Introduction to MATLAB

MATLAB

- MATLAB is a program for doing numerical computation. It was originally designed for solving linear algebra type problems using matrices.
- It's name is derived from MATrix LABoratory.
- MATLAB has since been expanded and now has built-in functions for solving problems requiring data analysis, signal processing, optimization, and several other types of scientific computations. It also contains functions for 2-D and 3-D graphics and animation.

MATLAB

- The MATLAB environment is command oriented somewhat like UNIX. A prompt appears on the screen and a MATLAB statement can be entered. When the <ENTER> key is pressed, the statement is executed, and another prompt appears.
- If a statement is terminated with a semicolon (;), no results will be displayed. Otherwise results will appear before the next prompt.
- The following slide is the text from a MATLAB screen.

MATLAB

To get started, type one of these commands: helpwin, helpdesk, or demo

```
» a=5;
» b=a/2

b =

    2.5000

»
```

MATLAB Variable Names

- Variable names ARE case sensitive
- Variable names can contain up to 63 characters (as of MATLAB 6.5 and newer)
- Variable names must start with a letter followed by letters, digits, and underscores.

MATLAB Special Variables

ans	Default variable name for results
pi	Value of π
eps	Smallest incremental number
inf	Infinity
NaN	Not a number e.g. 0/0
i and j	i = j = square root of -1
realmin	The smallest usable positive real number
realmax	The largest usable positive real number

MATLAB Math & Assignment Operators

Power	^	or	.^	a^b	or a.^b
Multiplication	*	or	.*	a*b	or a.*b
Division	/	or	./	a/b	or a./b
or	\	or	.\	b\a	or b.\a

NOTE: `56/8 = 8\56`

– (unary) + (unary)			
Addition	+	a + b	
Subtraction	–	a – b	
Assignment	=	a = b	(assign `b` to `a`)

Other MATLAB symbols

>>	prompt
. . .	continue statement on next line
,	separate statements and data
%	start comment which ends at end of line
;	(1) suppress output
	(2) used as a row separator in a matrix
:	specify range

MATLAB Matrices

- MATLAB treats all variables as matrices. For our purposes a matrix can be thought of as an array, in fact, that is how it is stored.
- Vectors are special forms of matrices and contain only one row OR one column.
- Scalars are matrices with only one row AND one column

MATLAB Matrices

- A matrix with only one row AND one column is a scalar. A scalar can be created in MATLAB as follows:

```
» a_value=23

a_value =

    23
```

MATLAB Matrices

- A matrix with only one row is called a row vector. A row vector can be created in MATLAB as follows (note the commas):

```
» rowvec = [12 ,  14 , 63]

rowvec =

    12    14    63
```

MATLAB Matrices

- A matrix with only one column is called a column vector. A column vector can be created in MATLAB as follows (note the semicolons):

```
» colvec = [13 ; 45 ; -2]

colvec =

    13
    45
    -2
```

MATLAB Matrices

- A matrix can be created in MATLAB as follows (note the commas AND semicolons):

```
» matrix = [1 , 2 , 3 ; 4 , 5  ,6 ; 7 , 8 , 9]

matrix =

     1     2     3
     4     5     6
     7     8     9
```

Extracting a Sub-Matrix

- A portion of a matrix can be extracted and stored in a smaller matrix by specifying the names of both matrices and the rows and columns to extract. The syntax is:

```
sub_matrix = matrix ( r1 : r2 , c1 : c2 ) ;
```

where r1 and r2 specify the beginning and ending rows and c1 and c2 specify the beginning and ending columns to be extracted to make the new matrix.

MATLAB Matrices

- A column vector can be extracted from a matrix. As an example we create a matrix below:

```
» matrix=[1,2,3;4,5,6;7,8,9]

matrix =
     1     2     3
     4     5     6
     7     8     9
```

- Here we extract column 2 of the matrix and make a column vector:

```
» col_two=matrix( : , 2)

col_two =

     2
     5
     8
```

MATLAB Matrices

- A row vector can be extracted from a matrix. As an example we create a matrix below:

```
» matrix=[1,2,3;4,5,6;7,8,9]

matrix =

     1     2     3
     4     5     6
     7     8     9
```

- Here we extract row 2 of the matrix and make a row vector. Note that the 2:2 specifies the second row and the 1:3 specifies which columns of the row.

```
» rowvec=matrix(2 : 2 , 1 : 3)

rowvec =

     4     5     6
```

Reading Data from files

- MATLAB supports reading an entire file and creating a matrix of the data with one statement.

```
>> load mydata.dat;                % loads file into matrix.
% The matrix may be a scalar, a vector, or a
%  matrix with multiple rows and columns.  The
%  matrix will be named mydata.
>> size (mydata)                   % size will return the number
                                   % of rows and number of
                                   % columns in the matrix
>> length (myvector)               % length will return the total
                                   % no. of elements in myvector
```

Plotting with MATLAB

- MATLAB will plot one vector vs. another. The first one will be treated as the abscissa (or x) vector and the second as the ordinate (or y) vector. The vectors have to be the same length.
- MATLAB will also plot a vector vs. its own index. The index will be treated as the abscissa vector. Given a vector "time" and a vector "dist" we could say:

```
>> plot (time, dist)      % plotting versus time
>> plot (dist)            % plotting versus index
```

Plotting with MATLAB

- There are commands in MATLAB to "annotate" a plot to put on axis labels, titles, and legends. For example:

```
>> % To put a label on the axes we would use:
>> xlabel ('X-axis label')
>> ylabel ('Y-axis label')

>> % To put a title on the plot, we would use:
>> title ('Title of my plot')
```

Plotting with MATLAB

- Vectors may be extracted from matrices. Normally, we wish to plot one column vs. another. If we have a matrix "mydata" with two columns, we can obtain the columns as a vectors with the assignments as follows:

```
>> first_vector = mydata ( : , 1) ;          % First column
>> second_vector = mydata ( : , 2) ;         % Second one
>> % and we can plot the data
>> plot ( first_vector , second_vector )
```

Some Useful MATLAB commands

- who List known variables
- whos List known variables plus their size
- help >> help sqrt Help on using sqrt
- lookfor >> lookfor sqrt Search for keyword sqrt in m-files
- what >> what a: List MATLAB files in a:
- clear Clear all variables from work space
- clear x y Clear variables x and y from work space
- clc Clear the command window

Some Useful MATLAB commands

what	List all m-files in current directory
dir	List all files in current directory
ls	Same as dir
type test	Display test.m in command window
delete test	Delete test.m
cd a:	Change directory to a:
chdir a:	Same as cd
pwd	Show current directory
which test	Display directory path to ‘closest’ test.m

A Useless, But Interesting, MATLAB command

- why In case you ever needed a reason

MATLAB Relational Operators

- MATLAB supports six relational operators.

Less Than	<
Less Than or Equal	<=
Greater Than	>
Greater Than or Equal	>=
Equal To	==
Not Equal To	~=

MATLAB Logical Operators

- MATLAB supports three logical operators.

```
not     ~     % highest precedence
and     &     % equal precedence with or
or      |     % equal precedence with and
```

MATLAB Logical Functions

- MATLAB also supports some logical functions.

xor (exclusive or) Ex: xor (a, b)

Where a and b are logical expressions. The xor operator evaluates to true if and only if one expression is true and the other is false. True is returned as 1, false as 0.

any (x)	returns 1 if any element of x is nonzero
all (x)	returns 1 if all elements of x are nonzero
isnan (x)	returns 1 at each NaN in x
isinf (x)	returns 1 at each infinity in x
finite (x)	returns 1 at each finite value in x

Matlab Selection Structures

- An if - elseif - else structure in MATLAB.
  Note that elseif is one word.

```
if      expression1                % is true
        % execute these commands
elseif expression2      % is true
        % execute these commands
else                               % the default
        % execute these commands
end
```

MATLAB Repetition Structures

- A for loop in MATLAB for x = array

```
for ind = 1:100
    b(ind)=sin(ind/10)
end
```

Alternative:

x=0.1:0.1:10; b=sin(x); - Most of the loops can be avoided!!!

- A while loop in MATLAB while expression

```
while x <= 10
    % execute these commands
end
```

Scalar - Matrix Addition

```
» a=3;
» b=[1, 2, 3;4, 5, 6]
b =
     1     2     3
     4     5     6
» c= b+a            % Add a to each element  of b
c =
     4     5     6
     7     8     9
```

Scalar - Matrix Subtraction

```
» a=3;
» b=[1, 2, 3;4, 5, 6]
b =
     1     2     3
     4     5     6
» c = b - a %Subtract a from each element of b
c =
    -2    -1     0
     1     2     3
```

Scalar - Matrix Multiplication

```
» a=3;
» b=[1, 2, 3; 4, 5, 6]
b =
     1     2     3
     4     5     6
» c = a * b  % Multiply each element of b by a
c =
     3     6     9
    12    15    18
```

Scalar - Matrix Division

```
» a=3;
» b=[1, 2, 3; 4, 5, 6]
b =
     1     2     3
     4     5     6
» c = b / a      % Divide each element of b by a
c =
    0.3333    0.6667    1.0000
    1.3333    1.6667    2.0000
```