	Ministry of Higher Education and Scientific Research - Iraq Northern Technical University Technical Engineering College Kirkuk Department of Fuel and Energy Engineering	

MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية					
Module Title	**Computer Programming (MATLAB)**			**Module Delivery**	
Module Type	Core			☒ **Theory** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☒ **Seminar**	
Module Code	**FEK202**				
ECTS Credits	**4**				
SWL (hr/sem)	**100**				
Module Level	2		**Semester of Delivery**	1	
Administering Department	FEK		**College**	COGTEK	
Module Leader	Layth Ali Hussein		**e-mail**	Layth.ali@ntu.edu.iq	
Module Leader's Acad. Title	Ass.Lecturer		**Module Leader's Qualification**	M.Sc.	
Module Tutor	Name (if available)		**e-mail**	E-mail	
Peer Reviewer Name	Name		**e-mail**	E-mail	
Scientific Committee Approval Date	15/09/2024		**Version Number**	1.0	

Relation with other Modules العلاقة مع المواد الدراسية الأخرى			
Prerequisite module	None	**Semester**	
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Aims أهداف المادة الدراسية	1. Working with the MATLAB user interface. 2. Entering commands and creating variables. 3. Analyzing vectors and matrices. 4. Visualizing vector and matrix data 5. Working with data files. 6. Automating commands with scripts 7. Writing programs with branching and loops.
Module Learning Outcomes مخرجات التعلم للمادة الدراسية	1. Demonstrate use of mathematical based software to write basic programs 2. Employ computer programs to solve numerical methods problems.# 3. Demonstrate competency of creating computer programs to solve problems of ordinary differential equations, partial differential equations and optimization.
Indicative Contents المحتويات الإرشادية	Indicative content includes the following. <u>Part A - Software Engineering</u> Structural and Functional Modelling, Software Development Life cycle. Requirements determination, feasibility analysis, final specifications, hardware and software study system (design –implementation –evaluation– modification). Role of systems analyst – attributes of a systems analyst – tools used in system analysis. Types of information: operational, tactical, strategic and statutory – why do we need information systems – management structure – requirements of information at different levels of management – functional allocation of management – requirements of information for various functions – qualities of information – small case study. <u>Part B - Algorithms and Flowcharts</u>

	Introduction, Symbols, Types of flowcharts, Exercise introduction to Visual studio. Platform, Environment, Menu Bar, Toolbars, Tool Box, Project explorer, Properties window, Form designer, Form layout. Design time and run time Fundamentals. Graphical User Interface, Command Buttons, Label, text box, check box, option, list box, Timer. Constants and Variable, Arrays, Arithmetic operators, Expressions - Events, Properties, Methods - Procedures and Functions – Menus. Part C - Control Flow Statements: Condition Statement: If-Then, Select Case. Loop statement: For-Next, Do-while, Do-Loop While, Exit Loop. Exit and stop statements. Test phase Debugging, Error Handling Mashed edit control - Chart controls - Rich text box - Slider - Tabbed Dialog - Multiple forms - common dialog control. Creating executable file by Package & Deployment Wizard. Create the applications for Fluid calculation, Trial and error calculation, Enthalpy calculation, non-linear equations, and matrix inverse

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	Type something like: The main strategy that will be adopted in delivering this module is to encourage students' participation in the exercises, while at the same time refining and expanding their critical thinking skills. This will be achieved through classes, interactive tutorials and by considering type of simple experiments involving some sampling activities that are interesting to the students.

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	63	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	4
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	37	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	2
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	100		

Module Evaluation

تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	2	10% (10)	5, 10	LO #1, 2, 10 and 11
	Assignments	2	10% (10)	2, 12	LO # 3, 4, 6 and 7
	Projects / Lab.	1	10% (10)	Continuous	All
	Report	1	10%(10)		
Summative assessment	Midterm Exam	2 hr	10% (10)	7	LO # 1-7
	Final Exam	2hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)

المنهاج الاسبوعي النظري

	Material Covered
Week 1	Introduction, Environment of MATLAB
Week 2	Arithmetic Expressions, Mathematical functions, Logical Operators, Relational Operators.
Week 3	Vectors and Matrices: Matrix operations, transpose and inverse of Matrix
Week 4	Working with polynomials (manipulating polynomials, derivatives roots, eigen values).
Week 5	Working with polynomials (manipulating polynomials, derivatives roots, eigen values).
Week 6	Solve System of Linear Equations by Gauss Elimination Method
Week 7	Solve System of Linear Equations by Gauss Elimination Method,
Week 8	M-file: Create in an M-file, function calling in MATLAB Programming with MATTAB, Use of Built-in Functions, Input Output, Structured Programming, Nesting and Indentation
Week 9	M-file: Create in an M-file, function calling in MATLAB Programming with MATTAB, Use of Built-in Functions, Input Output, Structured Programming, Nesting and Indentation
Week 10	Dealing with Errors and Pitfalls.
Week 11	Dealing with Errors and Pitfalls: Syntax Errors. Incompatible vector sizes. Name hiding. Logic and Rounding Error.
Week 12	Graphic plot: Graphics two-dimensions plots, Log-log and semi-log plots, Histograms plots. Linear Regression, Curve fitting.
Week 13	Graphic plot: Graphics two-dimensions plots, Log-log and semi-log plots, Histograms plots. Linear Regression, Curve fitting.
Week 14	Conditions and loops statements: Functions: if, else, else if, while, for, switch, break

Week 15	Conditions and loops statements: Functions: if, else, else if, while, for, switch, break
Week 16	**Preparatory week before the final Exam**

Delivery Plan (Weekly Lab. Syllabus) المنهاج الاسبوعي للمختبر	
	Material Covered
Week 1	Lab 1: Introduction, Environment of MATLAB.
Week 2	Lab 2: Arithmetic Expressions.
Week 3	Lab 3: Vectors and Matrices.
Week 4	Lab 4: M-file: Create in an M-file.
Week 5	Lab 5: Graphic plot: Graphics two-dimensions plots.
Week 6	Lab 6: Dealing with Errors and Pitfalls.
Week 7	Lab 7: Conditions and loops statements.

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**
Required Texts	Mark E. Davis “Numerical method and modelling for chemical engineers”.	Yes
Recommended Texts	Mathew J.H., Numerical Methods for Mathematics, Science and Engineering	Yes
Websites	https://www.mathworks.com/help/matlab/creating_guis/apps-overview.html	

Grading Scheme مخطط الدرجات				
Group	**Grade**	التقدير	**Marks (%)**	**Definition**
Success Group (50 - 100)	**A** - Excellent	امتياز	90 - 100	Outstanding Performance
	B - Very Good	جيد جدا	80 - 89	Above average with some errors
	C – Good	جيد	70 - 79	Sound work with notable errors
	D - Satisfactory	متوسط	60 - 69	Fair but with major shortcomings

	E - Sufficient	مقبول	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX –** Fail	راسب (قيد المعالجة)	(45-49)	More work required but credit awarded
	F – Fail	راسب	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.

MATLAB (matrix laboratory) is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization and programming.

MATLAB - Basic Syntax

MATLAB environment behaves like a super-complex calculator. You can enter commands at the >> command prompt.

MATLAB is an interpreted environment. In other words, you give a command and MATLAB executes it right away.

Hands on Practice

Type a valid expression, for example,

```
5 + 5
```

And press ENTER

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is –

```
ans = 10
```

Let us take up few more examples –

```
3 ^ 2          % 3 raised to the power of 2
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is –

```
ans = 9
```

Another example,

```
sin(pi /2)    % sine of angle 90°
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is –

ans = 1

Genenal Notes

- **To type a acommand the cursor must be place next to the command prompt >> .once a command is type and enter key is pressed the command is executed.**

  >>8*(4+2)
  ans =48

- **.serveral commandcan be type in the same line .this is done by typing a comma(,) between the commands**
  >>a=3,b=5,c=a+b
  a=3
  b=5
  c=8
- **.if the command is too long to fit in the one line it can be continued to the next line by typing three periods (...)and pressing the Enter key .the continuation of the command is then type in the new line**
  >>d=2+3*4+10/5+...
  2*4+2..

  d=26
- **.if the semicolon(;)is typed at the end of a command the result is not display**
  >>x=5;
  >>y=20;
  >>x*y
  ans=100

- **It is not possible to go back to a pervious line that is display in the command windows .to make a correction recall he pervious command by (↑)and (↓)keys modify then re-execute it.**
- **When the symbol (%)is type at the beginning of a line the line is designated as a comment .this means that when the Eneter key is pressed the line not executed .**
  >>%this command line used to find the square root of x

>>x=9
>>sqrt(x)
ans=3

- **In case of wrong coding the program will produce and alert sound and the error will be show in red colour**
  >>sqr(x)
  ??? Undefined function or variable 'sqr'.(Error red)
- **The (clc)command clears all input and output from the command window display without removing previously defined variables if want to remove all defined variables then we use (clear)command**

>>clc
>>clear

Variables Names :
A variable can be named according to the following rules :

- **.Must begin with a letter .**
- **Cannot contain punctuation characters(period ,comma,colon,semicolon,...)**
  A#,A,A:,A?,A",A.
  Error: Missing operator, comma, or semicoerlon.

- **Matlab is case-sensitive :it distinguishes between uppercase and lowercase is desired)**
  - **A≠a and AA≠aa aA≠Aa**

- **No sapces are allowed between characters(use the underscore where a space is desired**
  B ⌂ 2 (error) B_2(√ correct)

- **Avoid using the name of a built-in function (cos,sin,sqrt....)**

Or predefined variables (ans,pi,inf)for a variable.Once a function name is used for a variable. name the function or the predefine variable cannot be used .

- **Keywords(for,while,if,else,end,...)cannot be uesd as variable names**
- **Predefined variables**
  Matlab includes a number of predefied variables.some of these variables

Predefine Variables
Matlab includes a number of predefine variable names some of these variable are summarized below :

Predefine Variables	Description
Pi	The number pi=3.14
info	Used for infinity
I	Defined as $\sqrt{-1}$
NaN	Stands for not –Number 0/0=NaN

Defining variable can be defined using the following formula:
Varaible name=A numerical value or a computable expression
X=10---------->Numerical value
X=10+5 ----------→computable expression
Y=15

Notes if a variable already defined typing the variables name and pressing the enter key will display the variable and its value
If a variable already defined assigning a new value to the variable will replace its original value
C=15
>>c=15
c=c-10
>>5
Arithmetic opration with Scalars IN MATLAB

operation	symbol	Example
Addition	+	5+2=7
subtration	-	5-2=3
Multiplication	*	5*2=10
Right division	/	5/2=2.5
Left division	\	5\2=0.4
Exponentiation	^	5^2=25

Order of precedence
Matlab executes the calculation according to the order of precedence as follows

precedence	Mathematical Opertion
Frist	Parentheses For nested Parentheses the innermost are executed first
Second	Exponentaition
Third	Multiplication,(equal precedence)
Fourth	Addition and subtraction

Examples:
3*4+5=12+5=17
3*(4+5)=3*9=27
2^3*5=8*5=40
2^(3*5)=2^15=32768
4*5/2=20/2=10
3*4-6/2=12-3=9

Problem
Calculate

Math	**matlab**
A) $\frac{22+5.1^2}{50-6.5^2}$	**>> (22+5.1^2)/(50-6.5^2)** **>> 6.1948**
B) $\frac{44}{7}+\frac{8^2}{5}+\frac{99}{3.9^2}$	**>> 44/7+8^2/5-99/3.9^2** **ans =** **12.5768**
c) $(2+7)^3+\frac{273^{2/3}}{2}+\frac{55^2}{3}$	**>> (2+7)^3+273^(2/3)/2+55^2/3** **ans =** **1.7584e+03**
H.W Define the variable x as x=6.7 then evaluate: $0.01x^5-1.4x^3+80x+16.7$	

Type equation here.

Elementary Built-in function .Afunction has name and an argument in parenthesses.

for exaple the function that calculate the a square root of a number is sqrt(x).its name is sqrt and the argument is x.

Function name sqrt(x) Argument (number,define variable,computable expression)some commonly used function are given in Table below

Elementary math Functions:

Function	Description	example
Sqrt(x)	Square root	Sqrt(81) Ans =9
Exp(x)	Exponential	EXP(5) Ans=148.4132
Abs(x)	Absolute value(|x|)	Abs(-24) >>24
Log(x)	Nature logarith base logarithm(ln)	Log(1000) Ans=6.9078
Log10(x)	Base 10 logarithm	Log10(1000) Ans=3
Factorial(x)	The factorial function x! x must be a postive integer	Factorial(5) Ans =120

Problem Caculate

A) $\frac{\sqrt{41^2-5.2^2}}{e^{5}-100.53}$

B)

$|\sqrt[3]{132}+\frac{\ln(500)}{8}$

c) $\frac{3^7\log(76)}{7^{3+546}}+3!$

Solve
A
>> sqrt(41^2-5.2^2)/(exp(5)-100.53)

ans = 0.8493

B)
>> abs(132^(1/3)+log(500)/8)

ans = 5.8685

C)

>> (3^7*log10(76))/(7^3+546)+factorial(3)

ans = 10.6269

function	Description	Example
Deg2rad	**Convert angle from degrees to radians**	**Deg2rad(60)** **>> deg2rad(60)** **ans =** **1.0472**
Rad2deg	**Convert angle radians to degrees**	**>> sin(pi/6)** **ans =** **0.5000**
Sin(x)	**Sine of angle x in radians**	**>> sin(pi/6)** **ans =** **0.5000**
Cos(x)	**Cosine of angle x in radians**	**>> cos(pi/6)** **ans =** **0.8660**
Tan(x)	**Tangent of angle x in radians**	**Tan(pi/6)**
Sec(x)	**Secant of angle x in radians**	**>> tan(pi/6)** **ans =** **0.5774**
Sec(x)	**Cosecant of angle x in radians**	**>> Sec(pi/3)** **ans =** **2.0000**
Csc(x)	**Cosecantof angle x in radians**	**>> CSC(pi/3)** **ans =** **1.1547**

Cot(x)	**Cotangent of angle x in radians**	**>> cot(pi/3)** **ans =** **0.5774**
asin(x)	**Inverse cosine**	**>> asin(0.5)** **ans =** **0.5236**
acos(x)	**Inverse cosine(result in radians**	**>> acos(0.866)** **ans =** **0.5236**
atan(x)		**>> atan(0.5774)** **ans =** **0.5236**
Sinh(x)	**Hyperbolic sine of angle x in radians**	**>> sinh(pi/6)** **ans =** **0.5479**
Cosh(x)	**Hyperbolic cosine of angle x in radians**	**>> cosh(pi/6)** **ans =** **1.1402**
Tanh	**Hyperbolic of tangent of angle x in radians**	**>> tanh(pi/6)** **ans =** **0.4805**

Problem :calculate

a)cos$\left(\left(\frac{7}{9}\pi\right)+\boldsymbol{tan}\left(\frac{7}{15}\boldsymbol{\pi}\right)\right.$

solve

```
>> cos(7*pi/9)+tan(7*pi/15)

ans =

   8.7483
```

b)sin²(30)+$\boldsymbol{cos^2}\ (\mathbf{30})$

solve

```
>> sin(30)^2+cos(30)^2

ans =

    1
```

H.W

1)cos$\left(\frac{5\pi}{6}\right)^2\ \boldsymbol{sin^2}\left(\frac{7\pi}{8}\right)+\frac{tan(\frac{\pi ln8}{6}}{\frac{35}{2}}$

2)sin($\boldsymbol{cos^{-1}}\frac{3}{5}$ **)**

3)Trigonometric identity is given by :

$\boldsymbol{cos^2}\frac{x}{2}=\frac{tanx+sin(x)}{2tanx}$

For each part verify that identity is correct by calculate the valuee of the left and right side of the equation substitution x=$\frac{\pi}{5}$

Rounding function

round	**Round to the nearest integer**	**>>Round(18/5) 4**
Fix(x)	**Round toward zero**	**>>Fix(13/5) 2**
Ceil(x)	**Round toward infinity**	**>>ceil(11/5) 3**
Floor(x)	**Round toward minus infinity**	**>>floor(-9/4) -3**
Rem(x,y)	**Return the remainder after x is divided by y**	**>>rem(13,5) 3**
Sign(x)	**Return 1 if x >0 and-1 If x<0 and 0 if x=0**	**>>sign(5) 1**

Other function

function	**description**	**example**
rand	**Random number**	**>> rand ans = 0.9501**
date	**Current date**	**>> date ans = 09-Oct-2023**
who	**List variables in workspace**	**>> who Your variables are: ans x y**
whos	**List variables in worksapce withtheir sizes and types**	**X=4,y=5 >> whos Name Size Bytes Class x 1x1 8 double y 1x1 8 double**
Clear variable name	**Remove selected variables**	

MATLAB® is a high-performance language for technical computing.

It include

Math and computation

MATLAB System

Development Environment.

- graphical user interfaces

-MATLAB desktop

- Command Window,

-a command history,

-an editor and debugger,

-and browsers for viewing help, the workspace, files, and the search path.

MATLAB System

MATLAB Mathematical Function Library.

- **computational algorithms ranging from elementary functions, like sum, sine, cosine,**
- **complex arithmetic,**
- **sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.**

MATLAB System

- **The MATLAB Language. high-level matrix/array language with control flow statements, functions,**
- **data structures, input/output,**
- **object-oriented programming features.**
- **to create large and complex application programs**

MATLAB System

- **Graphics.**
- **displaying vectors and matrices as graphs,. functions for two-dimensional and three-dimensional**
- **data visualization, image processing, animation, and presentation graphics**

Prompt >>

- command line, indicated by the prompt (>>).

Matrix 3x3

A(I , j)

	Column 1	Column 2	Column 3
Row 1	A(1,1)	A(1,2)	A(1,3)
Row 2	A(2,1)	A(2,2)	A(2,3)
Row 3	A(3,1)	A(3,2)	A(3,3)

A(1,1) represent first row and first column in matrix A
A(3,1) represent third row and first column in matrix A and so on
A(1,1)and A(2,2) and A(3,3) represent Diagonal in Matrix A

To show content of matrix according I and j A(I,j).

$\mathcal{F}_x$>> A(1,1) Or

$\mathcal{F}_x$>>disp(A(1,1))

$\mathcal{F}_x$>>disp(A(1:3,3))

$\mathcal{F}_x$>>disp(A(3,1:3))

Summation Matrices

- **Enter matrix , A=[1 2 3;3 4 5;6 7 8]**
- **The element in row i and column j of A is denoted by A(i,j). I and j is Index of matrix**
- **Sum (matrix,2) summation rows**
- **Sum(matrix) it will summation columns**
- **Sum(diag(A))it will summation diagonal of matrix**
- **sum(diag(fliplr(A))) inverse of diag**

Transpose matrix

- **A’ = transpose matrix**
- **Make column row and row column**
- **It will be useful in Multiply Matrices**

Calculating the Determinant

- **For a 2×2 Matrix**

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$|A| = ad - bc$$

For a 3 x 3 Matrix

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

$$|A| = a(ei - fh) - b(di - fg) + c(dh - eg)$$

$$|A| = \begin{vmatrix} 6 & 1 & 1 \\ 4 & -2 & 5 \\ 2 & 8 & 7 \end{vmatrix} = -306$$

Mathematical Expressions

Y= e^{2x} in math

Y= exp(2*x) in matlab

$F=(1+\sqrt{5})/2$

Factoring Quadratics

- **$x^2 + 3x - 4$**
- (x+4)(x-1)
- Roots(x)

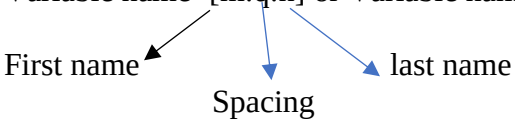
Row Vector
To create a column vector type the left squre bracket [and then enter the element with **Comma** between them or press the enter key after each element. type the right squre braket]after the last element

```
> t=[45,46,47,88]

t =

   45    46    47    88

>>
```

Column Vector
To create a column vector type the left squre bracket [and then enter the element with **semicolon** between them or press the enter key after each element. type the right squre braket]after the last element
Example :

```
>> t=[45; 46; 74; 88]

t =

   45
   46
   74
   88
```

Create a vector with constant sapcing by specifying the first term , the spacing and the last term :
Variable name=[m:q:n] or Variable name =m:q:n=First term



Examples

```
>> x=[1:2:10]

x =

    1     3     5     7     9

y=[20:-2:6]

y =

   20    18    16    14    12    10     8     6
```

Z=[0:0.1:1]

Z =

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Note :the last element in the vector will be the last number that does not exceed n.

n=[2:3:15]

n =

2 5 8 11 14

Create a vector with **equal spacing** by specify the first and last term and number of term

Variable name =linspace (xf,xl,n) xf is first element and xl last element and n number of elemens

a=linspace(-5,16,6)

a =

-5.0000 -0.8000 3.4000 7.6000 11.8000 16.0000

p=linspace(0,10,6)

p =

0 2 4 6 8 10

Create A two-Dimensional Array (Matrix)

A matrix is Created by typing the elements raw inside square brackets[type the left braket [then type the first row separating the elements with spaces or commas .To type the next row type a semicolon or press Enter .type the right bracket]at the end of last row .

Variable name =[1st row elements ;2nd elements ; 3rd row elements ;last elements]

A=[3 100 4 ; 5 8 10;22 5 20]

A =

3 100 4
5 8 10
22 5 20

Notes

- All the rows must have the same number of elements
- If an element is zero it has to be enter as such.
- The elements that are enter can be numbers or mathematical expressions that may include numbers predefined variables and functions.

```
d= 6; e=3 ; h=4 ;
>> M=[e, d*h, cos(pi/3); h^2, sqrt(h*h/d), 14]

Mat =

   3.0000   24.0000    0.5000
  16.0000    1.6330   14.0000
```

Rows of a matrix can be created from vectors

```
a=[2 3 4];
b=[4 12 7];
c=[9 12 4];
>> z=[a;b;c]

z =
    2     3     4
    4    12     7
    9    12     4
```

• Rows of a matrix can also be entered as vectors using the notation for creating vectors with constant spacing, or the linspace command.

```
>> A=[0:4:12; linspace(10,40,4); 67 2 43 68 ]

A =

    0     4     8    12
   10    20    30    40
   67     2    43    68
```

Creating matrices using the zeros, ones, eye and, rand Commands:
Example
Zr=zeros(2,3)

```
>> Zr=zeros(2,3)
Zr =
   0    0    0
   0    0    0
```

The ones (m,n)command creates a squre matrix with n row and n columns in which the diagonal the elements are equal to 1

```
V=ones(2,2)
V =

    1     1
    1     1
```

The eye (n) command creates a square matrix with n rows and n columns in which the diagonal elements are equal to 1 and the rest of the elements are 0

Example:

```
y=eye(3,3)
y =

    1     0     0
    0     1     0
    0     0     1
```

H.Ws:

1-Define the variables x = 0.85 , y = 12.5 , and then use them to create a column vector that has the following elements: y , yx , In (y/x) , and x+y.

2-Create a row vector with 11 equally spaced elements in which the first element is 96 and the last element is 2.

3- Create three row vectors: a = [3 -1 5 11 -4 2] , b = [7 -9 2 13 1 -2] , c = [-2 4 -7 8 0 9] Use the three vectors in a MATLAB command to create a 3 x 6 matrix in which the rows are the vectors a, b, and c

4)Create a row vector that has the following elements:

$$25.4, 2.1^2+11, 6!, 2.4^4, 0.03, \pi$$

The Transpose Operator:

The transpose operator (**'**), when applied to a vector, it switches a row vector into a column vector and vice versa. When applied to a matrix, it switches the rows to columns (columns to rows).

```
>> a=[1,2,3]

a =

    1    2    3

>> a'

ans =

    1
    2
    3
```

Array Addressing (Array Indexing):

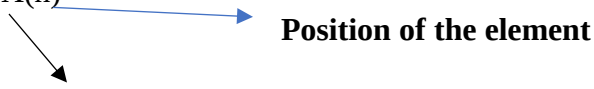
Array addressing is a means of recalling an element or a set of elements from the array based on their position in the array

Vector Addressing:

Vectors can be addressed by using the following formula:

X=[1,4,6,8]

X(n)



Position of the element

Vector name

```
>> v=[100,20,30,50,70,2^2]

v =

  100    20    30    50    70     4

>> v(4)

ans =

    50
```

Notes:

- It is possible to **change** the value of only one element of a vector by assigning a new value to a specific address. This is done by typing: v(k) = value.
- A single element can also be used as a variable in a mathematical expression.

>> t=[5;7;8;-1;3;9] ↲
t = 5 7 8 -1 3 9
>> t(4)=2 ↲ (**assigning a new value**)

t = 5 7 8 2 3 9
>> t(2)+t(5) ↲ (**Using elements in a mathematical expression)**
ans = 10
>> t(1)^t(4)+sqrt(t(6)) ↲
ans = 28

• A colon can be used to address a range of elements in a vector as follows:
v(:) → Refers to all the elements of the vector v (either a row or a column vector).
v(p:q) → Refers to elements p through q of the vector v.
• New vectors can be created from existing ones by using a range of elements.
>> u=[4 15 8 12 34 2 50] ↲
u = 4 15 8 12 34 2 50
>> u(:) ↲
ans = 4 15 8 12 34 2 50
>> u(3:6) ↲
ans = 8 12 34 2
>> p=u(1:3) ↲ **(creating new vector p)**
p = 4 15 8

H.W
1) Create a row vector with 11 elements such that: (Do not type the vector explicitly.)

V= 0 2 4 6 8 10 12 9 6 3 0

Matrix Addressing:

Matrices can be addressed by using the following formula:

Matrix_name=M(r,c)
where (**r**) is number of rows and (**c)** number if column

```
>> M=[3 11 6 5; 4 7 10 2; 13 9 0 8] ↲
M =
 3 11 6 5
 4 7 10 2
 13 9 0 8
```

To display the s[nd] row and th[rd] postion on Mtrix (M)

```
>> M(2,3)

ans =

   10
```

```
>> M

M =

    3    11     6     5
    4     7    10     2
   13     9     0     8
```

>> M(3,3)=20 ↲ (**assigning a new value**)

```
M =

    3    11     6     5
    4     7    10     2
   13     9    20     8
```

```
>> M(3,2)-M(1,4)

ans =

    4
```

Notes:

- A colon can be used to address a range of elements in a matrix as follows:
- **A(:)** → Refers to all the elements of the matrix **A**.
- **A(:,c)** → Refers to the elements in all the rows of column **c** of the matrix **A**.
- **A(r,:)** → Refers to the elements in all the columns of row **r** of the matrix **A**.
- **A(:,c:m)** → Refers to the elements in all the rows between columns **c** and **m** of the matrix **A.**
- **A(r:n,:)** → Refers to the elements in all the columns between rows **r** and **n** of the matrix **A**.
- **A(r:n,c:m)** → Refers to the elements in rows **r** through **n** and columns **c** through **m** of the matrix **A**.

>> A=[1 3 5 7 ; 2 4 6 8 ; 3 6 9 12] ↵

A =

1	3	**5**	7
2	4	**6**	8
3	6	**9**	12

>> A**(:,3**)

ans =

5
6
9

>> c=A(2,:)

c =

2 4 6 8

>> D=A(:,1:3)

D =

1 3 5
2 4 6
3 6 9

E=A(2:3,:)

E =

2	**4**	**6**	**8**
3	**6**	**9**	**12**

>> F=A(2:3,3:4)

F =

6	**8**
9	**12**

>> G= [A(2:3,1:2), A(1:2,3:4)]

G =

2	**4**	**5**	**7**
3	**6**	**6**	**8**

>>H= [A(2:3,1:2); A(1:2,3:4)]

H =

2	**4**
3	**6**
5	**7**
6	**8**

Adding Elements to Existing Arrays:

```
>> v=[2 4 6 8]

v =

    2     4     6     8

>> v(5)=10

v =

    2     4     6     8    10

>> v(9:12)=[6 -4 8 1]

v =

    2     4     6     8    10     0     0     0     6    -4     8     1
```

Adding elements to a matrix:

Rows and/or columns can be added to an existing matrix by assigning values to the new rows or columns. This can be done by assigning new values, or by appending existing variables. This must be done carefully since the size of the added rows or columns must fit the existing matrix.

```
>>  E=[1 5 8;-1 3 0;4 12 -2]

E =

    1     5     8
   -1     3     0
    4    12    -2

>> E(4,:)=[3 -4 16]

E =

    1     5     8
   -1     3     0
    4    12    -2
    3    -4    16
```

E(:,4)=[3 2 5 4]

E =

1	**5**	**8**	**3**
-1	**3**	**0**	**2**
4	**12**	**-2**	**5**
3	**-4**	**16**	**4**

Deleting Elements of an array:

An element, or a range of elements, of an existing array can be deleted by reassigning nothing to these elements. This is done by using square brackets with nothing typed in between them [ ].

>> t=[2 40 65 55 23 15 80]

t =

2 40 65 55 23 15 80

t(4)=[]

t =

2 40 65 23 15 80

Using Arrays in MATLAB Built-in Functions:

The built-in functions in MATLAB are written such that when the argument (input) is an array, the operation that is defined by the function is executed on each element of the array. The result (output) from such an operation is an array in which each element is calculated by entering the corresponding element of the argument (input) array into the function

```
>> x=[1 2 3 4 5]

x =

   1    2    3    4    5

>> factorial(x)

ans =

   1    2    6   24  120


>> a=0:pi/6:pi

a =

        0    0.5236    1.0472    1.5708    2.0944    2.6180    3.1416

>> b=sin(a)

b =

        0    0.5000    0.8660    1.0000    0.8660    0.5000    0.0000
```

Mathematical Operations with Arrays

• Addition and Subtraction:

Addition (+) and subtraction (-) can be used to add (subtract) arrays (vectors or matrices) of identical size (have the same numbers of rows and columns). In general, if A and B are two arrays (for example, 2 x 3 matrices) :

$$\mathbf{A}=\begin{matrix}\boldsymbol{A11} & \boldsymbol{A12} & \boldsymbol{A13}\\ \boldsymbol{A21} & \boldsymbol{A22} & \boldsymbol{A23}\end{matrix}$$

$$\mathbf{B}=\begin{matrix}\boldsymbol{B11} & \boldsymbol{B12} & \boldsymbol{B13}\\ \boldsymbol{B21} & \boldsymbol{B22} & \boldsymbol{B23}\end{matrix}$$

then the matrix that is obtained by adding A and B is:

$$A+B=\begin{bmatrix}A11+B11 & A12+B12 & A13+B13\\ A21+B21 & A22+B22 & A23+B23\end{bmatrix}$$

>> A=[5 -3 8; 9 2 10]

A =

```
   5   -3    8
   9    2   10
```

>> B=[10 7 4;-11 15 1]

B =

```
   10    7    4
  -11   15    1
```

>> C=A+B

C =

```
   15    4   12
   -2   17   11
```

>> E=[4 7;3 -2;5 10]

E =

4 7
3 -2
5 10

>> F=A+E
Matrix dimensions must agree.

• **Multiplication**:

The multiplication operation (*) is executed by MATLAB according to the rules of linear algebra. This means that if A and B are two matrices, the operation A* B can be carried out only if the number of columns in matrix A is equal to the number of rows in matrix B. The result is a matrix that has the same number of rows as A and the same number of columns as B.

C=A(m,n)*B(p,q)
n=p
C=(m,q)

For example, if A is a 4 x 3 matrix and B is a 3 x 2 matrix:

```
   1   4   3
A=2   6   1
   5   2   8
```

```
   5   4
B=1   3
   2   6
```

A(3,3)*B(3,2)

n=p=3
C=(3,2)=3*2

Exmple

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

A =

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

>> B=[5 4;1 3;2 6]

B =

```
   5   4
   1   3
   2   6
```

>> A*B

ans =

15 34
18 32
43 74

>> B*A
**Error using *
Inner matrix dimensions must agree.**

Notes:
- Two vectors can be multiplied only if they **have the same number** of elements, and one is a row vector and the other is a column vector.
- The multiplication of a row vector by a column vector gives a 1 x 1 matrix, which is a scalar.

>> p=[2 5 1]

p =

2 5 1

>> q=[3;5;8]

q =

3
5
8

>> res=p*q

res =

39 (1*1)

Notes
The result of the multiplication of two square matrices (A & B) is a square matrix of the same size. However, the multiplication of matrices is not commutative (A*B ≠ B*A)

• When an array is multiplied by a number (Scalar) each element in the array is multiplied by this number.

```
>> A=[2 5 7 0; 10 1 3 4; 6 2 11 5]

A =

    2    5    7    0
   10    1    3    4
    6    2   11    5

>> B=A*5

B =

   10   25   35    0
   50    5   15   20
   30   10   55   25
```

• **Division**: The division operation is more complex and can be explained with the help of the identity matrix, determinant and the inverse operation

Determinant of a matrix (|A|) : The determinant is a useful value that can be computed from the elements of a square matrix. The determinant of a matrix A is denoted |A|.

$|A|=\begin{matrix} a11 & a12 \\ a21 & 22 \end{matrix}$ a11*a22-a12*a21 for exmple $\begin{matrix} 6 & 5 \\ 3 & 9 \end{matrix}$

6*9-5*3=39

We can calculater the result by matlab by using fuction det(A)

```
>> A=[6 5;3 9]

A =

    6    5
    3    9

>> det(A)

ans =

   39
```

Inverse of a matrix (A^{-1}):

The inverse of a square matrix A, sometimes called a reciprocal matrix, is a matrix A^{-1} .

If $A=\begin{matrix} a & b \\ c & d \end{matrix}$ then $A^{-1} = \frac{1}{|A|} = \begin{matrix} d & -b \\ -c & a \end{matrix} = \frac{1}{a*d-b*c}$

For exmple $A=\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$ $A^{-1} = \frac{1}{1*4-2*3} * \begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix} = \frac{1}{-2} * \begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}$

In matlab

```
>>  A=[1 2;3 4]

A =

    1    2
    3    4

>> inv(A)

ans =

   -2.0000    1.0000
    1.5000   -0.5000
```

Not every matrix has an inverse.

- **A matrix has an inverse only if it is square and**
- **its determinant is not equal to zero** .

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

A =

    2    4    5
    6    8    9

>> inv(A)
Error using inv
Matrix must be square.
```

```
>> B=[2 1;4 2]

B =

    2    1
    4    2

>> det(B)

ans =

    0

>> c=inv(B)
Warning: Matrix is singular to working precision.

c =

  Inf   Inf
  Inf   Inf
```

If the matrix B is the inverse of the matrix A, when the two matrices are multiplied, the product is the identity matrix (I).

A * B = I

```
>> A=[2 1 4; 4 1 8; 2 -1 3];
>> B=inv(A);
>> A*B

ans =

    1    0    0
    0    1    0
    0    0    1
```

Solving Linear Equations by Matrix Operations:

Linear Equations can be solved by Matrix Operations (Right division, left division

1- Right division (/):

XA = B

X = B / A

or

X*A = B

$\mathbf{X*A*A^{-1} = B * A^{-1}}$

$\mathbf{X*I = B * A^{-1}}$

$\mathbf{X = B * A^{-1}}$

X=B*inv(A)

Example

$4x - 2y + 6z = 8$

$2x + 8y + 2z = 4$

$6x + 10y + 3z = 0$

Solution: Using the rules of linear algebra, the above system of equations can be solved by following methods:

1) **Right division & Inverse :**
   The above system of equations can be written in the matrix form
   XA = B as follows:

ملاحظة /ترتب معاملات(x) بشكل افقي وكذلك ال(y) و(z)

$$[X\ Y\ Z]\begin{bmatrix} 4 & 2 & 6 \\ -2 & 8 & 2 \\ 6 & 2 & 3 \end{bmatrix} = [8\ \ 4\ \ 0]$$

```
>> A=[4 2 6; -2 8 10; 6 2 3];
>> B=[8 4 0] ;
>> X=B/A

X =

   -1.8049    0.2927    2.6341
```
x y z

Or

```
>> X=B*inv(A)
```
↲ (Solving by using the inverse of A)

```
X = -1.8049  0.2927    2.6341
```

2) **Left division & Inverse** :
The above system of equations can also be written in the matrix form

AX = B

X = A \ B

Or
A*X = B
A^{-1} *A*X = A^{-1} * B
I*X = A^{-1} * B
X = A-1* B

X=inv(A)*B

Note use The transpose operator (')

$$\begin{matrix} X \\ [y] \\ z \end{matrix} \begin{matrix} 4 & -2 & 6 \\ 2 & 8 & 2 \\ 6 & 10 & 3 \end{matrix} = \begin{bmatrix} 8 \\ 4 \\ 0 \end{bmatrix}$$

>> A=**[4 2 6; -2 8 10; 6 2 3];**

>> A=A '

A =

```
   4   -2    6
   2    8    2
   6   10    3
```

```
>> B=B'

B =

     8
     4
     0

>> A\B

ans =

   -1.8049
    0.2927
    2.6341
```

Or

X=inv(A)*B

```
ans =

   -1.8049
    0.2927
    2.6341
```

H.W

$$-4x + z + 3y = -18.2$$
$$6y + 5x - 2z = -48.8$$
$$4.5z + 2x - 5y = 92.5$$

Element-by-Element Operations

Many situations require element-by-element operations. These operations are carried out on each of the elements of the array (or arrays). Addition and subtraction are by definition already element-by-element operations.

Notes:

- Element-by-element operations can be done only with arrays of the same size.
- Element-by-element multiplication, division, or exponentiation of two vectors or matrices is entered in MATLAB by typing a dot in front of the arithmetic operator

Symbol Description

.*	Element byelement multiplication
./	Element by element right division
.\	Element byelement left division
.^	Element byelement exponentiation

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

>> A.*B (Element by element ion multiplication)

>> A

A =

2 6 3

5 8 4

>> B

B =

1 4 10

3 2 7

>> A.*B

ans =

2 24 30

15 16 28

>> C=**A./B** ↲ (Element-by-element right division)

C =

2.0000 1.5000 0.3000

1.6667 4.0000 0.5714

>> C=A.\B ↲ (Element-by-element left division)

C =

0.5000 0.6667 3.3333

0.6000 0.2500 1.7500

>> C=**A.^B** ↲ (Element-by-element exponentiation)

C =

2 1296 59049

125 64 16384

>> X=[1:8]

x =

1 2 3 4 5 6 7 8

>> **y=x.^2+4*x**

y =

5 12 21 32 45 60 77 96

Problem:

Classroom Work

For the function y =$\frac{x^3 + 5x}{4x^2 - 10}$, calculate the value of y for the following values of x using element-by-element operations: 1, 3, 5, 7, 9, and 11

Solve)

y = -1.0000 1.6154 1.6667 2.0323 2.4650 2.9241

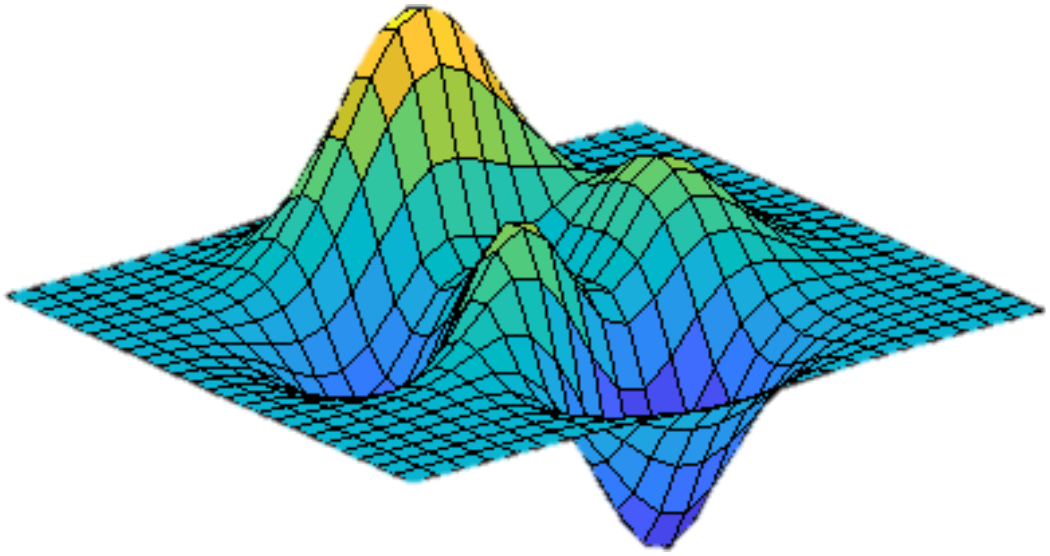
Problem

H.W

1)For the function y=$\frac{(x+7)^4}{(x+1)\sqrt{x}}$, calculate the value of y for the following values of x using element-by-element operations: 1.5 , 2.5 , 3.5 , 4.5 , 5.5 , 6.6.

2)Define x and y as the vectors x = [1 2 3 4 5] and y = [2 4 6 8 10]. Then use them in the following expressions to calculate z and w using element-by-element calculations.

$$Z=\frac{(x+y)^2}{x-y}$$



Chapter Three
Plotting in MATLAB

To plot the graph of a function, you need to take the following steps −

- Define **x**, by specifying the **range of values** for the variable **x**, for which the function is to be plotted
- Define the function, **y = f(x)**
- Call the **plot** command, as **plot(x, y)**
- Following example would demonstrate the concept. Let us plot the simple function **y = x** for the range of values for x from 0 to 100, with an increment of 5.
- Create a script file and type the following code −

```
x = [0:5:100];
y = x;
plot(x, y)
```

```
x = [1 2 3 4 5 6 7 8 9 10];
x = [-100:20:100];
y = x.^2;
plot(x, y)
```

Adding Title, Labels, Grid Lines and Scaling on the Graph

MATLAB allows you to add title, labels along the x-axis and y-axis, grid lines and also to adjust the axes to spruce up the graph.

- The **xlabel** and **ylabel** commands generate labels along x-axis and y-axis.
- The **title** command allows you to put a title on the graph.
- The **grid on** command allows you to put the grid lines on the graph.
- The **axis equal** command allows generating the plot with the same scale factors and the spaces on both axes.
- The **axis square** command generates a square plot.

Example

Create a script file and type the following code −

```
x = [0:0.01:10];
y = sin(x);
plot(x, y), xlabel('x'), ylabel('Sin(x)'), title('Sin(x) Graph'),
grid on, axis equal
```

Drawing Multiple Functions on the Same Graph

You can draw multiple graphs on the same plot. The following example demonstrates the concept −

Example

Create a script file and type the following code −

```
x = [0 : 0.01: 10];
y = sin(x);
g = cos(x);
plot(x, y, x, g, '.-'), legend('Sin(x)', 'Cos(x)')
```

Plots :

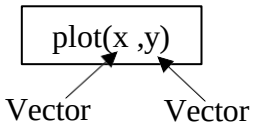
Plots are a very useful tool for presenting information. This is true in any field, but especially in science and engineering, where MATLAB is mostly used.

MATLAB has many commands that can be used for creating different types of plots such as: standard x-y plots, plots with logarithmic and semi-logarithmic axes, polar plots and three-dimensional mesh and surface plots, and many more.

This chapter describes how MATLAB can be used to create and format many types of two-dimensional plots and three-dimensional plots.

Two-dimensional Plots:

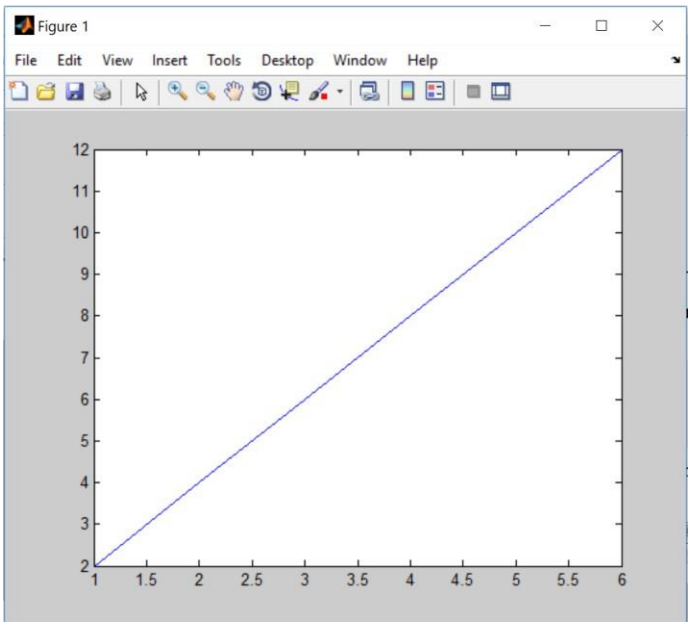
The ***plot*** command is used to create two-dimensional plots. The simplest form of the command is:



Note: The arguments x and y are each a vector (one-dimensional array). The two vectors must have the same number of elements.

Example:

```
>> x=[1  2  3  4  5  6];        ↵

>> y=[2  4  6  8  10  12];      ↵

>> plot(x,y)                    ↵
```



***Plot* Command with Line Specifiers:**

The plot appears on the screen in blue, which is the default line colour. The plot command has additional, optional arguments that can be used to specify the colour and style of the line and the colour and type of markers. With these options, the ***plot*** command has the following form:

```
Plot(x , y, 'line specifiers')
```

Line specifiers are:

- The line style specifiers are :

Line Style	Specifier
solid (default)	-
dashed	--
dotted	:
dash-dot	-.

- The line colour specifiers are:

Line Colour	Specifier
red	r
green	g
blue	b
cyan	c
magenta	m
yellow	y
black	k

- The marker type specifiers are:

Marker Type	Specifier
plus sign	+
circle	o
asterisk	*
point	.
cross	x
square	s
diamond	d
star	p

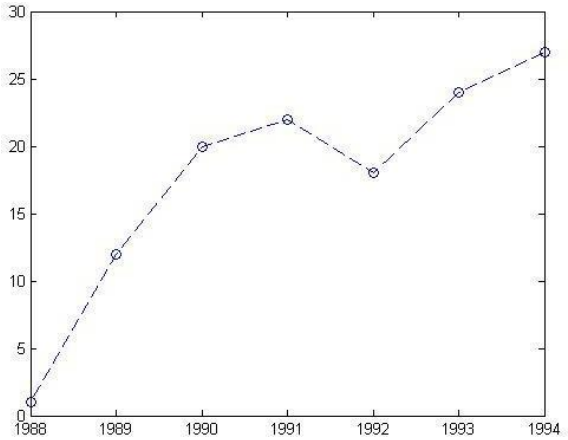
Notes:

- The specifiers are typed inside the plot command as strings using ‘ ‘.
- Within the string, the specifiers can be typed in any order.
- The specifiers are optional. This means that none, one, two, or all three types can be included in a command.

Example:

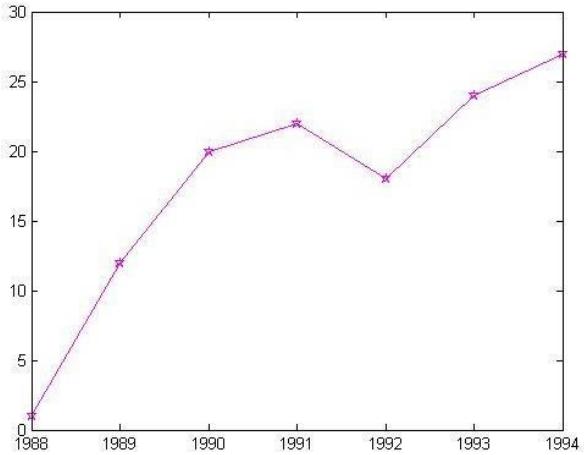
```
>> yr=[1988:1:1994];                ↲
>> sal=[1 12 20 22 18 24 27];       ↲

>> plot(yr,sal,'--o')               ↲
```

(Plotting with dashed line and circle type markers)




```
>>plot(yr,sal,'mp-')              ↲
```

(Plotting with solid magenta line and star type markers)



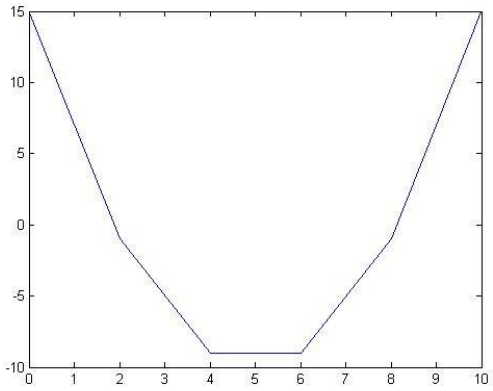
Plot of a Function:

In many situations, there is a need to plot a given function. This can be done in MATLAB by using the ***plot*** or the ***fplot*** command.

- **Using *plot* command:**

Example:

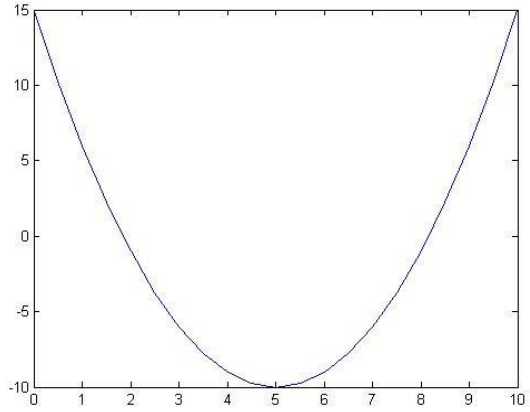
```
>> x=0:2:10                    ↵
x =
    0    2    4    6    8   10

>> y=x.^2-10*x+15              ↵
y =
   15   -1   -9   -9   -1   15

>> plot(x,y)                   ↵
```



Note: Since the plot is made up of segments of straight lines that connect the points, to obtain an accurate plot of a function, the spacing between the elements of the vector x must be appropriate.

Example:

```
>> x=0:0.5:10;                 ↵

>> y=x.^2-10*x+15;             ↵

>> plot(x,y)                   ↵
```



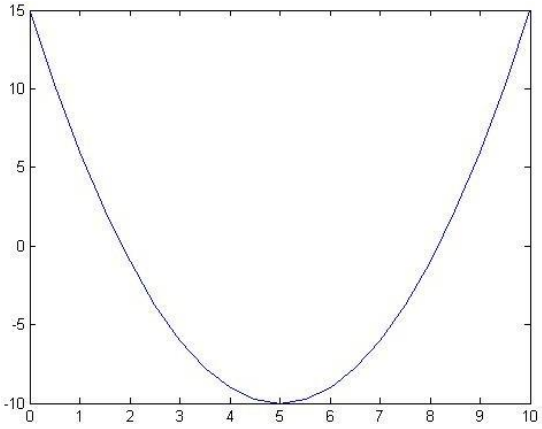
- **Using *fplot* command:**

  The ***fplot*** command plots a function with the form y = f(x) between specified limits. The command has the following form:

```
fplot( 'function', [ limits ] , 'line specifiers')
                                  optional
```

Example:

```
>> fplot( 'x^2-10*x+15' , [0  10])  ↵
```



Polar Plots:

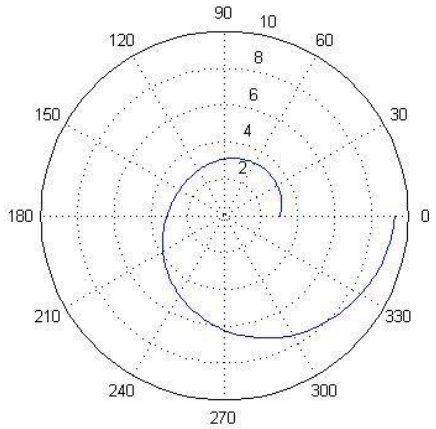
The ***polar*** command is used to plot functions in polar coordinates. The command has the following form:

```
polar( theta , radius , 'line specifiers')
```

vector vector optional

Example:

```
>> t=linspace(0,2*pi,200);  ↵
>> r=3*cos(0.5*t).^2+t;     ↵
>> polar(t,r)               ↵
```



Plots with Logarithmic Axes:

Many science and engineering applications require plots in which one or both axes have a logarithmic scale. MATLAB commands for making plots with log axes are:

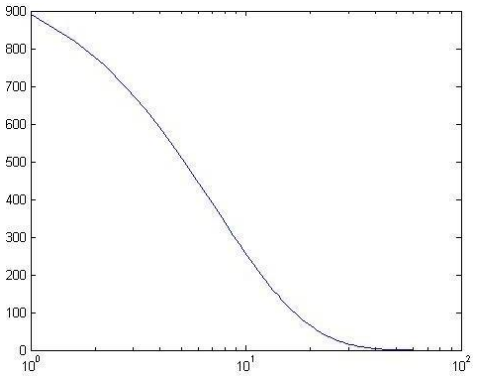
Commands	Description
semilogx(x, y)	Plots y versus x with a log (base 10) scale for the x axis and linear scale for the y axis.
semilogy(x, y)	Plots y versus x with a log (base 10) scale for the y axis and linear scale for the x axis.
loglog(x, y)	Plots y versus x with a log (base 10) scale for both axes.

Notes:

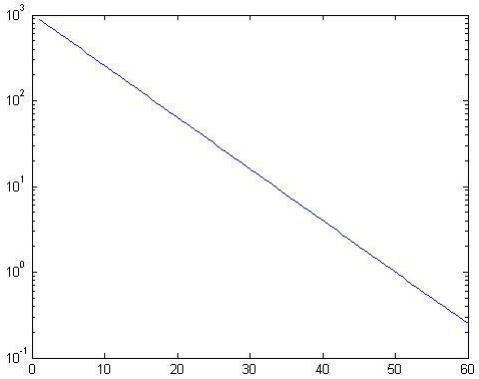
- The number zero cannot be plotted on a log scale (since a log of zero is not defined).
- Negative numbers cannot be plotted on log scales (since a log of a negative number is not defined).

Example:

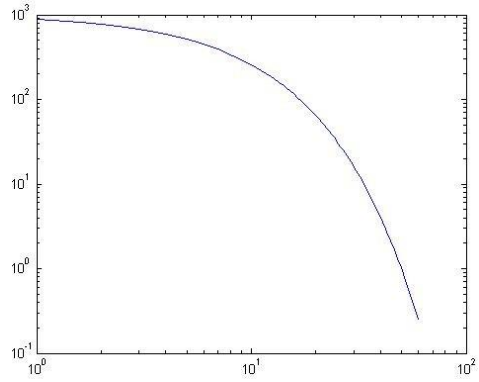
```
>> x=linspace(1,60,100);        ↲
>> y=2.^(-0.2*x+10);            ↲
>> semilogx(x,y)                ↲
```



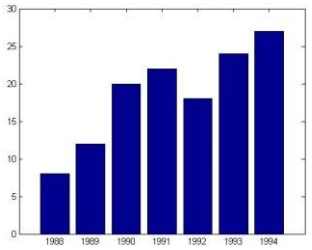
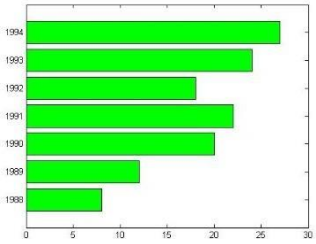
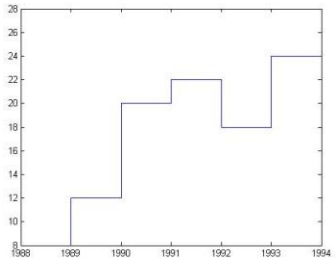
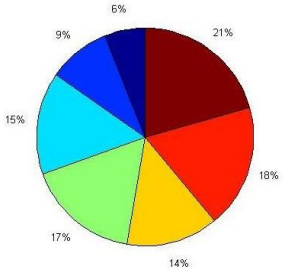
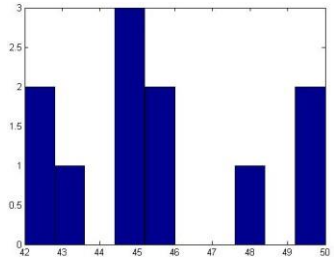

```
>> semilogy(x,y)                ↲
```




```
>> loglog(x,y)                  ↲
```



Plots with Special Graphics:

Plots	Function	Example	Figure
Vertical Bar Plot	bar(x,y)	>> yr=[1988:1994]; >> sal=[8 12 20 22 18 24 27]; >> bar(yr,sal)	
Horizontal Bar Plot	barh(x,y)	>> barh(yr,sal,'g')	
Stairs Plot	stairs(x,y)	>> stairs(yr,sal)	
Pie Plot	pie(x)	>>pie(sal)	
Histograms	hist(y)	>> temp=[45 46 50 42 43 42 45 48 50 46 45]; >> hist(temp)	

Three-dimensional Plots:

Three-dimensional (3-D) plots can be a useful way to present data that consists of more than two variables. MATLAB provides various options for displaying three-dimensional data. They include line, surface, mesh plots, and many others.

Line Plots:

A three-dimensional line plot is a line that is obtained by connecting points in three-dimensional space. A basic 3D plot is created with the ***plot3*** command, which is very similar to the plot command and has the following form:

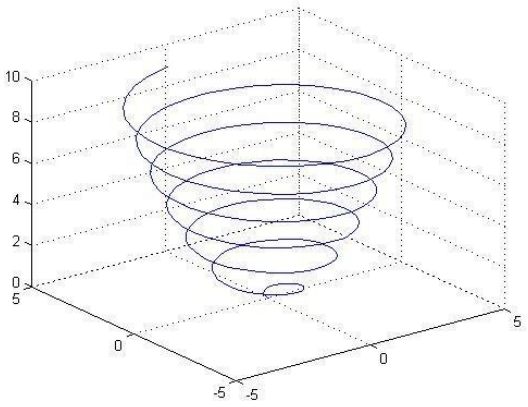
```
plot3(x, y, z, 'line specifiers')
                  optional
```

Note:

- x, y, z are vectors of the coordinates of the points.
- The three vectors with the coordinates of the data points must have the same number of elements.

Example:

```
>> t=0:0.1:6*pi;                    ↲
>> x=sqrt(t).*sin(2*t);             ↲
>> y=sqrt(t).*cos(2*t);             ↲
>> z=0.5*t;                         ↲
>> plot3(x,y,z)                     ↲
>> grid on                  ↲ (display grids)
```



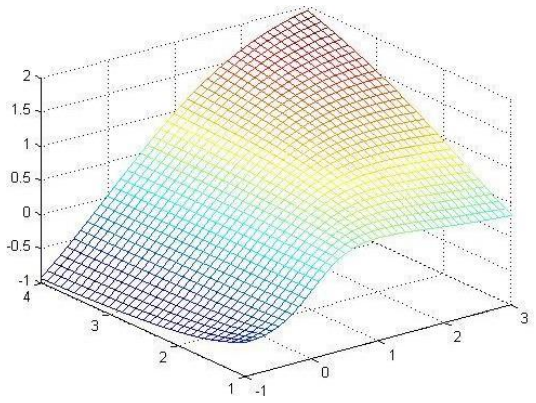
Mesh and Surface Plots:

Mesh and surface plots are three-dimensional plots used for plotting functions of the form z = f(x,y). Mesh and surface plots are created in three steps.

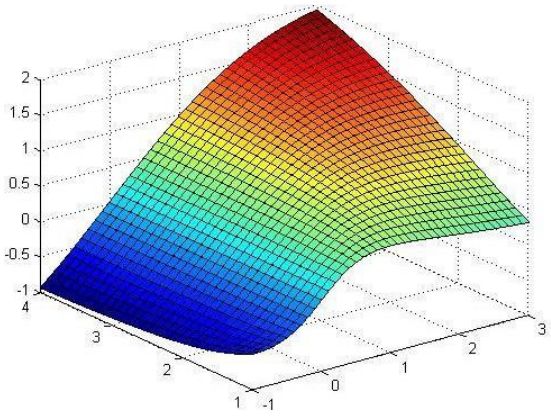
- 1st : Create a grid in the x y plane using meshgrid function.
- 2nd: Calculate the value of z at each point of the grid.
- 3rd: Create the plot using mesh and surf functions.

Example: Plotting the function $z = \frac{xy^2}{x^2+y^2}$ over the domain -1 ≤ x ≤ 3 and 1 ≤ y ≤ 4.

```
>> x=-1:0.1:3;                    ↲
>> y=1:0.1:4;                     ↲

>> [X,Y]=meshgrid(x,y);           ↲ (1st step)

>> Z=X.*Y.^2./(X.^2+Y.^2);        ↲ (2nd step)

>> mesh(X,Y,Z)                    ↲ (3rd step)




>> surf(X,Y,Z)                    ↲
```



Mesh plot of function z



Surface plot of function z

Plotting Multiple Graphs in The Same Plot:

In many situations, there is a need to make several graphs in the same plot. This is can be done by :

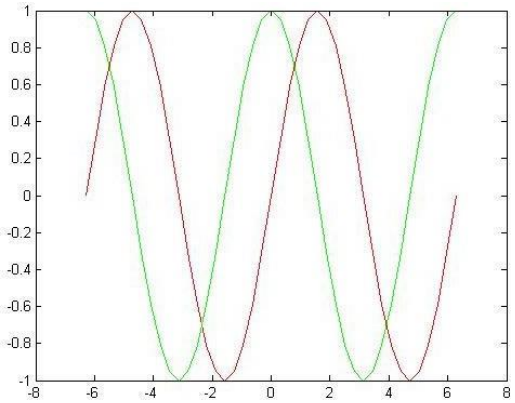
- **Using the *plot* Command:**

  Two or more graphs can be created in the same plot by typing pairs of vectors inside the ***plot*** command as following:

```
plot( x, y, u, v, t, h )
```

Example:

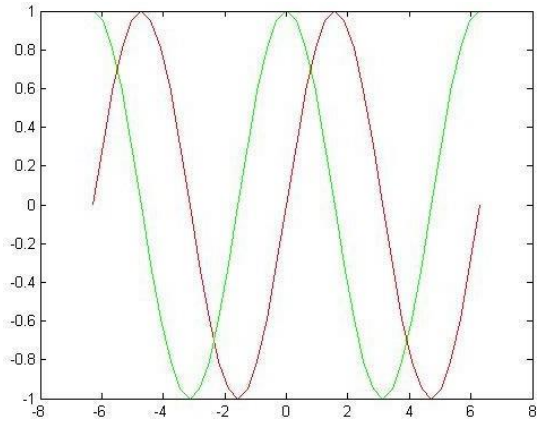
```
>> x=[-2*pi : pi/10 : 2*pi];
>> y1=sin(x);
>> y2=cos(x);
>> plot(x,y1,'r',x,y2,'g')
```



- **Using the *hold on* and *hold off* Commands:**

Example:

```
>> x=[-2*pi : pi/10 : 2*pi];
>> y1=sin(x);
>> y2=cos(x);
>> plot(x,y1,'r')
>> hold on
>> plot(x,y2,'g')
>> hold off
```



<u>Creating Multiple Plots on The Same Window:</u>

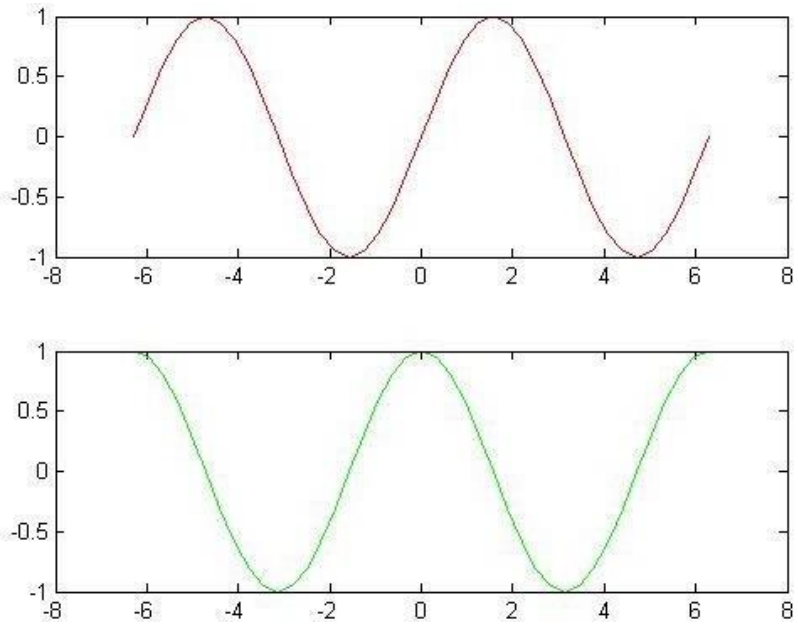
Multiple plots can be created on the same window with the ***subplot*** command, which has the form bellow:

```
subplot(m, n, p)
```

This command divides the figure window into m *(rows)* × n *(columns)* equal sized regions, and selects the *p*th region to receive all plotting commands. The subplots are numbered from left to right and from top to bottom.

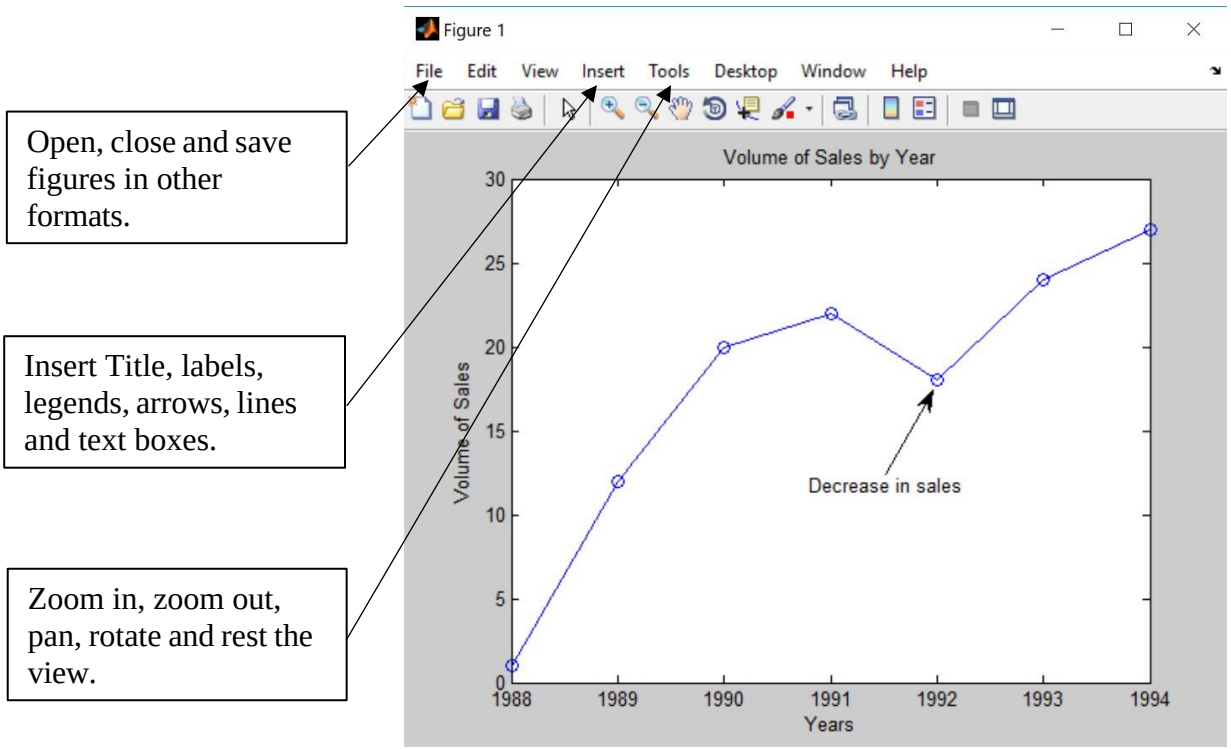
Example:

```
>> x=[-2*pi : pi/10 : 2*pi];
>> y1=sin(x);
>> y2=cos(x);
>> subplot(2,1,1)
>> plot(x,y1,'r')
>> subplot(2,1,2)
>> plot(x,y2,'g')
```



Formatting a Plot:

A plot can be formatted interactively in the figure window by clicking on the plot and/or using the menus.



H.w's:

2) Use the **fplot** command to plot the function

$$f(x) = (3cosx - sinx)e^{-0.2x}$$ in the domain $-4 \le x \le 9$.

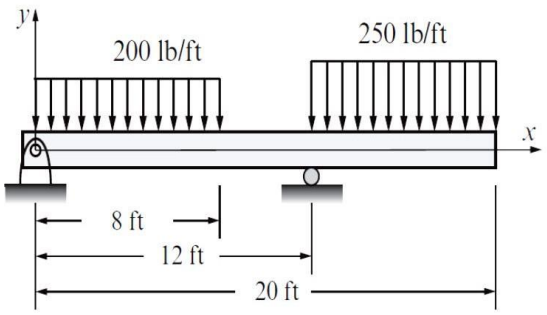
3) Plot the function $f(x) = -3x^4 + 10x^2 - 3$ and its derivative for $-4 \le x \le 3$ in one figure. Plot the function with a red solid line, and the derivative with a green dashed line.

4) Make a polar plot of the function $r = 2\sin(3\theta)\, sin\theta$ for $0 \le \theta \le 2\pi$.

5) Make a 3D mesh plot of the function $z = \frac{y^2}{4} - 2\sin(1.5x)$ in the domain $-3 \le x \le 3$ and $-3 \le y \le 3$.

6) Make a 3D surface plot of the function $= \frac{-\cos(2R)}{e^{0.2R}}$, where $R = \sqrt{x^2 + y^2}$ in the domain $-5 \le x \le 5$ and $-5 \le y \le 5$.

7) A simply supported beam is loaded as shown. The shear force V and bending moment M as a function of x are given by the following equations:



$V(x) = 400 - 200x$ lb

$M(x) = -100x^2 + 400x$ lb-ft

for $0 \le x \le 8$ ft.

$V(x) = -1,200$ lb, $M(x) = -1,200x + 6,400$ lb-ft for $8 \le x \le 12$ ft

$V(x) = -250x + 5,000$ lb, $M(x) = -125(x-12)^2 + 2,000x - 32,000$ lb-ft

for $12 \le x \le 20$ ft.

Plot the shear force and the bending moment as a function of x (two figures on one page such that the shear force diagram is displayed above the bending moment diagram).

Programming in MATLAB

A program is a sequence of commands. In a simple program the commands are executed

one after the other in the order they are typed. Many situations, however, require more

sophisticated programs in which commands are not necessarily executed in the order

they are typed, or different commands (or groups of commands) are executed when the

program runs with different input variables.

In this chapter we will learn how to write programs in MATLAB and how to control

the flow of these programs.

Script Files:

So far all the commands were typed in the Command Window and were executed when the Enter key was pressed. But the commands in the Command Window cannot be saved and executed again. In addition, the Command Window is not interactive.

This means that every time the Enter key is pressed only the last command is executed, and everything executed before is unchanged.A better way of executing MATLAB commands is by using Script Files (which is a

list of MATLAB commands that can be edited and executed many times).

Notes About Script Files:

- **A script file is opened from (File → New → Script) or (Ctrl+N).**
- **A script file is executed by (save and run button) or (F5).**
- **A script file is also executed from command window by typing its name and**

press Enter.

• When a script file is executed, the output is displayed in the Command Window.

• Script files are also called M-files in older versions of MATLAB.

All previous examples can be executed by using script file:

Example : Define the variable x as x = 6.7, then evaluate:

$y = 0.01x^5 - 1.4x^3 + 80x + 16.7$

Sol:

(open a script file and type the following commands)

```
x=6.7;
y=0.01*x^5-1.4*x^3+80*x+16.7
(save and run)
```

(Results in the command window)

y =

266.6443

Example: For the function $y = \frac{x^3 + 5x}{4x^2 - 10}$

Calculate the value of y for the following

values of x using element-by-element operations: 1 , 3 , 5 , 7 , 9 , 11. Then plot x versus y.

Sol:

(open a script file)

```
x=[1:2:11]
y=(x.^3 + 5*x)./(4*x.^2 - 10)
plot(x,y,'r:')
(save and run)
```

(Results in the command window)

x =

1 3 5 7 9 11

y =

-1.0000 1.6154 1.6667 2.0323 2.4650 2.9241

The Input Command:

There are situations that require asking the user to enter some sort of information into

the program and to save that information into a variable. This is done by using the input

command which has the following form:

variable name =input('prompt message')

When the input command is executed, the message is displayed in the Command

Window prompting the user to enter a value that is assigned to the variable. The user

types the value and presses the Enter key. This assigns the value to the variable.

Example:

(open a script file)

```
x=[1:2:11]
y=(x.^3 + 5*x)./(4*x.^2 - 10)
plot(x,y,'r:')
```

(save and run)

variable name =input('prompt message')

```
% MATLAB Program to find the area of a triangle
b=input('Enter base of the triangle:')
h=input('Enter height of the triangle:')
area=0.5*b*h
Enter base of the triangle:10 ↲
b =
 10
Enter height of the triangle:15 ↲
h =
 15
area =
 75
```

Note: The input command can also be used to assign a string to a variable. This is can be done by adding an option in the input command that defines the characters that are entered as a string. Then the form of the input command will be as followings:

```
variable name =input('prompt message' , 's')

name=input('What is your name?','s')
```

Output Commands:

MATLAB automatically generates a display when some commands are executed the output is not displayed if a semicolon is typed at the end of the command). In additionto this automatic display, MATLAB has several commands that can be used to generate displays.

The display Command:

The display command is used to display the value of a variable with its name,

and to display text. The format of the display command is:

Note: To display the value of a variable without its name, the disp command is used instead of display.

Output Commands:

• The display Command:

The display command is used to display the value of a variable with its name,

and to display text. The format of the display command is:

Note: To display the value of a variable without its name, the disp command is used

instead of display.

```
% This program converts a length in feet to meters
% 1ft=0.305m
ft=input('Enter the length in feet (ft)? ');
m=ft*0.305;
```

```
display(' ')
display('length in feet (ft)')
disp(ft)
display('length in meters (m)')
disp(m)
```

Solve

Enter the length in feet (ft)? 45 ↲

length in feet (ft) 45

length in meters (m) 13.7250

The fprintf Command:

The fprintf Command used to display a mix of text and a number (value of a variable) in the same line, the fprintf command has the form:

fprintf ('text as string %g additional text' , variable Value)

The spot where the text is inserted

```
x=input('Enter the value of x ?');
y=x^2+x-4;
```

```
fprintf('The value of y is equal to %g',y)
```

Enter the value of x ?5 ↲

The value of y is equal to 26

Relational and logical operators

are operators that produce a true or false result. These operators are very important, because they control which code gets executed in MATLAB program.

- Relational Operators:

A relational operator compares two numbers (e.g., 5 < 8) by determining whether a comparison statement is true or false. If the statement is true, it is assigned a value of 1. If the statement is false, it is assigned a value of 0

Relational operator	Description	Example	
<	Less than	>> 3 < 4 ↲ ans =1	
<=	Less than or equal to	>> 4 <= 4 ↲ ans =1	
>	Greater than	>> 3 > 4 ↲ ans =0	
>=	Greater than or equal to	>> 5>=4 ↲ ans =1	
==	== Equal to	>> 3 == 4 ↲ ans =0	
~=	Not Equal to	>> ~5=4 ↲ ans = 1	

Notes: • Equal to (==) relational operator consists of two= signs (with no space between them), since one = sign is the assignment operator. • There is no space between the relational operators that consist of two characters (<=,>=, ~=). • Relational operators are used as arithmetic operators within a mathematical expression.

Example:

```
>>y=(6<10)+(7>8)+(5*3==60/4)
```

y = 2

Conditional Statements:

A conditional statement is a command that allows MATLAB to make a decision of whether to execute a group of commands that follow the conditional statement, or to skip these commands.

There are two types of conditional statements the

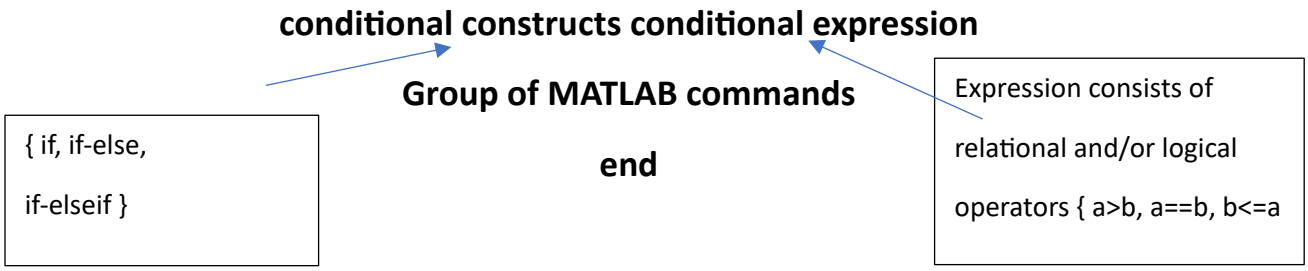
if statement and the switch-case

statement.

1-The if Statement:

In a conditional if statement, a conditional expression is stated. If the expression is true ,a group of commands that follow the statement are executed. If the expression is false ,the computer skips the group.

The basic form of a conditional if statement is:

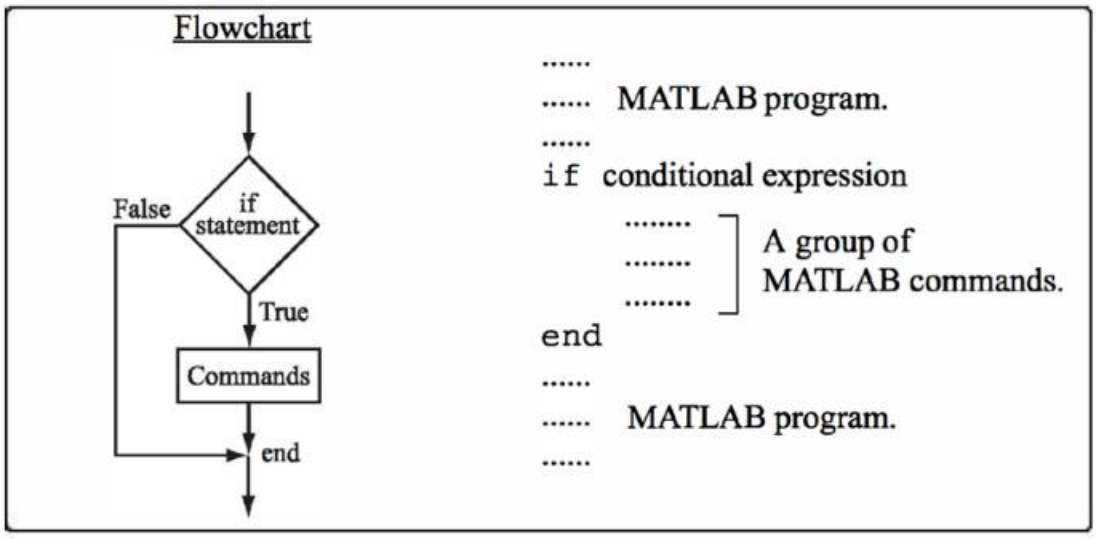


The if statement is commonly used in three structures,

- **if- end,**
- **if -else-end,**
- **if -elseif -else-end.**

• The if- end Structure:

The structure of the if- end conditional statement is shown in the figure below



As the program executes, it reaches the if statement. If the conditional expression in the

- **if statement is true (1), the program continues to execute the commands that follow the the end.**

- **if statement. If the conditional expression is false (0), the program skips the group of commands between the if and the end, and**
- **continues with the commands that follow**

Example % MATLAB program to test whether a number is positive or not

```
x=input('Enter a number: ');
if x>0
 display('The number is positive')
end
```

Problem:

A worker is paid according to his hourly wage up to 40 hours, and 50% more for overtime. Write a program in a script file that calculates the pay to a worker. The program asks the user to enter the number of hours (h) and the hourly wage (w). The program then displays the pay.

Solution

- **working hours ≤ 40 → pay = h*w**
- **working hours > 40 → pay=h*w+(h-40)*0.5*w**

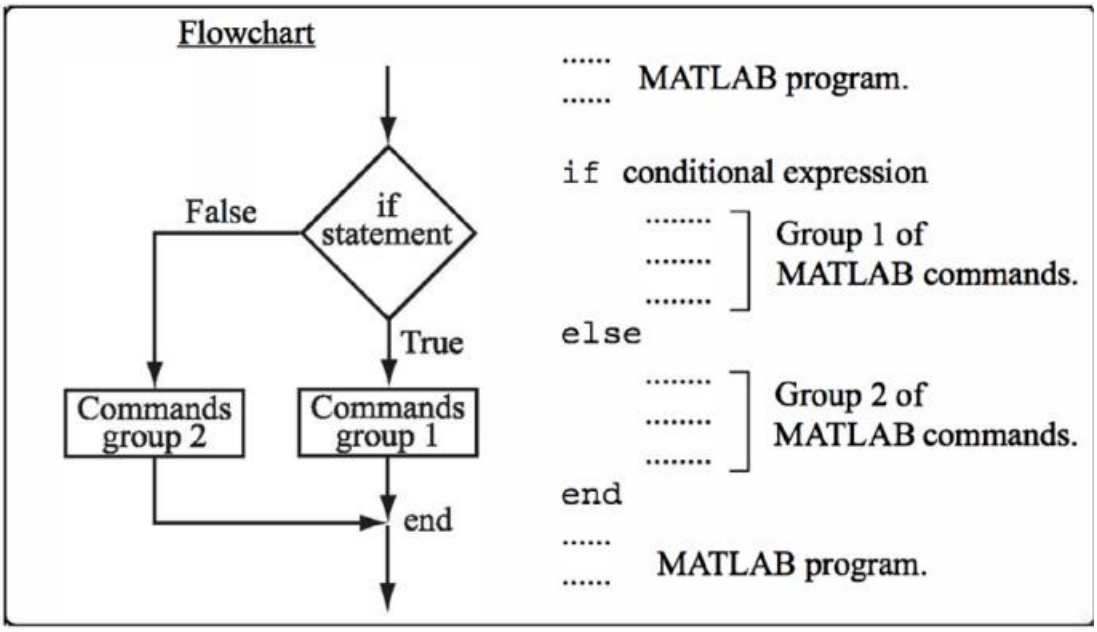
% MATLAB program that calculates the pay to a worker.

```
h=input('Please enter the number of hours worked: ');
w=input('Please enter the hourly wage in $: ');
pay=h*w;
if h>40
pay=h*w+(h-40)*0.5*w;
end
display(' ')
fprintf('The worker''s pay is $ %g\n',pay)
```

```
Please enter the number of hours worked: 35 ↲
Please enter the hourly wage in $: 8 ↲
The worker's pay is $ 280
(Re execute)
Please enter the number of hours worked: 50 ↲
Please enter the hourly wage in $: 10 ↲

The worker's pay is $ 550
```

• The if - else-end Structure:

The if - else-end structure is shown in Figure below:



As the program executes, if the conditional expression is true, the program executes group 1 of commands between the if and the else statements and then skips to the end. If the conditional expression is false, the program skips to the else and then executes group 2 of commands between the else and the end

```
% MATLAB program to test whether an integer number is
% even or odd
x=input('Enter an integer number:');
if rem(x,2)==0
 display('The number is even')
else
 display('The number is odd')
end
```

Enter an integer number:12 ↲

The number is even

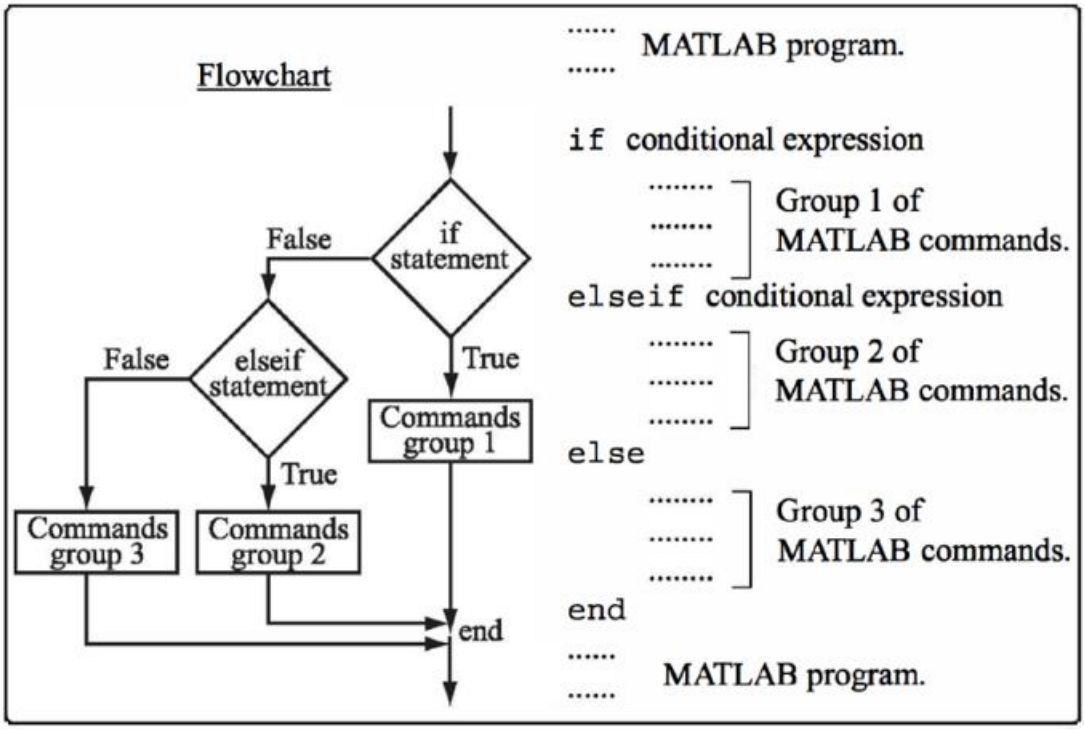
(Re execute)

Enter an integer number:3 ↲

The number is odd

• The if -elseif -else-end Structure:

The if -elseif -else-end structure is shown in Figure below:



If the conditional expression is true, the program executes group 1 of commands between the if and the elseif statements and then skips to the end. If the conditional expression in the if statement is false, the program skips to the elseif statement. If the conditional expression in the elseifstatement is true, the program executes group 2 of commands between the elseif and the else and then skips to the end. If the conditional expression in the elseif statement is false, the program skips to the else and executes group 3 of commands between the else and the end.

Notes:

- **Several elseif statements can be added.**
- **The else statement is optional**

```
% MATLAB program to check number is positive, negative
% or zero
x=input('Enter a number: ');
if x==0
 display('The number is zero')
elseif x>0
 display('The number is positive')
else
 display('The number is negative')
end
```

```
Enter a number: 4 ↲
The number is positive
(Re execute)
Enter a number: -6.5 ↲
The number is negative
```

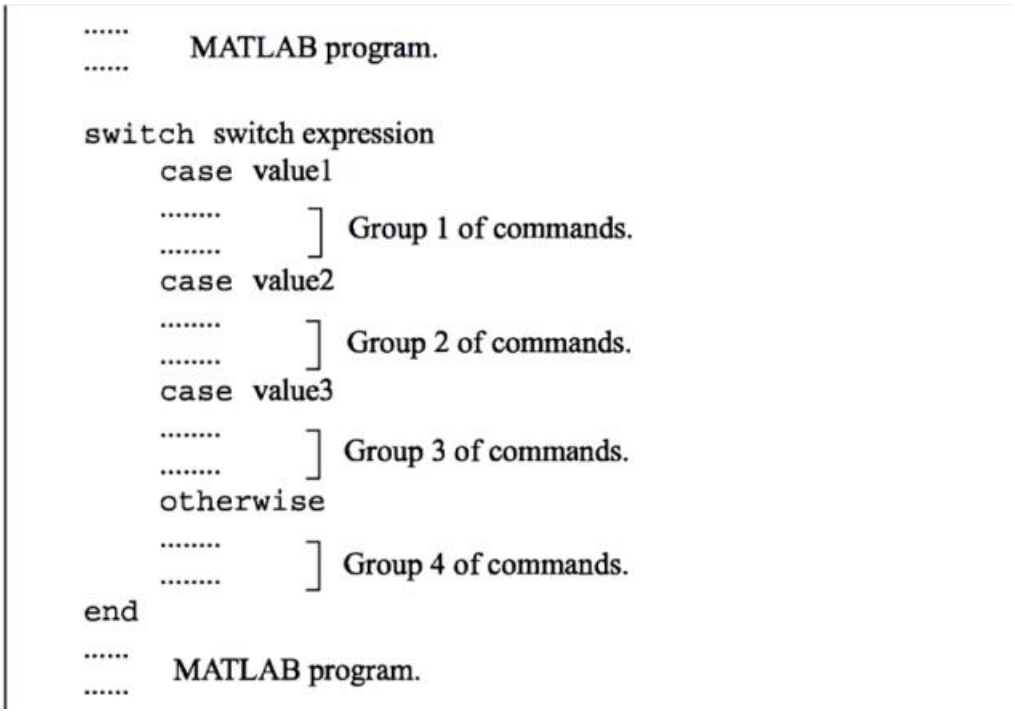
(Re execute)

Enter a number: 0 ↲

The number is zero

The switch-case statement:

The switch-case statement is another method that can be used to direct the flow of a program. The structure of the statement is shown in Figure below:



As the program executes, The value of the switch expression in the switch command is compared with the values that are next to each of the case statements. If a match is found, the group of commands that follow the case statement with the match are executed. If no match is found and the otherwise statement (which is optional) is present, the group of commands between otherwise and end is executed

% MATLAB program to check number is positive, negative % or zero

```
x=input('Enter a number: ');
s=sign(x);
switch s
 case 0
 display('The number is zero')
 case 1
 display('The number is positive')
 case -1
 display('The number is negative')
end
```

Enter a number: 24 ↲

The number is positive

(Re execute)

Enter a number: -15.5 ↲

The number is negative

(Re execute)

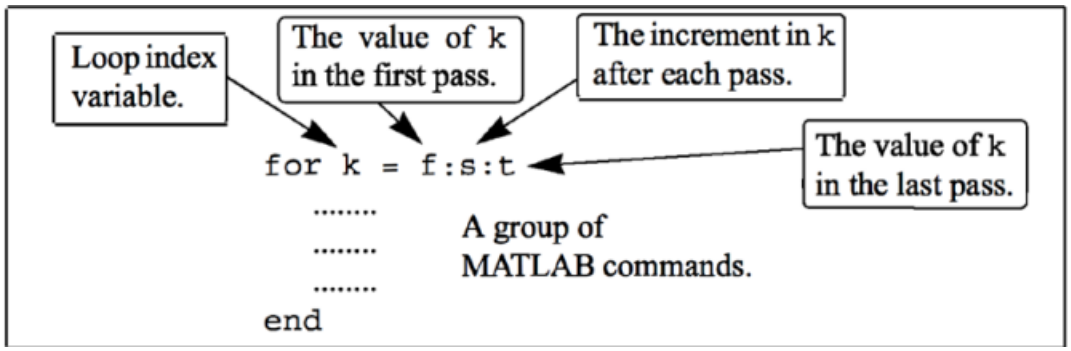
Enter a number: 0 ↲

The number is zero

Loop Statements:

There may be a situation when you need to execute a block of code many times. A loop statement allows us to execute a command or group of commands several times. MATLAB has two kinds of loops: for-end loops and while-end loops.

• for-end Loops: In for-end loops the execution of a command, or a group of commands, is repeated a predetermined number of times. The form of a for-end loop is shown in the Figure below:



This is an example of for loop statement

```
for x=1:3:10
y=x^2
end
```

y= 1

y = 16

y = 49

y = 100

Problem: Use a for-end loop in a script file to calculate the sum of the first n terms of the series:

$\sum_{i=0}^{n} \frac{(-1)^k k}{2^k}$

Execute the script file for n = 20

% MATLAB program to find the sum of series

```
n=input('Enter the number of terms:');
s=


0;
for k=1:n
 s=s+(-1)^k*k/(2^k);
end
fprintf('The sum of the series is: %g\n',s)
```

Enter the number of terms:20

The sum of the series is: -0.222216

Evaluate the following expressions without using MATLAB. Check the answers with MATLAB.

(*a*) $12 - 4 < 5 \times 3$ (*b*) $y = 8/4 > 6 \times 3 - 4^2 > -3$

(*c*) $y = -3 < (8 - 12) + 2 \times (5 > 18/6 - 4)^2$