MATLAB(MATrix LABoratory) - MathWorks, Inc.
an interactive, matrix-based system for scientific and enginnering computation

Invoking MATLAB

```
% matlab
```

to get out, type "exit"

```
>> exit
```

On-line Help

```
>> help
```

Objects in MATLAB
Matrices with integer, real, or complex entries

- scalar – 1×1 matrix
- vector – $m \times 1$ matrix or $1 \times m$ matrix
- scalar – square($n \times n$) or rectangular($m \times n$) ($m, n \geq 1$)

Entering matrices

- entered by an explicit list of elements
- generated by built-in statements and functions
- created in M-files
- loaded from external data files

  ```
  >> A = [1  2  3 ; 4  5  6 ; 7  8  9];
  ```

  creates a 3×3 matrix A

- $rand(n)$ creates a random $n \times n$ matrix
- $rand(m, n)$ creates a random $m \times n$ matrix Any line can be continued onto the next by using "..."

  ```
  >> A = [1  2  3 ; 4  5  6 ;...
          7  8  9];
  ```

  is the same as

  ```
  >> A = [1  2  3 ; 4  5  6 ; 7  8  9];
  ```

Expressions

Expression typed by the users are interpreted and evaluated by MATLAB

$$variable = expression$$

or

$$expression$$

Variable names begin with a letter, are up to 19 characters long, and are case sensitive. Expressions are composed of operators, special characters, functions, variable names. Evaluation of expression produces a matrix, which is displayed on the screen and assigned to a variable for future use. If "variable=" is omitted, a variable "ans" is created.

```
>> 1900/81
   ans = 23.4568
```

A statement is terminated with the carriage return. If the last character of a statement is ";", the printing is suppressed.

```
>> x = 1+2+3;
>> x = 1+2+3

 x = 6
```

"who" lists variables in the workspace

```
>> who

ans x
```

Arithmetic Expressions

+	addition	-	subtraction
*	multiplication	/	right division
\	left division	^	power
'	transpose	( )	parentheses

Hardcopy

```
>> diary filename
>> ...
>> diary off
```

Everything that appeared on the screen after 'diary filename' until 'diary off' is written into the file named in *filename*.

Graphics

Planar plots

'plot' creates $x-y$ plots. If x and y are vectors of the same length, *"plot(x,y)"* opens a graphics window and draws an $x-y$ plot of the elements of x versus the elements of y.

```
ex1. graph of y=sin(x) over [-4,4]
>> x = -4:.001:4; y=sin(x); plot(x,y)
ex2. graph of y=exp(-x^2) over [-1.5,1.5]
>> x = -1.5:.01:1.5; y=exp(-x.^2); plot(x,y)
```

Overlap of two graphs
plot two functions $sin(x)$ and `x^2/100` in the interval $[-15, 15]$

```
>> x = -15:.05:15;
>> y = sin(x);
>> z = x.^2/100;
>> plot (x,y,x,z)
```

title - adds a title to the graph
grid - turns on the grid lines
xlabel - adds a label to x-axis
ylabel - adds a label to y-axis

```
>> title ('survey')
>> xlabel ('year')
>> ylabel ('income')
```

Creating a hardcopy of MATLAB figures(graphs)
You can generate a PostScript file of the contents of any MATLAB figure window using the 'print' command. Then you can print this file on a PostScript printer to get a hardcopy.

```
>> print -dps filename.ps
```

and then at the UNIX prompt after exiting MATLAB

```
% lpr -Pprintername filename.ps
```

where printname is the name of a PostScript printer and filename is the name of the file - you must have a ".ps" extension on the file.

3-D mesh plots
'mesh(z)' creates a three-dimensional perspective plot of the elements of the matrix z. The mesh surface is defined by the z-coordinates of points above a rectangle rid in the $x - y$ plane. To draw a graph of $z = f(x, y)$:

1. define vectors xx and yy which give partitions of the sides of a rectangle
2. `[x,y] = meshgrid(xx,yy)`
3. compute z, and apply mesh

```
ex. z=exp(-x^2 - y^2) over [-2,2] x [-2,2]
>> xx=-2:.1:2;
>> yy=xx;
>> z=exp(-x.^2 - y.^2)
>> [x,y]=meshgrid(xx,yy);
>> mesh(z)
```

Relations and Logical Operators

<table>
<tr><td><</td><td>less than</td><td>></td><td>greater than</td></tr>
<tr><td><=</td><td>.LE.</td><td>>=</td><td>.GE.</td></tr>
<tr><td>==</td><td>equal</td><td>~=</td><td>not equal</td></tr>
<tr><td>\</td><td>end</td><td>|</td><td>or</td></tr>
<tr><td>~</td><td>not</td><td></td><td></td></tr>
</table>

FOR

```
>> for i = 1:n, x(i)=i^2, end
```

or

```
>> for i=1:n
x(i)=i^2
end ;
```

WHILE

```
>> n = 0;
>> while 2^n < a
n = n+1;
end;
>> n
```

IF

```
>> if n < 0
x =  0;
elseif n == 0
x = 1;
else
x = 2;
end
```

Polynomials

MATLAB represents polynomials as row vectors containing the coefficients ordered by descending powers.

```
ex. x^3 - 6x^2 + 11x - 6
>> p = [1  -6  11  -6];
>> r = roots(p)
 r = 3.0000
     2.0000
     1.0000
```

Data Analysis

generate a $1 \times n$ random vector x

```
>> x=rand(1,n)
```

sort the list x in increasing order

```
>> s=sort(x);
```

find the maximum value in the list x

```
>> max_value = max(x);
```

Find the minimum value in the list x

```
>> min_value = min(x);
```

sum all the elements in x

```
>> sum_all = sum(x);
```

compute the mean of the list x

```
>> mean_value = mean(x);
```

compute the median of the list x

```
>> median_value = median(x);
```

compute the standard deviation

```
>> std_dev = std(x);
```

M-files
MATLAB can execute a sequence of statements stored in files, called "M-files" because they have a ".m" extension. There are two types of M-files:

1. Script files - a script file consists of a sequence of MATLAB statements. If the file name is *file1.m*, then the MATLAB command

   ```
   >> file1
   ```

   will execute the statements in *file1.m*

2. Function files - you can create new functions. Variables in a function file are local.

   ```
   ex. in file "stat.m"

   function [mean, stdev] = stat(x)
   % For a vector x, stat(x) returns the mean and stdev of x
   [m,n] = size(x);
   mean = sum(x)/m; % compute the mean
   stdev = sqrt(sum(x.^2)/m-mean^2); % compute the standard deviation
   ```

   in MATLAB, typing

   ```
   >> [xm,xd]=stat(x)
   ```

   will assign the mean and standard deviation of the entries in the vector x to `xm` and `xd`, a `%` indicates that the rest of the line is a comment.

```
>> help stat
```

For a vector $x, stat(x)$ returns the mean and stdev of x

Entry-wise operation
$*$, ^, /, \$ can be made to operate entry-wise by preceding them by a period.

```
ex. [1 2 3 4].*[1 2 3 4] = [1 2 3 4].^2=[1 4 9 16]
```

Flops
flops - floating point operations, 1 flop is roughly 1 addtion or 1 multiplication.
flops(0) will reset flops to 0

Text Strings
text strings are surrounded by single quotes

```
ex. s='This is a test'
```

text strings can be displayed with "disp"

```
>> disp('This is a test')
```