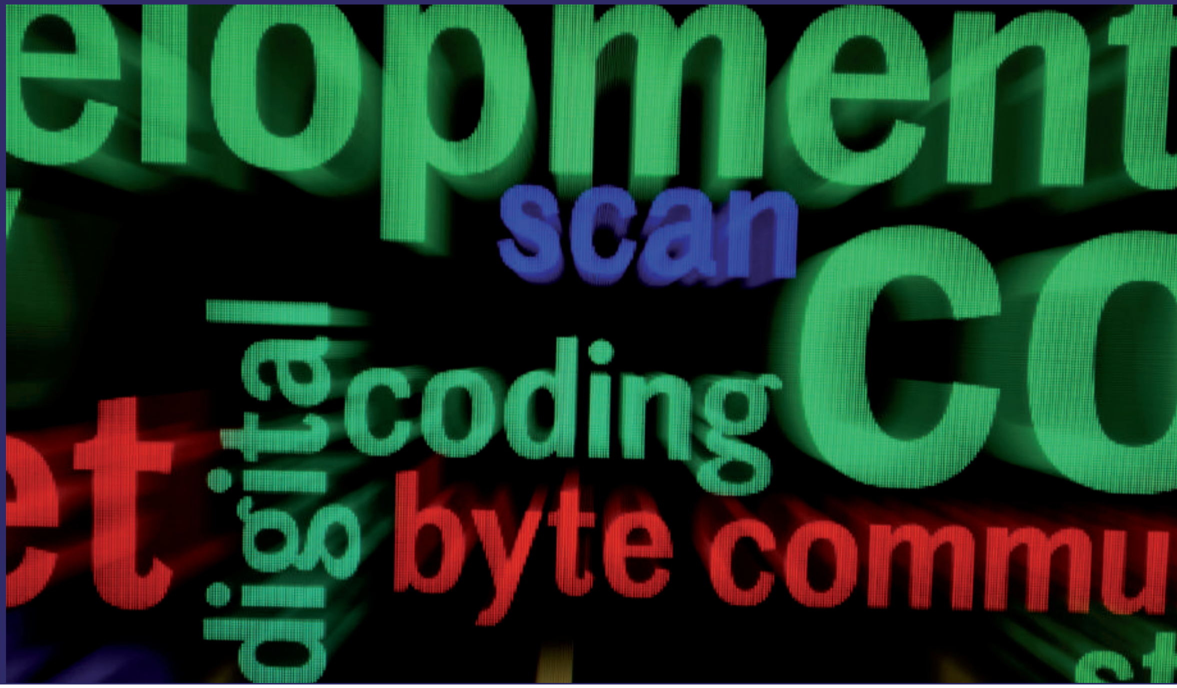
In this book, an introduction of the Technical computing language MATLAB, for mechanical engineering students was introduced. Many of engineers and scientists use MATLAB to analyze and design the systems and products transforming the world, where the matrix- based MATLAB language is the most world's most natural way to express computational mathematics. The main objective of this book is to provide the students with the opportunity to improve their skills using MATLAB environment to implement algorithms and to teach the use of MATLAB as a tool in solving problems in mechanical engineering. This book includes the coverage of basics of MATLAB and applications of MATLAB software to solve problems in statics and dynamics mechanical systems.

Enass H. Flaieh, 1975, Baghdad-Iraq, Single, Female, B. Eng. 1997, Mechanical Engineering, M. Sc. 2005, Applied Mechanics, Ph. D. 2015, Applied Mechanics. Member of Iraqi Engineers Association.

978-3-659-90366-3

MATLAB FOR MECHANICAL ENGINEERING

H. Flaieh, J. Dawood, Jaafer Habeeb



Enass H. Flaieh
Al-Khafaji Ali J. Dawood
Laith Jaafer Habeeb

Matlab for Mechanical Engineering

Beginner and Intermediate level

LAP LAMBERT Academic Publishing

Enass H. Flaieh
Al-Khafaji Ali J. Dawood
Laith Jaafer Habeeb

Matlab for Mechanical Engineering

Enass H. Flaieh
Al-Khafaji Ali J. Dawood
Laith Jaafer Habeeb

Matlab for Mechanical Engineering

Beginner and Intermediate level

LAP LAMBERT Academic Publishing

Impressum / Imprint
Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Bibliographic information published by the Deutsche Nationalbibliothek: The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Coverbild / Cover image: www.ingimage.com

Verlag / Publisher:
LAP LAMBERT Academic Publishing
ist ein Imprint der / is a trademark of
OmniScriptum GmbH & Co. KG
Bahnhofstraße 28, 66111 Saarbrücken, Deutschland / Germany
Email: info@omniscriptum.com

Herstellung: siehe letzte Seite /
Printed at: see last page
ISBN: 978-3-659-90366-3

MATLAB FOR MECHANICAL ENGINEERING

Beginner and Intermediate level

By

Enass H. Flaieh

Al-Khafaji Ali J. Dawood

Laith Jaafer Habeeb

Acknowledgement

The authors thank MATLAB staff on their wonderful software package and we hope that it spread widely around the world. MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

In 2004, MATLAB had around one million users across industry and academia. MATLAB users come from various backgrounds of engineering, science, and economics.

Contents

Chapter Four
2D Plotting

Chapter Five
Decision Making

Chapter Six
Loops

Chapter Seven
Applications

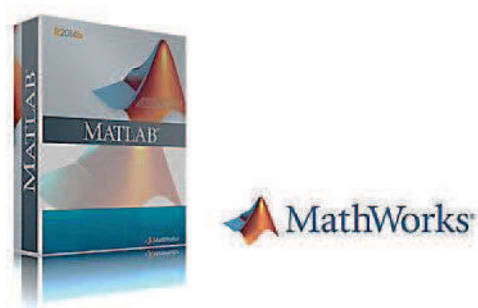
Chapter One: Installation MATLAB

1-1 Instruction

Using the MathWorks installer, you can install and activate MathWorks products on a computer running for Microsoft Windows operating system (32-bit or 64-bit).

Step 1: Start the Installer

Insert MATLAB CD in CD drive connected to your system. The installer usually starts automatically.



Step 2: Install With a File Installation Key

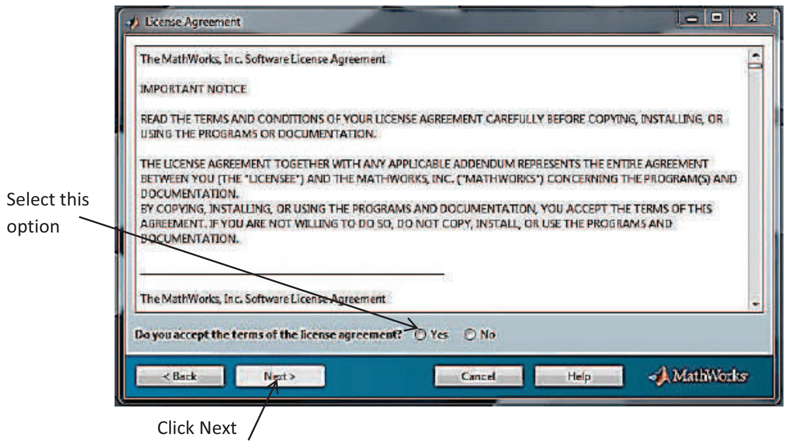
After running the CD, we get this dialog box, we select the "**Use a File Installation Key**" option (No Internet connection required) and click "**Next**".

Select this option

Click Next

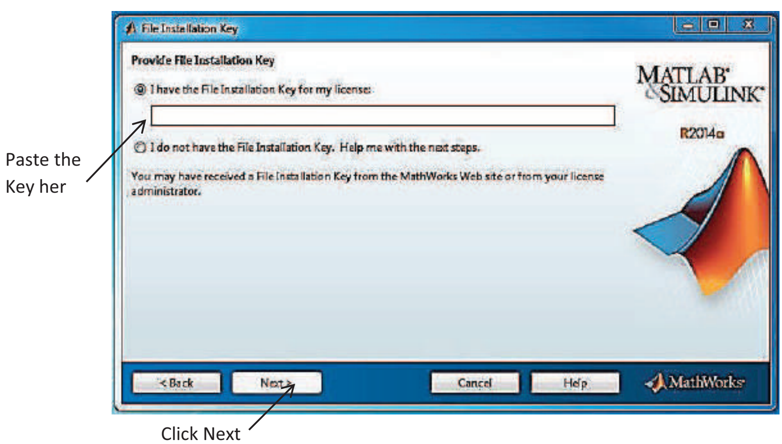
Step 3: Review the License Agreement

Review the software license agreement and, if you agree with the terms, select “**Yes”** and click “**Next**”.



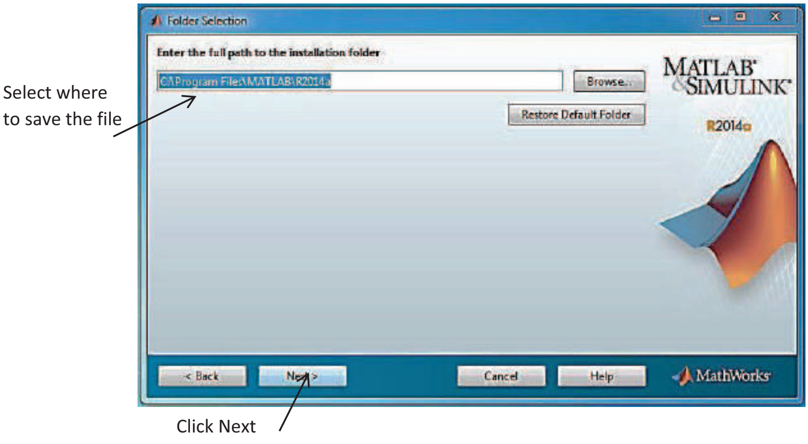
Step 4: Specify the File Installation Key

In this stage, we get this dialog box; the installer displays the File Installation Key dialog box. A File Installation Key identifies the products you can install. Select the “**I have the File Installation Key for my license**” option, enter the File Installation Key (You can get key from MATLAB CD > Serial > Key) copy the key and paste it in the key box and then click “**Next**”.



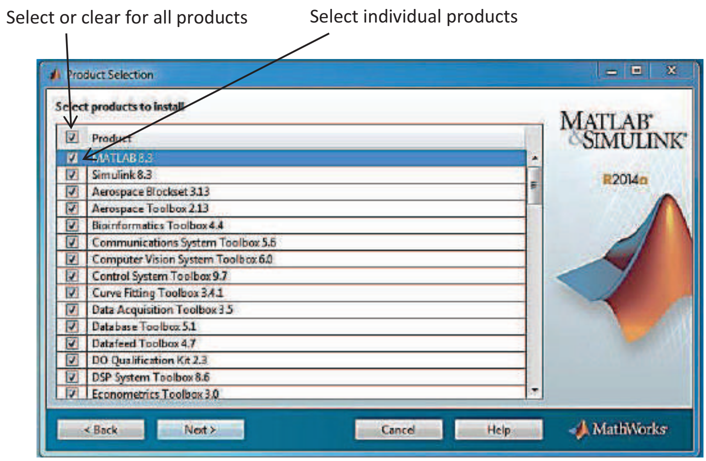
Step 5: Chose Installation Place

Specify the name of the folder where you want to install Math Works products. Accept the default installation folder or click "**Browse"** to select a different one. If the folder doesn't exist, the installer creates it. When specifying a folder name, you can use any alphanumeric character and some special characters, such as underscores. If you make a mistake while entering a folder name and want to start over, click "**Restore Default Folder"**. After making your selection, click "**Next"**.



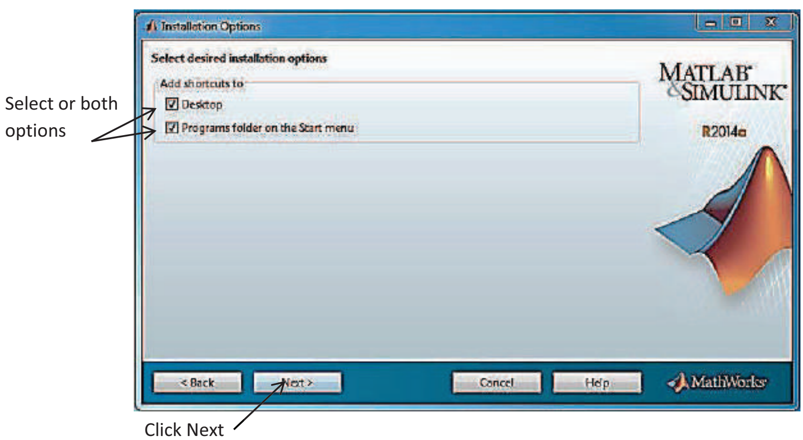
Step 6: Specify Products to Install

Specify which products you want to install in the Product Selection dialog box. This dialog box lists all the products associated with the license you selected or with the Activation Key you specified. In the dialog box, all the products are preselected for installation. If you do not want to install a particular product, clear the check box next to its name. After selecting the products you want to install, click **Next** to continue with the installation.



Step 7: Select Desired Installation Options

In this stage we get this dialog box, the installer displays add shortcuts of the program to two place "**Desktop**" and "**Programs folder on the Start menu**" choose both options and click "**Next**".

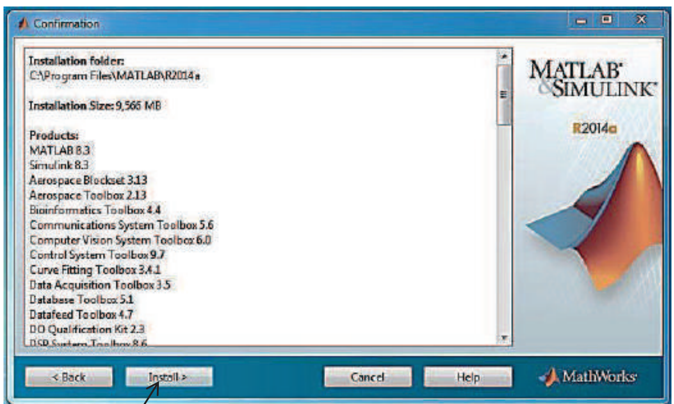


Step 8: Product Configuration Note

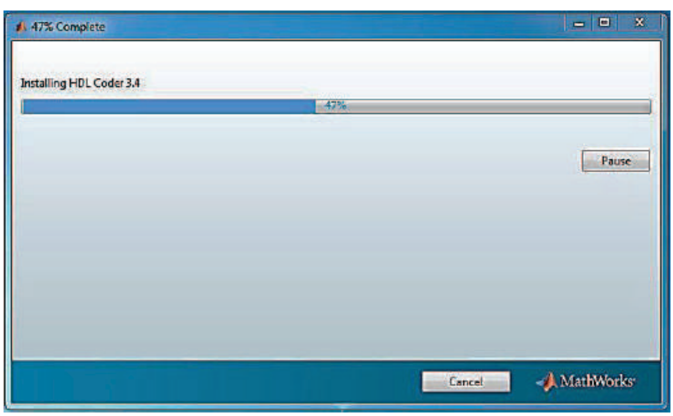
The installer displays "**Configuration**" dialog box show:-

- Installation folder: - Specify the name of the folder where you want to install Math Works products.
- Installation Size.
  - Products: - products you choose to install.

Click "**Next**". The installation will start.

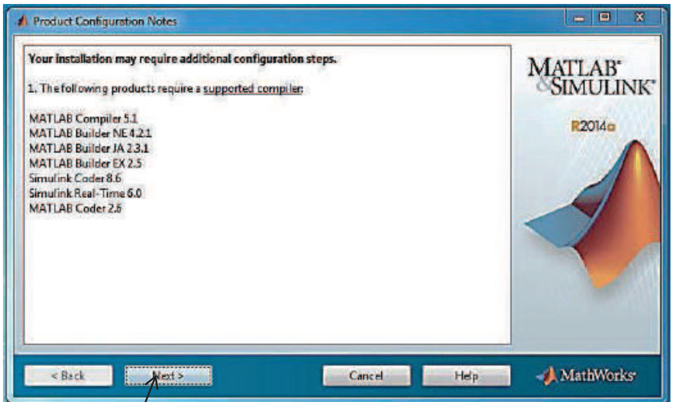


Click Next



Step 9: Product Configuration Note

When the installation successfully completed, the installer displays the Installation Complete dialog box and then "**Product Configuration Note"** dialog box show (Your installation may require additional configuration steps) click "**Next"**.

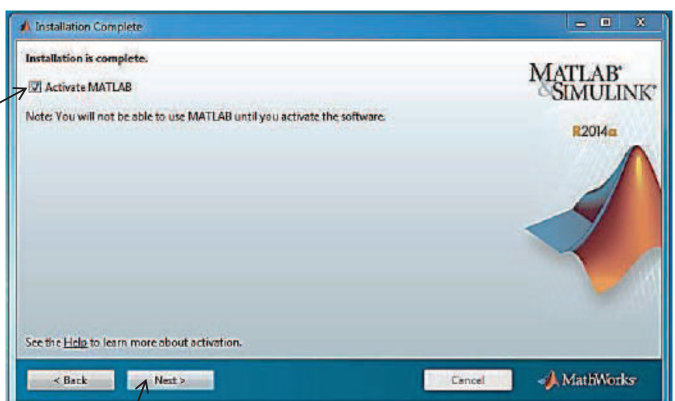


Click Next

Step 10: Activate MATLAB

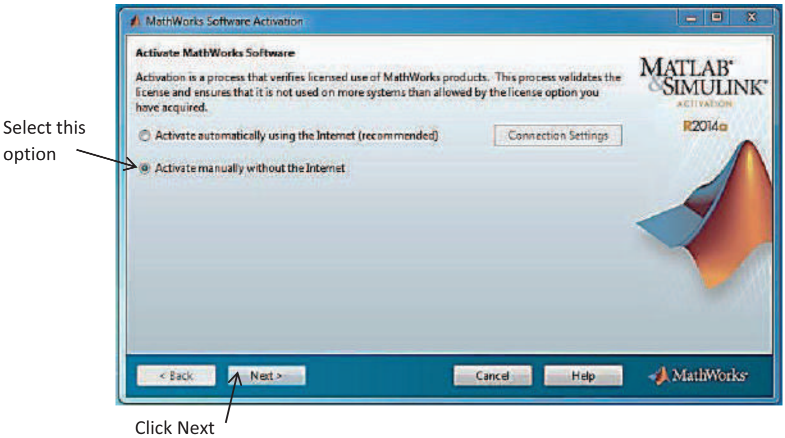
In this stage, we get this dialog box, the installer displays that the installation is complete and now you must activate MATLAB select "**Activate MATLAB**" and click "**Next**".

Select this option

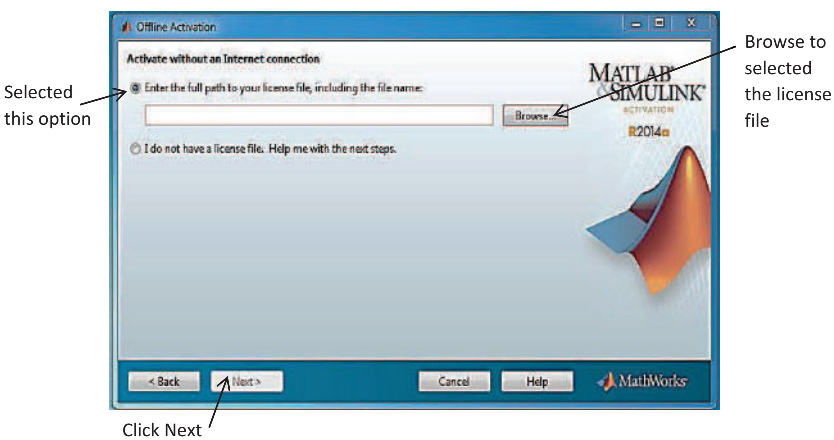


Click Next

Then you get dialog box show you the way of activation "Activation with and without Internet" choose "**Activate manually without the Internet**" and click "**Next**".

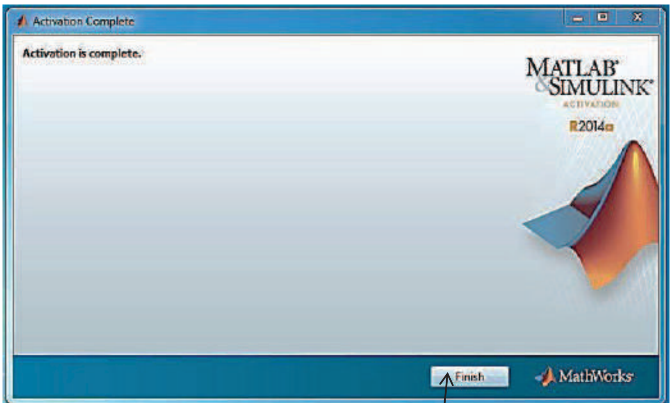


Then you get dialog box show two options choose "**Enter the full path to your license file, including the file name**" click "**Browse**" to select a the name of the folder where you license located and click "**Next**".



Step 11: Complete the Installation and Activations

Now the installation and Activations successfully completed, Click Finish and enjoy the program



Click Finish

Chapter Two: Introduction to MATLAB

2-1 Introduction

MATLAB is high –performance language of technical computing. It integrates computations, visualization, and programming in an –easy-to use environment where problems and solutions are expressed in familiar mathematical notations.

The name MATLAB stands for matrix laboratory. MATLAB is produced by Math Works .MATLAB was originally written to provide easy access to matrix.

2-2 Starting and Quitting MATLAB

1- To Start MATLAB, double –click the MATLAB short icon. On your window desktop .You will know MATLAB is running when you see the special '>>' prompt in the MATLAB command window.

2-To end your MATLAB session, select Exit MATLAB from the File menu in the desktop, or type quite (or Exit) in the command window, or by click on close button

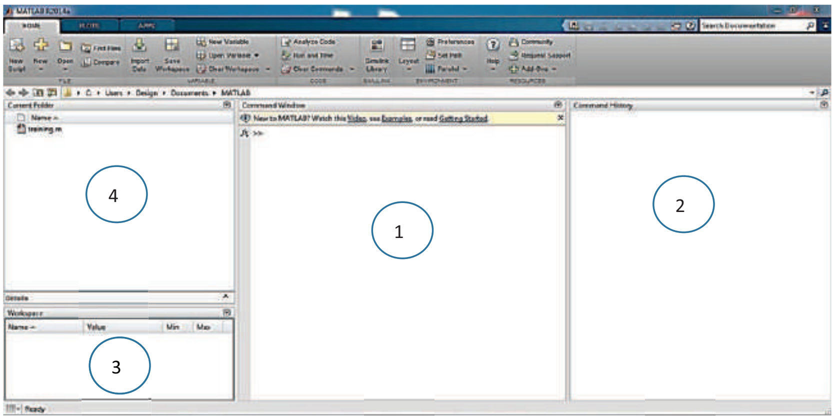
2-3 Desktop Tools

1- Command window: This is the main window, and contain the command prompt (>>) .This is where you will type all commands, enter variables and run functions and M-files.

2-Command History: Displays a list of previously typed commands, You can double –click to run them again.

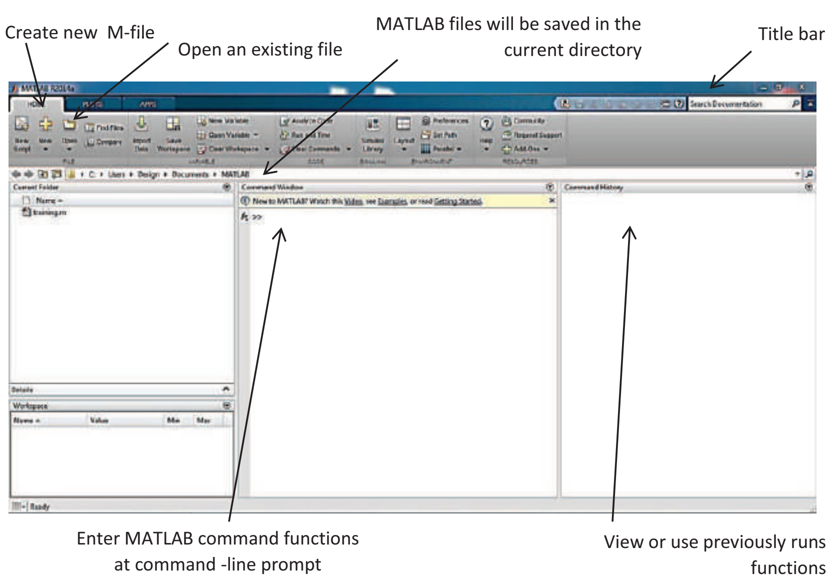
3-Workspace: Lists all variables you have generated in the current session. It shows the type and size of variables.

4- Current directory: Shows the files and folders in the current directory .The path to the current directory is listed near the top of the MATLAB desktop .By default, a MATLAB folder is created in your home directory on your M-drive, and this is where you should save your work.



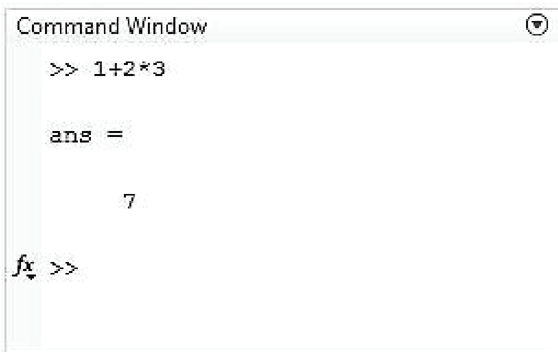
It could be exchange between the MATLAB different windows where,

Ctrl+0	Command window
Ctrl+1	Command history
Ctrl+2	Current directory
Ctrl+3	Work space

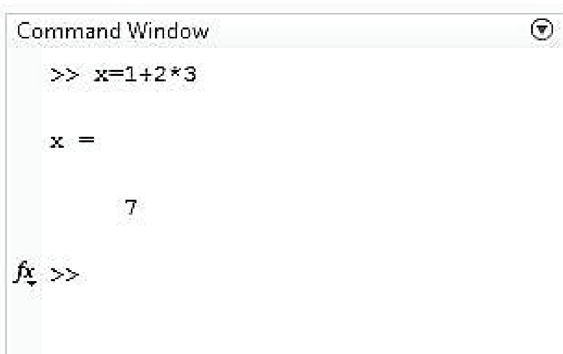


2-4 Using MATLAB as Calculator

MATLAB can perform basic calculations such as those you are used to doing on your calculator. Let's start at the very beginning, for example, let's suppose you want to calculate the expression, $1 + 2 \times 3$.You type it at the prompt command (>>) as follows,

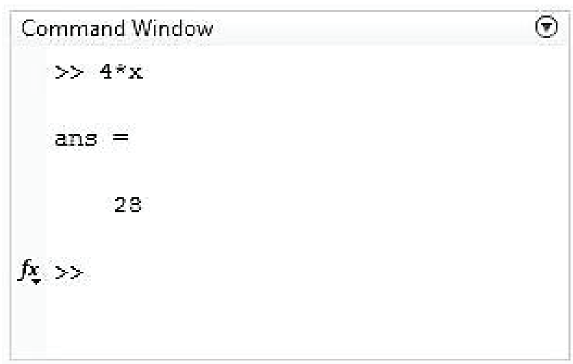


You noticed that if you don’t specify an output variable, MATLAB uses a default variables Ans., short for answer, to store the results of the current calculations. Note that the variable Ans. is created (or overwritten, if it is already existed) .To avoid this, you may assign a value to a variable or output argument name. For example,



Will result in x being given the value $1 + 2 \times 3 = 7$

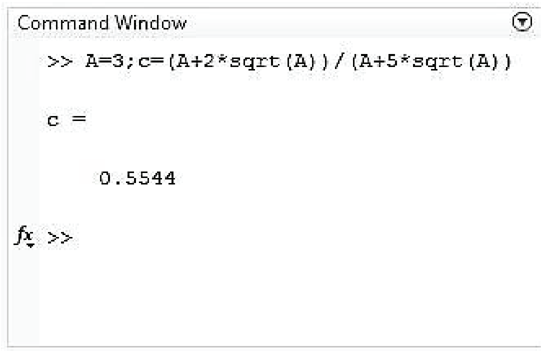
This variable name can be name can always be used to refer to the results of the previous computations. Therefore, computing 4*x will result in Table (1.1) gives the partial list of arithmetic operators



Basic arithmetic operators

Symbol	Operation	Example	answer
+	addition	2+3	5
-	subtraction	2-3	1-
*	multiplication	2*3	6
/	division	2/3	0.66667
^	Power	3^2	9
()	Specify evaluation order	(2+3)*3	15
sqrt	Square root	$\sqrt{144}$	12

You can do several operations in one calculation, such as



The default order of operation is (1) Exponential, multiplication and division, (3) addition and subtraction. Operation of equal priority is performed left to right.

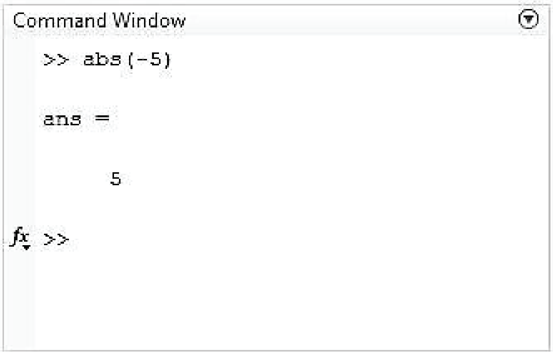
abs(x)	absolute value
Cos (x), sin(x), tan(x)	Cosine, sine, tangent of angle x in radians.
Sec (x), csc (x),cot(x)	Secant, cosecant, cotangent of angle x in radians
Log (x)	Natural logarithm : ln (x)
Log10(x)	Common (base10) logarithm :$\log_{10}$ (x)
exp(x)	Exponential function e^x

Examples

1-absolute value

Syntax

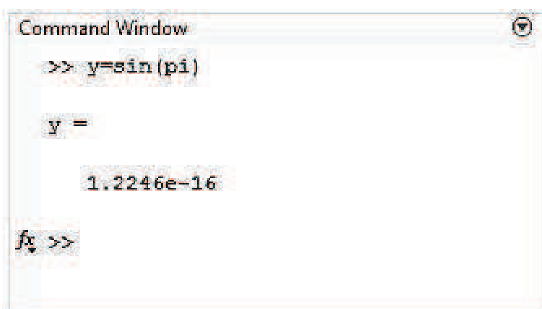
abs (x)



2-cos (x), sin (x), tan (x)

Syntax

y=sin(x)



3-sec(x), csc(x), cot(x)

Syntax

Y= csc (x)

```
Command Window
>> y=csc(pi/4)

y =

    1.4142

fx >>
```

4-log (x)

Syntax

Y=log (x)

```
Command Window
>> y=log(20)

y =

    2.9957

>> y=log10(20)

y =

    1.3010

fx >>
```

5- exponential (e^x)

Syntax

Y=exp(x)

6-square root ($\sqrt{x}$)

Syntax

Y=sqrt (x)

Exercises

Compute the values of

[Ans. =1.0323] 1. $\frac{2^5}{2^5-1}$ *and compare it with* $(1-\frac{1}{2^5})^{-1}$

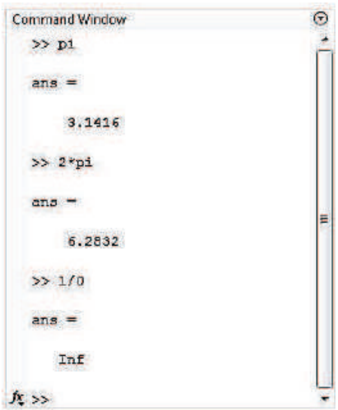
[Ans. = 0.5, 0.8536] 2. $\sin\left(\frac{\pi}{6}\right), cos^2(\pi/8)$

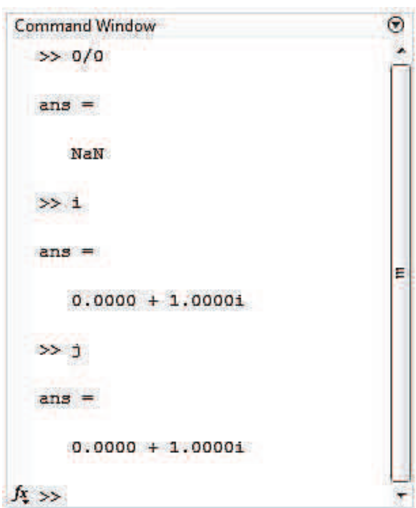
[Ans. = 3.0323] 3. $\frac{2^5}{2^5-1} + 4\sin(\pi/6)$

2-5 Some Predefined Variables in MATLAB

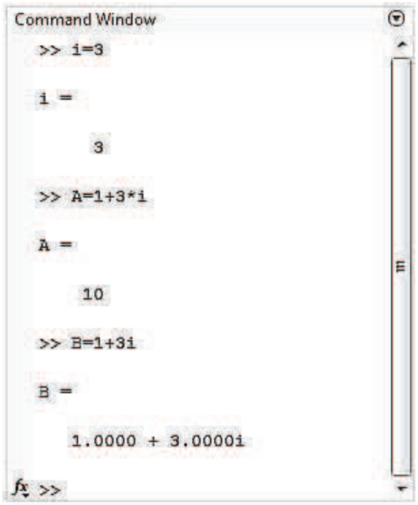
Predefined variable	Stand for
pi	$\pi = 3.1416$
Inf	$\infty = infinity$
NAN	Not a number
i	The complex variable $\sqrt{-1}$
j	The complex variable $\sqrt{-1}$

Examples





It could be overwrite the (i) value as



2-6 Managing the Workspace

The contents of the workspace persist between the executions of separate commands, it is possible for the results of one problem to have an effect on the next one .To avoid this possibility, it is a good idea to issue a clear command at the start of each new independent .It is a good to issue a clear command at start of each new independent calculation.

```
>>clear
```

The command (clear) removes all variables from the workspace .This frees the up system memory.

Notes

1-A semicolon ";" at the end the end of MATLAB statement suppresses printing of results.

2-Insert "%" before the statement that you want to use it as comment; the statement will appear in green color.

3-**clc command,** clears the command window and homes the cursor, i.e. remove all text.

Now try to do the following:

>>a=3

Can you see the effect of semicolon

>>a=3;

Just a comment >> % summation of 2 and 3

Can you see the effect of clear command

>>clear a

Clean the screen

>> clc

Comment

It is useful, for both yourself and others, to put comments in your script files. A comment is always precedes with a percent sign (%) which tells MATLAB not to execute the rest of the line as command.

2-7 Inverse Trigonometric Function

Built in function	Inverse Trigonometric Function
asin	Inverse sine
acos	Inverse cosine
atan	Inverse tangent
asec	Inverse secant
acsc	Inverse cosecant
acot	Inverse Cotangent

Examples

```
Command Window
>> %by defining the inverse sin function
>> a=asin(1)

a =

    1.5708

>>
```

At which angle the sine will be (1) ? It is (pi/2) =1.5708

By the same way:

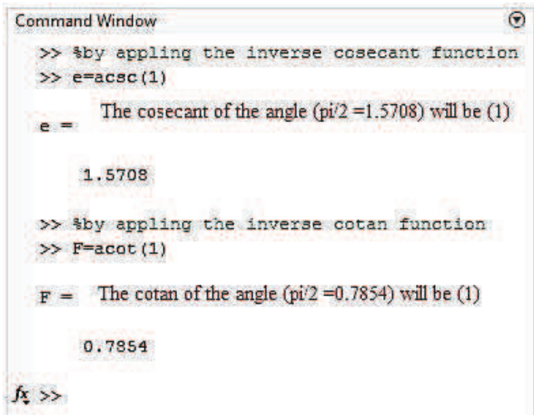
```
Command Window
>> %by appling the inverse cosine function
>> b=acos(1)

b =

     0

>> %by appling the inverse tan function
>> c=atan(1)

c =

    0.7854

>> %by appling the inverse secant function
>> d=asec(1)

d =

     0

>>
```

If we take the inverse cosine of (1)

We will get the angle (0)

The tan of angle (pi/4) = 0.7854

The secant of angle (0) will be 1



2-7.1 Hyperbolic Functions

Built in functions	Inverse Hyperbolic functions
sinh	Hyperbolic sine
cosh	Hyperbolic cosine
tanh	Hyperbolic tangent
sech	Hyperbolic secant
Csch	Hyperbolic cosecant
Coth	Hyperbolic cotangent

According to the relation

$$sinsh\ (z) = \frac{e^{z} - e^{-z}}{2}$$

We can check the answer by MATLAB as below:

```
Command Window
>> % comparing the result of (sinh) and the value  of (exp(x)-exp(-x))/2
>> x=1

x =

     1

>> A=sinh(x)

A =

    1.1752

>> B=(exp(1)-exp(-1))/2

B =

    1.1752

fx >>
```

$$\cosh(z) = \frac{e^{z} + e^{-z}}{2}$$

By using MATLAB

```
Command Window
>> % comparing result of (cosh) and the value of (exp(x)+exp(-x))/2
>> x=1

x =

     1

>> A=cosh(1)

A =

    1.5431

>> B=(exp(1)+exp(-1))/2

B =

    1.5431

fx >>
```

$$\tanh(z) = \frac{\sinh(z)}{\cosh(z)}$$

```
Command Window
>> %by getting (sinh) function
>> x=1;
>> A=sinh(x)

A =

    1.1752

>> % by getting (cosh) function
>> x=1;
>> B=cosh(x)

B =

    1.5431

>> C=A/B

C =

    0.7616

>> % by getting the(tanh)function
>> D=tanh(x)

D =

    0.7616
```

$$\mathrm{sech}(z) = \frac{1}{\cosh(z)}$$

```
Command Window
>> % by getting the (cosh) function
>> B=cosh(x)

B =

    1.5431

>> C=1/B

C =

    0.6481

>> % by getting the hyperbolic secant function
>> D=sech(x)

D =

    0.6481
```

$$\operatorname{csch}(z) = \frac{1}{\sinh(z)}$$

```
Command Window
>> % by getting (sinh) function
>> x=1;
>> A=sinh(x)

A =

    1.1752

>> C=1/A

C =

    0.8509

>> % by getting hyperbolic (cosecant) function
>> D=csch(x)

D =

    0.8509
```

$$\coth(z) = \frac{1}{\tanh(z)}$$

```
Command Window
>> % by getting (tanzh) function
>> x=1;
>> D=tanh(x)

D =

    0.7616

>> E=1/D

E =

    1.3130

>> % by getting the hyperbolic cotangent function
>> F=coth(x)

F =

    1.3130
```

2-7.2 Inverse Hyperbolic Functions

Built in function	Invers hyperbolic function
Asinh	Inverse hyperbolic sine
Acosh	Inverse hyperbolic cosine
Atanh	Inverse hyperbolic tangent
Asec	Inverse hyperbolic secant
Acsc	Inverse hyperbolic cosecant
Acot	Inverse hyperbolic cotangent

Chapter Three: Arrays (Vectors and Matrices)

An array is a list of numbers arranged in rows and / or columns. A one-dimensional array is a row column of numbers and a two- dimensional array has a set of numbers arranged in rows and columns. An array operation is performed element by element.

3-1 Vectors

The basic data structure in MATLAB is the matrix .Even scalars (numbers) are considered to be matrices (1 row and 1 column).

Example

```
>> x=3
X=3
>>whos ('x')
```

Name	size	Bytes	Class	Attributes
x	1x1	8	double	

Matrix 1×1 Note that MATLAB recognize the size of variable x as

3-1.1 Row Vector

A Row vector has the form:

$$v = [v_1 \;\; v_2 \; v_3 \ldots \ldots v_n]$$

Are usually scalars (either real or complex numbers) , where $v_1, v_2, v_3 \ldots . v_n$

When you enter a row vector in MATLAB, you can separate the individual elements by commas.

Example

```
Command Window
>> v=[1,2,3,4,5]

v =

     1     2     3     4     5
```

```
Command Window
>> v=[1 2 3 4 5]

v =

     1     2     3     4     5
```

You can determine the length of a row vector with MATLAB's Length command.

Syntax

```
Command Window
>> length(v)

ans =

     5
```

You can access the element of v at position k with MATLAB 's indexing notation v(k) .For example ,the element of the third position of vector v is found as follows :

Syntax

```
Command Window
>> v(3)

ans =

     3
```

You can access the 1^{st}, 3^{rd} and 4^{th} entries of vector v as follows :

```
Command Window
>> v=[1,3,4]

v =

     1     3     4

fx >>
```

You can use indexing to change the entry in the 5[th] position of v as follows:

Syntax

```
Command Window
>> v(5)=500

v =

     1     3     4     0   500

fx >>
```

You can change the 1st, 3rd and 5th entries as follows .Note that the vector on the right must have the same length as the area to which it is assigned.

Syntax

```
Command Window
>> v([1,3,5])=[-10 -20 -30]

v =

   -10     3   -20     0   -30

fx >>
```

If you break the equal length rule, MATLAB will respond with an error message.

```
Command Window
>> v([2,4])=[100 200 300]
In an assignment  A(I) = B, the number of elements
in B and I must be the same.

fx >>
```

There is one exception to this rule .You may assign a single value to a range of entries in the following manner.

```
Command Window
>> v([1,3,5])=0

v =

     0     3     0     0     0

fx >>
```

3-1.2 Column Vector

A column vector has the form:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \cdot \\ \cdot \\ \cdot \\ v_n \end{bmatrix}$$

Are usually scalars (either real or complex numbers) where $v_1, v_2, v_3, \dots \dots, v_n$

In MATLAB ,use the semicolon to end a row and begin a new row .This is useful in building a column vectors.

```
Command Window
  >> w=[1;2;3]

  w =

       1
       2
       3

fx >>
```

The length of a columns vector is determined in exactly the same way the length of row vector id determined.

```
Command Window
  >> length(w)

  ans =

       3

fx >>
```

Indexing works the same with column vectors as it does with row vectors .You can access the second element of the vector w as follows:

```
Command Window
  >> w (2)

  ans =

       2

fx >>
```

You can use indexing notation to (change) the entry in the second position of w as follows:

```
Command Window
  >> w (2)=15

  w =

       1
      15
       3

fx >>
```

Example

```
Command Window
  >> A=[1;2;3;4;5;6;7;8;9;10];
  >> A(12)=12

  A =

       1
       2
       3
       4
       5
       6
       7
       8
       9
      10
       0
      12

fx >>
```

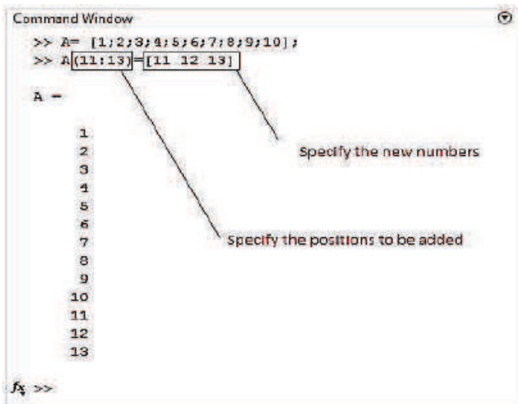
3-1.3 Adding an Element

To add an element to a vector as follows:

```
Command Window
  >> A=[1;2;3;4;5;6;7;8;9;10];
  >> A(11)=11

  A =

       1
       2
       3
       4
       5
       6
       7
       8
       9
      10
      11

fx >>
```

3-1.4 Adding Consecutive Elements

You can also use indexing to (add) the entry of consecutive elements as follows:



3-1.5 Transpose Operator

The transpose of a row vector is a column vector, and vice versa , the transpose of a column vector is a row vector .Mathematically , we use the following symbolism to denote the transpose of a vector .

$$\begin{bmatrix} v_1 \\ v_2 \\ \cdot \\ \vdots \\ v_3 \end{bmatrix}^T = [v_1\ v_2 \ldots v_n]$$

```
Command Window
>> u=[1;3;5;7]

u =

     1
     3
     5
     7

>> u'

ans =

     1     3     5     7

fx >>
```

3-1.6 Increment Notation

One can easily create vector with a constant increment between entries .You use Matlab's start: increment: finish construct for this purpose .In the case where no increment is supplied, the increment is understood to be 1.

Ex.1

```
Command Window
>> x=1:10

x =

     1     2     3     4     5     6     7     8     9    10

fx >>
```

Ex.2 create a vector starting at 0 and finish at 20 with increment of 5

```
Command Window
>> x=0:5:20

x =

     0     5    10    15    20

fx >>
```

Of course, if we'd rather store a column vector in x, we can use transpose operator.

```
Command Window
>> y=(0:0.2 :1)'

y =

         0
    0.2000
    0.4000
    0.6000
    0.8000
    1.0000

fx >>
```

Increment notation can be extremely useful in indexing .For example, consider the following vector v.

```
Command Window
>> v= 100:-10:10

v =

   100    90    80    70    60    50    40    30    20    10

fx >>
```

We can access every entry, starting at the 4th entry and extending to the last entry in the vector with the following notation.

```
Command Window
>> v (5:end)

ans =

    60    50    40    30    20    10

fx >>
```

We can access the entry in the even positions of the vector as follows.

```
Command Window
>> v(2:2:end)

ans =

    90    70    50    30    10

fx >>
```

3-2 Matrices

Manipulation of matrices is one of the most important tasks in Matlab.

Command	Meaning
[]	Matrix constructor
,	Separate matrix columns
;	Separate matrix rows
:	From to ,all

Matrices can be used to represent images, systems of linear equations and generally many types of data.

```
Command Window
>> A=[2,4;6,8]

A =

     2     4
     6     8

fx >>
```

The commas can be replaced by spaces

```
Command Window
>> A =[2 4; 6 8]

A =

     2     4
     6     8

fx >>
```

Commas and semicolon can be also used to separate statements. Commas will display the result, while semicolon will not .It is practical to insert a semicolon at the end of each statement, when working with large matrices.

```
Command Window
>> B =[1 ,3 ; 5 7] ,C=[12 ,13; 14, 15], A=[2 ,4; 6 , 8]

B =

     1     3
     5     7


C =

    12    13
    14    15


A =

     2     4
     6     8

fx >>
```

Matrices can be combined by using the comma and the semicolon in conjunction with the matrix constructor [ ].

```
Command Window
>> C=[A,B] , D=[A;B]

C =

     2     4     1     3
     6     8     5     7


D =

     2     4
     6     8
     1     3
     5     7

fx >>
```

If the matrices we are trying to combine are not of the correct shapes, i.e. the rows or columns that we are trying to combine do not match, and then the following error will be produced.

```
Command Window
>> C,D

C =

     2     4     1     3
     6     8     5     7


D =

     2     4
     6     8
     1     3
     5     7

fx >>
```

```
>> E=[C; D]
Error using vertcat
Dimensions of matrices being concatenated are not consistent.

fx >>
```

The problem in the construction of matrix E is that we are using the row separator (;), while the matrix C has 4 columns and matrix D has 2. The same problem would appear if we were trying to combine two matrices that do not have the same number of rows using the column separator (, or space).

```
Command Window
  >> E=[C,D]
  Error using horzcat
  Dimensions of matrices being concatenated are not consistent.

fx >>
```

To combine matrices with the row separator (;) the number of columns of each matrix has to be the same, while to combine matrices using the columns separator (,) the number of rows of each matrix has to be the same. The colon operator (:) can also be used in conjunction with the matrix constructor [ ].

```
Command Window
  >> F= [1:1:5 ; 11: 2: 20]

  F =

       1     2     3     4     5
      11    13    15    17    19

fx >>
```

3-2.1 Addressing and Assigning Elements

The first action we have to take in order to assign a value to a variable is to address the element of the variable we want .T address an element of a matrix the round brackets (a,b) are used .a and b are positive integers , the first element inside the brackets denotes the row and the second denotes the column .

Command Window

```
>> F

F =

     1     2     3     4     5
    11    13    15    17    19

>> F(2,3)

ans =

    15

>>
```

Vectors can also be used to address elements in a matrix

Command Window

```
>> F ([1;2] ,3)

ans =

     3
    15

>> F([1 2] , 3)

ans =

     3
    15

>> F (:,1)

ans =

     1
    11

>> F(1,:)

ans =

     1     2     3     4     5
```

The colon: can be used to address all the elements of a row or a column

Matrices can also be addressed as if they were a vector. The elements are numbered by first counting the elements of a column and then progressing to the next column. (as in figure below)

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

Numbering of matrix as a vector

```
Command Window
>> F

F =

     1     2     3     4     5
    11    13    15    17    19

>> F(3)

ans =

     2

>> F(4)

ans =

    13

fx >>
```

Elements can be modified in a matrix simply by addressing the particular element and then assigning it using the equal =.

```
Command Window
>> F

F =

     1     2     3     4     5
    11    13    15    17    19

>> F(1,2)=5

F =

     1     5     3     4     5
    11    13    15    17    19

>>
```

```
Command Window
>> F ([1 2] ,3) =[21 22]

F =

     1     5    21     4     5
    11    13    22    17    19

>> F

F =

     1     5    21     4     5
    11    13    22    17    19

>> F (:,4) =[14 ;16]

F =

     1     5    21    14     5
    11    13    22    16    19

>> F

F =

     1     5    21    14     5
    11    13    22    16    19
```

The following table has the most important matrix operations.

Matrix operations	Meaning
'	Transpose
+	Addition
-	Subtraction
*	Multiplication
^	Power

A' Or The transpose of a matrix A is a matrix A^T

```
Command Window
>> B=[10,20,30;40,50,60;70,80,90] ,A= [2,3,4;5,6,7;8,9,10]

B =

    10    20    30
    40    50    60
    70    80    90


A =

     2     3     4
     5     6     7
     8     9    10

>> A'

ans =

     2     5     8
     3     6     9
     4     7    10

fx >>
```

```
Command Window
>> A+B

ans =

    12    23    34
    45    56    67
    78    89   100

>> B-A

ans =

     8    17    26
    35    44    53
    62    71    80

fx >>
```

3-2.2 Matrix Multiplication

In Matrix multiplication the number of columns of the first matrix should be equal to the number of the rows of the second matrix.

$$A = \begin{bmatrix} 1 & 2 \\ 4 & 6 \\ 9 & 8 \end{bmatrix}_{3\times 2} \quad , \; B = \begin{bmatrix} 0 & 3 & 3 \\ 4 & 9 & 7 \end{bmatrix}_{2\times 3}$$

$$C = A \times B = \begin{bmatrix} 1 & 2 \\ 4 & 6 \\ 9 & 8 \end{bmatrix}_{3\times 2} \times \begin{bmatrix} 0 & 3 & 3 \\ 4 & 9 & 7 \end{bmatrix}_{2\times 3}$$

$$C = \begin{bmatrix} (1\times 0)+(2\times 4) & (1\times 3)+(2\times 9) & (1\times 3)+(2\times 7) \\ (4\times 0)+(6\times 4) & (4\times 3)+(6\times 9) & (4\times 3)+(6\times 7) \\ (9\times 0)+(8\times 4) & (9\times 3)+(8\times 9) & (9\times 3)+(8\times 7) \end{bmatrix}$$

$$C = \begin{bmatrix} 8 & 21 & 17 \\ 24 & 66 & 54 \\ 32 & 99 & 83 \end{bmatrix}_{3\times 3}$$

Now using MATLAB

```
Command Window
>> % By defining the matrix A
>> A=[1,2;4,6;9,8];
>> %By defining the matrix B
>> B=[0,3,3;4,9,7];
>> C=A*B

C =

     8    21    17
    24    66    54
    32    99    83

fx >>
```

3-2.3 Power of Matrix

Power of matrix is defined as followed

$$A^k = \underbrace{A \times A \times \dots . A}_{\text{k times}}$$

```
Command Window
>> B=[1,2,3;4,5,6;7,8,9]

B =

     1     2     3
     4     5     6
     7     8     9

>> B^2

ans =

    30    36    42
    66    81    96
   102   126   150

fx >>
```

3-2.4 Element by Element Operations

Element by element operations are also useful .These operations can only be done with arrays of the same size. Element by element multiplication ,division and exponentiation of two vectors or matrices is entered in MATLAB by typing a period in front of the arithmetic operator .The table below lists these operations.

Arithmetic operations			
Matrix operation		Array operation	
+	Addition	+	Addition
-	Subtraction	-	Subtraction
*	Multiplication	.*	Array Multiplication
^	Exponentiation	.^	Array Exponentiation
/	Right division	. /	Array Right division
\	Left division	. \	Array Left division

```
Command Window
>> A=[1 2 ; 4 6]  , B=[3 5;7 9]

A =

     1     2
     4     6


B =

     3     5
     7     9

>> A.*B

ans =

     3    10
    28    54

>> A./B

ans =

    0.3333    0.4000
    0.5714    0.6667
```

```
Command Window
>> A.^2

ans =

     1     4
    16    36

>> 
```

```
Command Window
>> A

A =

     1     2
     4     6

>> A+2

ans =

     3     4
     6     8

>> A-2

ans =

    -1     0
     2     4

>> A*2

ans =

     2     4
     8    12
```

```
Command Window
>> A/2

ans =

    0.5000    1.0000
    2.0000    3.0000

>>
```

3-2.5 Built –in Functions for Arrays

The table below lists some of the many build- in functions available in MATLAB for analyzing arrays:

Function	Description	Example
Mean (A)	If A is a vector ,returns the mean value of the elements.	>> A=[3 7 2 16]; >> mean (A) ans=7
C= max(A)	If A is a vector, C is the largest element in A. If A is a matrix ,C is a row vector containing the largest element of each column of A.	>> A=[3 7 2 16 9 5 18 13 0 4]; >> C=max(A) C=8
min(A)	The same as **max (A)** , but for the smallest element.	>> A =[3 7 2 16]; >> min (A) ans = 2
sum (A)	If A is a vector ,returns the sum of the elements of the vctor	>>A=[3 7 2 16]; >> sum (A) Ans =28
Det (A)	Returns the determinant of a square matrix A.	>> A=[1 2; 3 4]; >> det (A) Ans= -2
Inv (A)	Returns the inverse of a square matrix A.	>> a= [1 2 3 ;4 6 8 ; -1 2 3]; >>inv (A) Ans= -0.5000 0.0000 -0.5000 -5.0000 1.5000 1.0000 3.5000 -1.0000 -0.5000

3-3 Solving Linear Equations

One of the problems encountered most frequently in scientific computations is the solutions of the systems simultaneous linear equations .With matrix notation , a system of simultaneous linear equations is written:

Ax=b

Where there are as many equations as unknown. A is a given square matrix of order n ,b is a given column vector of n components ,and x is an unknown column vector of n components .In linear algebra we learn that the solution to Ax =b can be written as $x = A^{-1}b$,where A^{-1} is the inverse of A. For example consider the following system of linear equations

$$\begin{cases} x + 2y + 3z = 1 \\ 4x + 5y + 6z = 1 \\ 7x + 8y = 1 \end{cases}$$

The coefficient matrix A is

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} \qquad \text{and the vector} \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

With matrix notation ,a system of simultaneous linear equation is written :

Ax =b

This equation can be solved for x using linear algebra .The result is= $A^{-1}b$.

There are typically two ways to solve for x in MATLAB:

1-The first one is to use the matrix inverse, inv.

Lineareqn.m*

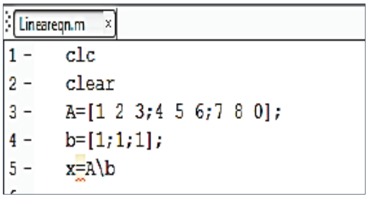
```
clc
clear
A=[1 2 3;4 5 6;7 8 0];
b=[1;1;1];
x=inv(A)*b
```

The output in the command window,

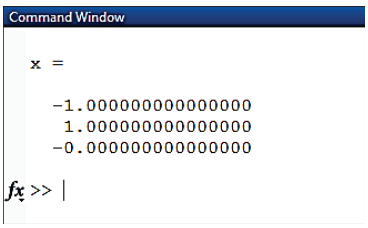
Command Window

```
x =

  -1.000000000000000
   1.000000000000000
  -0.000000000000000

fx >>
```

2-The second one is to use backslash(\) operator. The numerical algorithm behind this operator is computationally efficient. This is numerically reliable way of solving system of linear equations by using a well-known Gaussian Elimination.



The output in the command window will be,



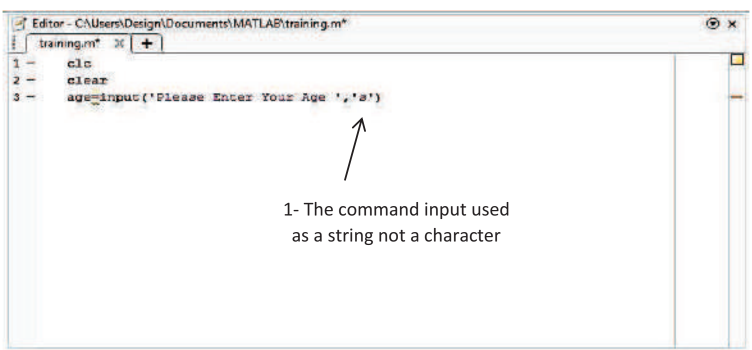
3-4 Str2num & num2str

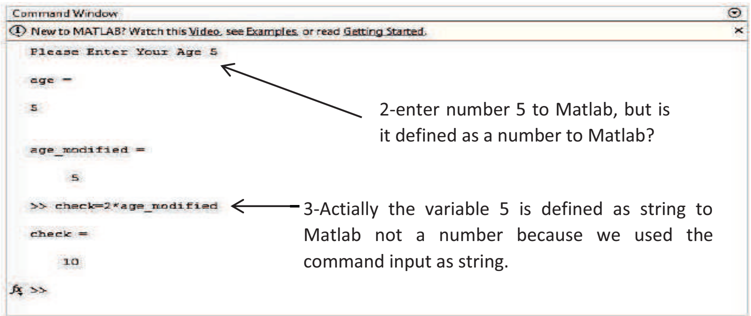
String: It is a letter or word in MATLAB,while **character** is a number or numbers

There is a command to change the string to character and vice versa as follows:

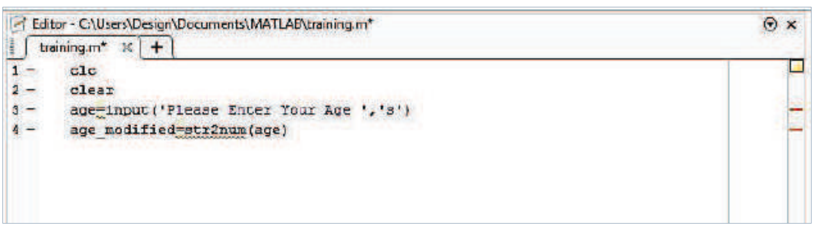
Num2str & str2num.

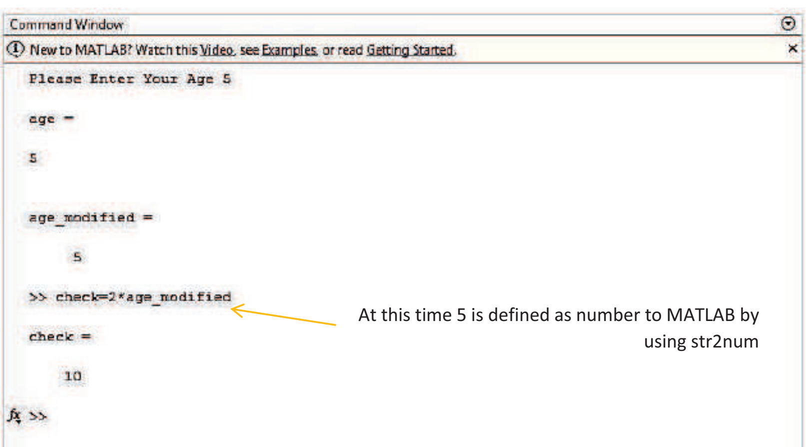
Now we will use the command **input** as a string ,



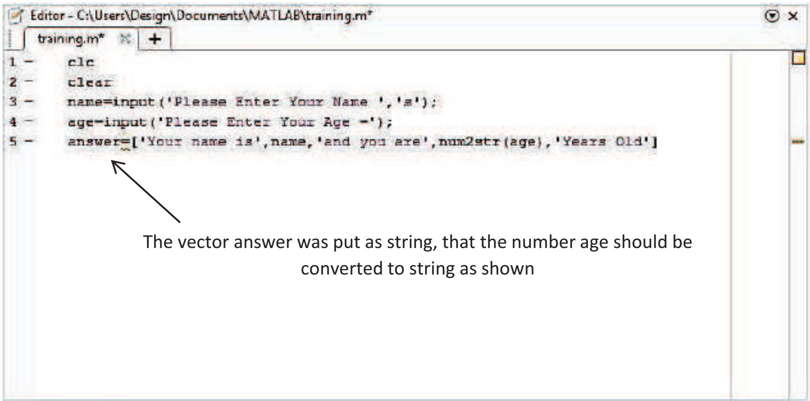


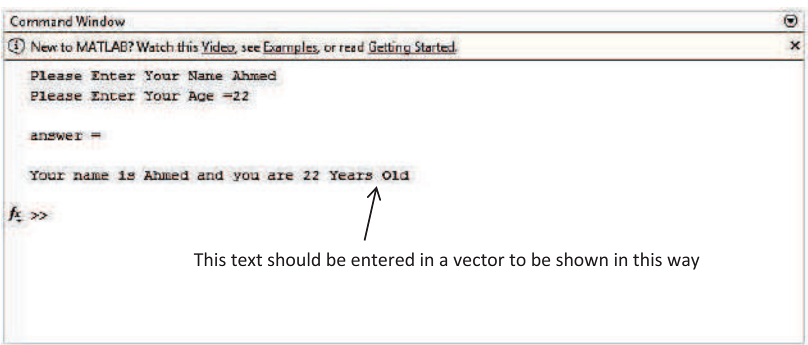
Using str2num to convert string to character as follows:





Num2str is used to convert the numbers to string .Now we will introduce the name and the age ,then put them within a vector ,that it should be noticed that *the vector should contained only a string or only characters and not both*.





Chapter Four: 2D Plotting

4-1 Plotting

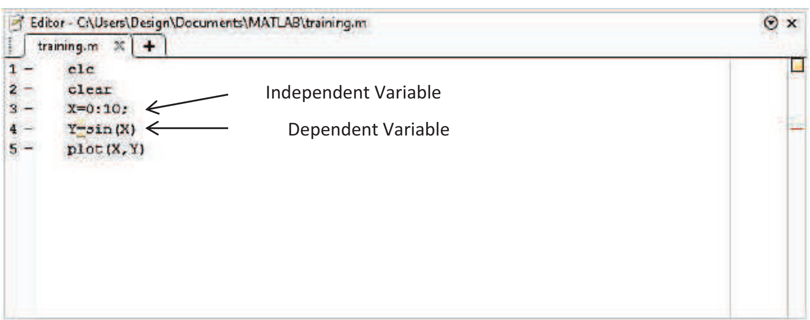
A very suitable feature of MATLAB is its capability to generate high quality two- and three-dimensional plots using simple and flexible functions.

The two-dimensional drawing is the process that relate tow element in one graph, the first called "Independent Variable" which has fixed values and the second called "dependent Variable" "which has values dependent on the values of independent variable.

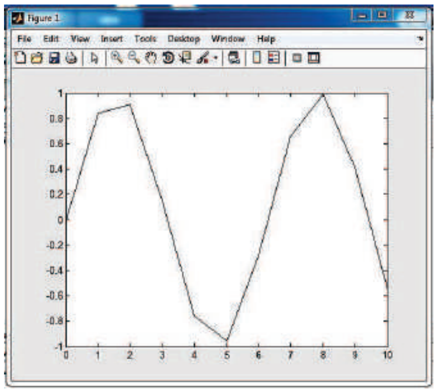
The syntax of 2D plotting command is:

Plot (Independent Variable, dependent Variable)

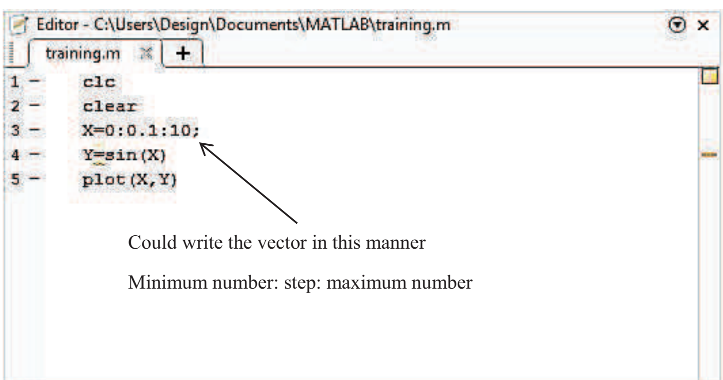
Example:



Note we select only 10 points to draw sin wave and this is a little number to draw this function, therefore the figure appears as below:

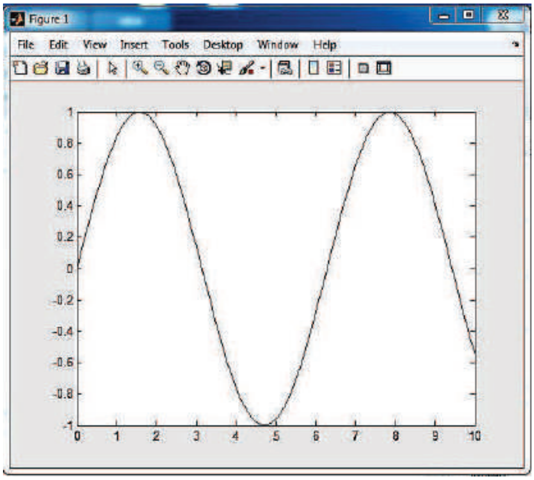


To solve this problem, we have to increase the number of points within the vector, as in the following figure:



You will notice that the drawing has improved a lot:

You will notice that the drawing has improved a lot:



4-1.1 Adding Graphics Properties Within MATLAB

Sometimes it may be necessary to change some of the characteristics of the graphics that we get such as changing colors, change the line drawing. Below set of characteristics:

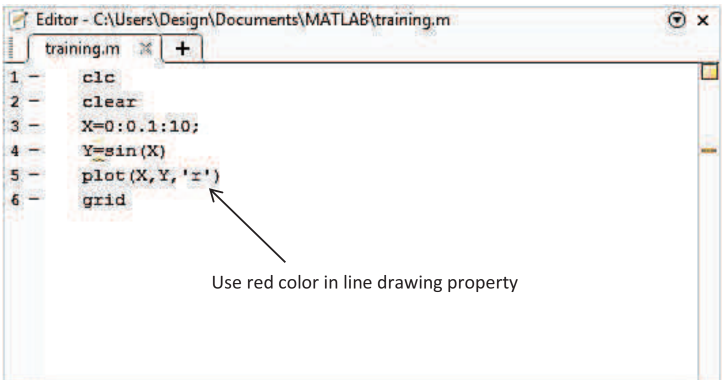
b	blue	.	point	-	solid	v	triangle (down)
g	green	o	circle	:	dotted	^	triangle (up)
r	red	x	x-mark	-.	dashdot	<	triangle (left)
c	cyan	+	plus	--	dashed	>	triangle (right)
m	magenta	*	star	(none)	no line	p	pentagram
y	yellow	s	square			h	hexagram
k	black	d	diamond				

Where it can use those characteristics within command plotting in the following formula:

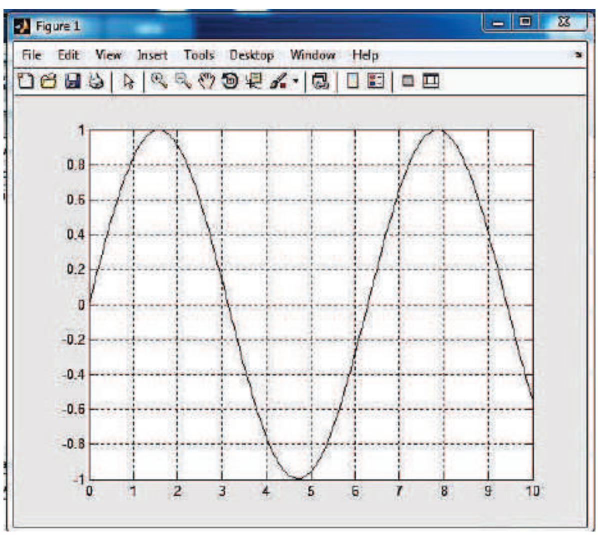
Plot (Independent Variable, dependent Variable, ' the property')

It further notes, property must be put between single citations after the dependent variable.

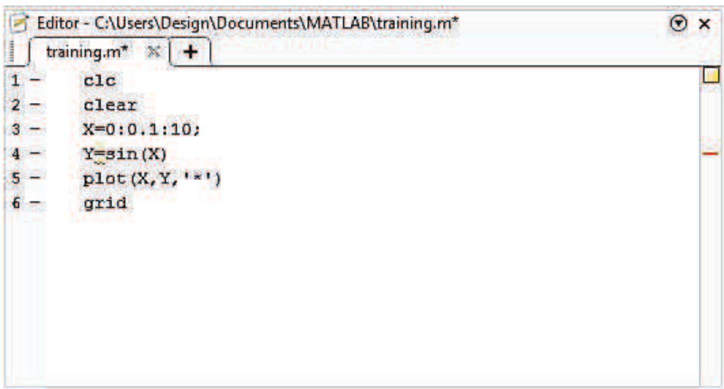
Depending on the previous example, will change the font color feature to red:



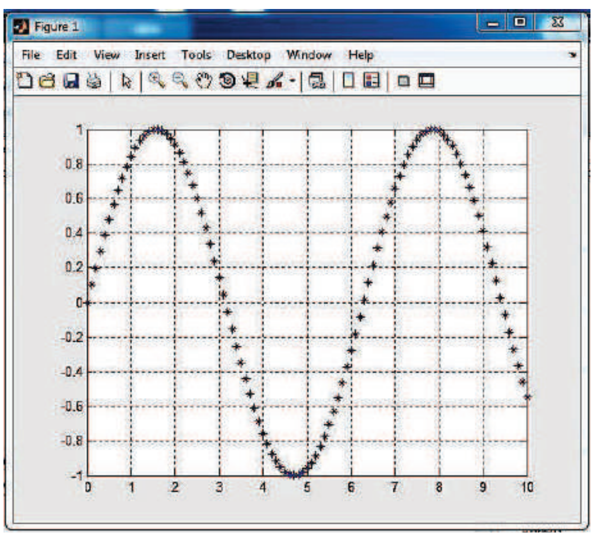
Thus, we get the following figure:



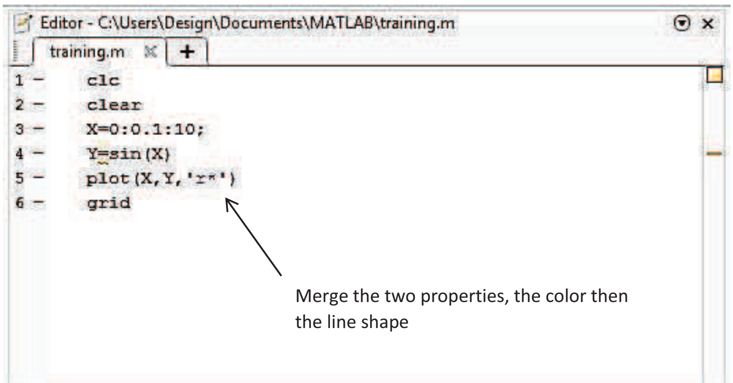
We are now adding a new property, change the font of the function from a straight line to the stars:



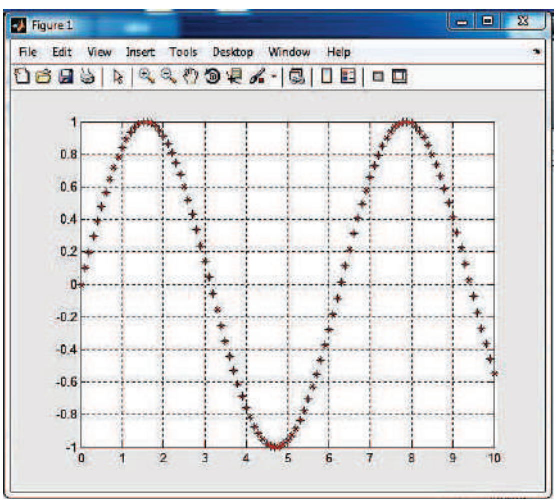
Thus, we get the following figure:



If we want to merge the two properties together (to change the color and shape of the line):



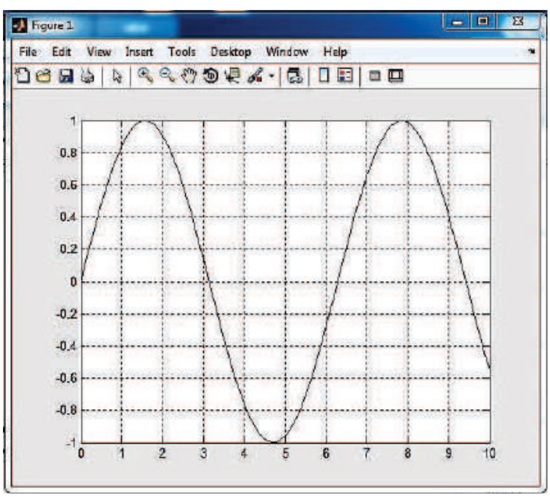
Thus, we get the following figure:



4-1.2 Use the Grid in Graphic

The MATLAB put a grid on the drawing, where it is easy to determine the set values by using the command **grid** after the command plot:

Thus, we get the following figure:



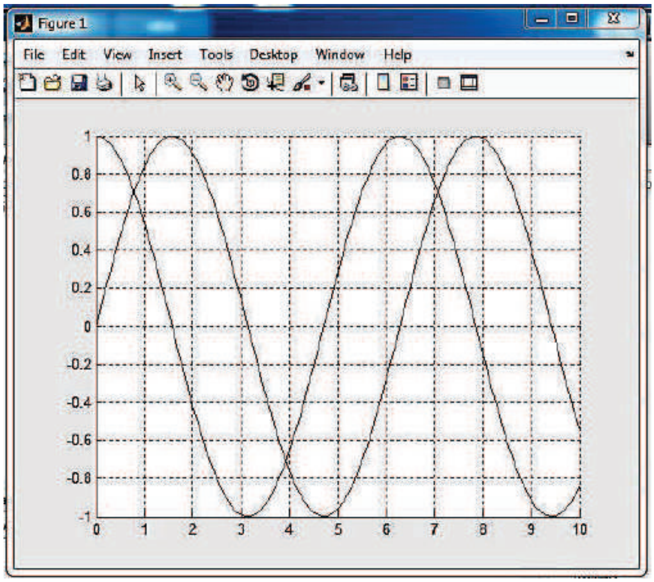
Now we will add another equation in addition to the first equation:

```
clc
clear
X=0:0.1:10;
Y=sin(X)
Z=cos(X)
plot(X,Y)
plot(X,Z)
grid
```

However, when execute the program, the program show only the last function. To appear the two functions in the same figure we use the command (Hold on Hold off) as show in the following example:

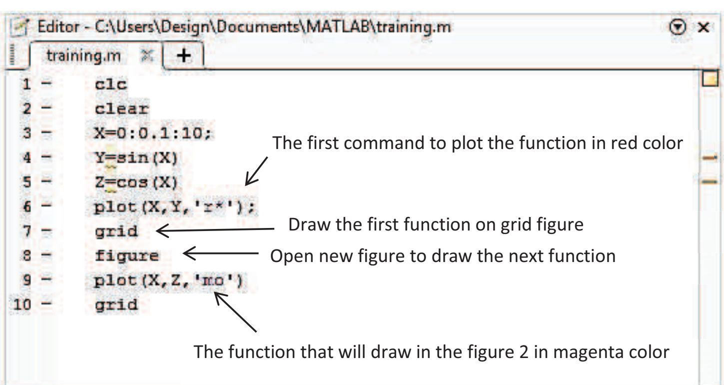
```
clc
clear
X=0:0.1:10;
Y=sin(X)
Z=cos(X)
hold on
plot(X,Y)
plot(X,Z)
grid
hold off
```

Thus it will appear as in the following figure:

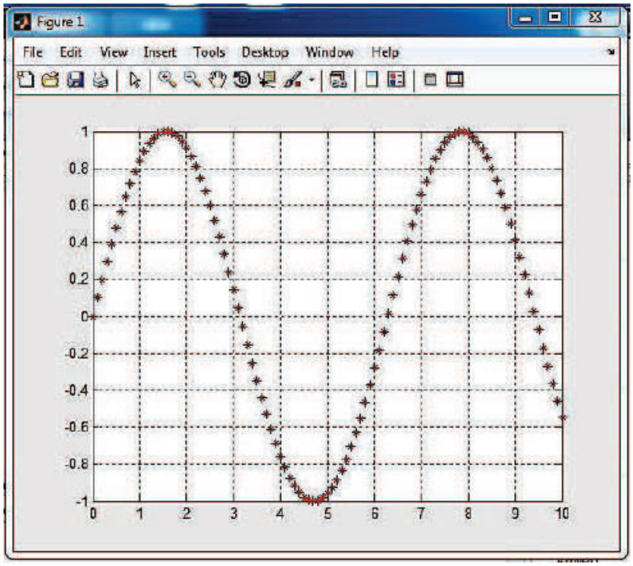


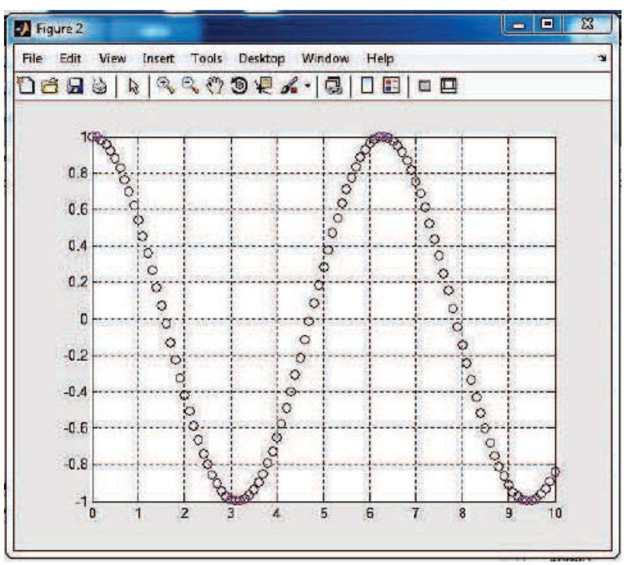
4-1.3 Display Functions in Separate Figures

If we want to appear, the functions in separate figures (each function in a figure) use the **figure** command as show in the following example:



Thus, it will appear each function in dependent figure:





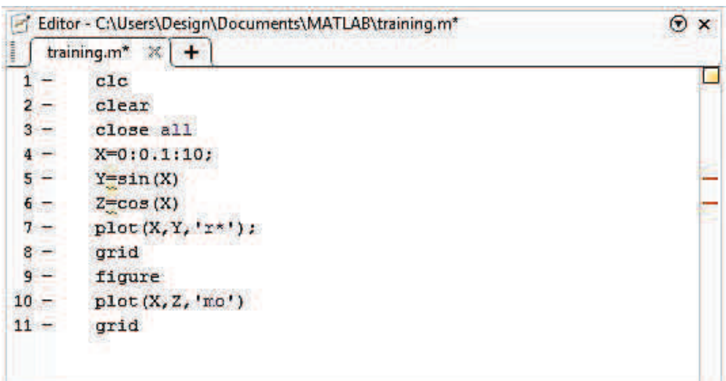
After each execution of MATLAB program must closed all windows which have results to avoid the appearance of new execution on the last figure, to do that used the command close all after command clear as shown in the following commands:

clc

clear

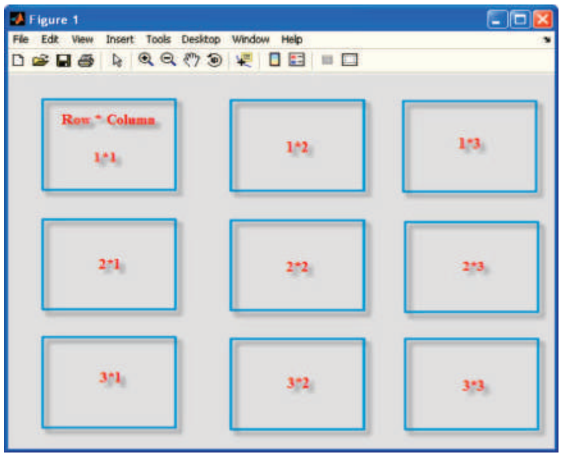
close all

Where command close all can be used in the previous example to read as follows:

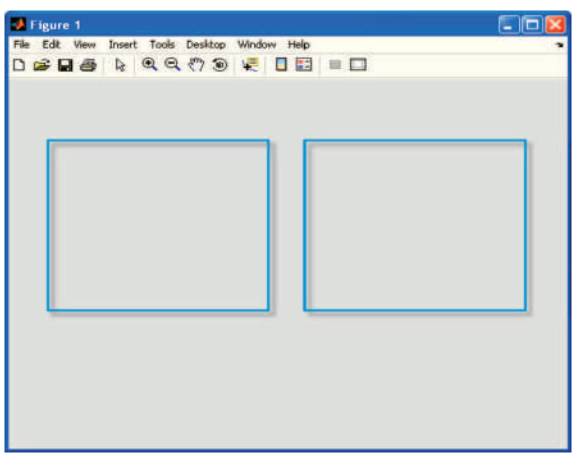


4-1.4 Create a Separate Graphics in a Single Window

MATLAB software provides the possibility of drawing several separate graphics in a single window, using the command subplot before each command plot, to used subplot command should be arrange the figures as matrix or vector and find out number and location of each figure in a widow as show in the following figure:



For example, we will take the equations must be drawn beside each other to be as in the following figure:



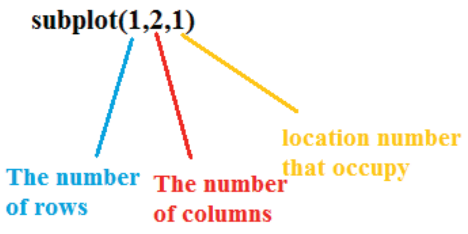
The two figures arranged as vector has one row and two columns, figure 1 has first location and the second figure has second location in a vector, this figure location should determine in a subplot command as in a following:

4-1.5 Subplot Command

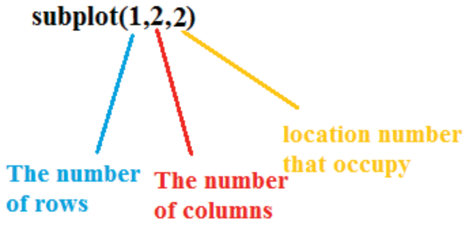
<u>Syntax</u>

Plot (number of rows, number of columns, the number of the matrix which occupy the figure)

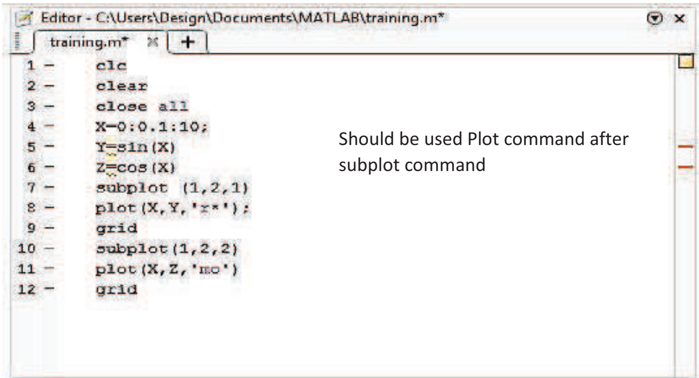
So the subplot command for the previous first figure can be written as:



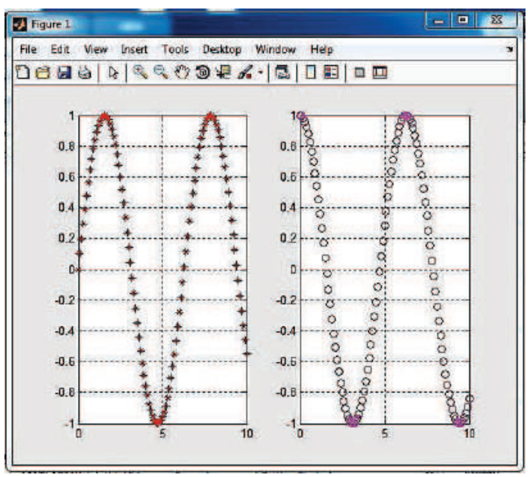
And the subplot command for the previous second figure can be written as:



Now we will write a full code:



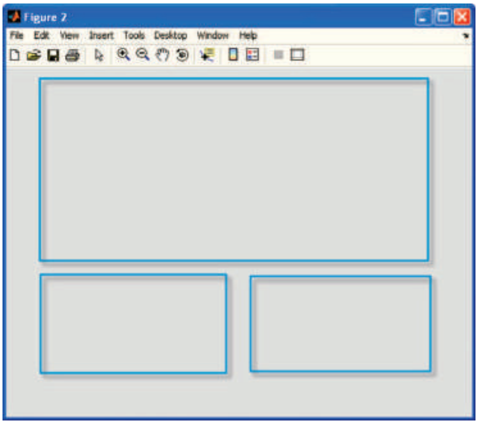
The execution will be as show below:



If the graphic occupies more than one location must use square brackets as the following formula:

[Number of all locations that occupies by figure]

For example, if we went the result as in the following figure:



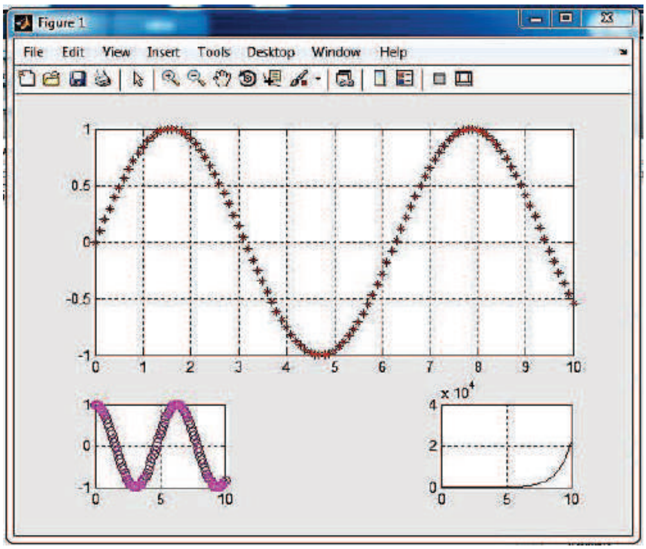
The number of rows 3 and number of columns 3, location numbers operated by first figure is 1, 2, 3, 4, 5, 6 sequentially, location 7 occupies the second figure and the third figure occupies location 9.the code of this example will be below:

Editor - C:\Users\Design\Documents\MATLAB\training.m*

training.m*

```
clc
clear
close all
X=0:0.1:10;
Y=sin(X)
Z=cos(X)
V=exp(X);
subplot (3,3,[1 2 3 4 5 6])
plot(X,Y,'r*');
grid
subplot(3,3,7)
plot(X,Z,'mo')
grid
subplot (3,3,9)
plot(X,V)
grid
```

In addition, the result will be in the following figure:



4-1.6 Naming Axes in MATLAB

The MATLAB program provides the ability of axis naming, for instance, if we want be naming the x-axis use the command **xlabel** and naming the y-axis use the command **ylabel** as show below:

Syntax:

xlabel(' the name of the axis')

ylabel(' the name of the axis')

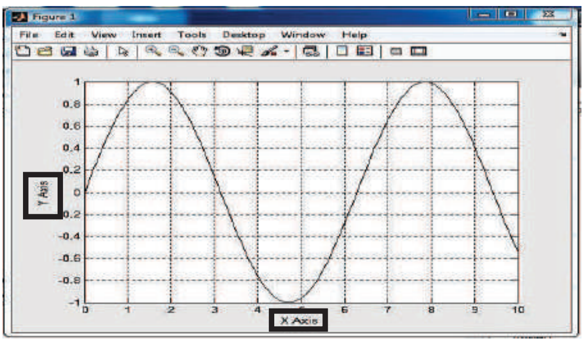
It must be the name between single quotations as shown in Fig.

Example:

Editor - C:\Users\Design\Documents\MATLAB\training.m

```
clc
clear
close all
X=0:0.1:10;
Y=sin(X)
plot(X,Y,'b');
xlabel('X Axis');
ylabel('Y Axis');
grid
```

The result is:



4-1.7 Put a Title of the Graph

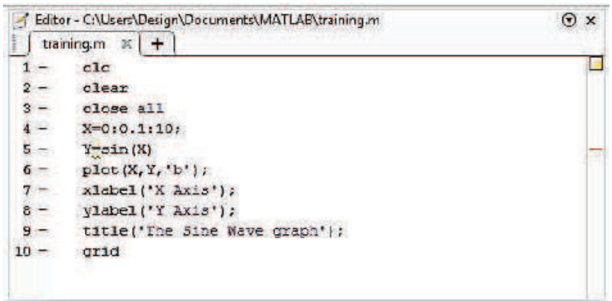
It could be the title of the graph by using the command **title**.

Syntax:

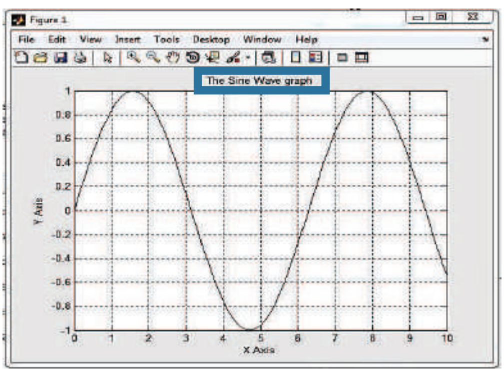
title('the title of the graph')

It must be the name between single quotations as shown in Fig.

By reference to the previous example and apply the title command:



The result is:

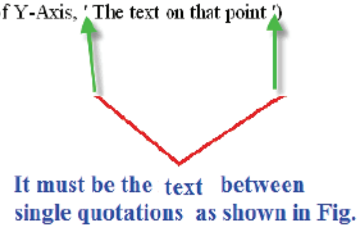


4-1.8 Put a Text to a Point or More Within the Graph

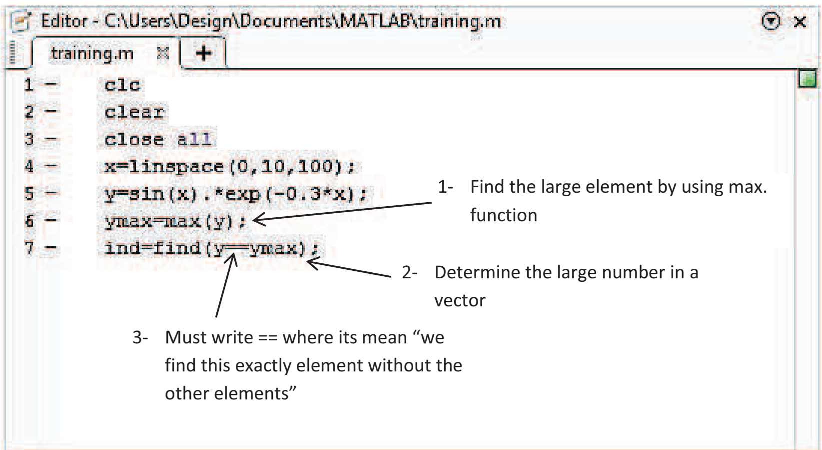
You can add text to point or more by using the command **text.**

Syntax

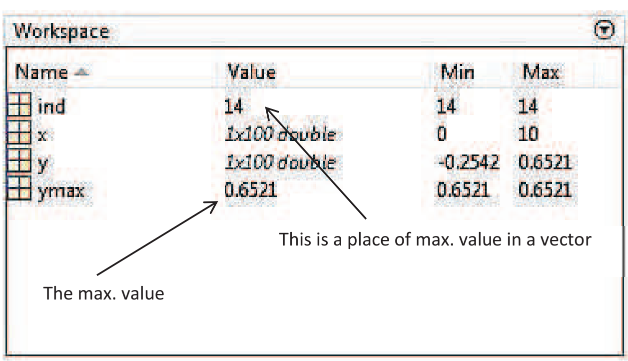
Text (position of the point at X-Axis, position of Y-Axis, ' The text on that point ')



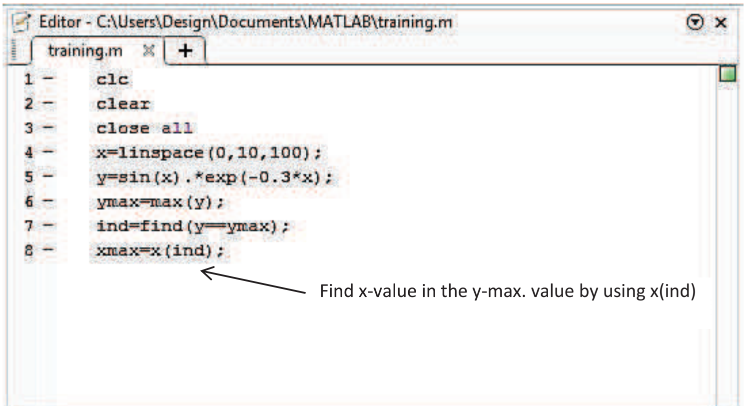
We will take a simple example of how to find the largest number, then put the red circle on max value and write maximum point. To find the coordinate of element in a vector we need command called **find** by determine the property of this element as show in this example:



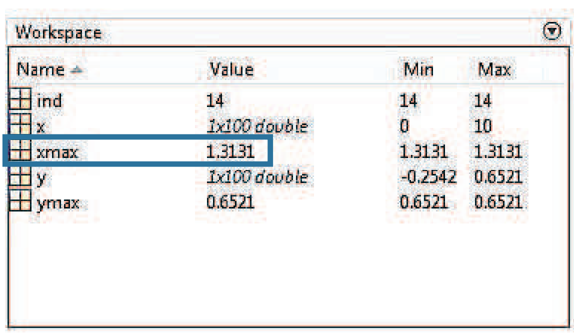
The result will appear as below:



If we need to find x-value in the y-max value write this code:



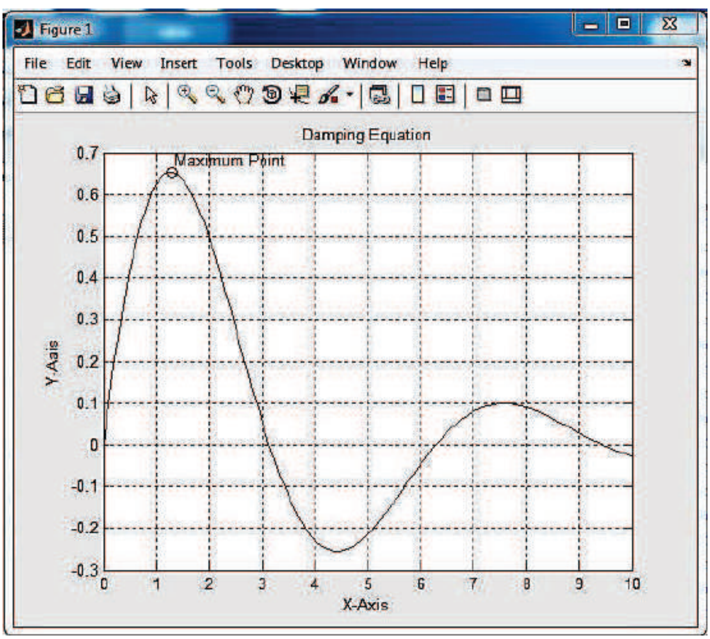
Also the x-value will appear in a workspace as below:



Now write the code and put "maximum value" on a max point

```
clc
clear
close all
x=linspace(0,10,100);
y=sin(x).*exp(-0.3*x);
ymax=max(y);
ind=find(y==ymax);
xmax=x(ind);
plot(x,y,xmax,ymax,'ro');
title('Damping Equation');
xlabel('X-Axis');
ylabel('Y-Aais');
grid
text(xmax+0.03,ymax+0.03,'Maximum Point');
```

The result is:



4-1.9 Legend Command

This command is used to explaining the meaning of each color in the graphic.

Syntax:

Legend ('the color reference')

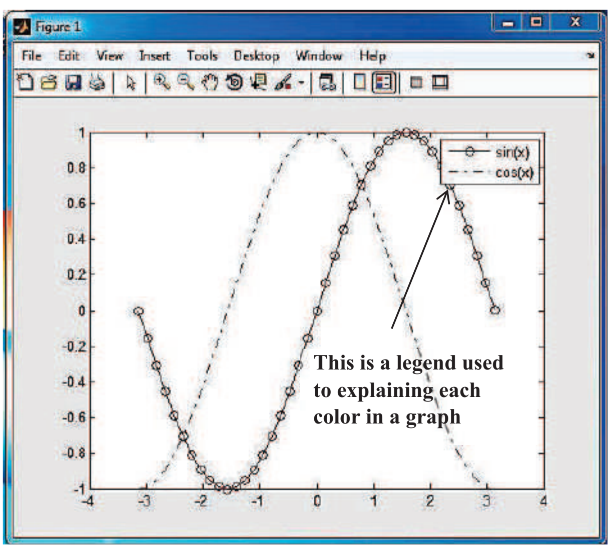
Example:

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
clc
clear
close all
x=-pi:pi/20:pi;
y1=sin(x);
y2=cos(x);
figure
plot(x,y1,'-ro',x,y2,'-.b')
legend('sin(x)','cos(x)')
```

The result is:

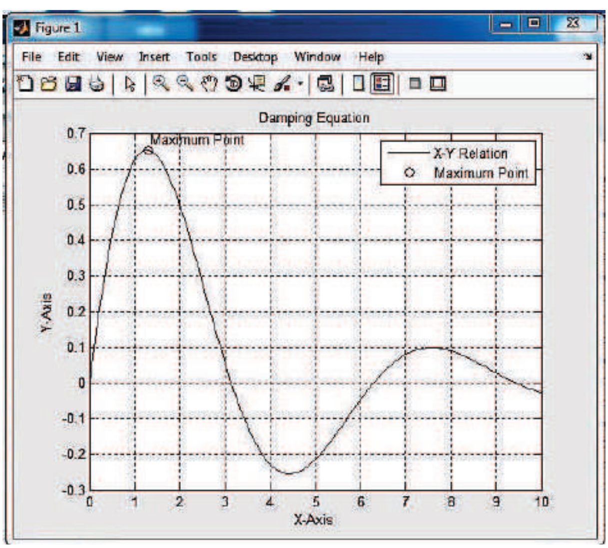


Example:

```
Editor - C:\Users\Design\Documents\MATLAB\training.m
training.m

clc
clear
close all
x=linspace(0,10,100);
y=sin(x).*exp(-0.3*x);
ymax=max(y);
ind=find(y==ymax);
xmax=x(ind);
plot(x,y,xmax,ymax,'ro');
title('Damping Equation');
xlabel('X-Axis');
ylabel('Y-Axis');
grid
text(xmax+0.03,ymax+0.03,'Maximum Point');
legend('X-Y Relation','Maximum Point');
```

1st relation | , | 2nd relation

The result is:



Note:

- Legend command depends on the number of drawn relationship in a graph.
- Legend command must be used after the plot command and not vice versa.

4-1.10 Open New Window and Determine its Resolution

MATLAB gives the ability to open a new window and determine the maximum and minimum values to the X-axis and the Y-axis by using **axis** command.

Syntax:

Axis ([minimum values of X, minimum values of X, minimum values of Y, minimum values of Y,])

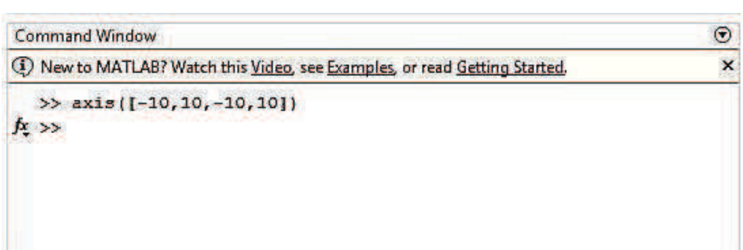
Example:

Open draw window has the following property:

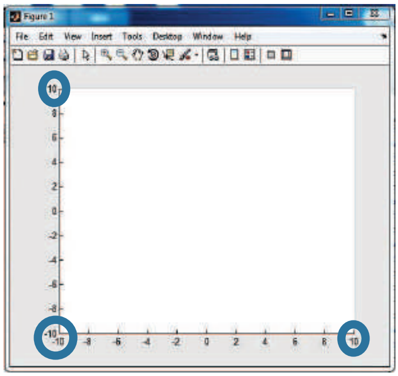
1- Minimum value of X-axis is 10.
2- Maximum values of X-axis is 10.
3- A minimum value of Y-axis is 10.
4- A maximum value of Y-axis is 10.

Solution steps:

In command window write the following code:



It will appear this window:

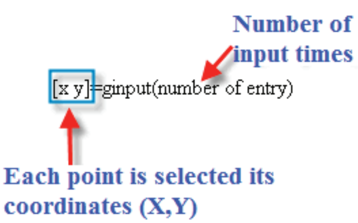


Now, you can put the property that you want on that window.

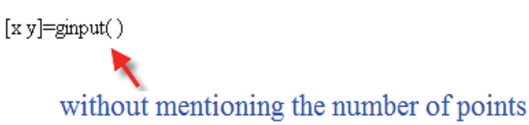
4-1.11 How to Enter the Points Through Mouse

MATLAB provides the ability to insert points using the mouse directly on the drawing window without writing the points in a vector or matrix. This trait is performed using **ginput** command.

Syntax:



However, if we want to enter an unlimited number of points we use the following formula (without mentioning the number of points):



When execution you can enter the points by using the mouse and when you're done, press the Enter key to exit.

Example:

We will open the drawing window, and then determine the minimum value for the X-axis is 0 and the maximum value of 10, as well as for the Y-axis. Then insert a large number of points using the command ginput, these points are printed in the form of red circles, it is as follows:

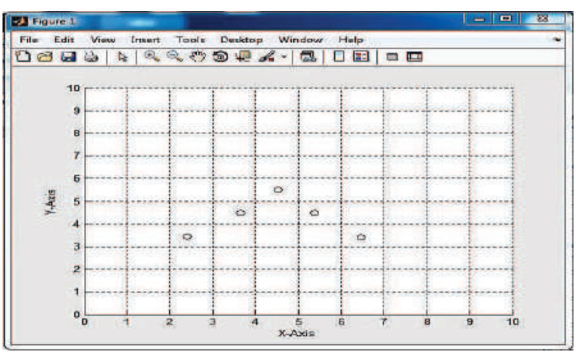
Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
clc
clear
close all
hold on
axis([0 10 0 10]);
xlabel('X-Axis');
ylabel('Y-Axis');
grid
[x y]=ginput(5);
plot(x,y,'ro')
```

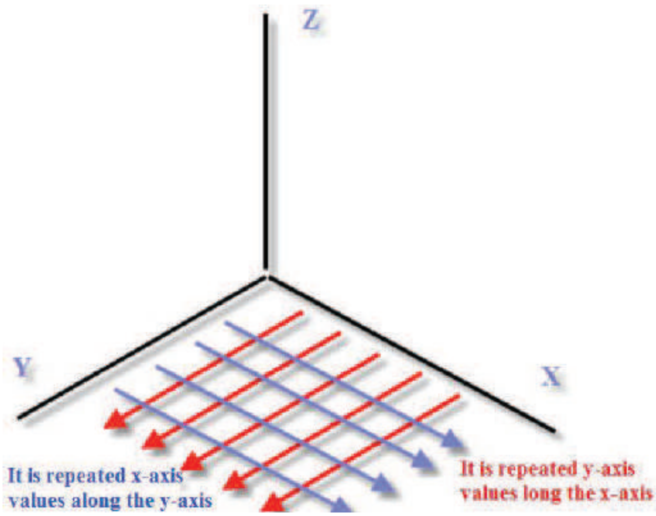
It must use the command hold on to be the points in the same window

Will show you a window to insert points, after the completion of the insertion of the desired points press the Enter key and window will appear as follows:



4-2 3D Plotting

As we have learned that the three-dimensional drawing based on three axes to draw it, X, Y and Z, where X, Y represent horizontal plane and Z represent vertical plane. But those points are the points on the axes, to it is drawing any point in the horizontal plane must be defined so as to MATLAB using the command **meshgrid**. Where the MATLAB product an array by repeat the x-axis value along y-axis and repeat the y-axis value along x-axis. Thus, the matrix produced in the Horizontal plane as show in the following figure:



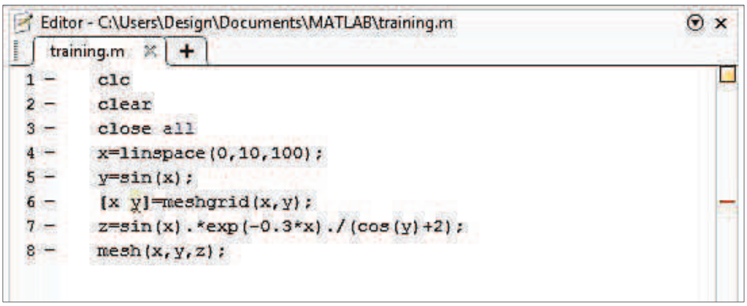
Syntax:

[x y]= meshgrid(x,y)

After using meshgrid, mech command is used as an alternative to plot command in the three-dimensional drawing.

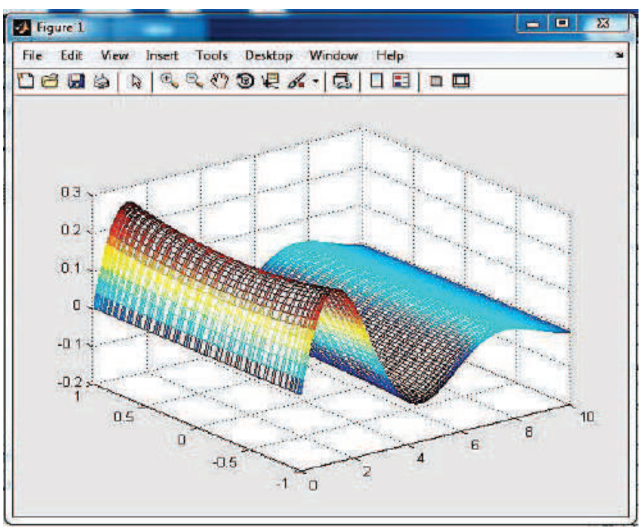
Example:

In this example we define the X-axis values, and define the equation that describes the Y-axis and its relationship to the X -axis, and then put the relationship between X -axis and Y -axis.

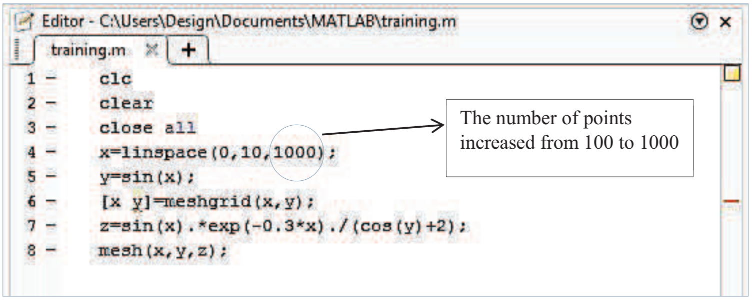



```
clc
clear
close all
x=linspace(0,10,100);
y=sin(x);
[x y]=meshgrid(x,y);
z=sin(x).*exp(-0.3*x)./(cos(y)+2);
mesh(x,y,z);
```

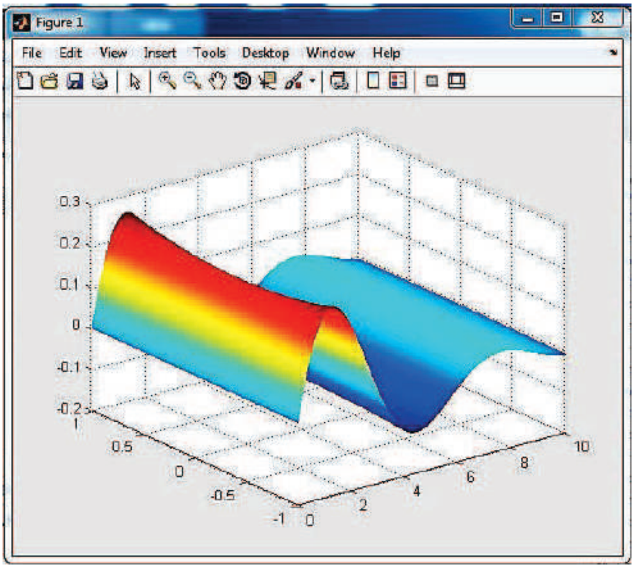
The result is as follows:



Graphic output is a grid based on a set of points X and Y, if the number of points of X increased that also increases points of Y.



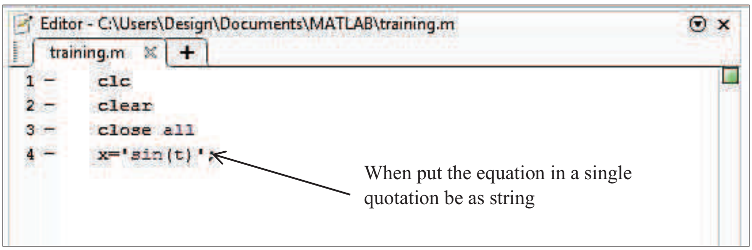
The result is as follows:



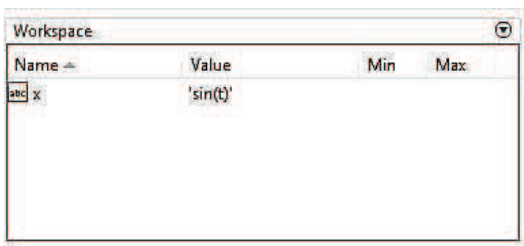
Note: When you increase the number of points will be delayed execution.

4-3 Eval Command

Before you start to explain this command, this example shows that the benefit of it. Let's say we have a sine equation, but has been write in the following shape:

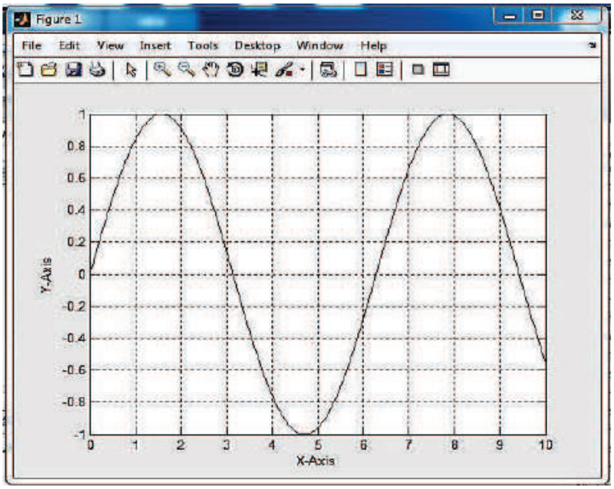


To make sure that the equation is the strings are going to workspace:



To draw the sin wave, must define t values and compensation in it, but its difficult because the equation between single quotation which is a barrier to compensate. Therefore, we need to break that barrier by using **eval** command, where this command search for equation between single quotation and search for the values that used in an equation to compensate in it. Eval command writes as in the following code:

The result is:



Chapter Five: Decision Making

All the problems you have solved so far have been problems with a straight-line logic pattern i.e. You followed a sequence of steps (defining variables, performing calculations, and displaying results) that flowed directly from one step to another. Decision making is an important concept in the programming and allows you to control which part of your code should execute depending on certain condition .This flow of control in your program can be performed by branching with it and else statements ,which will be discussed in this lecture ,or looping ,which will be discussed later.

5-1 Relational and Logical Operations

Relational and Logical operators are used in branching and looping to help making decisions. The result of using relational or logical operator will always be either true, given by 1, or false given by 0. Tables 8-1, 8-2 lists the most common relational and logical operators in MATLAB.

Relational operator

Operator	**Mathematical symbol**	**MATLAB symbol**
Equal	=	= =
Not Equal	≠	~ =
Less Than	<	<
Greater Than	>	>
Less than or equal	≤	<=
Greater than or equal	≥	>=

Logical Operator

Operator	Mathematical symbol	MATLAB symbol
And	AND	&
Or	OR	$\vert$
Not	NOT	~

Example

```
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
>> x=5;
>> y=10;
>> x<y

ans =

     1

>> x>y

ans =

     0

fx >>
```

Comments

- Lines 3 and 6 are called logical expressions because the result can only be either true represented by 1, or false represented by 0.

More examples of using relational expressions

```
Command Window
>> x=[1 5 3 7];y=[0 2 8 7];k=x<y

k =

     0     0     1     0

>> k=x<=y

k =

     0     0     1     1

>> k=x>y

k =

     1     1     0     0

>> k=x==y

k =

     0     0     0     1
```

```
>> k=x~=y

k =

     1     1     1     0

fx >>
```

5-2 Logical Operators

```
Command Window
>>  x=[1 5 3 7], y=[0 2 8 7],k=(x>y)&(x>4)

x =

     1     5     3     7


y =

     0     2     8     7


k =

     0     1     0     0

fx >>
```

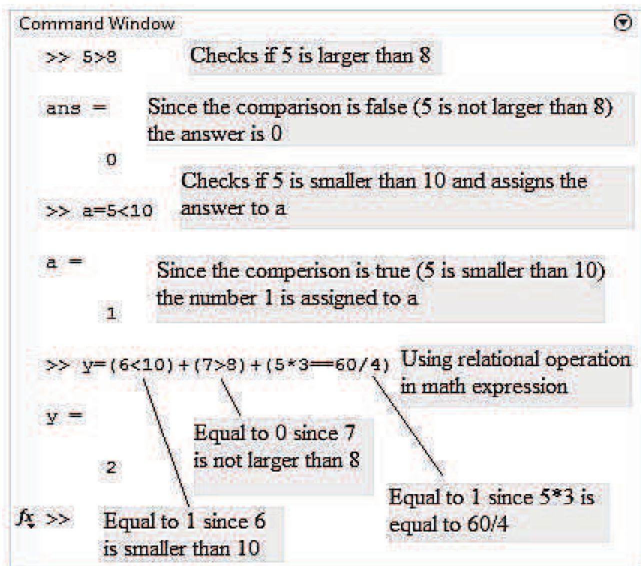
Comments

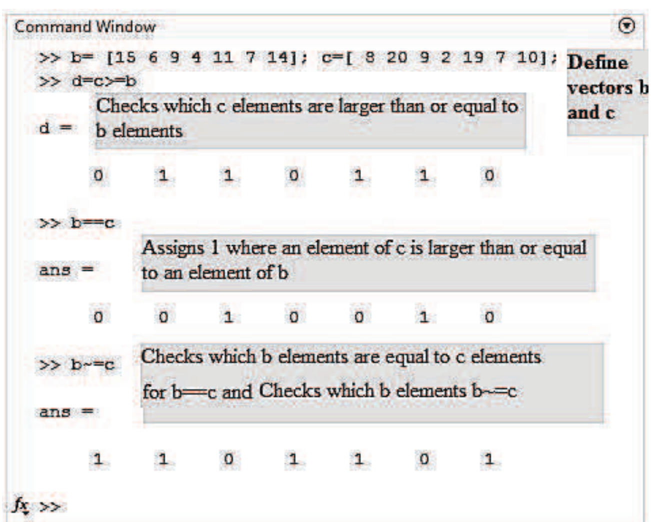
-the Relational and logical operators are used to compare, element by element, the vectors x and y.

Single and double

The difference between = and = = is often misunderstood .A single equals sign is used to assign a value to a variable e.g. x= 5. A double equal sign is used to test whether a variable is equal to given value e.g. my_ test = (x= = 5) means test if x is equal to 5, and if so assign the value 1 (true) to my _test.

Some examples





```
Command Window
B =

     1     0     0
     1     0     1
     0     0     1

fx >>
```

The results of relational operation with vectors ,which are vectors with 0s and 1s ,are called logical vectors and can be used for addressing vectors .When a logical vector is used for another vector ,it extracts from that vector the elements in the positions where the logical vectors has 1s.For example :

```
Command Window
>> r=[8 12 9 4 23 19 10];          Define a vector r
>> s=r<=10

s =              Checks which r elements are smaller than or equal to 10

     1     0     1     1     0     0     1
                 A logical vector s with 1s at positions where elemnts of r
>> t=r(s)        are smaller than or equal to 10
                 Use s for addresses in vector r to create vector t
t =

     8     9     4    10

>> w=r(r<=10)           Vector t consists of elements of r in positions
                        where s has 1s
w =              The same procedure can be done in one step

     8     9     4    10

fx >>
```

Exercises

1-Evaluate the following expressions without using MATLAB .Check your answer with MATLAB.

a) 14>15/3

b)$y = \frac{8}{2} < 5 \times 3 + 1 > 9$

c)$y = 8/(2 < 5) \times \ 3 + \ \ (1 > 9)$

d) $2 + 4 \times 3 \sim = 60/4 - 1$

2- Given a=4, b=7 .Evaluate the following expressions without using MATLAB .Check your answer with MATLAB.

a)$a + b \ >= a \times b$

b) $y = a + (b >= a) \times b$

c)$y = b - a < a < a/b$

3-Given v= [4 -2 -1 5 0 1 -3 8 2], and w= [0 2 1 -1 0 -2 4 3 2].Evaluate the following expressions without using MATLAB .Check your answer with MATLAB .

a) v < = w

b) w = v

c) v < w + v

d) (v< w)

5-3 The if-else Statements

The if –else and elseif statement in MATLAB provide methods of controlling which part s of your code should execute based whether certain conditions are true or false. The syntax of the simplest form of an if statement is given as:

The Form of Conditional Statements

If conditional exepression

Consisting of relational and/ or logical operators

Examples

$if\ a < b$

$if\ c \geq 5$

$if\ a == b$

$if\ a \cong 0$

$if\ (d < h)\&(x > 7)$

$if\ (x\sim = 13)\ |\ (y < 0)$

All Variables must have assigned values

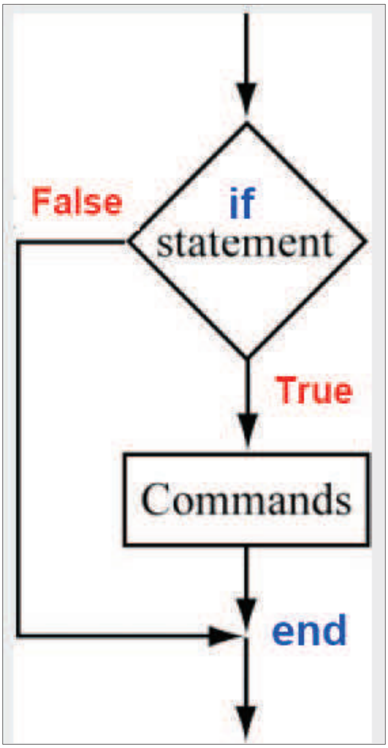
THREE FORMS OF THE if STATEMENT

```
If  conditional  statement
Commands
    end
```

```
if conditional statement
command group 1
else
command group 2
end
```

```
if conditional statement 1
command group 1
elseif conditional statement 2
nd group 2
else
command group 3
end
```

5-3.1 The if –end Statement



....

....

....

If Conditional expression

....

....

....

end

....

....

....

MATLAB Program

A group of MATALB Commands

MATLAB Program

Example of using the if- end statement

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

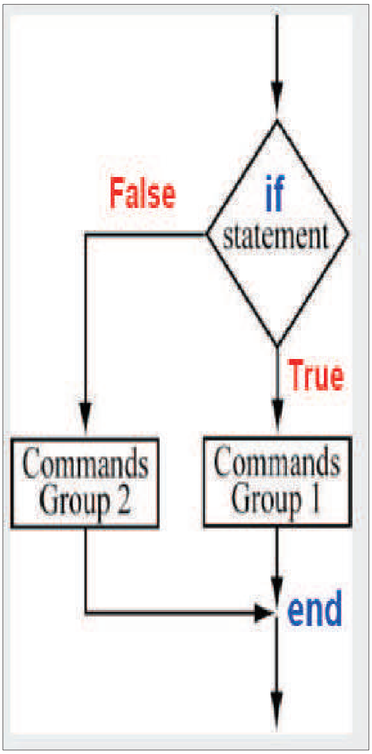
```
% A script file that demonstrates the use of the if-end statements
% The user is asked to enter three grades
% The program calculates the average of the grades
% if the average is less than 60, a massage:
% The student did not pass the course. is printed
score = input ('enter (as a vector ) the scores of the three tests');
ave_grade= (score(1)+score(2)+score(3))/3;
disp('The average grade is : ')
disp(ave_grade)
if ave_grade<60
    disp('The student did not pass the course.')
end
```

Executing the script file in Command window:

Command Window

```
>> training
enter (as a vector ) the scores of the three tests [ 78 61 85]
The average grade is :
   74.6667

>>
>> training
enter (as a vector ) the scores of the three tests [60 38 55]
The average grade is :
    51

The student did not pass the course.
fx >>
```

5-3.2 The if-else-end Statement



………

……….

MATLAB Program

……….

If Conditional expression

……….

……….

Group 1 of MATALB Commands

else

…….

……….

Group 2 of MATALB Commands

End

………

……….

Example of Using The if-else-end Statement

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

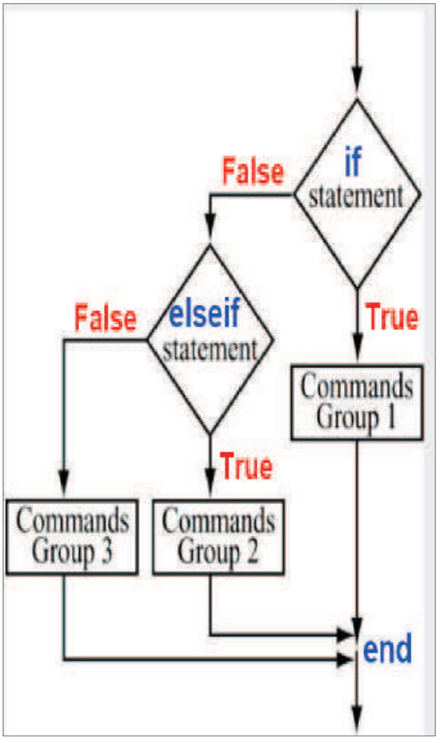
```
% A script file that demonstrates the use of the if-else-end statements
% The user is asked to enter three grades
% The program calculates the average of the grades
% if the average is less than 60, a massage:
% The student did not pass the course. is printed
score = input ('enter (as a vector ) the scores of the three tests');
ave_grade= (score(1)+score(2)+score(3))/3;
disp('The average grade is : ')
disp(ave_grade)
if ave_grade<60
    disp('The student did not pass the course.')
else
    disp('The student passed the course.')
end
```

Executing the script file in the command window:

Command Window

```
>> training
enter (as a vector ) the scores of the three tests [ 65 80 83]
The average grade is :
    76

The student passed the course.
>> training
enter (as a vector ) the scores of the three tests [60 40 55]
The average grade is :
   51.6667

The student did not pass the course.
>>
```

5-3.3 The if-elseif-else-end Statement



MATALB Program

....

If Conditional expression

....

Group 1 of MATALB Commands

....

Elseif Conditional expression

....

Group 2 of MATALB Commands

....

else

....

Group 3 of MATALB Commands

......

End

....

MATALB Program

....

Example of if-elseif-else-end statement

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
% A script file that demonstrates the use of
% the if-elseif-else-end statements
% The program calculates the tip in a restaurant
% according to the amount of the bill
% if the bill is less than 10$ the tip is 1.8$
% between 10$ and 60$ the tip is 18% of the bill
% Above 60$ the tip is 20% of the bill
bill = input ('Enter the amount of the bill (in dollars):');
if (bill<=10)
    tip=1.8;
elseif (bill>10)&(bill<=60)
    tip =bill*0.18;
else tip=bill*0.2;
end
disp('The tip is (in dollars);')
disp(tip)
```

Executing the script File of The Restaurant Tip Calculation

Command Window

```
>> training
Enter the amount of the bill (in dollars):15
The tip is (in dollars);
    2.7000

>> training
Enter the amount of the bill (in dollars):6
The tip is (in dollars);
    1.8000

>> training
Enter the amount of the bill (in dollars):100
The tip is (in dollars);
    20

fx >>
```

Comments about if-end Statements

-For every *if* command a computer program must have an *end* command.

- A program can have many *if ....end* statements following each other.

-A computer program can perform the same task using different combination of *if-end* ,*if-else-end*,and*if-elseif-else-end* statements

Some more examples

$$y = \begin{cases} \sqrt{x} & \text{for} \quad x \geq 0 \\ e^x - 1 & \text{for} \quad x < 0 \end{cases}$$

```
if x >= 0
   y = sqrt(x)
else
   y = exp(x) - 1
end
```

$$y = \begin{cases} \ln x & \text{if} \quad x \geq 5 \\ \sqrt{x} & \text{if} \quad 0 \leq x < 5 \end{cases}$$

```
if x >= 5
   y = log(x)
else
   if x >= 0
      y = sqrt(x)
   end
end
```

```
if x >= 5
   y = log(x)
elseif x >= 0
   y = sqrt(x)
end
```

5-4 The Input Function

The input function is used to request user input and assign it to a variable.*For example x=input ('Enter a number : ')* ; will display the text *Enter the number :*in the command window and wait until the user enters something .Whatever is entered will assigned to the variable *x*.

5-5 The disp Function

The disp function can be used display strings of texts to the command window e.g. disp ('I am a string of text'). You can also display numbers by converting them to string e.g. disp (num2str(10)).The num2str function simply converts the number 10 to a string that can be displayed by the disp function .You can also combine the display of text and numbers e.g.disp (['Factorial' num2str 'is' num2str (y)]) .Notice the use of spaces to denote the separate elements of the string, and square brackets around the string to concatenate it together.

Comments

- Output is displayed automatically if a statement does not end with a semicolon.

Output can also be displayed intentionally by using the **disp** command

Syntax

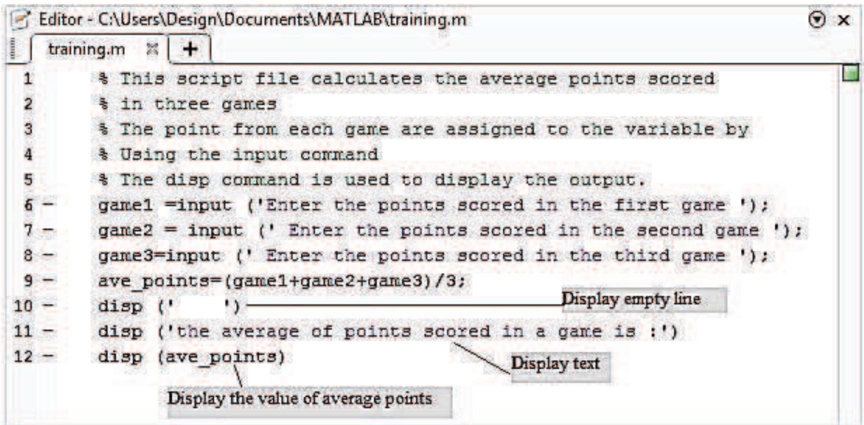
Disp (A) %Displays the content, but not the name, of the variable A

display ('text') %display the text (string) that is enclosed within the single quotes .

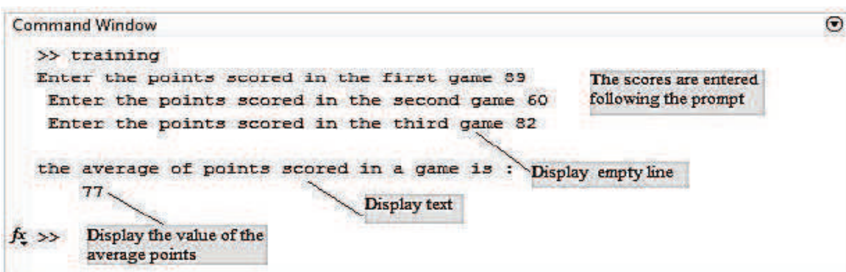
String

Every time a *disp* command executed, the display it generates appears In a new line.

Example of script file that uses the input and *disp* commands



Running the script file with the input and disp commands in the command window



5-6 fprintf Command

The *fprintf* (formatted print function) statement provides control that numeric and string data are printed to the command window or a file. The *fprintf* command gives you more control over the output than the *disp* command you can specify text and matrices to be printed. The general syntax :

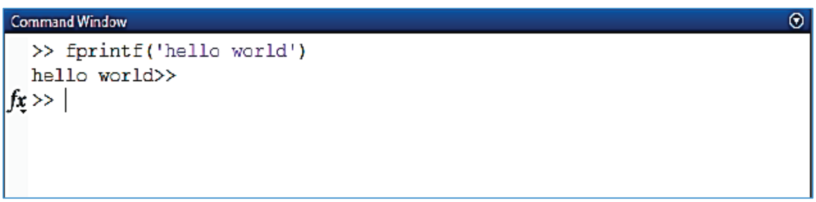
fprintf (' formatting structure commands ', values to be displaced) ;

5-6.1 Using fprintf command to display text.

The syntax is,

Fprintf ('text to be displaced')

Example



The following are list of useful space characters for **fprintf,**

Code	Result	Example
\ n	Jumps a line after the displayed text	fprintf('text to be type\ n')
\ b	Leaves backspace after the displayed text	fprintf('text to be typed \ b ')
\ t	Leaves tab after the displayed text	Fprintf('text to be typed \ t')

5-6.2 Using the fprintf to Display Numbers

Syntax

fprintf (' formatting structure commands', values to be displayed)

Inside the format statement the %f , %e , and %g specifiers are used to show where and how the output values are displayed .

code	Result
% f	Decimal format
% e	Exponential format
% g	Whichever is shorter

Example 1

practice.m

```
clc
clear
x=[1.1 2.2 3.3];
y=2*x;
fprintf('hello.(%1.3f,%1.3f),(%1.1f,%1.0f)\n',...
    x(1),y(1),x(3),y(3));
```

Executing the practice script file in the command window:

Command Window

```
hello.(1.100,2.200),(3.3,7)
fx >>
```

Example2

practice.m

```
% program Calculation
clc
clear
r= input('Please Enter the radius of the circumfrence in cm =');
A=pi*r^2
fprintf('the area of circumfrence is %f in cm^2 ',A);
```

This is the format

This is what will be displayed on the command screen :

Command Window

```
Please Enter the radius of the circumfrence in cm = 2

A =

  12.566370614359172

the area of circumfrence is 12.566371 in cm^2 >>
>> |
```

To control the number of decimal places displayed within the % f or % e specifies. In the following example, the 4.2 means allot four places for the value, 2 to the right of the decimal point.

practice.m*

```
% program Calculation
clc
clear
r= input('Please Enter the radius of the circumfrence in cm =');
A=pi*r^2
fprintf('the area of circumfrence is %4.2f in cm^2 ',A);
```

Formatting the number

This is what will be printed:

```
Command Window
Please Enter the radius of the circumfrence in cm = 2

A =

  12.566370614359172

the area of circumfrence is 12.57 in cm^2 >>
fx >>
```

Using the Scientific notation with the % e specifier

```
practice.m
% program Calculation
clc
clear
r= input('Please Enter the radius of the circumfrence in cm =');
A=pi*r^2
fprintf('the area of circumfrence is %4.2e in cm^2 ',A);
```

The display is

```
Command Window
Please Enter the radius of the circumfrence in cm = 2

A =

  12.566370614359172

the area of circumfrence is 1.26e+01 in cm^2 >>
fx >> |
```

Exercises

1-The tank in a water tower has the geometry shown in the figure below, the lower part is a cylinder and the upper part is an inverted frustum cone. Inside the tank there is a float that indicates the level of the water .Write a user defined function that determines the volume of the water in the tank from the position (height) of the float. The volume for the cylindrical section of the tank is given by:

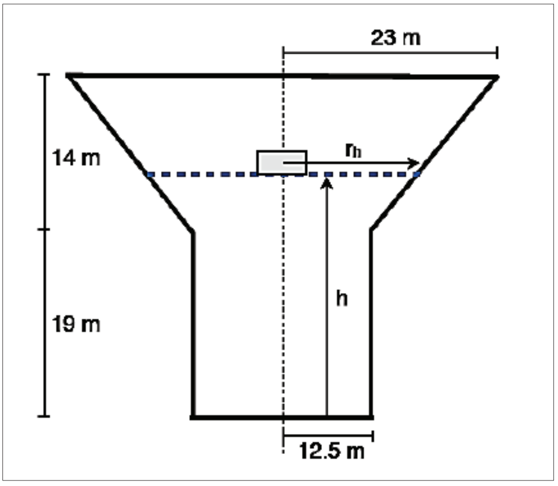
$$V = \pi.12.5^2.h$$

The volume for the cylindrical and conical sections of the tank is given by:

$$V = \pi.12.5^2.19 + \frac{1}{3}\pi(h - 19)(12{,}5^2 + 12.5r_h + r_h{}^2)$$

Where r_h=12.5+$\frac{10.5}{14}$ $(h - 19)$

Ans.[h=8m ,V=3927m^3; h=25.7m ,V=14115 m^3]



Water level in the in a water tower

2- Write a function to evaluate f(x,y)for any two user specified values x and y .The function f(x , y) is defined as :

$$f(x,y) = \begin{cases} x + y & x \geq 0 \; and \; y \geq 0 \\ x + y^2 & x \geq 0 and \; y < 0 \\ x^2 + y & x < 0 \; and \; y \geq 0 \\ x^2 + y^2 & x < 0 \; and \; y < 0 \end{cases}$$

3-write a script file that asks the user for the input of a number and returns the natural logarithm of the number if the number is positive, and displays error message otherwise.

5-7 Roots of Polynomials

In many engineering problems, there is a need to find the roots of a polynomials *P(s)*, which are the values of s for which *P(s) = 0* .When P(s) is of degree N, then there are exactly N roots.

Syntax

roots (a)

Returns as a vector the roots of the polynomial represented by the coefficient vector (a).

Example

Find the roots of polynomial

D(s) =$s^2 + 6s + 9$

Sol.

```
Command Window
>> d=[1 6 9];roots(d)

ans =

  -3.0000 + 0.0000i
  -3.0000 - 0.0000i

fx >>
```

Example

Find the roots of the cubic polynomial

$$P(s) = s^3 - 2s^2 - 3s + 10$$

Sol.

```
Command Window
>> P=[1 -2 -3 10];
>> R=roots(P)

R =

  -2.0000 + 0.0000i
   2.0000 + 1.0000i
   2.0000 - 1.0000i

>>
```

5-7.1 Value of Polynomial

MATLAB can also compute the value of a polynomial at point x using function

Syntax

Polyval (p,x)

P= is a vector with the coefficients of the polynomial.

X= is a number, variable or expression.

Example

F(x) = $5x^3 + 6x^2 - 7x + 3$

```
Command Window
>> P=[5 6 -7 3]

P =

     5     6    -7     3

>> x=2

x =

     2

>> Y=polyval(P,x)

Y =

    53

fx >>
```

5-7.2 Derivatives of Polynomials

MATLAB can also take the derivatives of polynomials

Syntax

k=polyder (p)

Where:

p=is the coefficient vector of the polynomial

k= is the coefficient vector of the derivative

Example

F (x) = $3x^2 - 2x + 4$

```
Command Window
>> p=[3 -2 4];
>> k=polyder(p)

k =

     6    -2

>> % dy/dx= 6x -2
```

Exercises

1-Find the roots of the polynomial $p(x) = x^3 - 3x^2 + 1$

Ans =2.8797, 0.6527, -0.5321

2- For the polynomial $(x) = x^4 + 7x^2 - x$, find q(x) value at x= -1

Ans =0,0.0712+2.6486i,0.0712-2.6486i,-0.1424,-7

3- Find the derivative of the polynomial $p(x) = x^2 - 3x + 5$

Ans=2 -3

Chapter Six: Loops

Loops are another way of altering the flow of control in your program, and provide methods for repeatedly executing commands .You might want to repeat the same commands, changing the value of a variable each time, for a fixed number of iterations. Alternatively, you may want to repeat the same command, changing the value of a variable each time, continually until a certain condition is reached. Two of the most common types of loops, **for** and **while**, will be examined.

6-1 For Loops

A *for* loops is used to repeat a command, or set of commands, a fixed number of times.

Syntax

1 variable = f :s:t

2 statements

3 end

F: is the value of the loop counter on the first iteration of the loop.

S: is the step size or increment

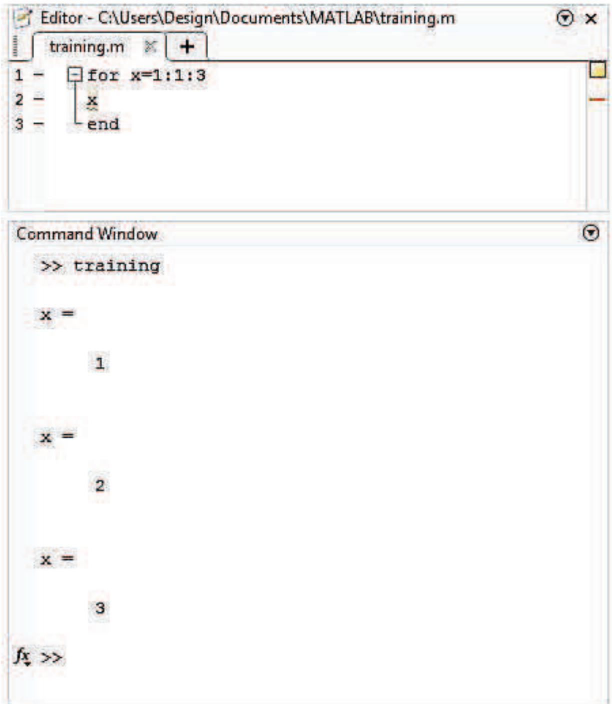
T: is the value of the loop counter on the final iteration of the loop

Line 1 contains the four command, followed by the loop counter variable which is defined by an expression. As an example if the loop counter is defined by the expression n=0:5:15 Which means n= [0 5 10 15], on the first iteration of the loop n=0,on the second iteration n=5, on the third iteration n=10,on the fourth and final iteration n=15.

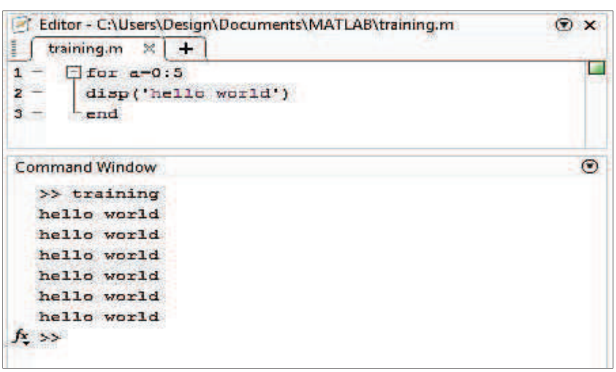
Line 2 contains the body of the loop which can be a command or series of commands that will be executed on each iteration of the loop.

Line 3 contains the *end* command which must always be used at the end of a loop to close it.

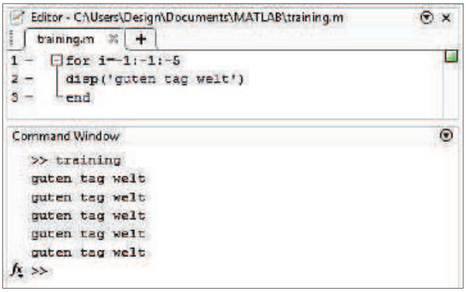
Example



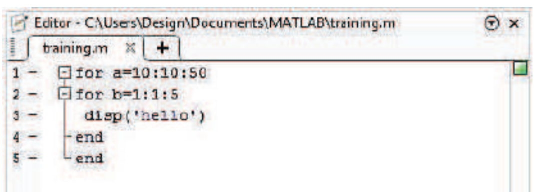
1- How many times will this code print hello world, check your answer by MATLAB



2- How many times will this code print Guten tag welt

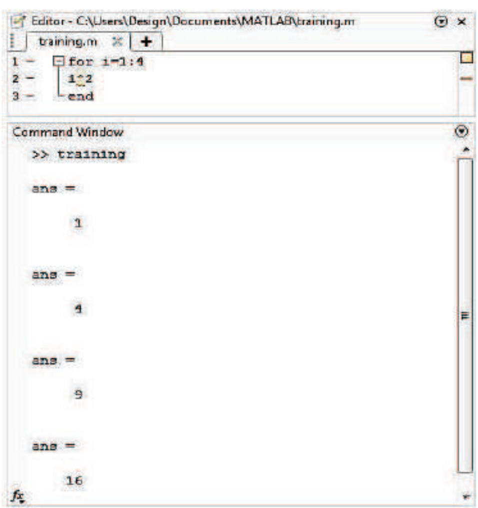


3- How many times this code print hello



<u>Example</u>

Squaring the numbers 1through 10in a loop is performed as:



Loop variable are used to index into arrays and matrices

Example

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
a= [3,40,1,8,0];
b= [-7,5,-0.4,16,1];
c= [ ];
for i=1:5
    c(i)=a(i)+b(i)
end
```

Sol.

Command Window

```
>> training

c =

    -4


c =

    -4    45


c =

   -4.0000   45.0000    0.6000


c =

   -4.0000   45.0000    0.6000   24.0000


c =

   -4.0000   45.0000    0.6000   24.0000    1.0000

>>
```

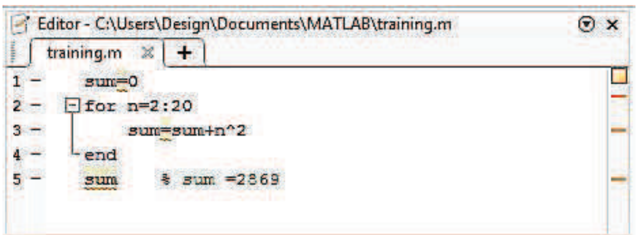
Example

Write a *for* loop to compute the sum of squares of all integers from 2 to 20.

$2^2 + 3^2 + 4^2 + \cdots .. + 20^2$

Sol.

We will initialize a variable sum for cumulative sum throughout the loop (the initial value is zero)



6-2 While Loops

A *while* loop is similar to for loop in that it is used to repeat a command or set of command, the key difference between a *for* loop and a while loop is that the while loop will continue to execute until specified condition become false.

Syntax

while condition is true

statements

end

Line 1 contain the while command, followed by a condition e.g. x>10 .this means as long as the condition, x >10 remains true the loop will continually repeat.

Line 2 contains the body of loop, which can be a command or seat of commands that will be executed on each iteration of the loop.

Line 3contains the end command, which must always be used at the end of the loop to close it.

Example

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
x=1
while x<5
    x
    x=x+1
end
```

```
>> training

x =

     1


x =

     1


x =

     2


x =

     2


x =
x =

     3


x =

     4


x =

     4


x =

     5

>>
```

Line 1 assigns the value of 1 to the variable x. notice this is outside of the while loop. If you don't do this you will get an error because you are testing weather x<5, but x has never been defined.

In line 2 the condition , x<5 ,is specified .in this case the loop will continue to repeat as long as x is less than 10 .as soon as x is equal to 10 .as soon as x=10 , execution of the loop is stopped.

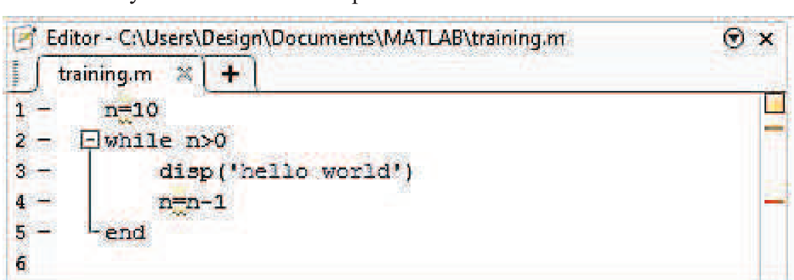
Line 3-4 contain the body of the loop ,in this case the value of the loop counter variable x is printed , then the value of x is incremented by 1 .the value of x must be explicitly incremented otherwise x will always be 1 ,the condition x<5 will always be true , and the loop will therefore execute continuously.

6-3 Breaking Out of the Loop

If you end up stuck in an infinitely repeating loop, use *CTRL+C* to force MATLAB to break out of the loop. however, under certain conditions you may want to break out of a loop before it is finished .to do this you can use *break* command .statements in your loop after the *break* command will not be executed and passes control to the next statement after the end of the loop.

Examples

1- How many times will this code print *hello world*?



2- How many times will this code print *hello world*?

```
n=1
while n>0
    disp('hello world')
    n=n+1
end
```

3-What values will this code print?

```
n=1
while n<100
    disp('hello world')
    n=n*2
end
```

Example

Calculate the summation of 1+2+3+….+100 .

Sol.

```
sum=0
current=1
while current <=100
sum =sum +current
current=current+1
end
```

6-4 Converting the for Loop to a While

Essentially every for loop can be written as while loop .this is a three steps process:

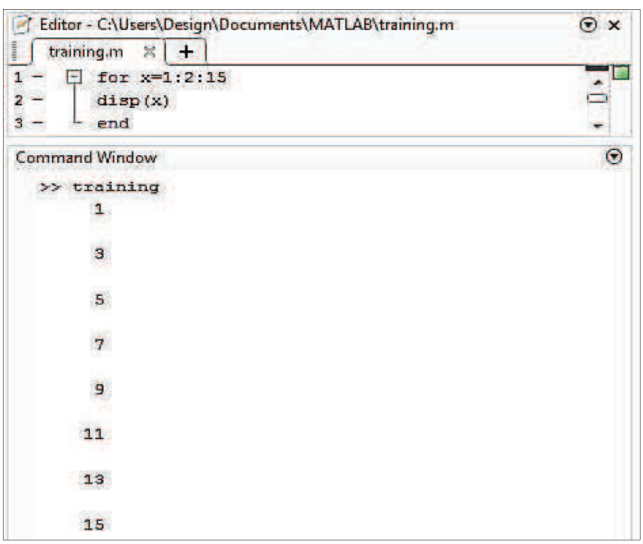
1-notice that we need to initialize a loop variable (a while loop does not do this automatically). If our loop began for x = 1:2:15, we must state that x=1 initially, before our *while* loop begins.

2- you must create a condition that is true while you want the loop to keep looping ,and that became false when you want the loop to stop, usually ,this is the upper (or the lower) bound on your loop variable .in our example ,we'd want to keep looping while *x* is less than or equal to 15 :x<=15

3-Finally, before each iteration of our *while* loop ends, we must increment our loop variable .Notice that a for loop did that for us .right before your *end* statement, you 'll likely place something like x=x+2 if you were converting the above example.

<u>Example written as both a for loop and equivalent while loop</u>

For loop



While loop

```
x=1   % step 1
while(x<=15)   % step 2
    disp(x)
    x=x+2   % step 3
end
```

6-5 Nested Loops

You can also put loops (for or while) inside of each other, in what are called *nested loops*. These are very powerful, especially when working with matrices

Syntax

```
for m=1:j
for n=1:k
statements
end
end
```

The syntax for a nested while loop in MATLAB as follows

```
while  expression 1
while  expression 2
statements
end
end
```

Example

```
Editor - C:\Users\Design\Documents\MATLAB\training.m
training.m

for i=1:4;
for j=1:4;
X(i,j)=i*j
end
end
```

```
Command Window
>> training

X =

     1     2     3     4
     2     4     6     8
     3     6     9    12
     4     8    12    16
```

Example

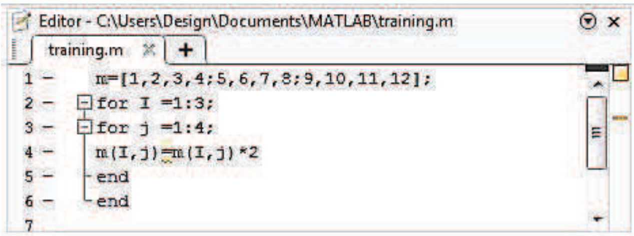
Form a 5x6 matrix, where each element in the matrix is the row number of a power of column number.

Sol.

```
Editor - C:\Users\Design\Documents\MATLAB\training.m
training.m

clear
clc
n=5;
m=6;
for i=1:n;
for j=1:m;
c(i,j)=i^j
end
end
```

While a vector has one dimension over which a loop variable can iterate, a matrix has two dimensions: rows and columns. To iterate over an entire matrix, we need to iterate over every row and for each row over every column. This is called a nested loop i.e. a loop within a loop.

Example



The resulting matrix will be :

$$\begin{bmatrix} 2 & 4 & 6 & 8 \\ 10 & 12 & 14 & 16 \\ 18 & 20 & 22 & 24 \end{bmatrix}$$

Exercises

Write your own scripts to perform the following tasks:

A) A for loop that multiplies all even numbers from 2 to 10.

B) A while loop that multiplies all even numbers from 2 to 10.

C) Given the vector x= [1 8 39 0 1] use a for loop to:

1-add up the values of all elements in x.

2-compute the cumulative sum i.e. 1,9,12,21,21,22 of the elements in x.

Chapter Seven: Applications

7- 1 Problems in Statics:

Are presented on the following common topics: forces on a particle, rigid bodies and equivalent forces, equilibrium of rigid bodies, truss analysis, beams, friction and distributed forces (centroids).

Example 1: Write a MATLAB program to determine the magnitude and direction of the resultant of 3- coplanar forces applied at point A in Fig. ES7.1. Use the following values:

F1 = 20 kN, F2 = 40 kN, F3 = 200 kN, α1 = 40º, α2 = 25º and α3 = 58º.

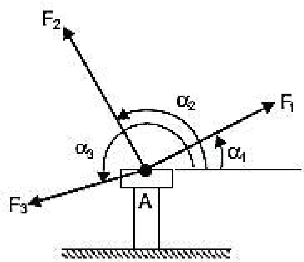


Fig. 1

Solution: We know that for a coplanar force system

$$R_x = \sum_{i=1`}^{n} F_i \cos \propto_i \qquad R_{xy} = \sum_{i=1`}^{n} F_i \sin \propto_i \qquad \dots\dots(1)$$

Therefor $$R = \sqrt{R_x^2 + R_y^2} \qquad \dots\dots(2)$$

And $$\propto_R = \tan^{-1}\frac{R_y}{R_x} \qquad \dots\dots(3)$$

Let $\propto_R^*$ be the value defined by Eq. (3) and such that $-90^{\circ} \leq \propto_R^* \leq 90^{\circ}$. Then we have

If $R_x \geq 0 \; and \; R_y \geq 0 \qquad \propto_R = \propto_R^* \qquad \dots\dots(4)$

If $R_x \geq 0 \; and \; R_y < 0$ $\propto_R = 360^{\circ} + \propto_R^{*}$ $\ldots \ldots .(5)$

If $R_x < 0$ $\propto_R = 180 + \propto_R^{*}$ $\ldots \ldots .(6)$

MATLAB Solution:

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
n=3; % Number of forces
alpha= [40 25 58];
alpha1=alpha*pi/180;
force= [20 40 200];
sumx=0;
sumy=0;
for i=1:n
sumx=sumx + force (i)*cos (alpha1 (i));
sumy=sumy + force (i)*sin (alpha1(i));
end
r=sqrt(sumx^2+sumy^2);
alphar = atan2 (sumy, sumx);
alphar=alphar*180/pi;
if alphar< 0
  alphar = alphar + 360;
end
fprintf ('The resultant R is %4.2f kN\n', r);
fprintf ('The angle between the resultant and x axis is % 4.2f degrees\n', alphar);
```

Command Window

```
>> training
The resultant R is 254.11 kN
The angle between the resultant and x axis is  51.68 degrees
>>
```

Example 2: Figure 2 shows two forces, one 500 N and the other P applied by cables on each side of the obstruction A in order to remove the spike. Write a MATLAB program to determine:

(a) The magnitude of P necessary to such that the resultant T is directed along the spike

(b) The magnitude of T

(c) Plot P and T as a function of d. (Range of d between 1 and 20 mm).

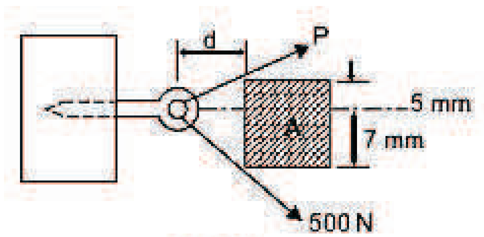


Fig. 2

Solution: Free-b

is shown in Fig. 2a.

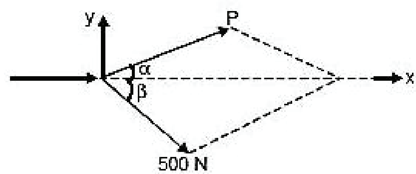


Fig.2 a

From equilibrium, resolving the forces along positive x and y directions:

$$T = \sum F_x = P\cos\propto + 500\cos\beta$$

$$0 = \sum F_y = P\sin\propto - 500\sin\beta$$

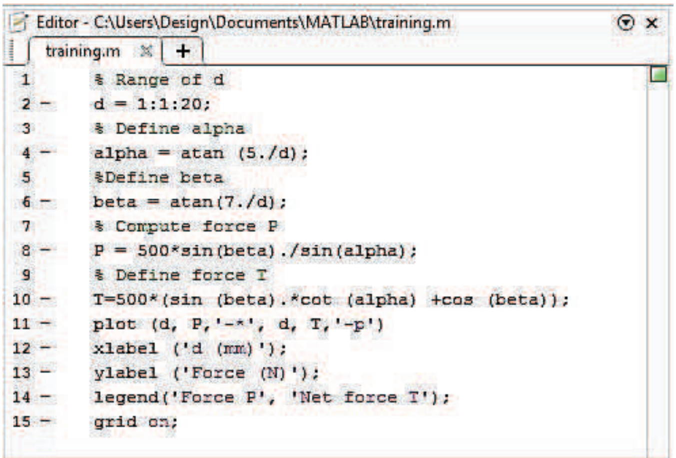
Solving the above two equations, we obtain

$$P = \frac{500\sin\beta}{\sin\alpha}$$

And $T = 500\,(\sin\beta\cot\alpha + \cos\beta\,)$

From the geometry $\propto = \tan^{-1}\left(\frac{5}{d}\right)$ *and* $\beta = \tan^{-1}(\frac{7}{d})$

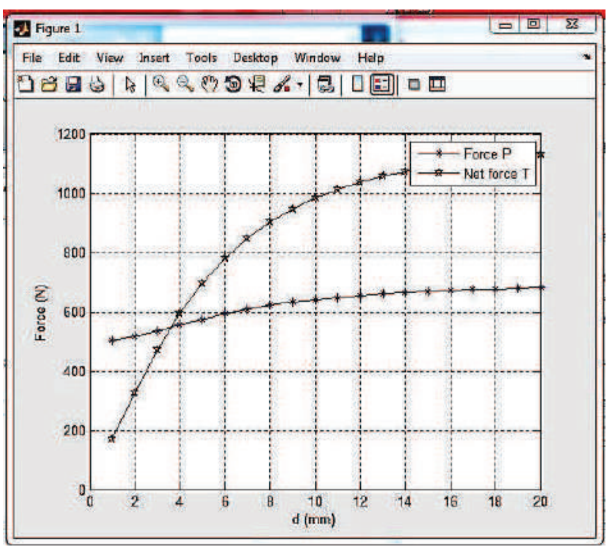
Complete MATLAB program is given below:



Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
% Range of d
d = 1:1:20;
% Define alpha
alpha = atan (5./d);
%Define beta
beta = atan(7./d);
% Compute force P
P = 500*sin(beta)./sin(alpha);
% Define force T
T=500*(sin (beta).*cot (alpha) +cos (beta));
plot (d, P,'-*', d, T,'-p')
xlabel ('d (mm)');
ylabel ('Force (N)');
legend('Force P', 'Net force T');
grid on;
```



Example 3: Figure 3 shows the two cables MO and NO tied together at O and the loadings are also shown. The magnitude of F is 150 N.

(a) Derive the expressions relating the tension in each cable as a function of α.

(b) Write a MATLAB program to plot the tension in each cable for 0° ≤ α ≤ 90º.

(c) Determine the smallest value of α for which both cables are in tension.

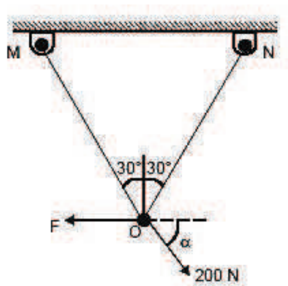


Fig. 3

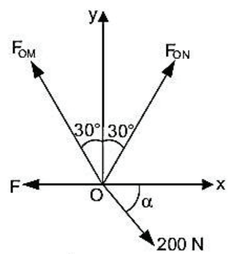


Fig. 3 a

Solution: Free-Body diagram is shown in fig. 3a.

Applying the equations of equilibrium:

$$(a)\ \sum F_x = 0\ \rightarrow (F_{ON} - F_{OM}) \sin 30^{\circ} - F + 200 \cos \propto = 0$$

$$or\ (F_{ON} - F_{OM}) = 2F - 400\ \cos \propto$$

Substituting F= 150 N

$$(F_{ON} - F_{OM}) = 300 - 400\ \cos \propto \qquad \dots \dots (1)$$

$$\sum F_y = 0\ \rightarrow (F_{OM} + F_{ON}) \cos 30^{\circ} - 200 \sin \propto = 0$$

$$or\ \ (F_{OM} + F_{ON}) = 200\ \times \frac{2}{\sqrt{3}} . \sin \propto = 230.94 \sin \propto \quad \dots \dots (2)$$

Solving eqn. (1) and (2).

$$F_{ON}\ = 150 - 200 \cos \propto + 115.47 \sin \propto$$

$$F_{OM} = 115.47 \sin \propto + 200 \cos \propto -150$$

For finding range of $\propto$ at which tension are positive equate $F_{ON} = 0$ and $F_{OM} = 0$ and find $\propto$.

(b) MATLAB Program:

Editor - C:\Users\Design\Documents\MATLAB\training.m*

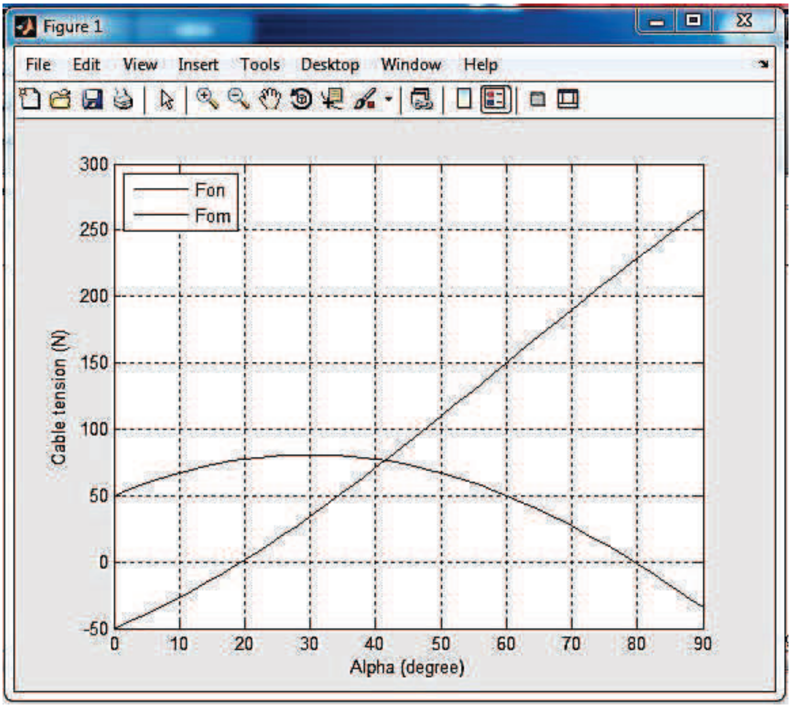
training.m*

```
Alpha= [0:2:90];
alpha=Alpha*pi/180;
fon =150-200*cos (alpha) +115.47*sin (alpha);
fom=-150+115.47*sin (alpha) +200*cos (alpha);
ton=abs (fon);
tom=abs (fom);
[ton_min, i]=min (ton);
[tom_min, j]=min (tom);
Ang1_min=Alpha (i);
Ang2_min=Alpha (j);
plot (Alpha, fon, Alpha, fom);
legend ('Fon', 'Fom', 2)
xlabel ('Alpha (degree)')
ylabel ('Cable tension (N)')
grid on
fprintf ('(c)\n')
fprintf ('Smallest value of Alpha for which the tensions are positive is
                from %g to %g degrees\n', Ang1_min, Ang2_min)
```

Command Window

```
>> training
(c)
Smallest value of Alpha for which the tensions are positive is from 20 to 80 degrees
>>
>>
```

The output is shown in Fig. below



Example 4: Figure 4 shows the location of the center of gravity of a 5000 N truck for the unloaded condition. The location of the added load WL is at a distance of x inches behind the rear axle. Write a MATLAB program and plot WL as a function of x for x ranging from 0 to 60 mm.

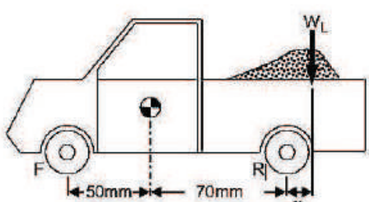


Fig. 4

Solution: Free-body diagram of the System is shown in Fig. 4a

The equilibrium equations can be written as

Moment about rear wheel axis:

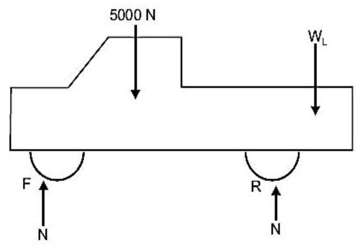


Fig. 4 a

$$\sum M_R = 0 = 5000\,(70) - N\,(120) - W_L x = 0$$

$$\sum F_y = 0 = N + N - 5000 - W_L = 0$$

Solving the above equations for W_L, we obtain

$$W_L = \frac{5000}{(60 + x)}$$

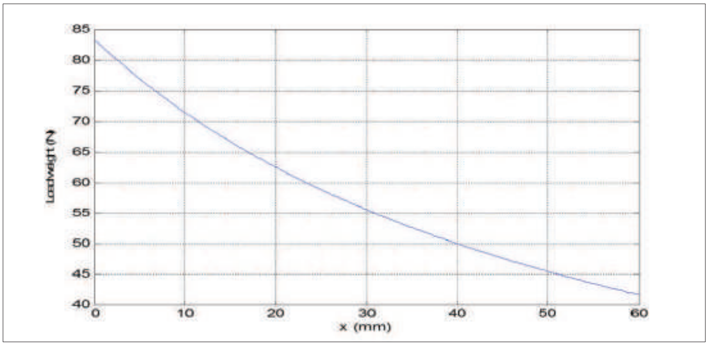
The plot of W_L as a function of x is shown in Fig. 5a from the following program.

MATLAB Solution

MATLAB solution

```
practice.m
% Define the range of x for the plot
x=0:0.05:60;
%Define W1
W1=5000./(60+x);
plot(x,W1);
%Labels
xlabel ('x (mm)')
ylabel ('Load weight (N)')
grid on
```

The output will be :



Example 5: Figure 5 shows the members CJ and CF of the loaded truss cross which are not connected to members BI and DG. Determine the values of α for which the truss cannot be in equilibrium. Write a MATLAB program to plot the forces in members BC, JC, IC and IG as a function of α.

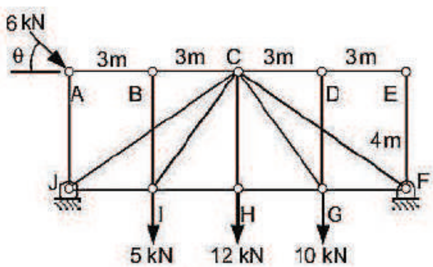


Fig. 5

Solution: Free-body diagram is shown in Fig. 5 a.

First we determine the reaction force at J from a free-body diagram for the entire truss.

$\Sigma MF = 0 = 6 \sin \theta\ (12) - 6 \cos \theta(4) + 4(9) + 10(6) + 8(3) - Jy\ (12)$

$Jy = 10 + 6 \sin \theta - 2 \cos \theta$

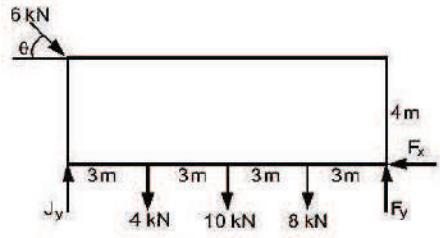


Fig. 5 a

Note that the rocker can only exert an upward force at A. Thus, to be in equilibrium, Jy must be positive. Since sin θ and cos θ vary between plus and minus one, the above equation indicates that Jy will be positive for all θ. Thus, the truss will be in equilibrium for all values of θ.
To obtain the required forces we now consider free-body diagrams for joints A, J and I. Note that each member is assumed to be in tension. Thus, positive answers will imply tension and negative answers compression. Also note the order in which the joints are analyzed. In each case there are only two unknown forces.

Joint A:

$$\sum F_x = 0 = AB + 6\cos\theta, \quad \sum F_y = 0 = -6\sin\theta - AJ$$

$$AJ = -6\sin\theta, AB = -6\cos\theta$$

Note that AB = BC since BI is a zero-force member.

Joint J:

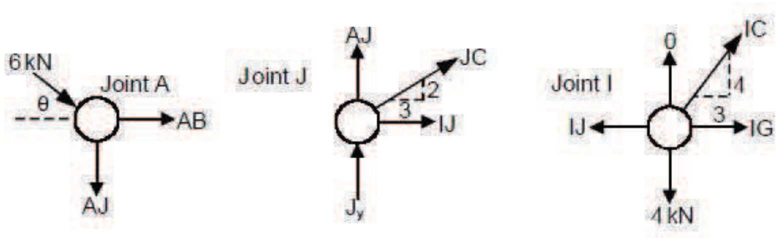
$$\sum F_x = 0 = IJ + -\frac{\sqrt{13}}{2}(AJ + J_y) = -\frac{\sqrt{13}}{2}(10 - 2\cos\theta)$$

$$IJ = -\frac{3}{\sqrt{13}} JC = 15 - 3\cos\theta$$

Joint I:

$$\sum F_x = 0 = GI - IJ + \frac{3}{5} IC, \sum F_y = 0 = \frac{4}{5} IC - 4$$

$$IC = 5\ kN. IG = IJ - \frac{3}{5}5 = 12 - 3\cos\theta$$



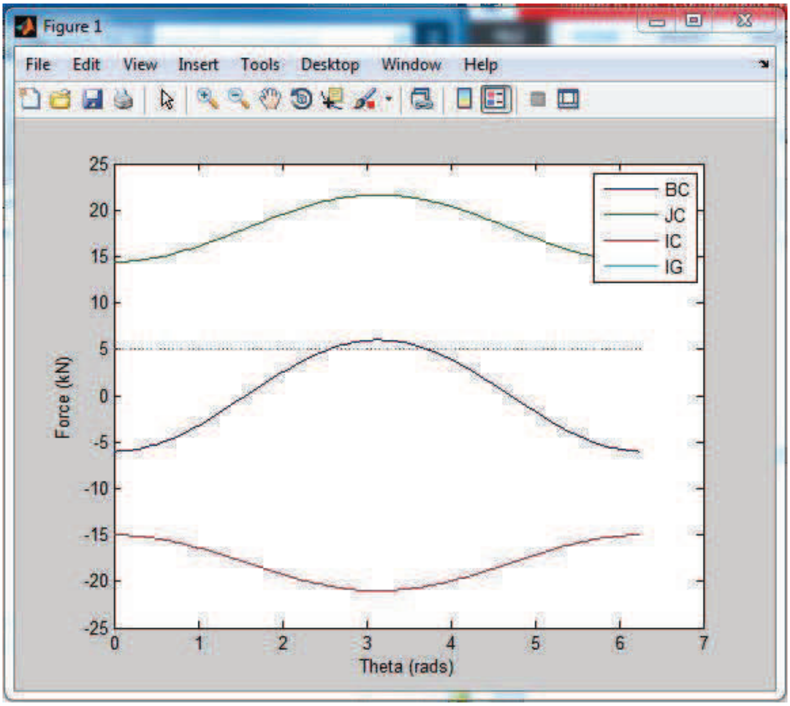
Note in the above that substitutions have been made in order to express each force explicitly in terms of θ. This has been done primarily for completeness. The program below shows that the explicit substitution is not necessary. The computer will substitute automatically provided each force is expressed in terms of functions that have been previously defined.

MATLAB Program:

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
th=0:0.05:2*pi;
Jy=10+6*sin (th)-2*cos (th);
AJ=-6*sin (th);
BC=-6*cos (th);
JC=sqrt (13)/2*(AJ + Jy);
IJ=-3/sqrt (13)*JC;
IC=5;
IG=IJ-3;
plot(th,BC,th,JC,th,IC,th,IG)
legend ('BC','JC','IC','IG')
xlabel ('Theta (rads)')
ylabel ('Force (kN)')
```



Example 6: In Fig. 6 rod CB is held by a cord AC that has a tension T.

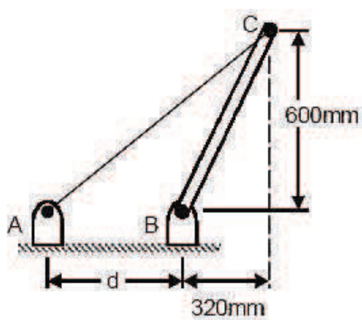


Fig. 6

Write a MATLAB program to determine,

(a) The moment about B of the force exerted by the cord at point C as a function of the tension T and the distance d.

(b) Plot the moment about B for 300 mm ≤ d ≤ 1000 mm when

(i) T = 60 N,

(ii) T = 80 N,

(iii) T = 110 N.

Solution: Free-body diagram is shown in Fig. 6 (a).

The length of AC is from the figure,

$$AC = \sqrt{(600)^2 + (320 + c)^2}$$

For the angle α we have

$$\cos \propto = \frac{320 + c}{AC}$$

$$\sin \propto = \frac{600}{AC}$$

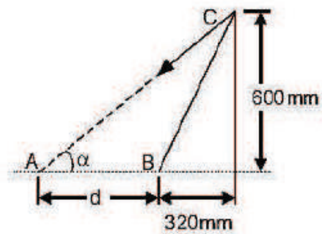


Fig. 6 a

The tension T is given by

T = –T cos αi – T sin j

And the position vector from point B as

r = 320i + 600j

The moment about point B is

$M_B = r \times T = (320i + 600j) \times (-\cos \alpha i - \sin \alpha j)\, T$

$= T\,(600 \cos \alpha - 320 \sin \alpha)\, k$

The magnitude of the moment is

$$M_B = \frac{600\, T_C}{\sqrt{c^2 + 640\, c + 462400}}$$

$$M_B = \frac{T}{AC}\,[600 * (320 + c) - 320 * (600)]$$

$$M_B = \frac{600\, Tc}{\sqrt{(600)^2 + (320 + c)^2}}$$

Or

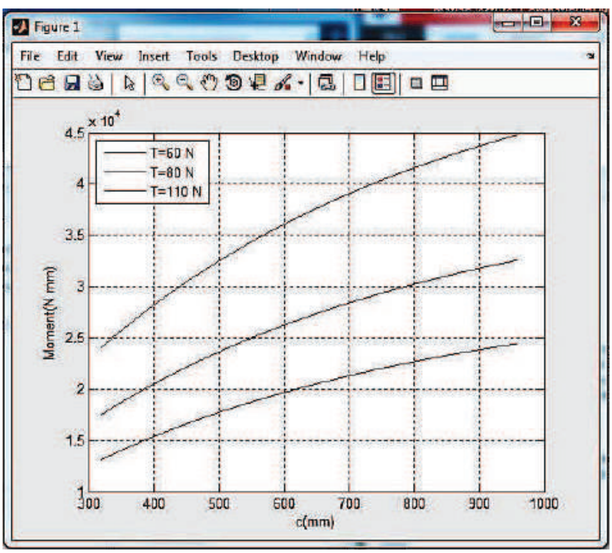
$$M_B = \frac{600\, Tc}{\sqrt{c^2 + 640\, c + 4624000}}$$

MATLAB Program:

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
c=[320:1:960];
Mb1=600*60*c./(sqrt(c.^2+640*c+462400));
Mb2=600*80*c./(sqrt(c.^2+640*c+462400));
Mb3=600*110*c./(sqrt(c.^2+640*c+462400));
plot(c,Mb1,c,Mb2,c,Mb3)
legend('T=60 N','T=80 N','T=110 N',2)
xlabel('c(mm)')
ylabel('Moment(N mm)')
grid on
```



Example 7: Write a MATLAB program to plot the shear and bending moment diagrams for the beam shown in Fig. 7. The length of the beam L = 4 m and w_0 = 20 kN/m.

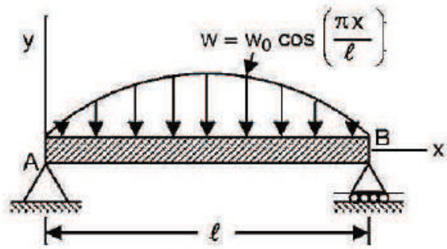


Fig. 7

Solution: It is known that

$$\frac{dV}{dx} = -w$$

$$\frac{dM}{dx} = V$$

Integrating equations (1) and (2), we get

$$V = -\int w dx = \int w_o \sin\frac{\pi x}{l}\, dx = w_o \frac{L}{\pi} \cos\frac{\pi x}{l} + C_1$$

$$M = \int V\, dx = \int \left(w_o \frac{l}{\pi} \cos\frac{\pi\, x}{l} + C_1\right) dx = w_o \left(\frac{l}{\pi}\right)^2 \sin\frac{\pi\, x}{l} + C_1 x + C_2$$

The boundary conditions are written as:

$At\ x = 0$: $M = 0 = w_o \left(\frac{l}{\pi}\right)^2 \sin\frac{\pi\,(0)}{l} + C_1(0) + C_2$

Which implies that C_2=0.

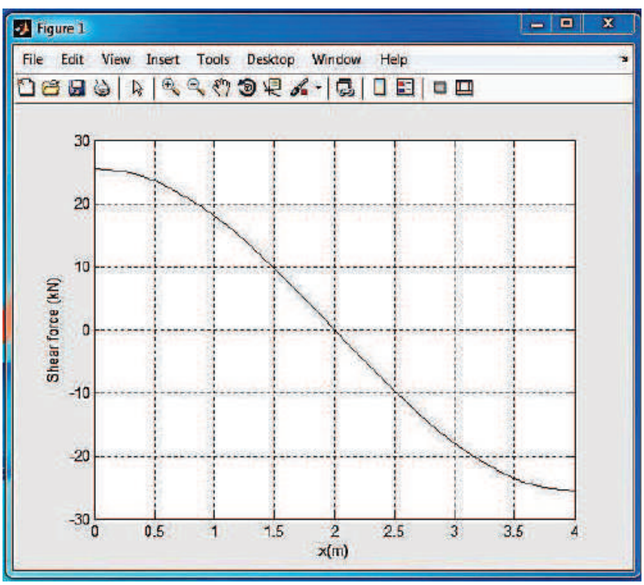
$At\ x = l$: $M = 0 = w_o \left(\frac{l}{\pi}\right)^2 \sin\frac{\pi\,(l)}{l} + C_1 l$

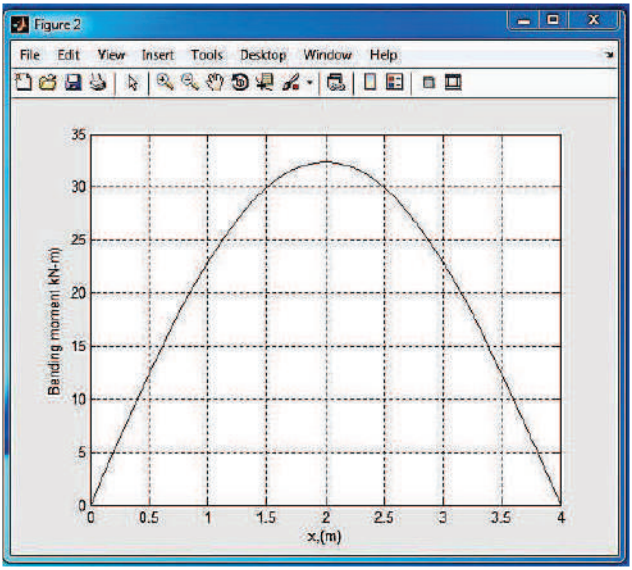
Which implies that C_1=0.

Hence, $V = w_o \frac{l}{\pi} \cos\frac{\pi\, x}{l}$, $M = w_o \left(\frac{l}{\pi}\right)^2 \sin\frac{\pi\, x}{l}$.

MATLAB Solution:

```
% Input w0 and l
w0=20;
l = 4;
x = [0:0.1:4];
v1 = w0*(1/pi)*cos(pi*x/l);
m1= w0*(1/pi)^2*sin(pi*x/l);
plot(x,v1)
xlabel('x(m)')
ylabel('Shear force (kN)')
grid on
figure
plot(x,m1)
xlabel('x, (m)')
ylabel('Bending moment kN-m)')
grid on
```





Example 8: Figure 8 shows three flat blocks positioned on an inclined oriented at angle α. Write a MATLAB program to plot the maximum value of P (if no slipping occurs) versus α. Assume only positive values of P and indicate the regions over which

(i) The 40 kg block slides alone,

(ii) The 40 kg and 30 kg blocks slide together.

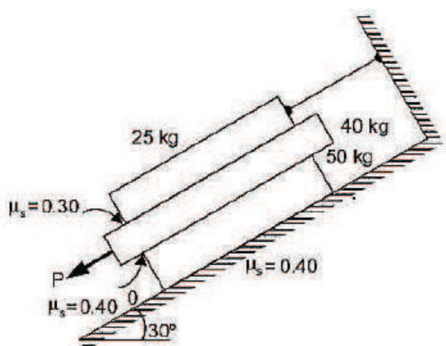


Fig. 8

Solution: The free-body diagrams for the three blocks are shown in Fig. 8 (a). To obtain the required plot it will be convenient to use a slightly different approach than that used in the sample problem in your text. We start by writing down the equilibrium equations without making any assumptions about where sliding occurs.

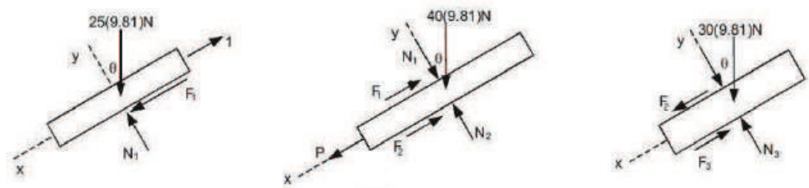


Fig. 8 a

[ΣFy = 0]: 25 kg ; N1 – 25(9.81) cos θ = 0
40 kg ; N2 – N1 – 40(9.81) cos θ = 0
30 kg ; N3 – N2 – 30(9.81) cos θ = 0

These equations can be readily solved for the normal forces.

N1 = 25(9.81) cos θ ; N2 = 65(9.81) cos θ ; N3 = 95(9.81) cos θ

[ΣFx = 0]: 40 kg ; P – F1 – F2 + 40(9.81) sin θ = 0
30 kg ; F2 – F3 + 30(9.81) sin θ = 0

Now we have two equations with four unknowns P, F1, F2 and F3. Note that we have not written the equation for the summation of forces in the x-direction for the 25 kg block. The reason is that this equation introduces an additional unknown (T) that we are not interested in determining.

The next step is to make assumptions about which block(s) slide. As will be seen, either of the two possible assumptions about impending motion will reduce two of the friction forces to functions of θ only.

This will result in two equations that may be solved for P and the remaining friction force. The forces calculated will be designated P1 or P2 to distinguish the two cases for impending slip.

Case 1: Only the 40 kg block slips.
Impending slippage at both surface of the 40 kg block gives F1 = 0.3N1 = 73.575 cos θ and F2 = 0.4N2 = 255.06 cos θ. Substituting these results into the equilibrium equations yields

$$P1 = 328.635 \cos\theta - 392.4 \sin\theta$$

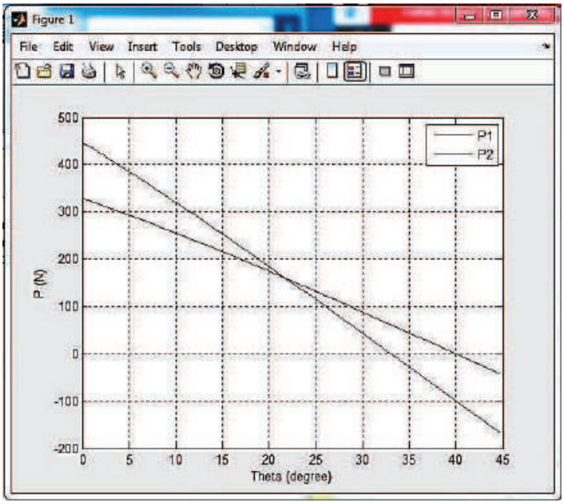
Case 2: The 30 and 40 kg blocks slide together.
Impending slippage at the upper surface of the 30 kg block and lower surface of the 40 kg block gives F1=0.3N1=73.575 cos θ and F3 = 0.4N3 = 372.78 cos θ. Substitution of these results into the equilibrium equations gives,

$$P2 = 446.355 \cos\theta - 686.7 \sin\theta$$

Which of these two values of P represents the maximum load that can be applied without slippage on any surface is best illustrated by plotting the two expressions as a function of θ. This plot will be generated in the script below. The basic idea is that at any specified angle θ, the critical or maximum value of P will be the smaller of two values calculated.

MATLAB Program:

```
Editor - C:\Users\Design\Documents\MATLAB\training.m
training.m

theta=0:0.01:pi/4;
P1=328.635*cos(theta)-392.4*sin(theta);
P2=446.335*cos(theta)-686.7*sin(theta);
plot(theta*180/pi, P1, theta*180/pi , P2)
legend('P1','P2')
xlabel('Theta (degree)')
ylabel('P (N)')
grid on
```



Example 9: The coefficient of friction μ, can be determined by μ = F/mg where F is the measured force (N), m is the mass (kg) and g = acceleration due to gravity (9.81 m/s2). The following table gives the experimental data. Determine

(a) The coefficient of friction in each test

(b) The average from all tests.

Test #	1	2	3	4	5	6	7
Mass m (kg)	2	4	5	10	20	50	100
Force F(N|)	12.4	23.2	30.5	60.8	116.5	293.8	597.3

Solution: System under equilibrium is shown in Fig.9.

This is simple application of mathematics.

Program is given below:

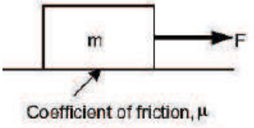


Fig. 9

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
% Force and mass values in vector form
F=[12.4 23.2 30.5 60.8 116.5 293.8 597.3];
m=[2 4 5 10 20 50 100];
%Coefficient of friction, mu
mu=F./(m*9.81)
% average mu=mean(mu)
SUM = sum(mu)
average=mean(SUM)/7
```

Command Window

```
>> training

mu =

    0.6320    0.5912    0.6218    0.6198    0.5938    0.5990    0.6089


SUM =

    4.2665


average =

    0.6095

fx >>
```

Example 10: Figure 10 shows a large turnbuckle which supports a cable tension of 12,000 N. The mean diameter of the two 1.0 mm screws is 1.15 mm and has five square threads per mm. Both screws have single start threads.

(a) Determine the moments MT and ML that must be applied to the body of the turnbuckle in order to tighten and loosen it respectively.

(b) Write a MATLAB program to plot the moments MT and ML as functions of μ for $0 \le \mu \le 1$, where μ is the coefficient of friction for the threads.

Fig. 10

Solution: M_T and M_L can be obtained as

$$M_T = 2 \tan(\varphi + \alpha)\ MT\ T\ r \ldots \qquad (1)$$

And

$$M_L = 2 \tan(\varphi - \alpha)\ ML\ T\ r \ldots \qquad (2)$$

Where T = 12,000 N and the lead L = 1/5 mm/rev.
The mean radius is given by r = 1.15/2 = 0.575 mm
We also have

$$\propto = \tan^{-1}\left(\frac{L}{2\,\pi\,r}\right) \ \ and\ \varphi = \tan^{-1}(\mu) \ldots \qquad (3)$$

From Eqs. (1), (2) and (3), we can compute ML and MT explicitly.

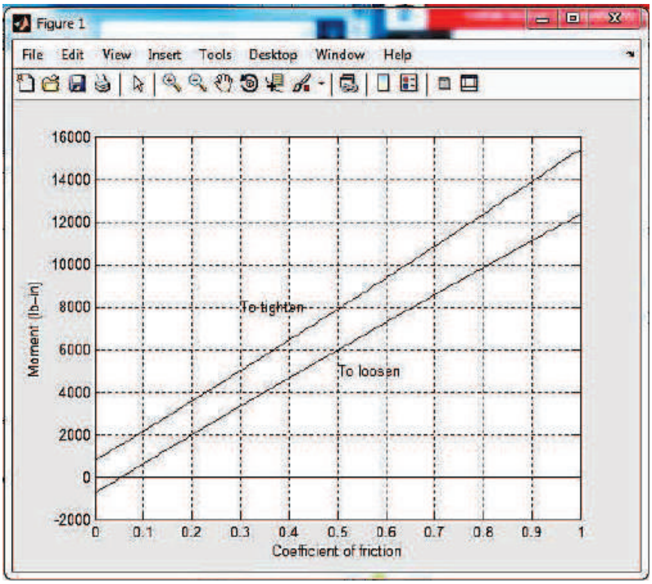
MATLAB Program:

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
% Input T,L, and r
T=12000;
L=1/5;
r=1.15/2;
alpha = atan(L/2/pi/r);
mu = 0:0.01:1;
phi = atan(mu);
MT = 2*T*r*tan(alpha + phi);
ML=2*T*r*tan(phi-alpha);
% Place a horizontal line at x = 0
M=0*mu;
plot(mu,MT,mu,ML,mu,M)
xlabel('Coefficient of friction')
ylabel('Moment (lb-in)')
text(0.5,5000,'To loosen')
text(0.3,3000,'To tighten')
grid on
```

The plot of moments MT and ML as functions of μ for $0 \le \mu \le 1$



Example 11: Figure 11 shows a flexible cable which supports the 100 kg load and passes over a circular drum and is subjected to a force P to maintain equilibrium.

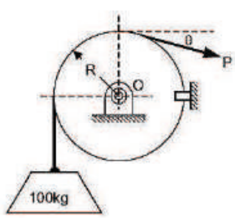


Fig.11

(a) For θ = 0, determine the maximum and minimum values of P may have in order to raise or lower the load.

(b) Write a MATLAB program to plot P_{max} and P_{min} versus μ for $0 \leq \mu \leq 1$, where μ is the coefficient of static friction between the cable and the fixed drum.

(c) For P = 550 N, determine the minimum value for which the angle θ may have before the load begins to slip.

(d) Plot θmin versus μ for $0 \leq \mu \leq 1$.

Limit the variation of θ between –60º and 360º.

Solution: Free-body diagram of the circular drum is shown in Figures 11 (a) and (b).

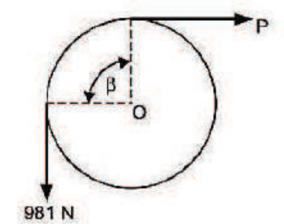


Fig. 11 a, θ= 0, β = π/2

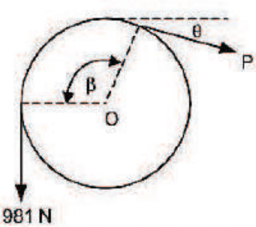


Fig. 11 b, P = 550 N, β=θ+π/2

Here we have $T_2 = T_1 e^{\mu\beta}$ (belt friction)

Recall that in deriving this formula it was assumed that T2 > T1.

(a) With θ = 0 the contact angle is β = π/2 rad. For impending upward motion of the load we have $T_2 = T_{max}$ and $T_1 = 981$ N. Hence

$$P_{max} = 981\, e^{\mu\pi/2}$$

For impending downward motion of the load we have T2 = 981 N and T1 = $P_{min.}$

$$981 = P_{min} e^{\mu\pi/2} \;\; or \;\; P_{min} = 981\, e^{-\mu\pi/2}$$

(b) With P = 550 N we have β = π/2 +θ, T2 = 981 N and T1 = P = 550 N. Therefore,

$$\frac{981}{550} = e^{\mu(\theta+\pi/2}$$

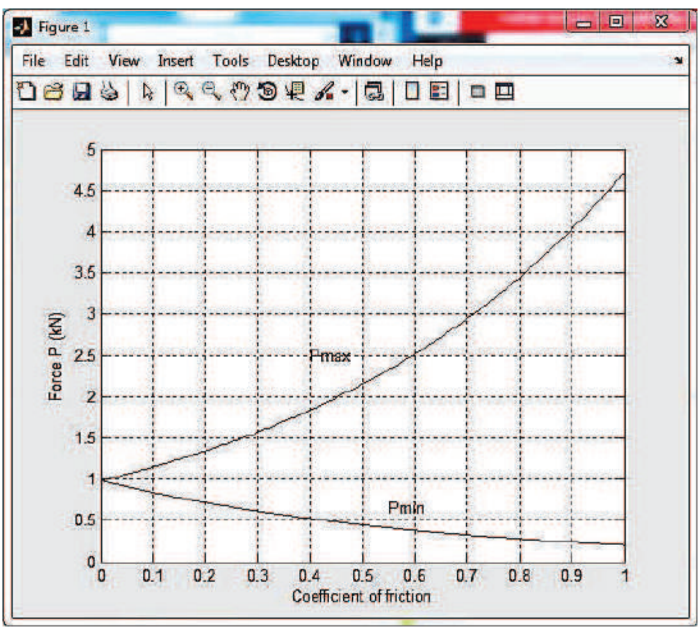
The plots are shown in Fig. below as output of following MATLAB program.

MATLAB Solution:

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
% Plots of Pmax and Pmin versus coefficient of friction
mu=0:0.005:1;
pmax=981*exp(mu*pi/2)/1000;
pmin=981*exp(-mu*pi/2)/1000;
plot(mu,pmax,mu,pmin)
grid on
xlabel('Coefficient of friction')
ylabel('Force P (kN)')
text(0.4,2.5,'Pmax')
text(0.55,0.65,'Pmin')
grid on
```



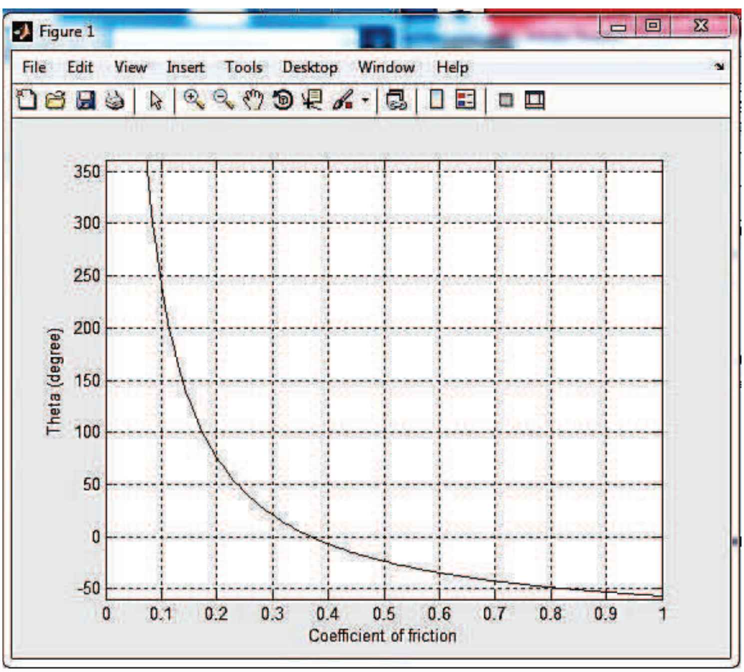
(c) Taking the natural logarithms on both sides of the above equation and solving for θ gives, $\theta = \frac{ln\left(\frac{981}{550}\right)}{\mu} - \frac{\pi}{2}$

(d) Following program plots minimum value of θ versus μ.

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
% Plot of minimum angle theta versus mu
mu=0:0.005:1;
thet=log(981/550)./mu-pi/2;
plot(mu,thet*180/pi);
axis([0 1 -60 360])
xlabel('Coefficient of friction')
ylabel('Theta (degree)')
grid on
```



Example 12: Figure 12 shows axle pulley system where the coefficient of friction between cable ABCD and the pulley varies between 0 and 0.60. Write a MATLAB program to determine,

(a) The values of α for the system to remain in equilibrium

(b) The reactions at A and D

(c) Plot α as a function of the coefficient of friction.

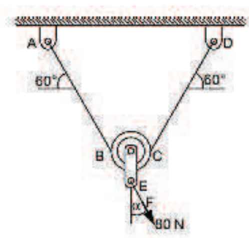


Fig.12

Solution: Free-body diagram and force triangle are given in Figs. 12 (a) and (b).

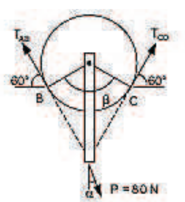


Fig.12 a

Fig.12 b

Since the 80 N force tends to rotate the pulley counterclockwise, the cable tends to slip relative to the pulley clockwise and we have

$T_1 = T_{CD}$, $T_2 = T_{AB}$, μ_s = static friction

$\beta = 120^o = 2\pi/3$ radians.

From the ratio of belt-tension relations:

$$\frac{T_2}{T_1} = e^{\mu,\beta} \quad \rightarrow \frac{T_{AB}}{T_{CD}} = e^{\frac{2\pi}{3}\mu}$$

or

$$T_{AB} = e^{\frac{2\pi}{3}\mu} . T_{CD}$$

From the force triangle, we have using the law of cosines

$$P^2 = T_{AB}^2 + T_{CD}^2 - 2T_{AB}T_{CD}\cos\beta$$

$$= \left(e^{\frac{2\pi}{3}\mu}\, T_{CD}\right) + T_{CD}^2 - 2\left(e^{\frac{2\pi}{3}\mu}\, T_{CD}\right)\left(-\frac{1}{2}\right)$$

$$\left[\left(e^{\frac{2\pi}{3}\mu}\right)^2 + 1 + e^{\frac{2\pi}{3}\mu}\right] T_{CD}^2$$

$$P^2 F T_{CD}^2$$

Where

$$F = \left[\left(e^{\frac{2\pi}{3}\mu}\right)^2 + 1 + e^{\frac{2\pi}{3}\mu}\right]$$

Hence, we have that

$$T_{CD} = \frac{1}{\sqrt{F}}\, P$$

(a) The corresponding values of α for the system to remain in equilibrium. Using the law of sines we have.

$$\frac{\sin\gamma}{T_{CD}} = \frac{\sin\beta}{P} \qquad \sin\gamma = \frac{T_{CD}}{P}\sin\beta$$

$$\sin\gamma = \frac{1}{\sqrt{F}}\sin\beta \qquad \gamma = \sin^{-1}\left(\frac{1}{\sqrt{F}}\, sin\beta\right)$$

$$\propto = 90^{\circ} - (60^{\circ} + \gamma)$$

(b) The reactions at A and D are as follows:
Substituting P = 80 N in Eq.(2), then

$$D = T_{CD} = \frac{1}{\sqrt{F}}\,(80)N$$

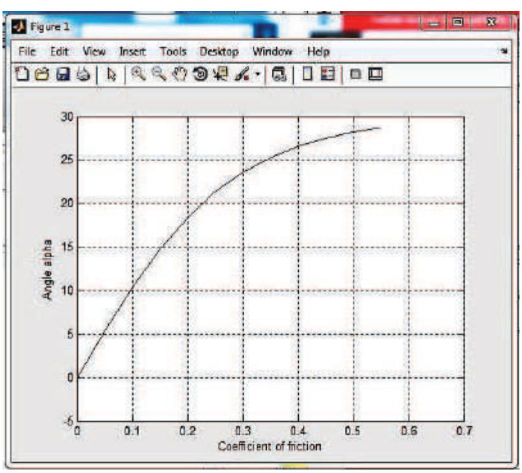
And from Eq. (1), $A = T_{CD} = e^{\frac{2\pi}{3}\mu}\left(\frac{1}{\sqrt{F}}\right)(50)N$

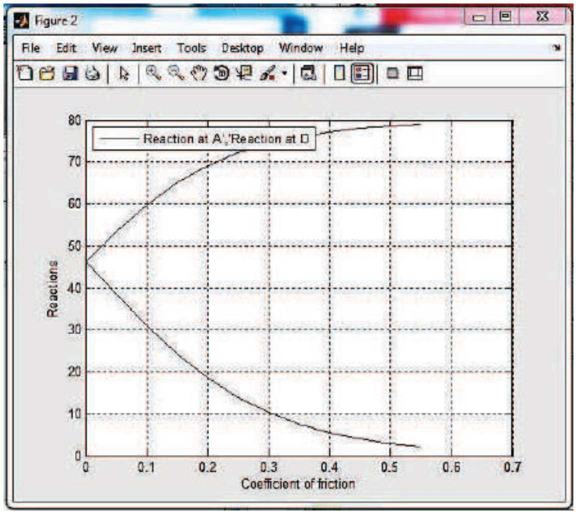
MATLAB program for this problem is given below:

```
Editor - C:\Users\Design\Documents\MATLAB\training.m*
training.m*

mu=0;
%Output Headings
fprintf('Friction   Angle\n')
fprintf('\n')
ang=120.*pi/180.;
N=1;
while mu<0.60
Y(N)=mu;
a=exp(pi*ang*mu);
E=sqrt(a^2+1.+a);
ooe=1./E;
sgamma=ooe*sin(ang);
Gamma=asin(sgamma);
alpha(N)=pi/2.-(1.0472+Gamma);
X(N)=alpha(N)*180/pi;
D(N)=ooe*80.;
A(N)=a*D(N);
fprintf('  %5.3f    %5.3f\n',Y(N),X(N))
mu=mu+.05;
N=N+1;
end%while
fprintf('\n')
fprintf('\n')
fprintf('Friction    Reac.A     Reac.D\n')
fprintf(' %5.3f      %5.3f      %5.3f\n',Y(I),A(I),D(I))
end
figure(1)
plot(Y,X)
xlabel('Coefficient of friction')
ylabel('Angle alpha')
grid on
figure(2)
plot(Y,D,Y,A)
xlabel('Coefficient of friction')
ylabel('Reactions')
grid on
legend('Reaction at A','Reaction at D',2)
```

Output is as follows:

```
Command Window
>> training
Friction   Angle

  0.000    -0.000
  0.050    5.377
  0.100    10.391
  0.150    14.780
  0.200    18.425
  0.250    21.332
  0.300    23.582
  0.350    25.286
  0.400    26.558
  0.450    27.497
  0.500    28.186
  0.550    28.687

Friction     Reac.A     Reac.D

 0.000       46.188     46.188
 0.050       53.481     38.488
 0.100       59.860     31.001
 0.150       65.068     24.251
 0.200       69.105     18.535
 0.250       72.125     13.922
 0.300       74.336     10.326
 0.350       75.934     7.591
 0.400       77.083     5.546
 0.450       77.907     4.033
 0.500       78.498     2.925
 0.550       78.921     2.116
fx >>
```





Example 13: Figure 13 shows a cylindrical silo where H = height of the cylindrical portion, r = radius of the cylindrical silo, R = radius of the spherical cap roof and V = volume of the silo.

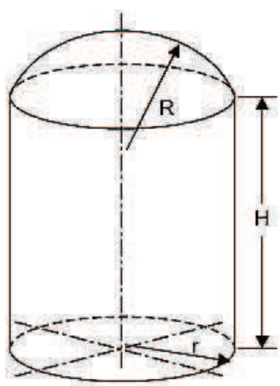


Fig. 13

Write a MATLAB program to compute

(a) The height for given values of r, R and V

(b) The surface area of the silo, S

Use the program to determine the height and surface area of a silo, given r = 32 cm, R = 50 cm and V = 125,000 cm3.

Solution: The total volume of the silo is the sum of the volumes of the cylindrical part and the spherical cap.

$$V_{total} = V_{cyl} + V_{cap}$$

$$= \pi r^2 H + \frac{1}{3}\pi h^2(3R - h)$$

Where,

h = R − R cosθ = R(1 − cosθ) (see Fig. 13(a))

and r = R sin θ or θ = sin−1 (r R)

The height H, of the cylindrical part is given by

$$H = \frac{V - V_{cap}}{\pi r^2}$$

The surface area of the silo is the sum of the surface areas of the cylindrical part and the spherical cap.

$S = S_{cyl} + S_{cap} = 2\pi rH + 2\pi RH$.

MATLAB Solution:

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
r=32;
R=50;
V=125000;
Theta=asin(r/R);
h = R*(1-cos(Theta));
Vcap=pi*h^2*(3*R-h)/3;
H=(V-Vcap)/(pi*r^2);
S=2*pi*(r*H+R*h);
fprintf('The height H = %f cm ',H);
fprintf('The surface area of the silo S = %f square cm.',S);
```

Output comes as follows:
The height H = 32.812738 cm. The surface area of the silo S = 10235.750764 square cm.

6- 2 Problems in Dynamics:

Particle Kinematics
Example 14: The motion of a particle is defined by the equation

$x = 35t^2 - 110t$

And $y = 115t^2 - 42t^3$

Where x and y = displacement of the particle (mm)

t = time (sec.) for the time interval $0 \le t \le 25$s

Write a MATLAB program to plot:

(a) The path of the particle in the x-y plane
(b) The components of the velocity v_x and v_y and the magnitude of the velocity v,
(c) The components of the acceleration ax and ay and the magnitude of the acceleration a.

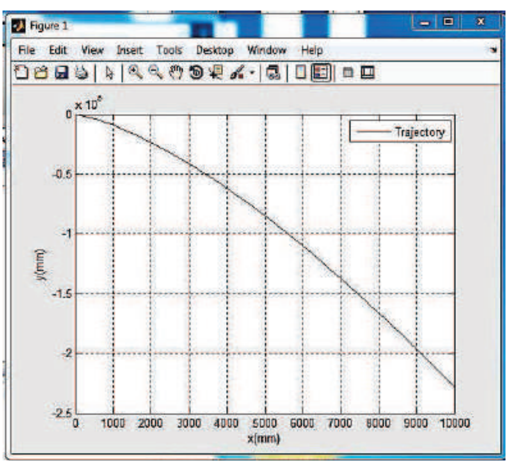
Solution:
Given $x = 35t2 - 110t$; $y = 115t2 - 42t3$ as the position vector of the particle.
Hence, $v_x = \dot{x} = 70t - 110$; $v_y = \dot{y} = 230t - 126t2$ are component of velocities and $a_x = \ddot{x} = 70$; $a_y = \ddot{y} = 230 - 252t$ are components of accelerations.
Here particle path refers to the plot of x and y positions at various instants of time.So first x and y are found by varying t from 0 to 25 seconds.

MATLAB Program is given below:

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
t=[0:0.5:25];% Vary the time from 0 to 25 in steps of 0.5
x=35*t.^2-110*t; % compute x
y=115*t.^2-42*t.^3; % compute y
v_x=70*t-110; % compute vx
v_y=230*t-126*t.^2; % compute vy
v=sqrt(v_x.^2+v_y.^2); % compute v
a_x=70;% compute ax
a_y=230-252*t; % compute ay
a=sqrt(a_x.^2+a_y.^2);
% compute a PLOTING THE CALCULATED DATA
figure(1)
plot(x,y);
xlabel('x(mm)')
ylabel('y(mm)')
legend('Trajectory')
grid on
axis([0 10000 -2.5e5 0])
figure(2)
plot(t,v_x,t,v_y,t,v)
xlabel('t(sec)')
ylabel('v_x,v_y,v(mmps)')
legend('v_x','v_y','v',2)
plot(t,a_x,t,a_y,t,a)
xlabel('t(sec)')
ylabel('a_x,a_y,a(mmps^2)')
legend('a_x','a_y','a',2)
grid on
```



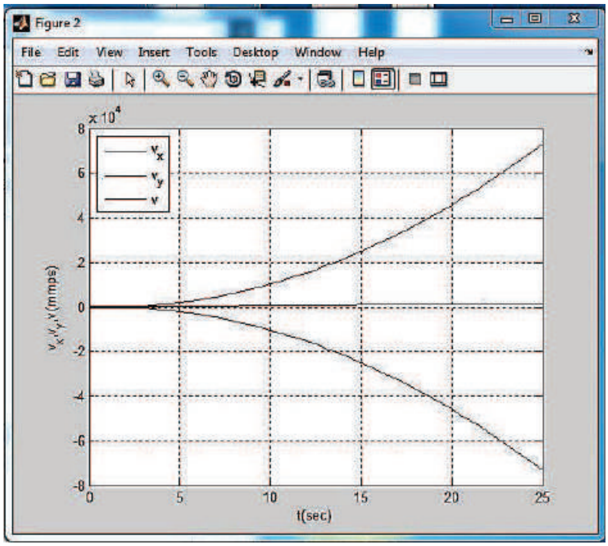


Figure 2
File Edit View Insert Tools Desktop Window Help
x 10^4
v_x
v_y
v
v_x,v_y,v (mmps)
t(sec)

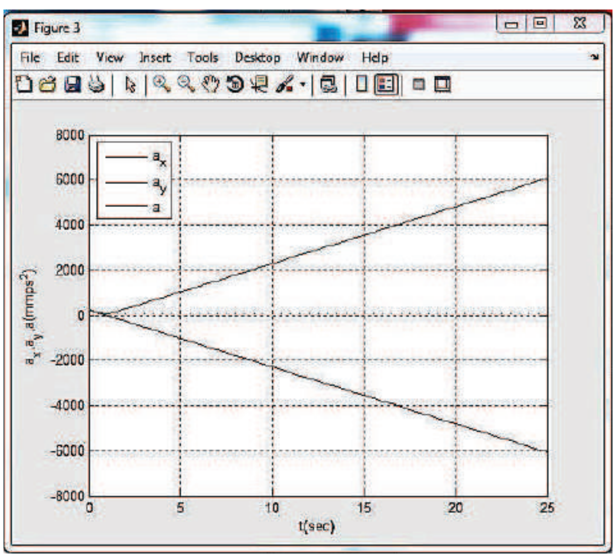


Figure 3
File Edit View Insert Tools Desktop Window Help
a_x
a_y
a
a_x,a_y,a(mmps^2)
t(sec)

Example 15: A particle is fired vertically downwards with a velocity 30 m/s in a fluid. Due to resistance of the fluid the particle experiences a deceleration equal to a = – (0.7v3) m/s2, where v is velocity in m/s. Plot a graph of velocity versus time and distance versus time using MATLAB.

Solution: Given a = f (v), so the velocity is determined as a function of time using a = dv/dt, since this equation relates v, a and t.
Thus a =dv/dt=0.7. v^3
Integrating both sides

$$-\int_{30}^{v} \frac{dv}{0.7\, v^3} = \int_{0}^{t} dt \; or \; t = \frac{1}{1.4}\left[\frac{1}{v^2} - \frac{1}{30^2}\right] \; or = \left(1.4\, t + \frac{1}{30^2}\right)^{-0.5}$$

Position s is give by

$$\frac{ds}{dt} = v = \left(1.4\, t + \frac{1}{30^2}\right)^{-0.5}$$

Or integration

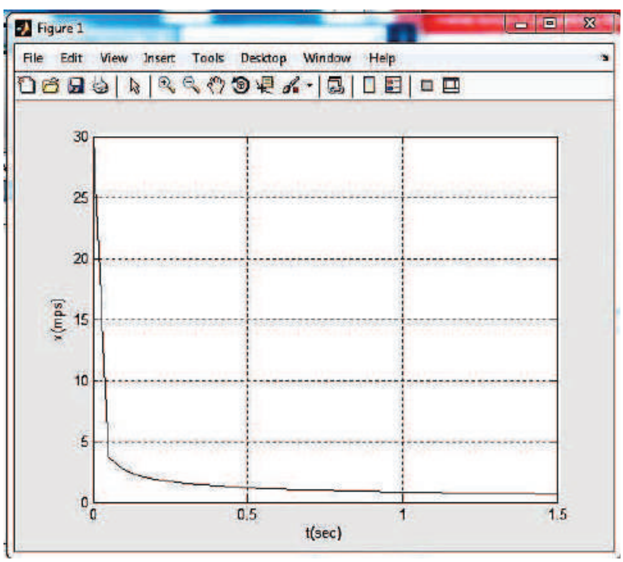
$$\int_{0}^{s} ds = \int_{0}^{t} \left(1.4\, t + \frac{1}{30^2}\right)^{-0.5} dt$$

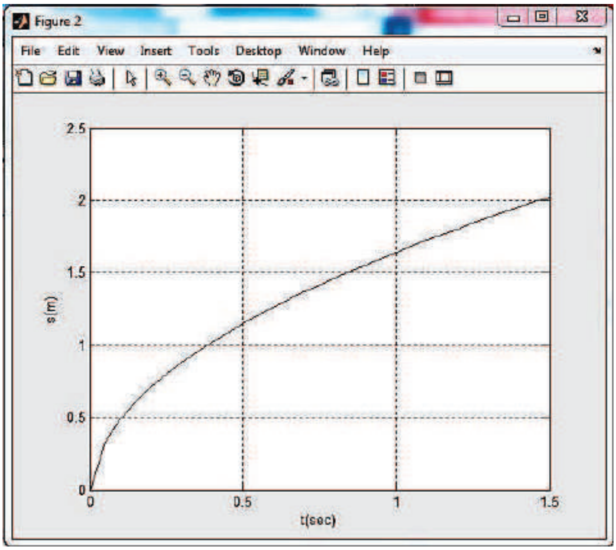
Or $s = \frac{2}{1.4}\left[\left(1.4\, t + \frac{1}{30^2}\right)^{0.5} - \frac{1}{30}\right]$

Using MATLAB, the graphical representation of velocity and displacement as a function of time t is given with following simple program.

MATLAB Program:

```
t=0:0.05:1.5;
v0=30; %INITIAL VELOCITY
v=(1.4*t+1/v0^2).^-0.5;
s=2/1.4*(sqrt(1.4*t+1/v0^2)-1/30);
figure(1)
plot(t,v);
xlabel('t(sec)')
ylabel('v(mps)')
grid on
figure(2)
plot(t,s)
xlabel('t(sec)')
ylabel('s(m)')
grid on
```





Example 16: Figure 16 shows the motion of a particle. The initial velocity and the angle at which the projectile is fired are known. Write a MATLAB program to calculate and plot the maximum height and distance. Use the program to calculate and plot the trajectory of a projectile that is fired at a velocity of 250 m/s at an angle of 40º.

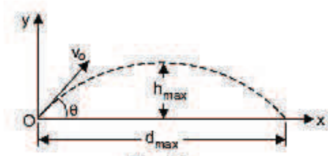


Fig. 16

Solution: Components of velocity
v_{ox} = vo. cos θ and v_{oy} = vo. sin θ.
Height h_{max}= v2oy /2 g
and corresponding time
t_{max}= v_{oy} / g
But to draw trajectory as a function of distance(x), use

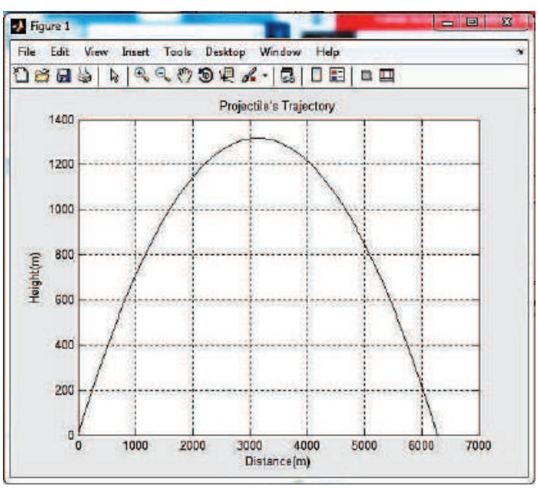
$$y = v_{oy}.t * -\frac{1}{2}\ gt^2 \ \ and \ \ x = v_{ox}t$$

MATLAB script is shown below:

Editor - C:\Users\Design\Documents\MATLAB\training.m*

training.m*

```
% Trajectory calculates the max height
% and distance of a projectile,
%Output arguments are:
%The function creates also a plot of the trajectory.
g=9.81;
%v0=initial velocity in(m/s) 250.
%theta=angle in degrees 40.
v0x=250*cos(40*pi/180);
v0y=250*sin(40*pi/180);
thmax=v0y/g;
%hmax=maximum height in (m).
hmax=v0y^2/(2*g);
ttot=2*thmax;
%dmax=maximum distance in(m).
dmax=v0x*ttot;
%Creating a trajectory plot
tplot=linspace(0,ttot,200);
x=v0x*tplot;
y=v0y*tplot-0.5*g*tplot.^2;
plot(x,y)
xlabel('Distance(m)')
ylabel('Height(m)')
title('Projectile's Trajectory')
grid on
```



Example 17: A 3 kg block is subjected to two forces as shown in Fig. 17 If block starts from rest, determine the distance it has moved when it attains a velocity of 10 m/s. Plot its distance as a function of coefficient of kinetic-friction between the block and floor.

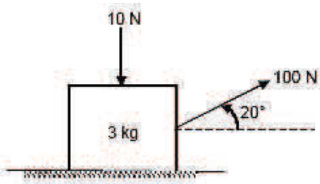


Fig. 17

Solution: This is an application of work-energy principle on the motion of a particle.

The work-energy principle is

$$T_1 + \Sigma U = T_2,$$

Where T_1 = Initial kinetic energy and

T_2 = Final kinetic energy of the particle.

Here, $T_1 = 0$ and $T_2 = 1/2\ .mv^2$

The forces doing the work on the particle are horizontal component of external force 100 N and friction acting on the floor. Work-done by the forces are

$$\Sigma U = (100 \cos 20° - \mu\ N) \times s,$$

Where $N = (10 + W - 100 \sin 10°)$,

$W = 3g$, the weight of block.

Thus $½.\ mv^2 = \Sigma U$, relates velocity and kinetic friction μ.

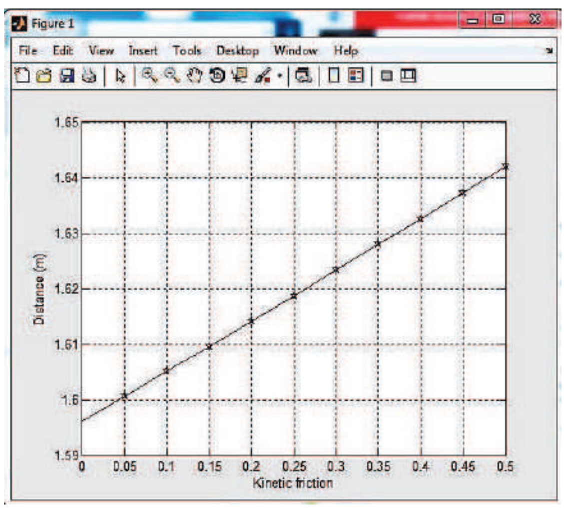
The program for this problem is written as follows:

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
% DEFINE ALL VARIABLES
m=3;
v=10;
g=9.81;
F1=100;
F2=10;
T=0.5*m*v^2;
mu=0:0.05:0.5;
s=150./(F1*cos(20*pi/180)-mu*(F2+m*g-F1*sin(20*pi/180)));
plot(mu,s,'-p')
xlabel('Kinetic friction');
ylabel('Distance (m)');
grid on;
```

Output is given in Fig. below



Example 18: Figure 18 shows a block B of mass m_B starts from rest and slides down an inclined plane of a wedge of mass mA which is supported by a horizontal surface.

(a) Obtain an expression for the speed of block B relative to wedge A.

(b) The speed of wedge A,

(c) Write a MATLAB program to plot the speed of B relative to A and the speed of A as function of s, where‘s’ is the distance traveled by the block B down the surface of the wedge for $0 \leq s \leq 1.0$ m. Neglect friction between all the surfaces. Given: $m_B = 10$ kg and mA = 16 kg.

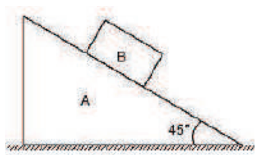


Fig. 18

Solution: Drawing velocity triangle as shown in Fig. 18 (a).

Applying principle of conservation of momentum to the particles A and B

$$m_B\, v_{BA} \cos è - (m_A + m_B) v_A = 0$$

Therefore, speed of block B relative to wedge A:

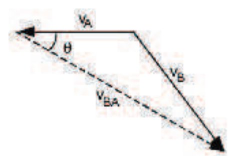


Fig. 18 a

$$v_{BA} = \frac{(m_A + m_B) v_A}{m_B \cos\theta}$$

Applying conservation of energy rule (as there is no friction)

$$m_B g h = \frac{1}{2}\, m_A v_A^2 + \frac{1}{2}\, m_B v_B^2$$

Where $v_B^2 = v_A^2 + v_{BA}^2 - 2 v_A v_{BA} \cos\theta$ (cosine rule from triangle Fig. 18 a

$$10 \times 9.81 \times s\ \cos\theta = \frac{1}{2}\, m_A m_A^2 + \frac{1}{2}\, m_B m_B^2$$

Here v_B and v_{BA} can be replaced in terms of v_A and plot of v_A and s can be drawn

$$m_B g.s\cos\theta = \frac{1}{2}\,m_A v_A^2 + \frac{1}{2}\,m_B v_{AB}^2\left(1 + \frac{(m_A+m_B)^2}{m_B^2\cos^2\theta} - \frac{(m_A+m_B)}{m_B}\right)$$

$$= 0.5\ \times\left[m_A + m_B\left(1 + \left\{\frac{m_A+m_B}{m_B\cos\theta}\right\}\right) - 2\left\{\frac{(m_A+m_B)}{m_B\cos\theta}\right\}\cos\theta\right] v_A^2$$

Now speed of wedge v_A and relative velocity v_{BA} are plotted as a function of s. Complete MATLAB program is given below:

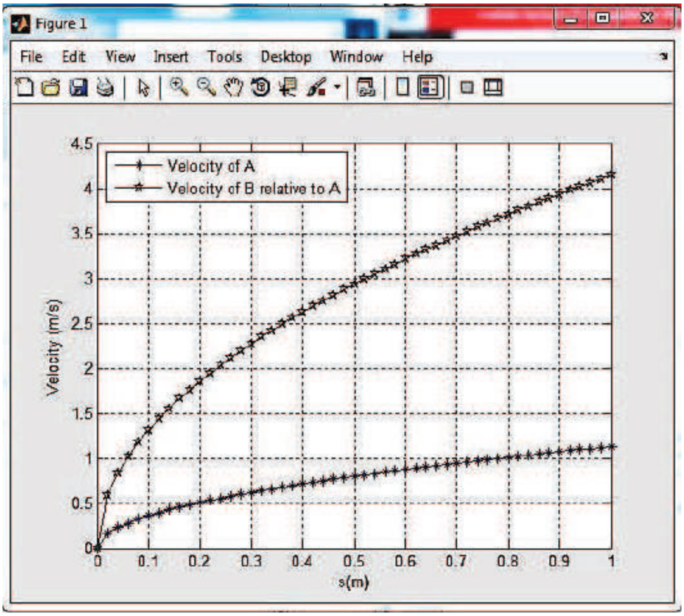
MATLAB Program:

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
%DEFAULT DATA GIVEN
g=9.81;
ma=16;
mb=10;
theta=45*pi/180; % ANGLE OF WEDGE
mul=(ma+mb)/(mb*cos(theta)); % MULTIPLICATION FACTOR
den=0.5*ma+0.5*mb*(1+mul^2-2*mul*cos(theta));
s=[0:0.02:1];
pe=mb*s*cos(theta)*g; % POTENTIAL ENERGY
va=sqrt(pe./den); % EXPRESSION FOR VELOCITY OF A
vba=mul*va; % RELATIVE VELOCITY
plot(s,va,'-*',s,vba,'-p')
xlabel('s(m)')
ylabel('Velocity (m/s)')
legend('Velocity of A','Velocity of B relative to A',2);
grid on
```

Output is shown in Fig. below.



Example 19: Figure 19 shows the slider crank mechanism. Write a MATLAB program that calculates and plots the position, velocity and acceleration of the piston for one full revolution of the crank. Assume that the crank is rotating at a constant speed of 550 rpm. Given radius of crank = 125 mm and radius of crank shaft = 250 mm.

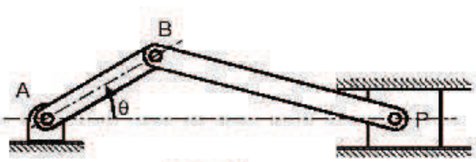


Fig. 19

Solution: This problem can be done with either absolute motion analysis or relative motion analysis. Let us do it with absolute motion analysis, where

the coordinates of points B and P are defined first with common origin A and then differentiated with respect to time to obtain velocities.
Figure 19 a shows the line diagram of the mechanism.

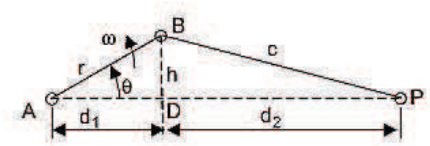


Fig. 19 a

The crank is rotating with a constant angular velocity $\omega = \dot{\theta}$
When t = 0, θ = 0º.
At time t, the angle θ is given by

$\theta = \omega t = \dot{\theta}\, t$, and

That $\ddot{\theta} = 0$ at all times.

The distances d_l and h are given by

$d_l = r \cos\theta$ and $h = r \sin\theta$

Knowing h, the distance d2 is obtained as:

$$d_2 = (c^2 - h^2)^{1/2} = (c^2 - r^2\ \sin^2\theta)^{1/2}$$

The position x of the piston P with respect to A (common origin) is given by

$$x = d_1 + d_2 = r\ \cos\theta + (c^2 - r^2\ \sin^2\theta)^{1/2}$$

The velocity of the piston is given by

$$\dot{x} = -r\,\dot{\theta}\sin\theta - \frac{r^2}{2\,(c^2 - \ r^2 \sin^2\theta)^{1/2}}$$

The acceleration of the piston is given by

$$\ddot{x} = -r\,\dot{\theta}^2 \cos\theta - \frac{4\,r^2\dot{\theta}^2 \cos 2\theta\,(c^2 - \,r^2 \sin^2\theta) + \left(\,r^2\,\dot{\theta} sin\,2\theta\right)^2}{4\,(c^2 - \,r^2 \sin^2\theta)^{3/2}}$$

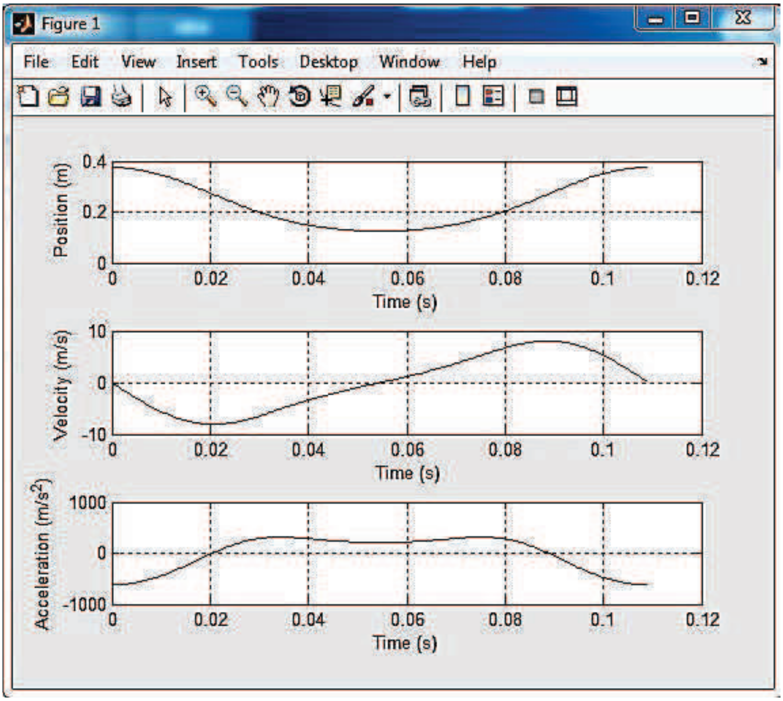
Complete MATLAB program for this problem is given below:

Editor - C:\Users\Design\Documents\MATLAB\training.m*

training.m*

```
% MATLAB Solution:
% Define TD, r, and c
N=550; % Speed in rpm
TD=N*2*pi/60; % Speed in radians/sec
tf=2*pi/TD;
r=0.125; % radius of crank in meters
c=0.250; % length of connecting rod in meters
t=linspace(0,tf,200); % Create a vector with 200 elements
TH=TD*t; % Compute Theta for each t
d2s=c^2-r^2*sin(TH).^2; % d2 squared
x=r*cos(TH)+sqrt(d2s); % Calculate x for each Theta
xd=-r*TD*sin(TH)-(r^2*TD*sin(2*TH))./(2*sqrt(d2s)); % Velocity
% Acceleration
xdd=-r*TD^2*cos(TH)-(4*r^2*TD^2*cos(2*TH)...
*d2s+(r^2*sin(2*TH)*TD).^2)./(4*d2s.^(3/2));
subplot(3,1,1)
plot(t, x)% Plot x versus t
grid
xlabel('Time (s)')
ylabel(' Position (m)')
subplot(3,1,2)
plot(t, xd) % Plot Velocity vs. t
grid
xlabel(' Time (s)')
ylabel('Velocity (m/s)')
subplot(3,1,3)
plot(t, xdd) % Plot Acceleration Vs. t
grid
xlabel('Time (s)')
ylabel('Acceleration (m/s^2)')
```

Output of the program is shown in Fig. below



Example 20: A 30 kg disk is pin-supported at its center. It is acted upon by a constant force F =10 N which is applied to a cord wrapped around its periphery and a constant couple 5 Nm. Plot the variation of angular speed with the number of revolutions it makes. Assume the system started from rest.

Solution: Figure ED7.24 shows the configuration of the system.

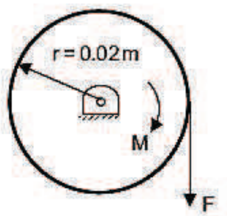


Fig. 20 **Configuration of the system**

Here as angular speed and displacement are involved, one can apply work-energy principle. But remember that it is rigid-body motion.

i.e., $T_1 + \Sigma U_{1\text{-}2} = T_2$

Here T_1 = initial kinetic energy of the system =0

T_2 = final kinetic energy = ½.Iω^2. Where I = ½ .mr^2

$\Sigma U_{1\text{-}2}$ = work done by force and moment = Mθ +Fs = (M +Fr) θ

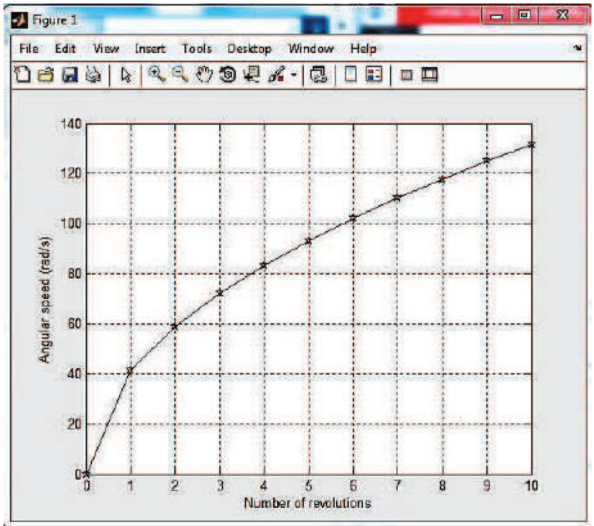
Hence $(M + Fr)\theta = \frac{1}{2}\, I\omega^2 = \frac{1}{2}\left(\frac{1}{2}m\omega r^2\right) = \frac{1}{4}m\omega^2 r^2$

A simple program that relates ω and θ is given below:

Editor - C:\Users\Design\Documents\MATLAB\training.m

training.m

```
% Initialize values
F=10;
M=5;
m=30;
r=0.02;
theta=0:10;
omega=2*sqrt((M+F*r).*theta/(m*r^2));
plot (theta,omega,'-p');
xlabel('Number of revolutions');
ylabel('Angular speed (rad/s)');
grid on
```



Example 21: A block of 0.8 kg mass moves within the smooth vertical slot as shown in Fig. 21. If it starts from rest when the attached spring is in unstretched position at A, plot a graph of velocity of block as a function of distanced moved by the block. Assume F =100 N, k = 100 N/m, $0 \leq s \leq 0.4$.

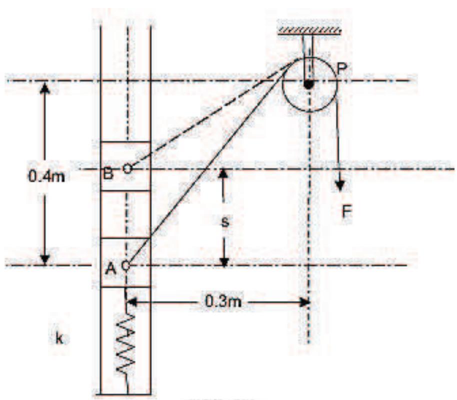


Fig. 21

Solution: This is an application of work-energy principle. As surfaces are smooth, no frictional force is possible. Thus, there are no non-conservative forces in the system. So work-energy principle reduces to principle of conservation of energy, mathematically

$T_1 + V_1 = T_2 + V_2$

Where initial state is rest. Thus, $T_1 = V_1 = 0$. Here, initially if spring is compressed then potential energy, $V_1 = ½\, k\partial^2_{ini}$

Where ∂_{ini} is initial compression of spring.

Final energies are written as, $T_2 = ½\ mv^2$,

Where v is velocity of block

And V_2 = elastic energy + gravitational energy + work due to external force F.

Thus $\int_{x=0}^{\partial} kxdx + mg \times s - F \times \Delta f$

Where Δf = AP– BP= stretch in string length, which can be expressed in terms of s. From geometry,

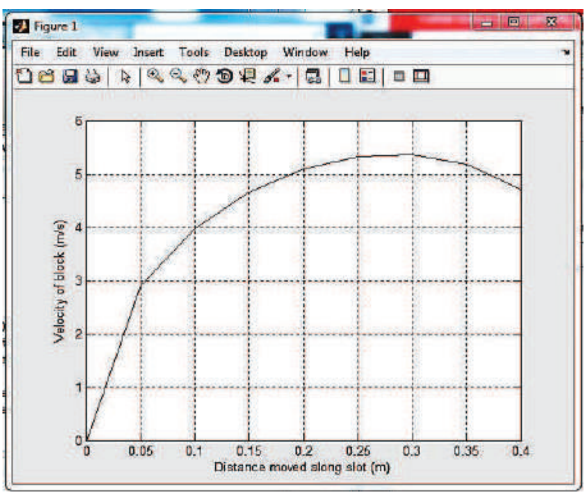
$$AP = \sqrt{0.4^2 + 0.3^2} = 0.5\, m$$

$$BP = \sqrt{(0.4^2 - s)^2 + 0.3^2}$$

Now the principle of conservation of energy can be applied to relate velocity v and distance s. i.e., $V_2 + T_2 = 0$

MATLAB program for this problem is generalized as follows:

```
% Set of default values
g=9.81;
m=0.8;
k=100;
F=100;
W=m*g;% weight of block
s=0:0.05:0.4;
df= 0.5-sqrt((0.4-s).^2+0.3^2); % distance moved by string
V2= (W.*s+0.5*k*s.^2-F.*df); % potential energy of block
v=sqrt((2/m).*(-V2)); % Velocity of block
plot(s,v);
xlabel('Distance moved along slot (m)');
ylabel('Velocity of block (m/s)')
grid on;
```



References

[1] D. Houcque, “Introduction to MATLAB for Engineering Students,” *Northwest. Univ. Version*, no. August, 2005.

[2] D. Hahn, Brian, Valentine, *Essential MATLAB for Engineers and Scientists*, 3rd ed. Newnes, 2007.

[3] M. Kalechman, *Practical MATLAB Basics for Engineers*, 1st ed. New York: CRC Press, 2008.

[4] R. Dukkipati, *MATLAB: An Introduction With Applications*, 1st ed. New Age International (P) Ltd ,Publushers, 2010.

[5] W. Palm, *Introduction to Matlab for Engineers, III ed*, 3rd ed. McGraw-Hill Education, 2010.

[6] E. B. Magrab, *An engineer’s guide to MATLAB : with applications from mechanical, aerospace, electrical, civil, and biological systems engineering*, 3rd ed. Prentice Hall, 2011.

[7] C. Warren, “An interactive introduction to Matlab,” 2012.

[8] S. Chapra, *Applied Numerical Methods with MATLAB for Engineers and Scientsts*, 3rd ed. New York: McGraw-Hill Companes, 2012.