

جامعة ديالى
كلية التربية المقداد
قسم الرياضيات

محاضرات مادة (الحاسبات المتقدمة)

المرحلة الثانية (صباحي + مسائي)

إعداد

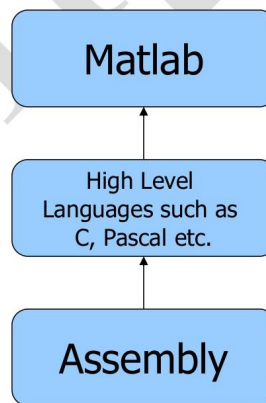
م.م هند ابراهيم محمد

Chapter one

MATLAB

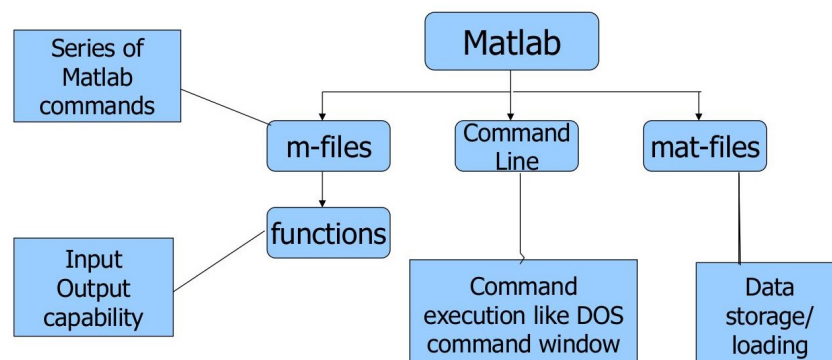
MATLAB Introduction

- 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 is basically a high level language which has many specialized toolboxes for making things easier for us
- How high?



What are we interested in?

- Matlab is too broad for our purposes in this course.
- The features we are going to require is



Matlab Screen

Command Window

- Type commands

Current Directory

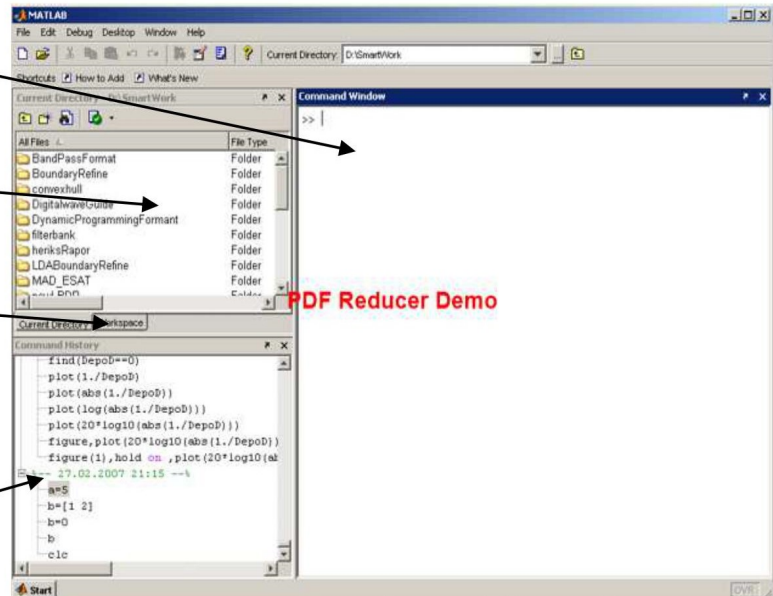
- View folders and m-files

Workspace

- View program variables
- Double click on a variable to see it in the Array Editor

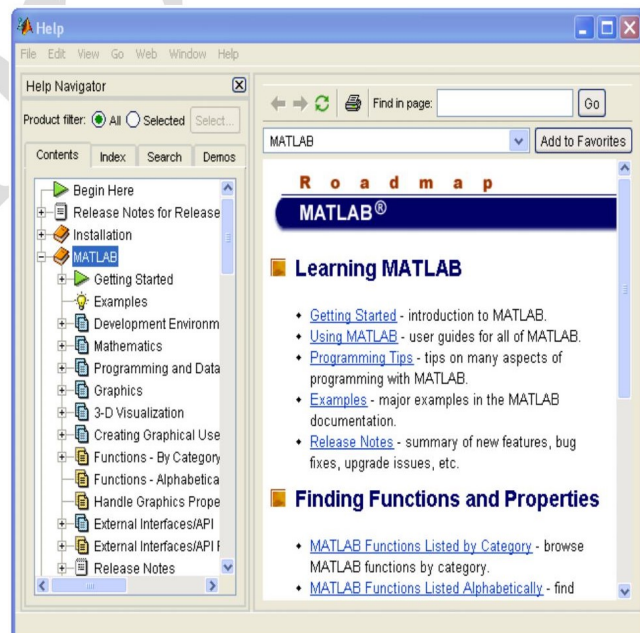
Command History

- View past commands
- Save a whole session using diary

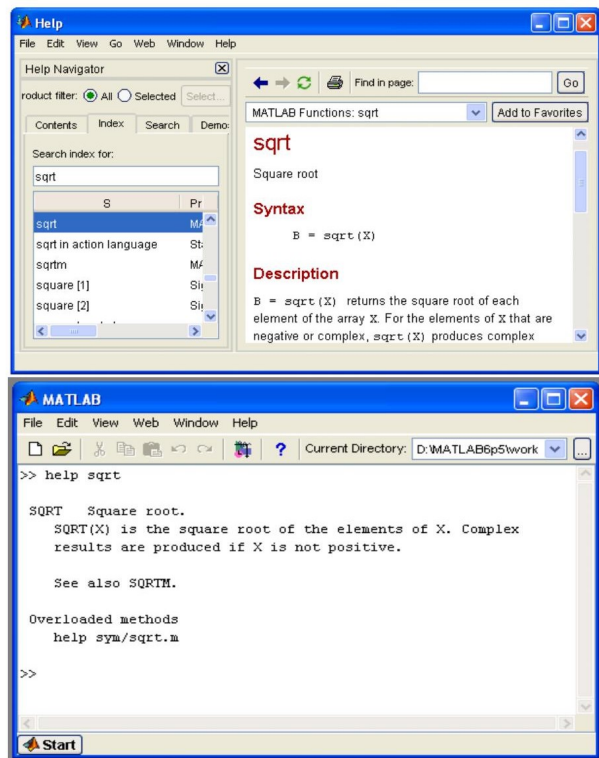


MATLAB Help

- MATLAB Help is an extremely powerful assistance to learning MATLAB
- Help not only contains the theoretical background, but also shows demos for implementation
- MATLAB Help can be opened by using the HELP pull-down menu



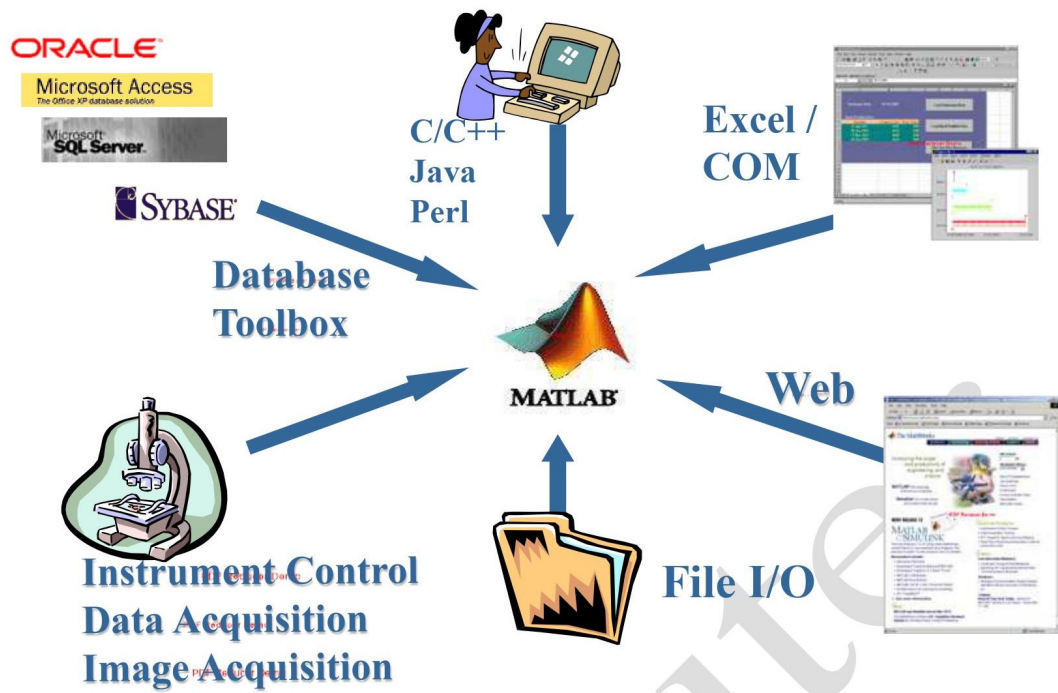
- Any command description can be found by typing the command in the search field
- As shown above, the command to take square root (sqrt) is searched
- We can also utilize MATLAB Help from the command window as shown



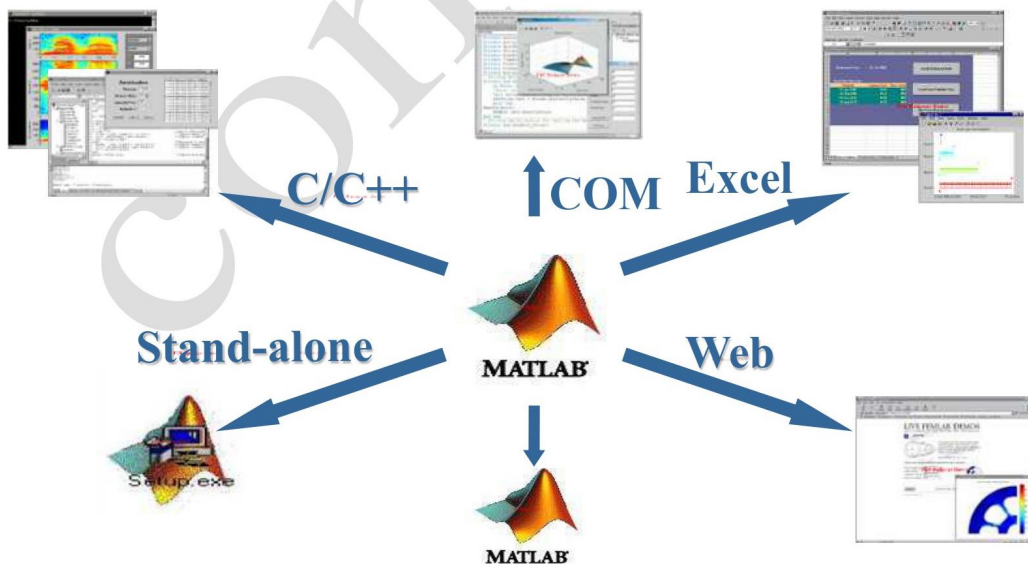
MATLAB Toolboxes

- Statistics Toolbox
- Optimization Toolbox
- Database Toolbox
- Parallel Computing Toolbox
- Image Processing Toolbox
- Bioinformatics Toolbox
- Fuzzy Logic Toolbox
- Neural Network Toolbox
- Data Acquisition Toolbox
- MATLAB Report Generator
- Signal Processing
- Communications
- System Identification
- Wavelet Filter Design
- Control System
- Robust Control

Connecting to MATLAB



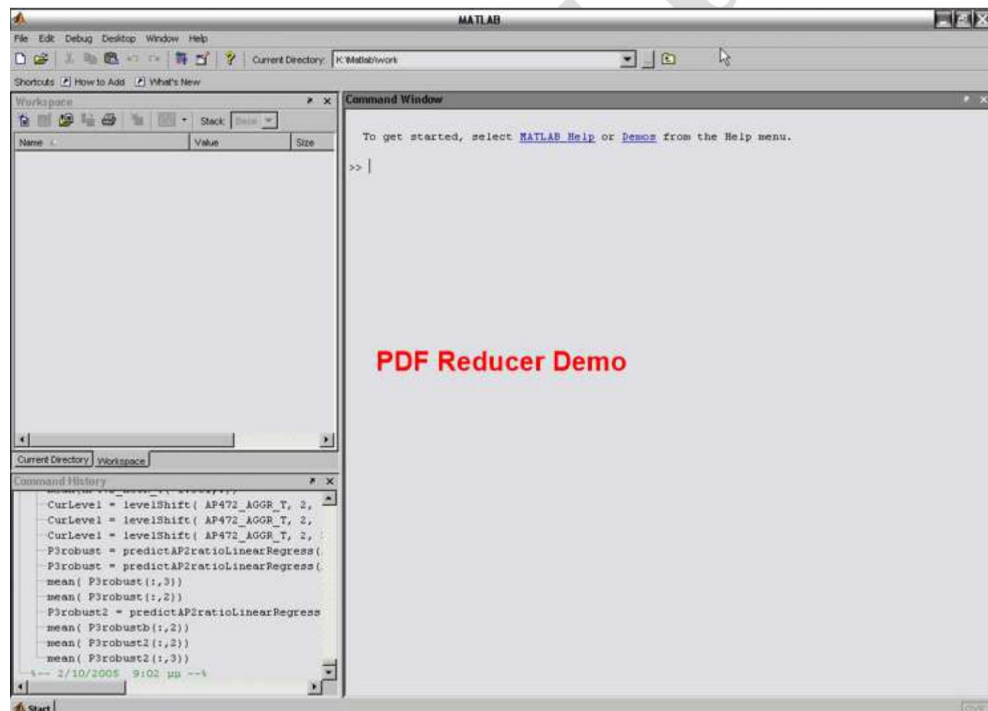
Deploying with MATLAB



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 MATLAB User Interface



More about the Workspace

- who, whos – current variables in the workspace
- save – save workspace variables to *.mat file
- load – load variables from *.mat file
- clear – clear workspace variables

chapter two

Everything in MATLAB is a matrix!

Starting MATLAB

To get started, type one of these commands:

helpwin, helpdesk, or demo

```
» a=5;
```

```
» b=a/2
```

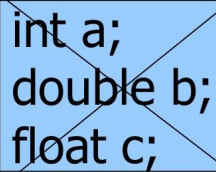
```
b =
```

```
2.5000
```

```
»
```

Variables

- No need for types. i.e.,



```
int a;  
double b;  
float c;
```

- All variables are created with double precision unless specified and they are matrices.

Example:

```
>>x=5;  
>>x1=2;
```

- After these statements, the variables are 1x1 matrices with double precision

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

- | | | |
|----|---------|--|
| 1. | ans | Default variable name for results |
| 2. | pi | Value of π |
| 3. | eps | Smallest incremental number |
| 4. | inf | Infinity |
| 5. | NaN | Not a number e.g. 0/0 |
| 6. | i and j | $i = j = \text{square root of } -1$ |
| 7. | realmin | The smallest usable positive real number |
| 8. | realmax | The largest usable positive real number |

Different format

```
>> e=1/3
```

```
e =
```

```
0.3333
```

```
%default
```

```
>> format long
```

```
>> e
```

```
e =
```

```
0.3333333333333333
```

```
%long decimal
```

```
>> format short e
```

```
>> e
```

```
e =
```

```
3.3333e-001
```

```
%long exponential
```

To clear a variable

» who

Your variables are:

D ans rho

NRe mu v

» clear D

» who

Your variables are:

NRe ans mu rho v

»

Complex Numbers

Complex number i or j stands for $\sqrt{-1}$

» i

ans =

0 + 1.0000i

» c1 = 2+3i

c1 =

2.0000 + 3.0000i

»

Some functions deal with complex number

>> c=1-2i

c = 1.0000 - 2.0000i

>> abs(c)

ans = 2.2361

>> real(c)

ans = 1

```
>> imag(c)
```

```
ans = -2
```

```
>> angle(c)
```

```
ans = -1.1071
```

Mathematical Functions

```
» x=sqrt(2)/2
```

```
x =
```

```
0.7071
```

```
» y=sin(x)
```

```
y =
```

```
0.6496
```

```
»
```

Built-in Functions

Trigonometric functions	sin, cos, tan, sin, acos, atan, sinh, cosh, tanh, asinh, acosh, atanh, csc, sec, cot, acsc, ...
Exponential functions	exp, log, log10, sqrt
Complex functions	abs, angle, imag, real, conj
Rounding and Remainder functions	floor, ceil, round, mod, rem, sign

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

- (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 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

Exercise

Q1/

1) $8+3*5$
ans =23

2) $(8+3)*5$

ans =55

3) $4^2-12-8/4*2$

ans =0

4) $4^2 -12-8/(4*2)$

ans =3

5) $(3*4)^2+5$

ans =149

6) $27^{(1/3)}+32^{(0.2)}$

ans =5

7) $27^{1/3}+32^{0.2}$

ans =11

Q2/ use Matlab to computer the following expenessions:

$$a) \quad 6\left(\frac{10}{13}\right) + \frac{18}{5(7)} + 5(9^2)$$

sol (a)

$$ans = 410.1297$$

$$6 * (10 / 13) + 18 / (5 * 7) + 5 * (9^2)$$

$$ans = 410.1297$$

$$b) \quad 6(35^{1/4}) + 14^{0.35}$$

sol (b)

$$ans = 17.1123$$

$$6 * 35^{(1/4)} + 14^{0.35}$$

$$ans = 17.1123$$

$$c) \quad \frac{1}{2+3^2} + \frac{4}{5} * \frac{6}{7}$$

Sol(c) 1/ (2+3 ^ 2) +

$$ans = 0.7766$$

$$4/5 * 6/7$$

$$d) \quad \frac{1}{2} + 3^2 + \frac{4}{5} * \frac{6}{7}$$

sol (d) 1/2 +3 ^

$$ans = 10.1857$$

$$2 + 4/5 + 6/7$$

e)

sol(e)

$$c = e^{-a} \sin(x) + 10\sqrt{y}$$

$$for, a = 5, x = 2, y = 8$$

$$ans = 28.2904$$

$$a = 5; x = 2; y = 8;$$

$$c = \exp(-a) * \sin(x) + 10 * \text{sqrt}(y)$$

$$ans = 28.2904$$

f) $\log(142)$

sol (f) $\log_{10}(142)$

Ans=4.9558

Q3/ consider the equation:

$$1) f = \frac{\log(ax^2 + bx + c) - \sin(ax^2 + bx + c)}{4\pi x^2 + \cos(x - 2) * (ax^2 + bx + c)}$$

x=9;

a=1;

b=3

c=5;

$$\text{Poly} = a * x^2 + b * x + c;$$

$$\text{denom} = 4 * \pi * x^2 + \cos(x - 2) * \text{poly};$$

$$f = (\log(\text{poly}) - \sin(\text{poly})) / \text{denom}$$

F=0.0044

Poly= 113

Denom = $1.1031 e^{+03}$

$$2) a = \frac{10 + 4x * 9/5 - \sin(b) + \tan(b^2)}{5 + 10b + 2\pi \log_{10} b}$$

Where =5;

So

$$a = (10 + 4 * x * (9/5) - \sin(b) + \tan(b^2)) / (5 + 1 * b + 2 * \pi * \log_{10}(b))$$

Ans=0.3035

$$3) a = \frac{5 + 6 * 7 / 3 - \sin(2^2)}{2 / 3 * 3 / 3 * \cos(0.5)}$$

$$\text{Ans} = 33.7691$$

$$\text{Sol/ } a = (5 + 6 * (7 / 3) - \sin(2 \wedge 2)) / ((2 / 3) * (3 / 3) * \cos(0.5))$$

$$4) a = 2^3 + \frac{4 + \cos(x + y)}{5 + 3x + 2\pi \log_{10}(x + y)}$$

$$\text{Where: } x=2; y=5;$$

Sol/

$$a = 2 \wedge 3 + (4 + \cos(x + y)) / (5 + (3 * x) + 2 * \pi * \log_{10}(x + y))$$

$$\text{Ans} = 9.2915$$

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:

```
» x=23
```

```
x =
```

```
23
```

- 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
```

- 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
```

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

» a = [1 , 2 , 3 ; 4 , 5 , 6 ; 7 , 8 , 9] Or

» a = [1 2 3 ; 4 5 6 ; 7 8 9] Or

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

a =

```
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: <pre>» matrix=[1,2,3;4,5,6;7,8,9]</pre> matrix = <pre>1 2 3 4 5 6 7 8 9</pre>	Here we extract column 2 of the matrix and make a column vector: <pre>» coltwo=matrix(: , 2)</pre> coltwo = <pre>2 5 8</pre>
A row vector can be extracted from a matrix. As an example we create a matrix below: <pre>» matrix=[1,2,3;4,5,6;7,8,9]</pre> matrix = <pre>1 2 3 4 5 6 7 8 9</pre>	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. <pre>» rowvec=matrix(2 : 2 , 1 : 3)</pre> rowvec = <pre>4 5 6</pre>

Special Matrices

$$\text{eye}(3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{ones}(3) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\text{zeros}(3,2) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\text{ones}(2,4) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Special Matrices functions

```
>> a=magic(4)
```

```
%magic matrix
```

```
a =
```

```
16  2  3 13
 5 11 10  8
 9  7  6 12
 4 14 15  1
```

```
>> b=rand(5)
```

```
%random matrix
```

```
b =
```

```
0.8147 0.0975 0.1576 0.1419 0.6557
0.9058 0.2785 0.9706 0.4218 0.0357
0.1270 0.5469 0.9572 0.9157 0.8491
0.9134 0.9575 0.4854 0.7922 0.9340
0.6324 0.9649 0.8003 0.9595 0.6787
```

Some matrix building functions

```
>> a
```

```
a =
```

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

```
>> diag(a)
```

```
ans =
```

```
1
5
9
```

```
>> triu(a)
```

```
ans =
```

```
1  2  3
0  5  6
0  0  9
```

```
>> tril(a)
```

```
ans =
```

```
1  0  0
4  5  0
7  8  9
```

Concatenation of Matrices

- $x = [1 \ 2]$, $y = [4 \ 5]$

$$A = [x \ y]$$

1 2 4 5

$$B = [x ; y]$$

1 2

4 5

Matrices Operations

Given A and B:

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

A =

1	2	3
4	5	6
7	8	9

```
>> B = [3 5 2; 5 2 8; 3 6 9]
```

B =

3	5	2
5	2	8
3	6	9

Addition

Subtraction

Product

Transpose

```
>> X = A + B
```

X =

4	7	5
9	7	14
10	14	18

```
>> Y = A - B
```

Y =

-2	-3	1
-1	3	-2
4	2	0

```
>> Z = A * B
```

Z =

22	27	45
55	66	102
88	105	159

```
>> T = A'
```

T =

1	4	7
2	5	8
3	6	9

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
```

The use of “.” – “Element” Operation

Given A:

```
A =
     3     5     3
     6     8     2
     2     7     3
```

Divide each
element of A by 2

```
>> A./2

ans =

    1.5000    2.5000    1.5000
    3.0000    4.0000    1.0000
    1.0000    3.5000    1.5000
```

Multiply each
element of A by 3

```
>> A.*3

ans =

     9    15     9
    18    24     6
     6    21     9
```

Square each element
of A

```
>> A.^2

ans =

     9    25     9
    36    64     4
     4    49     9
```

Mean and Median

Mean: Average or mean value of a distribution

Median: Middle value of a sorted distribution

$M = \text{mean}(A), \quad M = \text{median}(A)$

$M = \text{mean}(A, \text{dim}), \quad M = \text{median}(A, \text{dim})$

$M = \text{mean}(A), M = \text{median}(A)$: Returns the mean or median value of vector A.

If A is a multidimensional mean/median returns an array of mean values.

Example:

$A = [0 \ 2 \ 5 \ 7 \ 20]$

$B = [1 \ 2 \ 3$
 $3 \ 3 \ 6$
 $4 \ 6 \ 8$
 $4 \ 7 \ 7];$

$\text{mean}(A) = 6.8$

$\text{mean}(B) = 3.0000 \ 4.5000 \ 6.0000$ (column-wise mean)

$\text{mean}(B, 2) = 2.0000 \ 4.0000 \ 6.0000 \ 6.0000$ (row-wise mean)

Examples:

$A = [0 \ 2 \ 5 \ 7 \ 20]$

$B = [1 \ 2 \ 3$
 $3 \ 3 \ 6$
 $4 \ 6 \ 8$
 $4 \ 7 \ 7];$

Mean:

$\text{mean}(A) = 6.8$

$\text{mean}(B) = 3.0 \ 4.5 \ 6.0$ (column-wise mean)

$\text{mean}(B, 2) = 2.0 \ 4.0 \ 6.0 \ 6.0$ (row-wise mean)

Median:

$\text{median}(A) = 5$

$\text{median}(B) = 3.5 \ 4.5 \ 6.5$ (column-wise median)

```
median(B,2) = 2.0
            3.0
            6.0
            7.0 (row-wise median)
```

Standard Deviation and Variance

- Standard deviation is calculated using the std() function
- std(X) : Calculate the standard deviation of vector x
- If x is a matrix, std() will return the standard deviation of each column
- Variance (defined as the square of the standard deviation) is calculated using the var() function
- var(X) : Calculate the variance of vector x
- If x is a matrix, var() will return the standard deviation of each column

```
X = [1 5 9;7 15 22]
s = std(X)
s = 4.2426  7.0711  9.1924
```

Insert element to matrix

```
» B=[1 3 7 8;2 6 5 11;12 14 15 13]
```

```
B =
```

```
1   3   7   8
2   6   5  11
12  14  15  13
```

To insert element 42 to row 2 and column 5

```
» B(2,5)= 42
```

```
B =
```

1	3	7	8	0
2	6	5	11	42
12	14	15	13	0

» B=[1 3 7 8;2 6 5 11;12 14 15 13]

B =

1	3	7	8
2	6	5	11
12	14	15	13

To insert elements 11,13,54,31 to row 4 and columns 1,2,3,4

» B(4,1:4)= [31 54 13 11]

B =

1	3	7	8
2	6	5	11
12	14	15	13
31	54	13	11

Replace element in a matrix

» B=[1 3 7 8;2 6 5 11;12 14 15 13]

B =

1	3	7	8
2	6	5	11
12	14	15	13

To replace element in row 3 and column 1 by number 0

» B(3,1)= 0

B =

1	3	7	8
2	6	5	11
0	14	15	13

» B=[1 3 7 8;2 6 5 11;12 14 15 13]

B =

1	3	7	8
2	6	5	11
12	14	15	13

To replace many elements in row 1,2 and columns 1,2,3 by number 0

» B(1:2,1:3)= 0

B =

0	0	0	8
0	0	0	11
12	14	15	13

Delete elements in a matrix

In Matlab , you can delete row or column in a matrix

» B=[1 3 7 8;2 6 5 11;12 14 15 13]

B =

1	3	7	8
2	6	5	11
12	14	15	13

To delete row 3

» B(3,:)= []

B =

1	3	7	8
2	6	5	11

To delete column 4

```
» B(:,4) = [ ]
```

B =

```
1   3   7
2   6   5
12  14  15
```

Find elements in a matrix

To find element in row 1 and column 3 in matrix B

```
» B(1,3)
```

ans =

```
7
```

To find elements in row 2 and columns from 2 to 4

```
» B(2,2:4)
```

ans =

```
6   5  11
```

To find last element in a row or column

```
» B(2,end) =
```

ans =

```
11
```

Equality of Matrices

- Equality of matrices: Matrices A, B are said to be equal if A and B have the same size and corresponding elements are equal; i.e., A and B have dimension $m \times n$, and $A = [a_{ij}]$, $B = [b_{ij}]$, with $a_{ij} = b_{ij}$, $1 \leq i \leq m$, $1 \leq j \leq n$.

Matrix Addition

- Addition of matrices: Let $A = [a_{ij}]$ and $B = [b_{ij}]$ be of the same size. Then

$$A + B = [a_{ij}] + [b_{ij}] = [a_{ij} + b_{ij}]$$

is the matrix obtained by adding corresponding elements of A and B.

Scalar Multiple of a Matrix

- Scalar multiple of a matrix: Let $A = [a_{ij}]$ and t be a number (scalar). Then tA is the matrix obtained by multiplying all elements of A by t ; that is

$$tA = t[a_{ij}] = [ta_{ij}]$$

Negative of a Matrix

- Negative of a matrix: Let $A[a_{ij}]$. Then $-A$ is the matrix obtained by replacing the elements of A by their negatives; that is

$$-A = -[a_{ij}] = [-a_{ij}]$$

Matrix Subtraction

- Subtraction of matrices: Matrix subtraction is defined for two matrices $A[a_{ij}]$ and $B[b_{ij}]$ of the same size, in the usual way; that is

$$A - B = [a_{ij}] - [b_{ij}] = [a_{ij} - b_{ij}]$$

Matrix Operations

- The matrix operations of addition, scalar multiplication, negation and subtraction satisfy the usual laws of arithmetic.
- s, t are scalars and A, B, C are matrices of the same size.

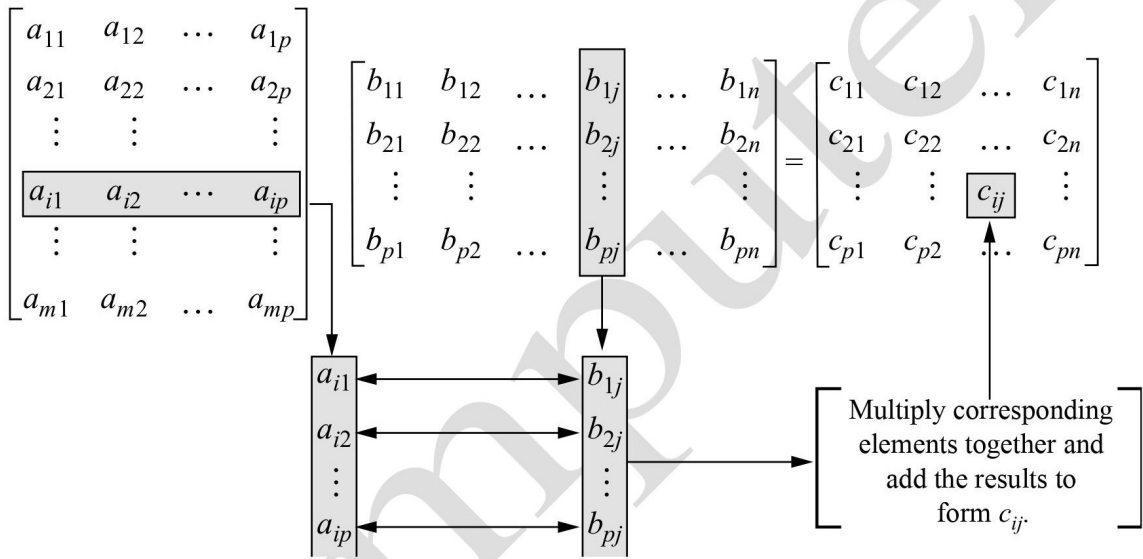
1. $(A+B)+C=A+(B+C)$;
2. $A+B=B+A$;
3. $0+A=A$;
4. $A+(-A) = 0$;
5. $(s+t)A=sA+tA, (s-t)A=sA-tA$;
6. $t(A+B) = tA+tB, t(A-B) = tA-tB$;

Matrix Multiplication

8. $1A = A, 0A = 0, (-1)A = -A$;
9. $tA = 0 \Rightarrow t = 0$ or $A = 0$

(Matrix product) Let $A = [a_{ij}]$ be a matrix of size $m \times p$ and $B = [b_{jk}]$ be a matrix of size $p \times n$; (that is the number of columns of A equals the number of rows of B). Then AB is the $m \times n$ matrix $C = [c_{ik}]$ whose (i, j) -th element is defined by the formula

$$c_{ij} = \sum_{k=1}^p a_{ik} b_{kj} = a_{i1}b_{1j} + \cdots + a_{ip}b_{pj}$$



Examples of Matrix Multiplication

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1 \times 5 + 2 \times 7 & 1 \times 6 + 2 \times 8 \\ 3 \times 5 + 4 \times 7 & 3 \times 6 + 4 \times 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 23 & 34 \\ 31 & 46 \end{bmatrix} \neq \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

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

$$\begin{bmatrix} 3 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 11 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Laws of Matrix Multiplication

Matrix multiplication obeys many of the familiar laws of arithmetic apart from the commutative law.

1. $(AB)C = A(BC)$ if A, B, C are $m \times n, n \times p, p \times q$, respectively;
2. $t(AB) = (tA)B = A(tB), A(-B) = (-A)B = -(AB)$;
3. $(A+B)C = AC + BC$ if A and B are $m \times n$ and C is $n \times p$;
4. $D(A+B) = DA + DB$ if A and B are $m \times n$ and D is $p \times m$.

System of Linear Equations

- System of m linear equations in n unknowns:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

- Equivalent matrix equation $Ax=b$.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

- A is the coefficient matrix

- Vector or unknowns: $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

- Vector of constants (right-hand side): $b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$

Equivalent Equation for $Ax = b$

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Example of a Linear System

- System
$$\begin{aligned} x + y + z &= 1 \\ x - y + z &= 0 \\ 3x + 5y - z &= 2 \end{aligned}$$

- Matrix form
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 3 & 5 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$$

- Alternative form
$$x \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix} + y \begin{bmatrix} 1 \\ -1 \\ 5 \end{bmatrix} + z \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$$

- Solution
$$\begin{bmatrix} 0.0000 \\ 0.5000 \\ 0.5000 \end{bmatrix}$$

$$(1) 0.0000 + (1) 0.5000 + (1) 0.5000 = 1$$

$$(1) 0.0000 - (1) 0.5000 + (1) 0.5000 = 0$$

$$3(0.0000) + 5(0.5000) - (1) 0.5000 = 2$$

The Trace

If A is an $n \times n$ matrix, the trace of A , written $\text{trace}(A)$, is the sum of the diagonal elements; that is

$$\text{trace}(A) = a_{11} + a_{22} + \cdots + a_{nn} = \sum_{i=1}^n a_{ii}.$$

Example of the Trace

$$\text{If } A = \begin{bmatrix} 5 & 8 & 12 & -1 \\ 7 & 4 & -8 & 7 \\ 0 & 3 & -6 & 5 \\ -1 & -9 & 4 & 3 \end{bmatrix}, \text{ then } \text{trace}(A) = 5 + 4 + (-6) + 3 = 6.$$

Properties of the Trace

- $\text{trace}(A + B) = \text{trace}(A) + \text{trace}(B)$
- $\text{trace}(cA) = c \text{trace}(A)$, where c is a scalar.
- If A is an $n \times n$ matrix and B is an $n \times n$ matrix, then $\text{trace}(AB) = \text{trace}(BA)$.

MATLAB

- When performing vector or matrix operations using the operators '*', '/', and '^', it may be necessary to use the dot operator ('.') to perform element by element operations.

MATLAB Examples

```
>> A = [1 5 1; 2 -1 6; 1 0 3]
```

```
A =
```

```
1    5    1
```

```
2   -1    6
```

```
1    0    3
```

```
>> B = [2 3 0; 3 -1 7; 4 8 9]
```

```
B =
```

```
2    3    0
```

```
3   -1    7
```

```
4    8    9
```

```
>> 5*A - 10*B + 3*A*B
```

```
ans =
```

```
48   13  137
```

```
55  170  101
```

```
7    1    6
```

```
>> trace(A + B)
```

```
ans =
```

```
13
```

```
>> 7*trace(A + B)
```

```
ans =
```

```
91
```

```
>> trace(A*B)
```

```
ans =
```

```
103
```

```
>> trace(B*A)
```

```
ans =
```

```
103
```

```
>> A.*B
```

```
ans =
```

```
2   15    0
```

```
6    1   42
```

```
4    0   27
```

Another MATLAB Example

- It is a common mistake to forget using "." when dividing an expression into a constant.

```
>> v = [1 2 3 4 5]';
```

```
>> 1./v
```

```
ans =
```

```
1.0000
```

```
0.5000
```

```
0.3333
```

```
0.2500
```

```
0.2000
```

MATLAB Exponentiation Operator

- Operator '^'. When vectorization is used, you must use it in the form '.*'. For instance, if t is the vector

```
t = [1.0:0.5:3.0]
```

```
then
```

```
y = t.^2
```

is the vector whose elements are the squares of those in t.

```
>> t = [1.0:0.5:3.0]
```

```
t =
```

```
1.0000 1.5000 2.0000 2.5000 3.0000
```

```
>> y = t.^2
```

```
y =
```

```
1.0000 2.2500 4.0000 6.2500 9.0000
```

Powers of Matrices

- The $n \times n$ identity matrix has ones on its diagonal and zeros in all other locations.

$$I = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}$$

- If A is an $n \times n$ matrix, $A * I = I * A = A$.

The Matrix Inverse

- An $n \times n$ matrix A is invertible, or nonsingular, if there exists an $n \times n$ matrix denoted by A^{-1} such that

$$A * A^{-1} = A^{-1} * A = I$$

- For example if $A = \begin{bmatrix} 1 & 0 & 2 \\ 1 & 3 & 0 \\ 2 & 1 & 5 \end{bmatrix}$, then

$$A^{-1} = \begin{bmatrix} 3 & \frac{2}{5} & -\frac{6}{5} \\ -1 & \frac{1}{5} & \frac{2}{5} \\ -1 & -\frac{1}{5} & \frac{3}{5} \end{bmatrix}$$

Properties of the Matrix Inverse

- A matrix A can have only one inverse.
- If B is a matrix such that $BA = I$, then $AB = I$. Similarly, if $AB = I$, then $BA = I$; in other words $B = A^{-1}$.
- $(AB)^{-1} = B^{-1} A^{-1}$
- $(B^{-1} A^{-1})(AB) = B^{-1}(A^{-1}A)B = B^{-1}(I)B = B^{-1}B = I$
- If the coefficient matrix, A , of an $n \times n$ linear system $Ax = b$ has an inverse then the unique solution to the system is

$$x = A^{-1}b$$

Computing the Inverse using MATLAB

- Use the command `inv`.

```
>> A = [6 15 -1;12 9 -8;-1 2 3]
```

```
A =
```

```
6    15   -1
```

```
12    9   -8
```

```
-1    2    3
```

```
>> B = inv(A)
```

```
B =
```

```
-0.220512820512820  0.241025641025641  0.569230769230769
```

```
0.143589743589744 -0.087179487179487 -0.184615384615385
```

```
-0.169230769230769  0.138461538461538  0.646153846153846
```

```
>> A*B
```

```
ans =
```

```
1.000000000000000 -0.000000000000000 0
```

```
0 1.000000000000000 0
```

```
0.000000000000000 -0.000000000000000 1.000000000000000
```

The Matrix Transpose

- If A is an $m \times n$ matrix, the transpose A^T is the $n \times m$ matrix obtained by exchanging rows and columns. If A is the matrix $[a_{ij}]$, then

$$(A^T)_{ij} = a_{ji}.$$

$$\begin{bmatrix} 1 & 3 & -1 & 7 \\ 2 & 5 & 8 & -9 \\ 4 & 0 & 1 & 12 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 5 & 0 \\ -1 & 8 & 1 \\ 7 & -9 & 12 \end{bmatrix}$$

Properties of the Transpose

1. $(A^T)^T = A$;
2. $(A \pm B)^T = A^T \pm B^T$ if A and B are $m \times n$;
3. $(sA)^T = sA^T$ if s is a scalar;
4. $(AB)^T = B^T A^T$ if A is $m \times k$ and B is $k \times n$;
5. A is nonsingular, then A^T is also nonsingular and $(A^T)^{-1} = (A^{-1})^T$.
6. $x^T x = x_1^2 + \dots + x_n^2$ if $x = [x_1, \dots, x_n]^T$ is a column vector.

A Symmetric Matrix

- An $n \times n$ matrix A is symmetric if $A^T = A$

$$A = \begin{bmatrix} 1 & 8 & 12 & 3 \\ 8 & 5 & 1 & 10 \\ 12 & 1 & 6 & 9 \\ 3 & 10 & 9 & 2 \end{bmatrix} = A^T$$

- Symmetric matrices appear very frequently in engineering and science problems.
- Computations with symmetric matrices can normally be done faster than with nonsymmetric matrices.

A Very Important Symmetric Matrix

- If A is an $m \times n$ matrix then $A^T A$ is an $n \times n$ matrix.
- A^T is $n \times m$, and an $(n \times m)$ times an $(m \times n)$ matrix has dimensions $n \times n$.
- If A is an $m \times n$ matrix, then $A^T A$ is an $n \times n$ symmetric matrix.

$$(A^T A)^T = A^T (A^T)^T = A^T A$$

The Transpose in MATLAB

- Use the operator `'` to transpose a matrix

```
A =
```

```
    1     8    -1
    3    -9    15
   -1     5     3
```

```
>> A_TA = A'*A
```

```
A_TA =
```

```
    11    -24    41
   -24    170   -128
    41   -128   235
```

```
>> A_TA - A_TA'
```

```
ans =
```

```
    0     0     0
    0     0     0
    0     0     0
```

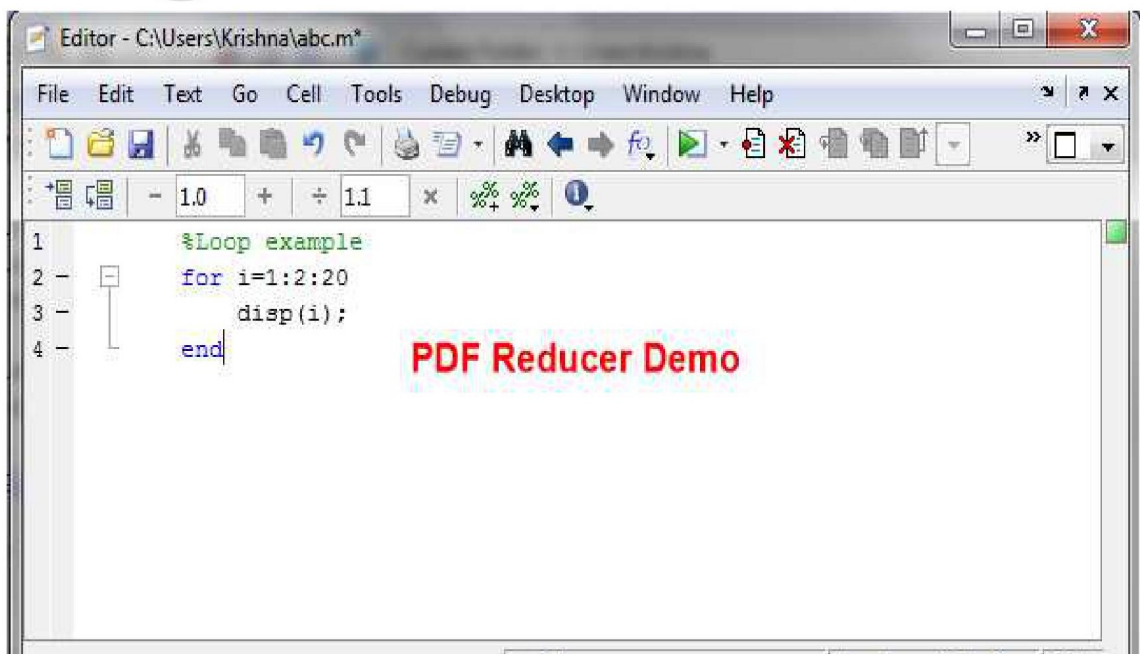
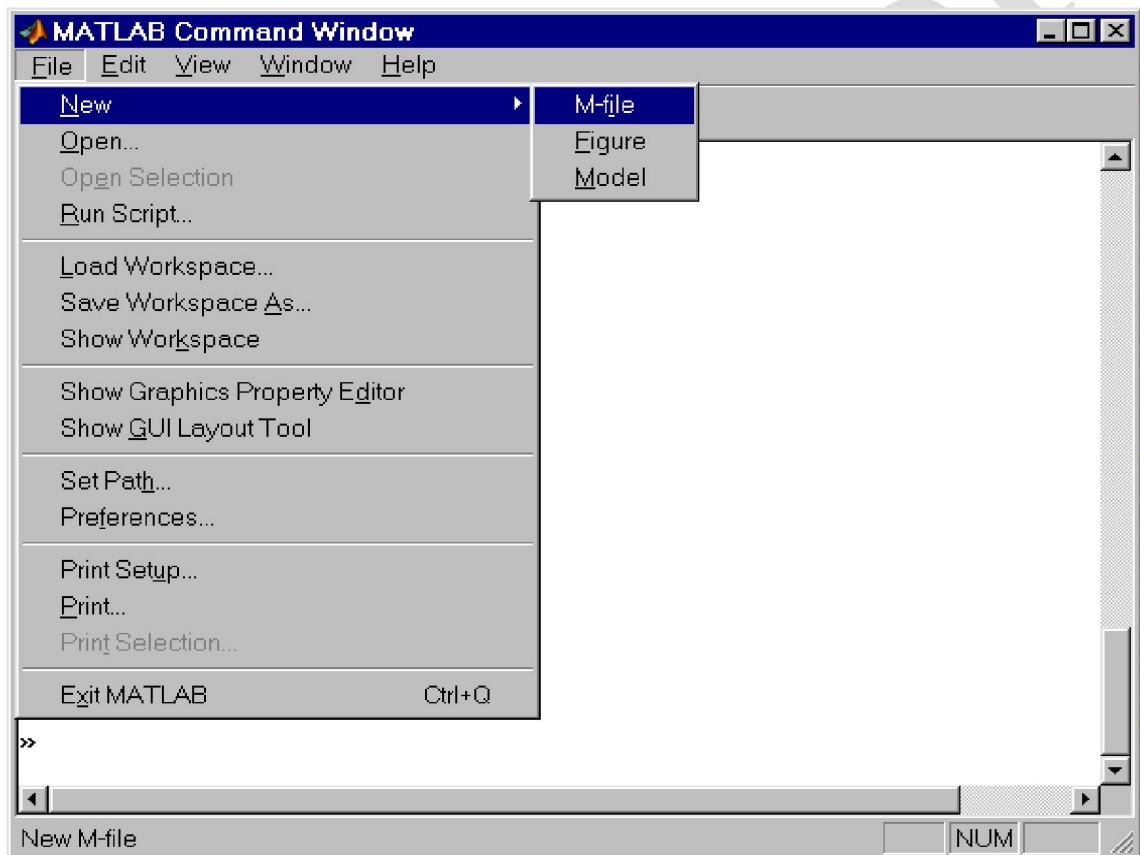
Chapter three

M-Files

- **Script file:** a collection of MATLAB commands
- **Function file:** a definition file for one function

Script Files

- Any valid sequence of MATLAB commands can be in the script files.
- Variables defined/used in script files are global, i.e., they present in the workspace.



M-file Example

%test.m

```
A=[1 2 3;4 5 6;7 8 9]
B=sum(sum(A))
C=prod(sum(A))
```

%output

```
>> test
A=
     1     2     3
     4     5     6
     7     8     9
B=
    45
C=
   3240
```

%test.m

```
A=[11 22 33;41 51 61;78 88 98]
B=sum(sum(A))
C=prod(sum(A))
```

%output

```
>> test
A=
    11    22    33
    41    51    61
    78    88    98
B=
   483
C=
  4018560
```

%test.m

```
clc
A=[1 2 3 ;4 5 6 ;7 8 9]
B=sum(sum(A))
C=prod(sum(A))
```

%output

```
>> test
A=
     1     2     3
     4     5     6
     7     8     9
B=
    45
C=
   3240
```

%test.m

```
clc
A=[11 21 31 ;42 52 62 ;73 38 39]
B=sum(sum(A))
C=prod(sum(A))
```

%output

```
>> test
A=
    11    21    31
    42    52    62
    73    38    39
B=
    369
C=
   1846152
```

%test.m

```
clc
A=[1 2 3 ;4 5 6 ;7 8 9]
B=sum(sum(A))
C=prod(sum(A))
```

%output

```
>> test
A=
     1     2     3
     4     5     6
     7     8     9
B=
    45
C=
   3240
```

%test.m

```
clc
D=[1 2 3 ;4 5 6 ;7 8 9]
E=sum(sum(D))
F=prod(sum(D))
```

%output

```
>> test
D=
     1     2     3
     4     5     6
     7     8     9
E=
    45
F=
   3240
```

%test.m

```
clc
clear
D=[1 2 3 ;4 5 6 ;7 8 9]
E=sum(sum(D))
F=prod(sum(D))
```

%output

```
>> test
D=
     1     2     3
     4     5     6
     7     8     9
E=
    45
F=
   3240
```

%example.m

```
A=[3 4 9;2 4 5]
B=size (A)
C= size(A,1)
D = size(A,2)
```

%output

```
>> example
A=
     3     4     9
     2     4     5
B=
     2     3
C=
     2
D=
     3
```

%example.m

```
A=[1 15 2 11;23 1 4 5;3 1 15 7;1 4 9 10]
B=max(A)
C=max(B)
```

%output

```
>> example
A=
     1    15     2    11
    23     1     4     5
     3     1    15     7
     1     4     9    10
B=
    23    15    15
    11
C=
    23
```


%example.m

```
A=[1 15 2 11;23 1 4 5;3 1 15 7;1 4 9 10]
B=min(A)
C=min(B)
```

%output

```
>> example
A=
     1    15     2    11
    23     1     4     5
     3     1    15     7
     1     4     9    10
B=
     1     1     2     5
C=
     1
```

%example.m

```
A=[1 2 3 4 5 6 7 8 9 10]
B= length(A)
```

%output

```
>> example
A=
     1     2     3     4     5     6     7     8     9    10
B=
    10
```

%example.m

```
A=[1 2 3;4 5 6;7 8 9]
B= fliplr(A)
```

%output

```
>> example
A=
     1     2     3
     4     5     6
     7     8     9
B=
     3     2     1
     6     5     4
     9     8     7
```

%example.m

```
A=[1 2 3;4 5 6;7 8 9]  
B= flipud(A)
```

%output

```
>> example  
A=  
    1  2  3  
    4  5  6  
    7  8  9  
B=  
    7  8  9  
    4  5  6  
    1  2  3
```

%example.m

```
A=[1 2 3;4 5 6;7 8 9]  
B= rot90(A)
```

%output

```
>> example  
A=  
    1  2  3  
    4  5  6  
    7  8  9  
B=  
    3  6  9  
    2  5  8  
    1  4  7
```

%example.m

```
A=[1 2 3 ;4 5 6 ;7 8 9;10 11 12]  
B= reshape(A,6,2)
```

%output

```
>> example  
A=  
    1    2    3  
    4    5    6  
    7    8    9  
   10   11   12  
B=  
    1    8  
    4   11  
    7    3  
   10    6  
    2    9  
    5   12
```

example.m

```
X = linspace(pi,-pi,8)
Y = sin(x)
```

%output

```
>> example
X = 3.14  2.24  1.35  0.45 -0.45 -1.35 -2.24 -3.14
Y = 0.00  0.78  0.97  0.43 -0.43 -0.97 -0.78  0.00
```

example.m

```
X = [ 7 5 2 1 3 6 4 8]
Y = sort(X)
```

%output

```
>> example
X = 7 5 2 1 3 6 4 8
Y = 1 2 3 4 5 6 7 8
```

Chapter four

MATLAB Graphics

- One of the best things about MATLAB is interactive graphics
- “plot” is the one you will be using most often

Plotting Commands

plot(independent variable, dependent variable)

plot(x,y) defaults to a blue line

plot(independent variable, dependent variable, 'property')

plot(x,y,'ro') uses red circles

plot(x,y,'g*') uses green asterisks

If you want to put two plots on the same graph, use “hold on”

plot(a,b,'r:') (red dotted line)

hold on

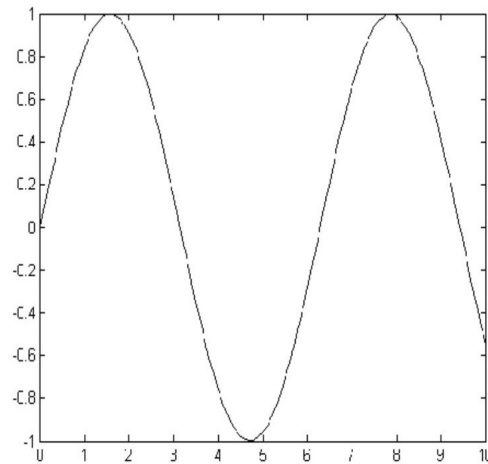
plot(a,c,'ko') (black circles)

Color, Symbols, and Line Types

Colors	Symbols	Line Types
b blue	. point	- solid
g green	o circle	: dotted
r red	x x-mark	-. dashsot
c cyan	+ plus	-- dashed
m magenta	* star	
y yellow	s square	
k black	d diamond	
	v triangle (down)	
	^ triangle (up)	
	< triangle (left)	
	> triangle (right)	
	p pentagram	

Example

```
clc
clear
x=0 : 0.1 : 10;
y= sin(x);
plot(x,y);
```



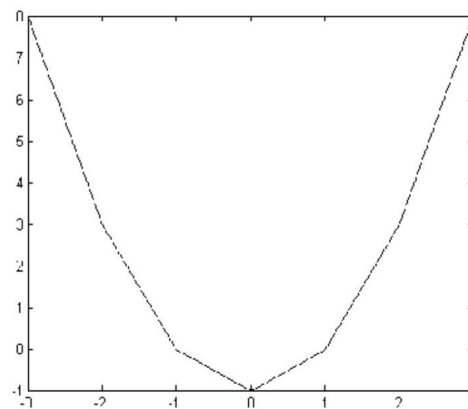
Plotting

PLOT Linear plot.

- PLOT(X,Y) plots vector Y versus vector X
- PLOT(X,Y,'S') with plot symbols and colors

Example

```
x = [-3 -2 -1 0 1 2 3];
y = (x.^2) -1;
plot(x, y,'r');
```



Plot Properties

XLABEL X-axis label.

- XLABEL('text') adds text beside the X-axis on the current axis.

YLABEL Y-axis label.

- YLABEL('text') adds text beside the Y-axis on the current axis.