y1$). Try to add the sounds together and listen to the result. NOTE: Addition is only defined for two arrays of the same size.

Final note: Feel free to explore Matlab's `help` features (use the “help” button at the top of the window). There are many cool demos!