A First Look at Matlab

At the end of this session, you should be able to:

- Define matrices and vectors as variables.
- Do some basic operations on matrices and vectors in the command window.
- Write a simple function with multiple inputs and multiple outputs. Be able to call the function from the command window.
- Produce a simple plot.
- Write a simple script file and run it from the command window.

Let's get started- I'll assume that you have already put the Matlab icon on your toolbar, and that you have a folder called `Modeling` and a subfolder called `Lab01` (no white space!).

A Quick Note if You Know Maple

Matlab (short for Matrix-Laboratory) was originally designed as a front end for numerical linear algebra routines (numerically finding eigenvalues and eigenvectors, etc.). Now it is an industry standard for fast development of numerical solutions to mathematics.

Matlab is different than Maple: Maple was designed to work symbolically (e.g., factor a polynomial, compute derivatives). If you need to do symbolic work, you should use Maple. If you need to do numerical work, use Matlab.

If you have had a class in computer programming, then Matlab will look very familiar. There are three ways of running Matlab, and we will consider each:

- Run Matlab "live", by typing commands in the command window.
- Run Matlab by first writing a script file with Matlab commands. A script file is just a text file with the Matlab commands typed in (so you don't have to type them "live".
- Run Matlab by calling programs that you have written. We'll see how to do this in just a moment.

Starting the Matlab Program

Matlab starts by selecting the Matlab icon we've already put on our toolbar. You should see a window with several subwindows, similar (but probably not the same) to the one shown in Figure 1.

Working in the Command Window

Here are some sample tasks together with the Matlab commands. Type them into the command window to verify your answer.

- Compute the value of $e^{-1.1} + \sin(4\pi)$

  SOLUTION: The exponential function is defined as: e^x =`exp(x)`, and the value of π is a built in value, `pi` (must be lowercase P). An asterisk is used to define multiplication:

  ```
  exp(-1.1)+sin(4*pi)
  ```

- Multiply the complex numbers together: $(1 - i)(4 + 2i)$

  SOLUTION: If you have not already used the letter i for something in an earlier line, then the default is to assume $i = \sqrt{-1}$ (same for the letter j- but that is for engineers, not mathematicians!). You should get $6 - 2i$ if you type:

  ```
  (1-i)*(4+2*i)
  ```

- Make a 3×5 matrix with some numbers in it. Assign the matrix to the variable `A`:

  SOLUTION: I'm just making up some numbers- Pay attention to how the matrix is defined:

  ```
  A=[1 2 3 4 5;5 4 3 2 1;0 0 0 1 1];
  ```

  NOTE: Matlab treats the equal sign as "assignment" (the thing on the right is assigned to the variable name on the left).

  For example, if we want to transpose our matrix and assign the result to the variable B, we would just type:

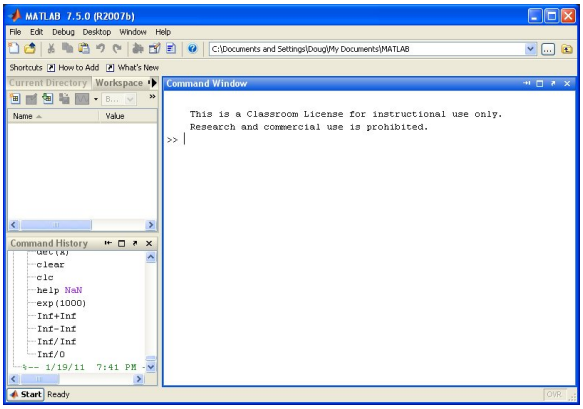

Figure 1: Matlab's main window. Take a look around, and find the directory bar and the command window.

```
B=A';
```

And now the variable B holds A^T.

NOTE: What happens if you don't put a semi-colon at the end of the line? (Answer: Matlab will print the result on the screen).

- What will the following commands do?

```
x=5;
x=x+3;
```

Answer: First, the value 5 is assigned to the variable x. Next, $x+3$ is computed as 8, and finally, the value of 8 is assigned to the variable x.

What happens if you type `5=x`? (try it!)

- Enter the following matrix and compute $B = A^T A$, C is the sine of each element of A.

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

SOLUTION:

```
A=[1 2 3 4; 5 6 7 8];
B=A'*A;
C=sin(A);
```

- Continuing with the example, let B be the matrix defined below. Compute the *element-wise* product of A and B:

```
B=[0 1 0 1;1 0 1 0];
C=A.*B
```

NOTE: The "dot" operator will be used if the operation is performed elementwise (as opposed to the operation as it is defined on a matrix). For example, if $x = [1, 2, 3, 4, 5]$, if we want to compute $[2^1, 2^2, 2^3, 2^4, 2^5]$, and $[1^2, , 2^2, 3^2, 4^2, 5^2]$, we would type:

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

- Notice what the following commands do:

```
A=[1 2 3 4;-1 2 1 0];
A(2,:)
A(:,3)
A(2,4)
```

Note: The colon is used to denote all rows (or all columns).

Creating arrays and vectors

- `a:b`

  Gives a row created by $a, a+1, a+2, \ldots, a+n$ where n is the largest integer so that $a+n \leq b$

  Example: `3.5:7`

- `a:b:c`

  Produces the numbers from a to c by adding b each time:

  $$\texttt{a:b:c} \quad \text{gives} \qquad [a, a+b, a+2b, \ldots, a+nb]$$

  where n is the largest integer so that $a+nb \leq c$. For example, $3 : 0.2 : 4.7$ returns what?

- `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.

- `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
```

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:

```
X=load clown.mat
whos
X(4:6, 6:10) %Look at some of the values in the array X.
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
```

If you want to re-do the clown again, you do not need to retype it! Use the up-arrow key to bring back the commands you typed. You can also type the first few characters, then use the up-arrow key:

```
X=(up arrow)
```

Solving $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$

To solve $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$, Matlab has two basic commands: `x=A\b` or `x=pinv(A)*b`. The command `pinv(A)` computes the *pseudoinverse* of A, which we will discuss later in the section dealing with the Singular Value Decomposition.

In linear algebra, there were three possible outcomes for solving $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$. They were:

1. A unique solution.
2. An infinite number of solutions.
3. No solution.

Matlab will always give exactly one solution. We need to interpret that solution in the second two cases. In the case of an infinite number of solutions (we have free variables in this case, also called *an underdetermined system*), the two methods may give different answers:

- `x=A\b` provides the most zeros in $\mathbf{x}$.
- `x=pinv(A)*b` gives $\mathbf{x}$ with the smallest norm.

Example: Let $A = \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 1 \end{bmatrix}$, with $b = \begin{bmatrix} 9 \\ 3 \end{bmatrix}$. Then the full solution is:

$$\mathbf{x} = \begin{bmatrix} 9 + 2t \\ 3 - t \\ t \end{bmatrix} \tag{1}$$

In Matlab, the result of typing `x=A\b` is $x = [0, -15/2, -9/2]^T$ and the result of typing `x=pinv(A)*b` is $x = [4, 11/2, -5/2]$.

(MATLAB HINT: You can get Matlab to return numbers as fractions by typing `format rat`)

In the case of an overdetermined system (a system with no solution), Matlab will automatically return the *least squares* solution- that is, the answer $\mathbf{x}^*$ will be the minimum of $\|A\mathbf{x} - \mathbf{b}\|$:

$$\|A\mathbf{x}^* - \mathbf{b}\| \leq \|A\mathbf{x} - \mathbf{b}\|, \text{ for all } \mathbf{x}$$

In general, you should always use the forward slash (for help, type `help slash`): `x=A\b` which automatically determines a best numerical method. That is, sometimes its best (numerically) not to explicitly compute the pseudoinverse first.

Exercise Set I

1. What is the Matlab command to create the array x which holds the integers: $2, 5, 8, 11, \ldots 89$
2. (Referring to the array above) What would the Matlab command be that zeros out the even-numbered indices (That is, $x(2), x(4), x(6), \ldots$)?

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=2.3:0.5:3.5;`

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);
```

7. What will Matlab do if you type in:

```
A=rand(3,4);
A(:)
A(7)
```

   NOTE: This is very bad programming style! Don't do it unless you know what you're doing!!

8. What is the Matlab command to perform the following:

   (a) Given an array x, add 3 to each of its values.

   (b) Given an array A, remove its first column and assign the result to a new array B.

9. What will the following code fragment do?

```
a=1:10;
for k=1:10
  h=ceil(length(a)*rand);
  b(k)=a(h);
  a(h)=[];
end
```

   Compare this with `a=ceil(10*rand(10,1))` and `a=randperm(10)`

10. Use the Quick Summary sheet to help you write a code fragment that takes a random matrix X and re-sorts the columns so that the first column has the smallest size and the last column has the greatest size.

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.

   Type `doc plot` to see all the different built in colors and symbols you can use in a plot.

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 function has a given input (from the domain), and produces something (from the range). Some functions are built-in (like sine, etc.). To extend the usefulness of Matlab, we can write our own functions. Here is a simple example that takes in two things- a constant and an array, and outputs three things- The sum of the array and the constant, the product and the difference (not a very useful function).

Open Matlab's editor (type `edit` in the command window), and type the following:

```
function [A,B,C]=myfunc(x,Z)
% [A,B,C]=myfunc(x,Z)
%  Inputs:  x is a constant, Z is an array
%  Outputs:  A=x+Z, B=x*Z and C=Z-x

A=x+Z; B=x*Z; C=Z-x;
temp=A+B+C
```

Save this file as the function name with a `.m.` suffix, or, `myfunc.m`. The line with `temp` is meaningless, but is there to show you something. But first, some things to notice about a function:

The first line should always begin with the word "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 myfunc
A=rand(3,2);
c=5;
[H1, H2, H3]=myfunc(c,A)
whos
```

Notice that the array `temp`, while computed during the execution of the function, is gone once the function is finished. Therefore, inside of a function call, you can write your variables as if nothing has been defined (except your input variables).

A **script file** is a file that includes Matlab commands. When Matlab executes a script file, it simply executes the commands that are written there as if you were typing them in.

Here's an example. Open the editor, type these lines in and save as `myscript.m`. It looks very similar to the function-

```
clear
clc
A=rand(3,2);
c=5;
H1=c+A;
H2=c*A;
H3=A-c;
temp=H1+H2+H3
```

To run the script, in the command window type `myscript`, then you can type `whos` to see what the new variables are.

Exercise Set II

1. Let x be a row. What happens if you type `plot(x)`?
2. Write a Matlab script file to plot $y = \sin(x)$ in red, $y = \sin(2x)$ in black, and $y = \sin(3x)$ in green, all on the same plot. You can assume that $x \in [-4, 8]$.
3. When we compute with numbers, some errors can occur. Try typing each of the following into Matlab, and see what happens:
   - `eps` (This is machine epsilon)
   - `1/0` (Think about what this means before trying it!)
   - `-1/0`
   - `0/0`
   - `1/Inf`
   - `Inf+Inf, Inf-Inf, Inf/Inf, Inf/0`
4. Try to reason out what you think Matlab will do with each of the following, then type it in and record what you get:

```
x=[1 3 2 1 3];
max(x)
[vals, idx]=sort(x)
find(x==max(x))
mean(x)
sum(x)
```

5. Write a Matlab function that will take in two matrices A, B of the same dimensions, and will output a matrix C so that

$$C(i,j) = \max\{A(i,j), B(i,j)\}$$

The function will use the built-in `max` function- See Matlab's help file by typing (in the command window) `doc max`

6. What does this function do (think about what each line does individually; use the help features in Matlab)? The input B is an $m \times n$ matrix.

```
function A=mystery(B)

[m,n]=size(B);
Temp=sqrt(sum(B.*B));
A=B./repmat(Temp,m,1);
```

(Hint: It is often very useful to have a matrix where each column has unit length).