COMPUTATIONAL MECHANICALENGINEERING LABORATORY

LAB MANUAL

Year	:	2017- 2018
Subject Code	:	AME106
Regulations	:	IARE-R16
Class	:	IV Semester
Branch	:	Mechanical Engineering

Prepared By

Mr. V. V. S. H. Prasad, Professor
Ms. T. Vanaja, Assistant Professor

MECHANICAL ENGINEERING

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal, Hyderabad - 500 043

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

Program Outcomes	
PO1	**Engineering Knowledge:** Capability to apply knowledge of Mathematics, Science Engineering in the field of Mechanical Engineering
PO2	**Problem Analysis:** An ability to analyze complex engineering problems to arrive at relevant conclusion using knowledge of Mathematics, Science and Engineering.
PO3	**Design/ Development of solution:** Competence to design a system, component or process to meet societal needs within realistic constants.
PO4	**Conduct investigation of complex problems:** To design and conduct research oriented experiments as well as to analyze and implement data using research methodologies.
PO5	**Modern Tool usage:** An ability to formulate solve complex engineering problems using modern engineering and information technology tools.
PO6	**The Engineer society:** To utilize the engineering practices, techniques, skills to meet needs of health, safety legal, cultural and societal issues.
PO7	**Environment and Sustainability:** To understand the impact of engineering solution in the societal context and demonstrate the knowledge for sustainable development.
PO8	**Ethics:** An understanding and implementation of professional and Ethical responsibilities.
PO9	**Individual Team work:** To function as an effective individual and as a member or leader in multi-disciplinary environment and adopt in diverse teams.
PO10	**Communication:** An ability to assimilate, comprehends, communicate, give and receive instructions to present effectively with engineering community and society.
PO11	**Project Management and Finance:** An ability to provide leadership in managing complex engineering project at multi-disciplinary environment and to become a professional engineer.
PO12	**Life-Long learning:** Recognition of the need and an ability to engage in lifelong learning to keep abreast with technological changes.
Program Specific Outcomes	
PSO1	**Professional Skills:** To produce engineering professional capable of synthesizing and analyzing mechanical system including allied engineering streams.
PSO2	**Design/ Analysis:** An ability to adapt and integrate current technologies in the design and manufacturing domain to enhance the employability.
PSO3	**Successful Career and Entrepreneurship:** To build the nation by imparting technological inputs and managerial skills to become a Technocrats.

ATTAINMENT OF PROGRAM OUTCOMES and PROGRAM SPECIFIC OUTCOMES

Exp. No.	Experiment	Program Outcomes Attained	Program Specific Outcomes Attained
1	Introduction to MATLAB.	PO1, PO3, PO5	PSO1, PSO2
2	Uses of MATLAB.	PO1, PO2, PO5	PSO1, PSO2
3	MATLAB program	PO1, PO2, PO3, PO5	PSO1, PSO2
4	MATLAB. program	PO1, PO3, PO5	PSO1, PSO2
5	MATLAB program	PO2, PO3,PO5	PSO1, PSO2
6	MATLAB program	PO1, PO2, PO3	PSO1, PSO2
7	MATLAB program	PO2, PO3, PO5	PSO1, PSO2

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

Certificate

This is to certify that it is a bonafied record of practical work done by Sri/Kum. ____ bearing the Roll No. _________________________ of class ______________ ____ branch in the ________________________________ laboratory during the academic year ______________________ under our supervision.

Head of the Department **Lecturer In-Charge**

External Examiner **Internal Examiner**

COMPUTATIONAL MECHANICAL ENGINEERING LABORATORY

IV Semester: ME								
Course Code	Category	Hours / Week			Credits	Maximum Marks		
AME106	Core	L	T	P	C	CIA	SEE	Total
		-	-	3	2	30	70	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 36				Total Classes: 36		

OBJECTIVES:

The courses should enable the students to:

I. Develop MAT LAB programs for simple and complex engineering problems.
II. Interprete the output graphical plots for the given governing equation.
III. Apply the MATLAB programming to real time applications.

LIST OF EXPERIMENTS

Week	Title
Week-1	INTRODUCTION TO MATLAB
Features of MATLAB.	
Week-2	MATLAB
Uses of MATLAB.	
Week-3	MATLAB PROGRAM
Analysis of kinematics in four bar mechanism.	
Week-4	MATLAB PROGRAM
Thermal stress analysis of Piston.	
Week-5	MATLAB PROGRAM
Formulation of ideal and real gas equations.	
Week-6	MATLAB PROGRAM
Dynamics and vibration analysis	
Week-7	MATLAB PROGRAM
Pipe flow analysis.	

Reference Books:

1. Delores M. Etter, David C. Kuncicky , Holly Moore, "Introduction to MATLAB 7", Pearson Education Inc, 1st Edition,, 2009.
2. Rao. V. Dukkipati , "MATLAB for ME Engineers" , New age Science, 1st Edition, 2008.
3. Agam Kumar Tyagi, "MATLAB and Simulink for Engineers", Oxford University Press 1st Edition, 2012.

Web References:

1. http://www.tutorialspoint.com/matlab/
2. http://in.mathworks.com/products/matlab/?requestedDomain=www.mathworks.com
3. http://www.iare.ac.in

COMPUTATIONAL MATHEMATICS LABORATORY

EXPERIMENT-I

1.1 OBJECTIVES

a. To Know the history and features of MATLAB
b. To Know the local environment of MATLAB

1.1.1 CONTENT

Introduction
MATLAB is a high-level language and interactive environment for numericalcomputation, visualization, and programming. Using MATLAB, you can analyse data, develop algorithms, and create models and applications. The language, tools, and built-inmath functions enable you to explore multiple approaches and reach a solution fasterthan with spread sheets or traditional programming languages, such as C/C++ or Java. You can use MATLAB for a range of applications, including signal processing andcommunications, image and video processing, control systems, test and measurement, computational finance, and computational biology. More than a million engineers andscientists in industry and academia use MATLAB, the language of technical computing.

History

- Developed primarily by Cleve Moler in the 1970'sDerived from FORTRAN subroutines LINPACK and EISPACK, linear and eigenvaluesystems.
- Developed primarily as an interactive system to access LINPACK and EISPACK.
- Gained its popularity through word of mouth, because it was not socially distributed.
- Rewritten in C in the 1980's with more functionality, which include plotting routines.
- The Math Works Inc. was created (1984) to market and continue development of
  MATLAB.

Strengths

- MATLAB may behave as a calculator or as a programming language
- MATLAB combine nicely calculation and graphic plotting.
- MATLAB is relatively easy to learn
- MATLAB is interpreted (not compiled), errors are easy to fix
- MATLAB is optimized to be relatively fast when performing matrix operations
- MATLAB does have some object-oriented elements

Weaknesses

- MATLAB is not a general purpose programming language such as C, C++, or
  FORTRAN
- MATLAB is designed for scientific computing, and is not well suitable for other applications

- MATLAB is an interpreted language, slower than a compiled language such as C++
- MATLAB commands are specific for MATLAB usage. Most of them do not have a direct equivalent with other programming language commands

Competition

One of MATLAB's competitors is Mathematica the symbolic computation program. MATLAB is more convenient for numerical analysis and linear algebra. It is frequently used in engineering community. Mathematica has superior symbolic manipulation, making it popular among physicists.
There are other competitors: Scilab, GNU Octave, and Rlab

Key Features

- It is a high-level language for numerical computation, visualization and application development.
- It also provides an interactive environment for iterative exploration, design and problem solving.
- It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
- It provides built-in graphics for visualizing data and tools for creating custom plots.
- MATLAB's programming interface gives development tools for improving code quality, maintainability, and maximizing performance.
- It provides tools for building applications with custom graphical interfaces.
- It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

MATLAB's Power of Computational Mathematics

MATLAB is used in every facet of computational mathematics. Following are some commonly used mathematical calculations where it is used most commonly:

- Dealing with Matrices and Arrays
- 2-D and 3-D Plotting and graphics
- Linear Algebra
- Algebraic Equations
- Non-linear Functions
- Statistics
- Data Analysis
- Calculus and Differential Equations
- Numerical Calculations
- Integration
- Transforms
- Curve Fitting
- Various other special functions

Uses of MATLAB

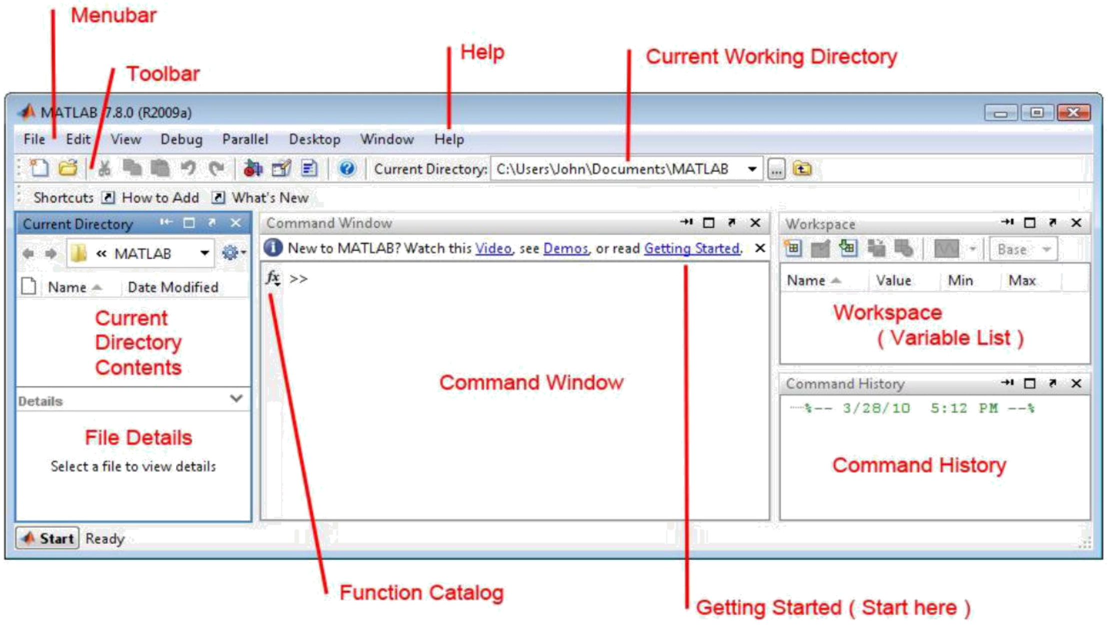
MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including:

- Signal processing and Communications
- Image and video Processing
- Control systems
- Test and measurement
- Computational finance
- Computational biology

Understanding the MATLAB Environment

MATLAB development IDE can be launched from the icon created on the desktop. The main working window in MATLAB is called the desktop. When MATLAB is started, the desktop appears in its default layout:

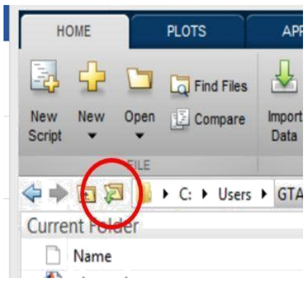
The MATLAB Work Environment



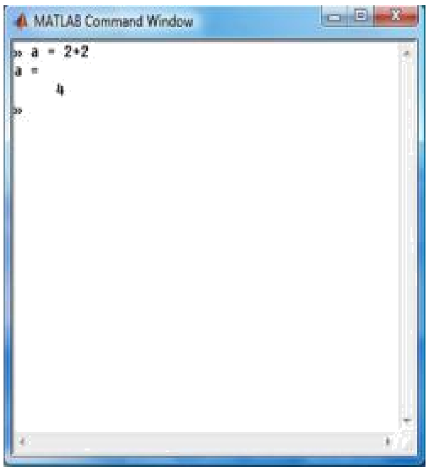
Menubar
Toolbar
Help
Current Working Directory
MATLAB 7.8.0 (R2009a)
File Edit View Debug Parallel Desktop Window Help
Current Directory: C:\Users\John\Documents\MATLAB
Shortcuts How to Add What's New
Current Directory
Current Directory Contents
Details
File Details
Select a file to view details
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Command Window
Workspace
Name Value Min Max
Workspace (Variable List)
Command History
--%-- 3/28/10 5:12 PM --%
Command History
Start Ready
Function Catalog
Getting Started (Start here)

The desktop includes these panels:

Current Folder - This panel allows you to access the project folders and files.

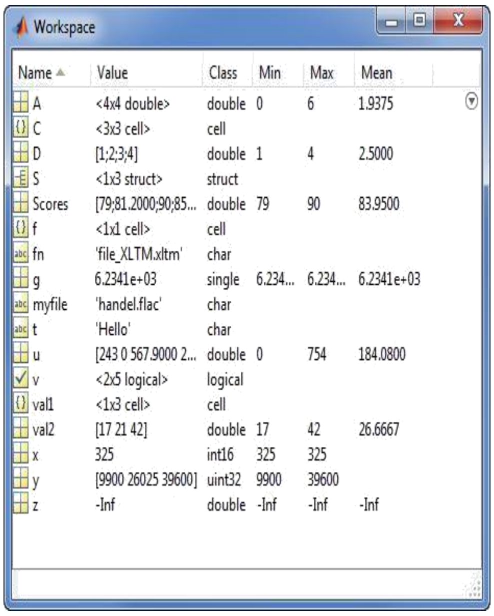


Command Window - This is the main area where commands can be entered at the command line. It is indicated by the command prompt (>>).



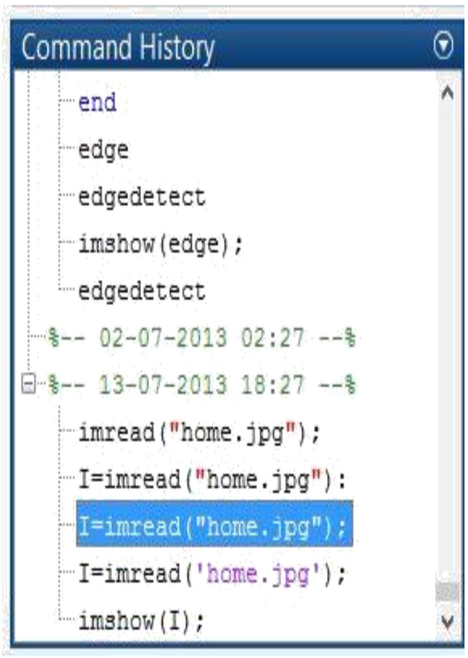
```
>> a = 2+2
a =

     4

>>
```

Workspace - The workspace shows all the variables created and/or imported from files.

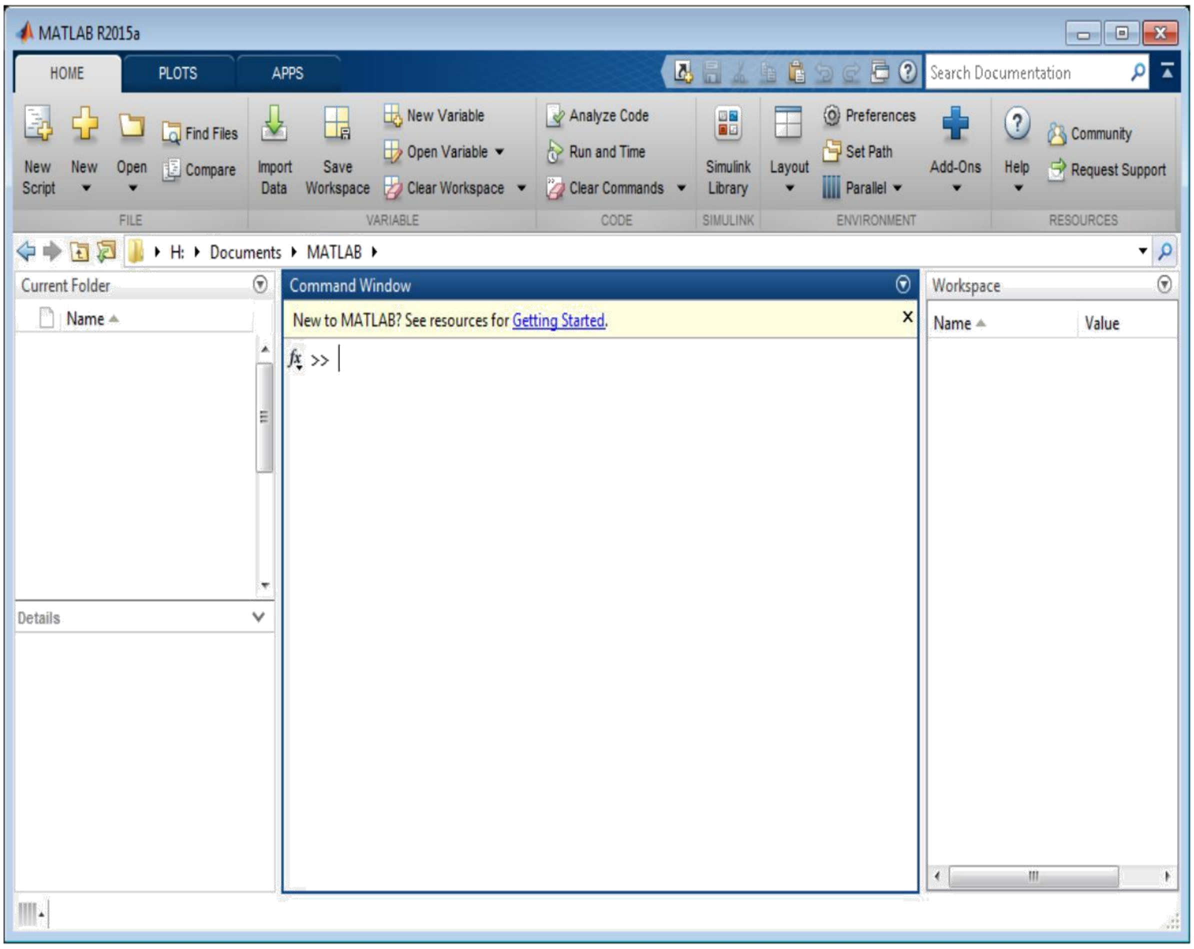


Name	Value	Class	Min	Max	Mean
A	<4x4 double>	double	0	6	1.9375
C	<3x3 cell>	cell			
D	[1;2;3;4]	double	1	4	2.5000
S	<1x3 struct>	struct			
Scores	[79;81.2000;90;85...	double	79	90	83.9500
f	<1x1 cell>	cell			
fn	'file_XLTM.xltm'	char			
g	6.2341e+03	single	6.234...	6.234...	6.2341e+03
myfile	'handel.flac'	char			
t	'Hello'	char			
u	[243 0 567.9000 2...	double	0	754	184.0800
v	<2x5 logical>	logical			
val1	<1x3 cell>	cell			
val2	[17 21 42]	double	17	42	26.6667
x	325	int16	325	325	
y	[9900 26025 39600]	uint32	9900	39600	
z	-Inf	double	-Inf	-Inf	-Inf

Command History - This panel shows or rerun commands that are entered at the command line.



```
end
edge
edgedetect
imshow(edge);
edgedetect
%-- 02-07-2013 02:27 --%
%-- 13-07-2013 18:27 --%
imread("home.jpg");
I=imread("home.jpg"):
I=imread("home.jpg");
I=imread('home.jpg');
imshow(I);
```



New to MATLAB? See resources for Getting Started.

```
fx >>
```

You are now faced with the MATLAB desktop on your computer, which contains the prompt (>>) in the Command Window. Usually, there are 2 types of prompt:

>>For full version
EDU> for educational version

Note:

1. To simplify the notation, we will use this prompt, >>, as a standard prompt sign, though our MATLAB version is for educational purpose.
2. MATLAB adds variable to the workspace and displays the result in the Command Window.

Managing workspace and file commands

Command	Description
cd	Change current directory
clc	Clear the Command Window
clear (all)	Removes all variables from the workspace
clear x	Remove x from the workspace
copy file	Copy file or directory
delete	Delete files

dir	Display directory listing
exist	Check if variables or functions are defined
help	Display help for MATLAB functions
look for	Search for specified word in all help entries
mkdir	Make new directory
move file	Move file or directory
pwd	Identify current directory
rmdir	Remove directory
type	Display contents of file
what	List MATLAB files in current directory
which	Locate functions and files
who	Display variables currently in the workspace
whos	Display information on variables in the workspace

Commonly used Operators and Special Characters

MATLAB supports the following commonly used operators and special characters:

Operator	**Purpose**
+	Plus; addition operator.
-	Minus; subtraction operator.
*****	Scalar and matrix multiplication operator.
.*	Array multiplication operator.
^	Scalar and matrix exponentiation operator.
.^	Array exponentiation operator.
\	Left-division operator.
/	Right-division operator.
.	Array left-division operator.
./	Array right-division operator.
:	Colon; generates regularly spaced elements and represents an entire row or column.
( )	Parentheses; encloses function arguments and array indices; overrides precedence.
[ ]	Brackets; enclosures array elements.
.	Decimal point.
…	Ellipsis; line-continuation operator
,	Comma; separates statements and elements in a row
;	Semicolon; separates columns and suppresses display.
%	Percent sign; designates a comment and specifies formatting.

_	Quote sign and transpose operator.
._	Non-conjugated transpose operator.
=	Assignment operator.

Note:

If you end a statement with a semicolon, MATLAB performs the computation, butsuppresses the display of output in the Command Window.

Special Variables and Constants

MATLAB supports the following special variables and constants:

Name	Meaning
ans	Most recent answer.
eps	Accuracy of floating-point precision.
i,j	The imaginary unit √-1.
Inf	Infinity.
NaN	Undefined numerical result (not a number).
pi	The number π

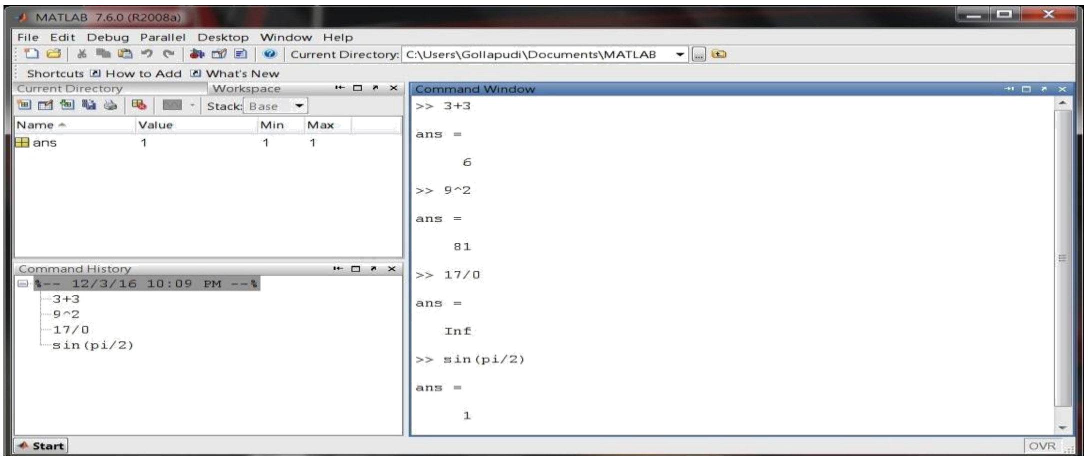
Naming Variables

Variable names consist of a letter followed by any number of letters, digits or underscore. MATLAB is **case-sensitive**.
Variable names can be of any length; however, MATLAB uses only first N characters, where N is given by the function **namelengthmax**.

Saving Your Work

The **save** command is used for saving all the variables in the workspace, as a file with .mat extension, in the current directory.
For example, save myfile
You can reload the file anytime later using the **load** command.
load myfile

Example 1:

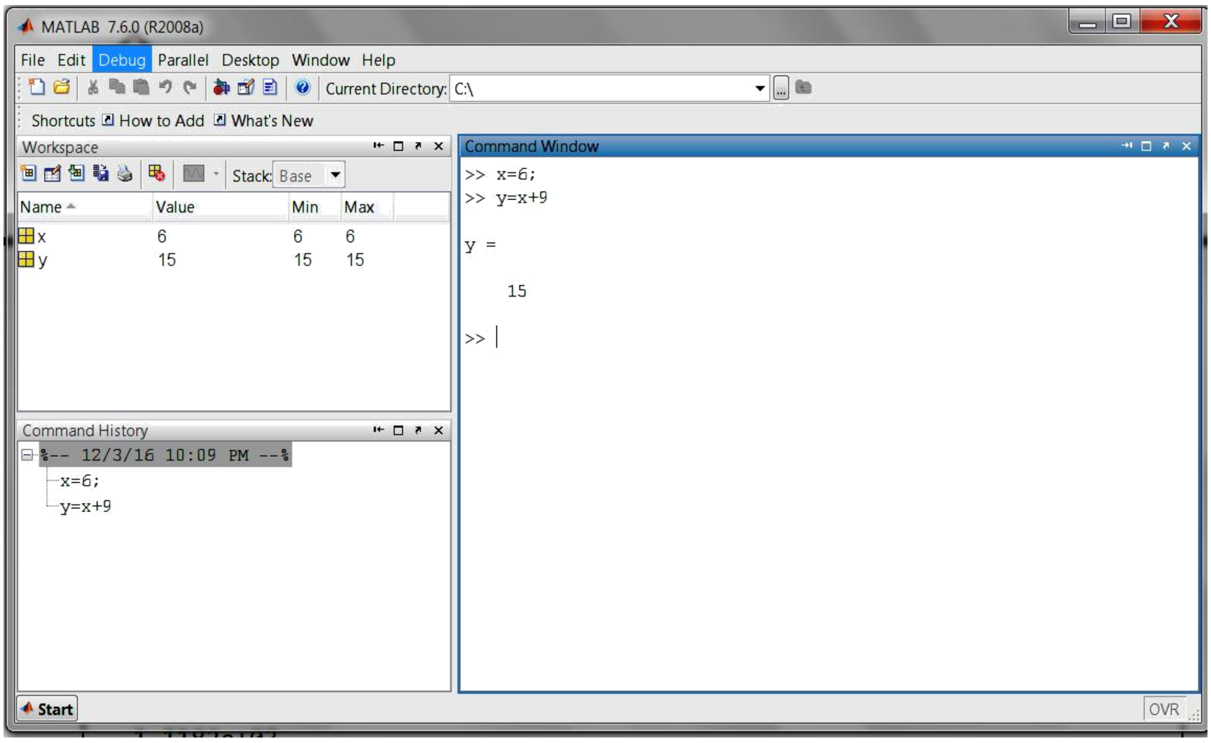

MATLAB 7.6.0 (R2008a)
File Edit Debug Parallel Desktop Window Help
Current Directory: C:\Users\Gollapudi\Documents\MATLAB
Shortcuts How to Add What's New
Current Directory Workspace
Stack: Base
Command Window
Command History
Start
OVR

Name	Value	Min	Max
ans	1	1	1

```
%-- 12/3/16 10:09 PM --%
   3+3
   9^2
   17/0
   sin(pi/2)
```

```
>> 3+3

ans =

     6

>> 9^2

ans =

    81

>> 17/0

ans =

   Inf

>> sin(pi/2)

ans =

     1
```

Example 2:



Workspace:

Name	Value	Min	Max
x	6	6	6
y	15	15	15

Command History:

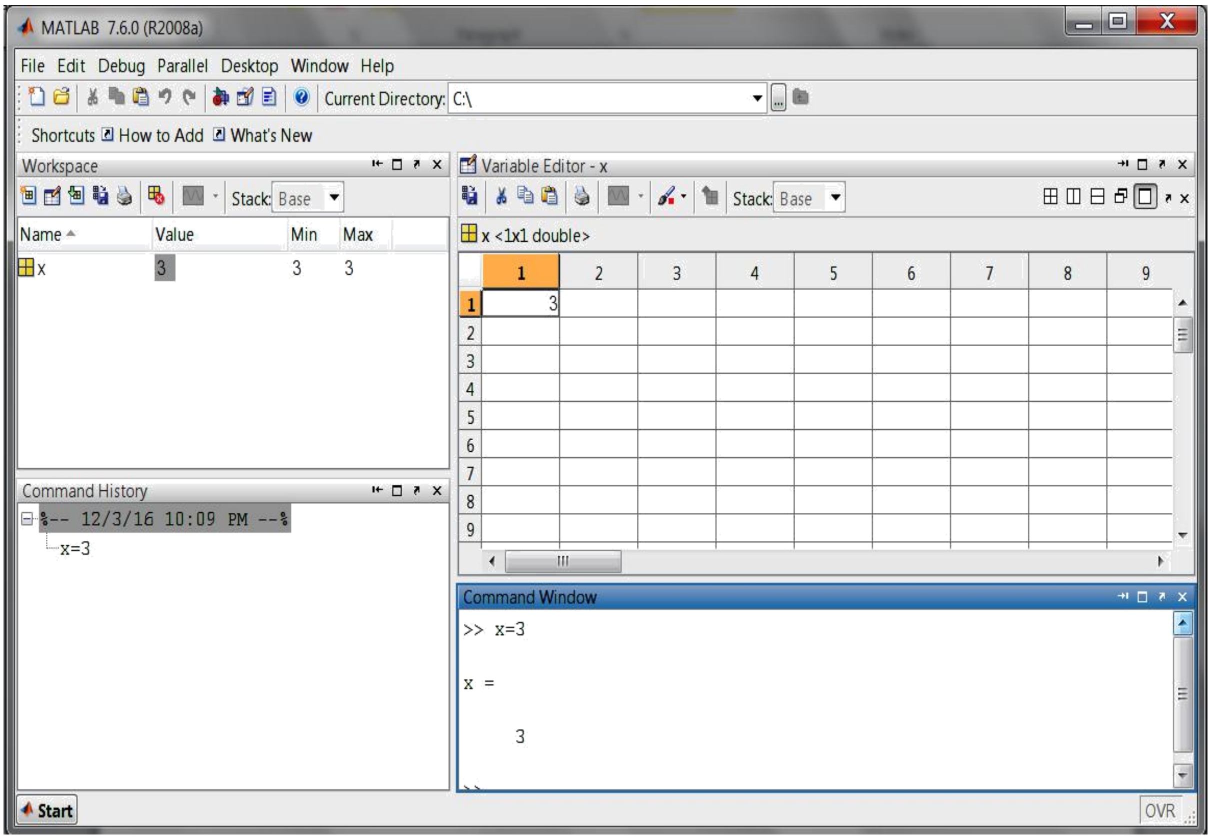
```
%-- 12/3/16 10:09 PM --%
x=6;
y=x+9
```

Command Window:

```
>> x=6;
>> y=x+9

y =

    15

>>
```

In MATLAB environment, every variable is an array or matrix.

Example 3:



Workspace:

Name	Value	Min	Max
x	3	3	3

Command History:

```
%-- 12/3/16 10:09 PM --%
x=3
```

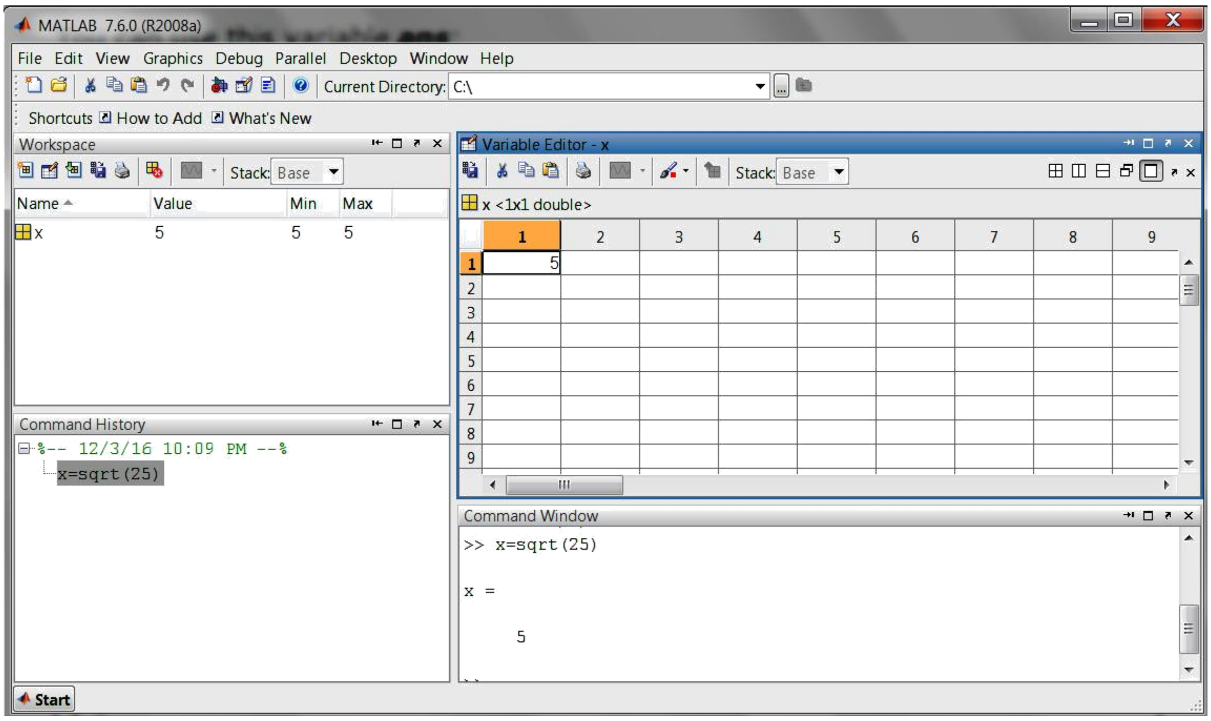
Variable Editor - x: x <1x1 double>, row 1 column 1: 3

Command Window:

```
>> x=3

x =

     3
```

In the above example it creates a 1-by-1 matrix named ‘x’and stores the value 3 in its element.

Example 4:


```
>> x=sqrt(25)

x =

     5
```

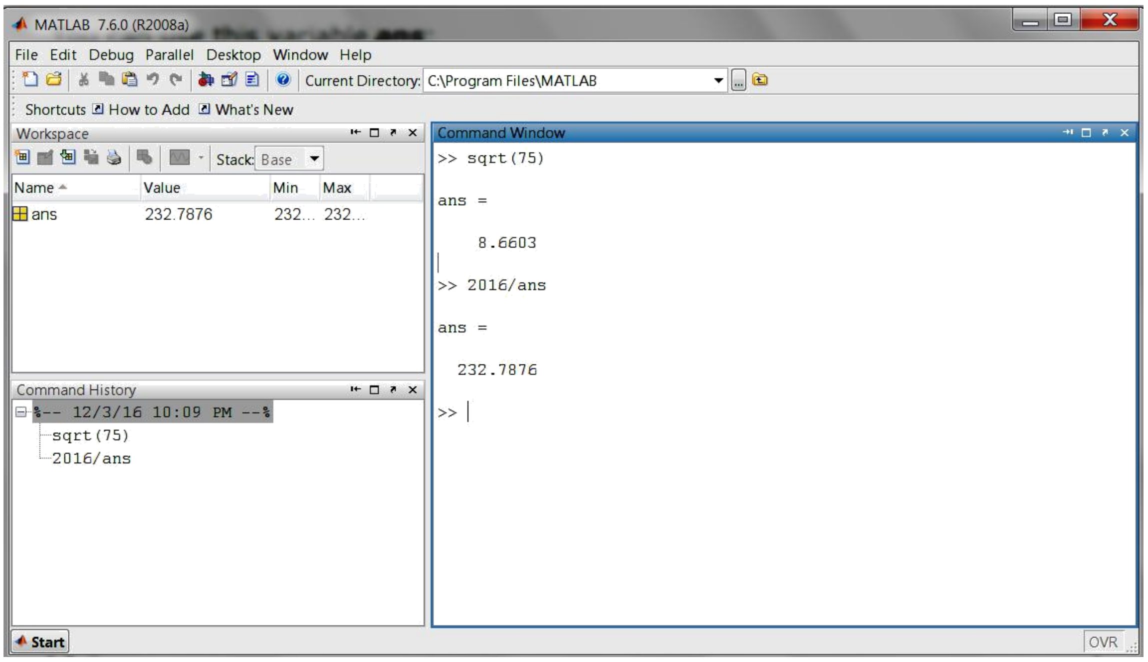
In this example x is to find the square root of 25 it creates a 1-by-1 matrix named ‘x’and stores the value 5 in its element

.

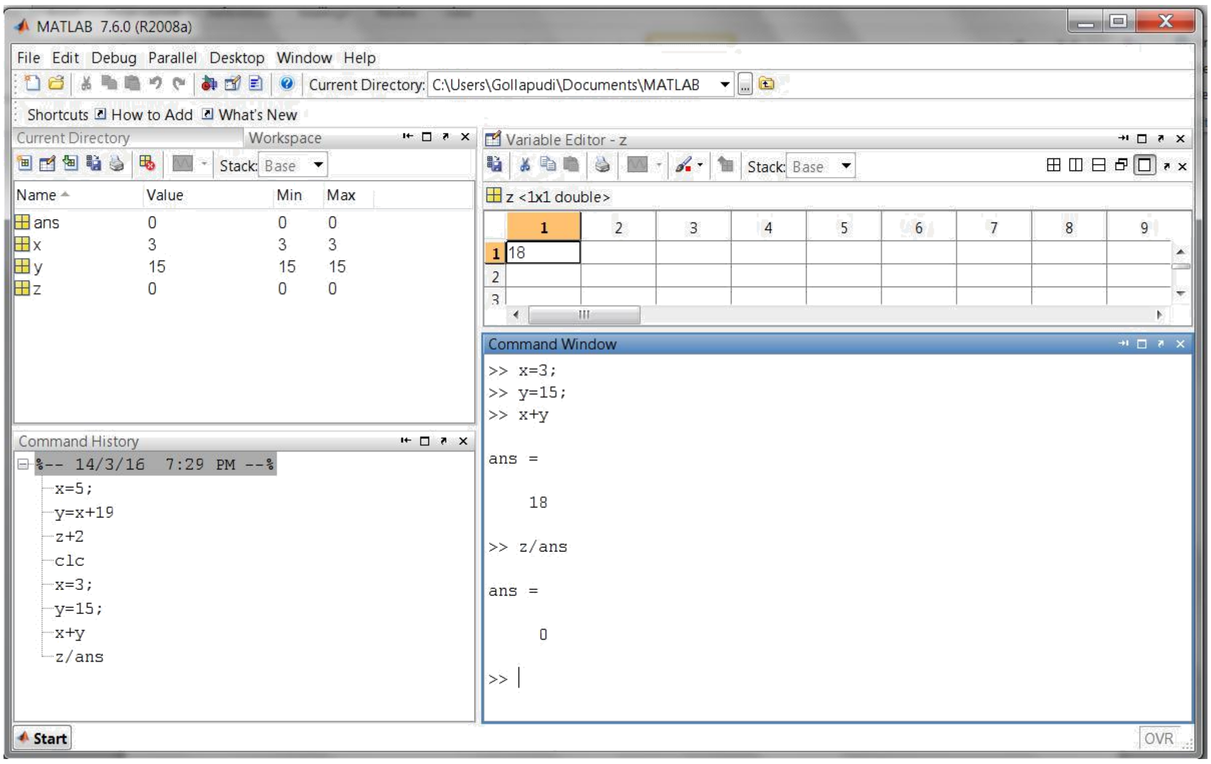
Note:

- Once a variable is entered into the system, you can refer to it later.
- Variables must have values before they are used.
- When you do not specify an output variable, MATLAB uses the variable ans, short for *answer*, to store the results of your calculation.

Example 6:


```
>> sqrt(75)

ans =

    8.6603

>> 2016/ans

ans =

  232.7876

>> 
```

Example 7:


MATLAB 7.6.0 (R2008a)
File Edit Debug Parallel Desktop Window Help
Current Directory: C:\Users\Gollapudi\Documents\MATLAB
Shortcuts How to Add What's New
Current Directory Workspace
Stack: Base
Variable Editor - z
z <1x1 double>
Command History
Command Window
Start
OVR

Name	Value	Min	Max
ans	0	0	0
x	3	3	3
y	15	15	15
z	0	0	0

	1	2	3	4	5	6	7	8	9
1	18								
2									
3									

```
%-- 14/3/16  7:29 PM --%
  x=5;
  y=x+19
  z+2
  clc
  x=3;
  y=15;
  x+y
  z/ans
```

```
>> x=3;
>> y=15;
>> x+y

ans =

    18

>> z/ans

ans =

     0

>>
```

In the above example we have multiple assignments

EXPERIMENT-2

2.1 OBJECTIVE

Analysis of kinematics in four bar mechanism.

2.2 SOFTWARE REQUIRED

1. MATLAB R2013a.
2. Windows 7/XP SP2.

2.3 PROCEDURE

1. Open MATLAB
2. Open new M-file
3. Type the program
4. Save in current directory
5. Compile and Run the program
6. For the output see command window\ Figure window

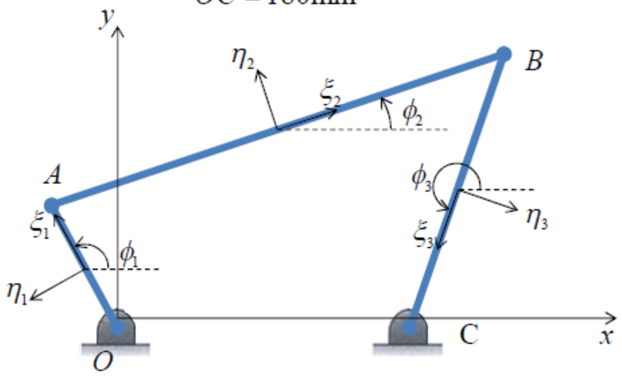
Example of a four-bar linkage mechanism

$\overline{OA} = 80\text{mm}$

$\overline{AB} = 260\text{mm}$

$\overline{BC} = 180\text{mm}$

$\overline{OC} = 180\text{mm}$



Constraint equations:

$$-x_1 + 40\cos\phi_1 = 0$$
$$-y_1 + 40\sin\phi_1 = 0$$
$$x_1 + 40\cos\phi_1 - x_2 + 130\cos\phi_2 = 0$$
$$y_1 + 40\sin\phi_1 - y_2 + 130\sin\phi_2 = 0$$
$$x_2 + 130\cos\phi_2 - x_3 + 90\cos\phi_3 = 0$$
$$y_2 + 130\sin\phi_2 - y_3 + 90\sin\phi_3 = 0$$
$$x_3 + 90\cos\phi_3 - 180 = 0$$
$$y_3 + 90\sin\phi_3 = 0$$
$$\phi_1 - 2\pi t - \pi/2 = 0$$

To solve the 9 equations for 9 unknown $\boldsymbol{q}^T = [x_1, y_1, \phi_1, x_2, y_2, \phi_2, x_3, y_3, \phi_3]$

2.4 PROGRAM

```
% Set up the time interval and the initial positions of the nine coordinates
T_Int=0:0.01:2;
X0=[0 50 pi/2 125.86 132.55 0.2531 215.86 82.55 4.3026];
global T
Xinit=X0;

% Do the loop for each time interval
for Iter=1:length(T_Int);
T=T_Int(Iter);
% Determine the displacement at the current time
[Xtemp,fval] = fsolve(@constrEq4bar,Xinit);
```

```

% Determine the velocity at the current time
phi1=Xtemp(3); phi2=Xtemp(6); phi3=Xtemp(9);
JacoMatrix=Jaco4bar(phi1,phi2,phi3);
Beta=[0 0 0 0 0 0 0 0 2*pi]';
Vtemp=JacoMatrix\Beta;

% Determine the acceleration at the current time
dphi1=Vtemp(3); dphi2=Vtemp(6); dphi3=Vtemp(9);
Gamma=Gamma4bar(phi1,phi2,phi3,dphi1,dphi2,dphi3);
Atemp=JacoMatrix\Gamma;

% Record the results of each iteration
X(:,Iter)=Xtemp; V(:,Iter)=Vtemp; A(:,Iter)=Atemp;

% Determine the new initial position to solve the equation of the next
% iteration and assume that the kinematic motion is with inertia
if Iter==1
Xinit=X(:,Iter);
else
Xinit=X(:,Iter)+(X(:,Iter)-X(:,Iter-1));
end

end
% T vs displacement plot for the nine coordinates
figure
for i=1:9;
subplot(9,1,i)
plot (T_Int,X(i,:))
set(gca,'xtick',[], 'FontSize', 5)
end
% Reset the bottom subplot to have xticks
set(gca,'xtickMode', 'auto')

% T vs velocity plot for the nine coordinates
figure
for i=1:9;
subplot(9,1,i)
plot (T_Int,V(i,:))
set(gca,'xtick',[], 'FontSize', 5)
end
set(gca,'xtickMode', 'auto')

% T vs acceleration plot for the nine coordinates
figure
for i=1:9;
subplot(9,1,i)
plot (T_Int,A(i,:))
AxeSup=max(A(i,:));
AxeInf=min(A(i,:));
if AxeSup-AxeInf<0.01
axis([-inf,inf,(AxeSup+AxeSup)/2-0.1 (AxeSup+AxeSup)/2+0.1]);
end
set(gca,'xtick',[], 'FontSize', 5)
end
set(gca,'xtickMode', 'auto')
% Determine the positions of the four revolute joints at each iteration
Ox=zeros(1,length(T_Int));
Oy=zeros(1,length(T_Int));
Ax=80*cos(X(3,:));
```

```
Ay=80*sin(X(3,:));
Bx=Ax+260*cos(X(6,:));
By=Ay+260*sin(X(6,:));
Cx=180*ones(1,length(T_Int));
Cy=zeros(1,length(T_Int));

% Animation
figure
for t=1:length(T_Int);
 bar1x=[Ox(t) Ax(t)];
 bar1y=[Oy(t) Ay(t)];
 bar2x=[Ax(t) Bx(t)];
 bar2y=[Ay(t) By(t)];
 bar3x=[Bx(t) Cx(t)];
 bar3y=[By(t) Cy(t)];

 plot (bar1x,bar1y,bar2x,bar2y,bar3x,bar3y);
 axis([-120,400,-120,200]);
 axis normal

 M(:,t)=getframe;
end
```

Initialization

```
1. % Set up the time interval and the initial positions of the nine coordinates
2. T_Int=0:0.01:2;
3. X0=[0 50 pi/2   125.86 132.55 0.2531   215.86 82.55 4.3026];
4. global T
5. Xinit=X0;
```

1. The sentence is notation that is behind symbol “%”.
2. Simulation time is set from 0 to 2 with $\Delta t = 0.01$.
3. Set the appropriate initial positions of the 9 coordinates which are used to solve nonlinear solver.
4. Declare a global variable T which is used to represent the current time t and determine the driving constraint for angular velocity.

Determine the displacement

```
10. [Xtemp,fval] = fsolve(@constrEq4bar,Xinit);
```

10. Call the nonlinear solver fsolve in which the constraint equations and initial values are necessary. The initial values is mentioned in above script. The constraint equations is written as a function (which can be treated a kind of subroutine in Matlab) as following and named as constrEq4bar. The fsolve finds a root of a system of nonlinear equations and adopts the trust-region dogleg algorithm by default.

```
function F=constrEq4bar(X)

global T

x1=X(1); y1=X(2); phi1=X(3);
x2=X(4); y2=X(5); phi2=X(6);
x3=X(7); y3=X(8); phi3=X(9);

F=[ -x1+40*cos(phi1);
    -y1+40*sin(phi1);
    x1+40*cos(phi1)-x2+130*cos(phi2);
    y1+40*sin(phi1)-y2+130*sin(phi2);
    x2+130*cos(phi2)-x3+90*cos(phi3);
    y2+130*sin(phi2)-y3+90*sin(phi3);
    x3+90*cos(phi3)-180;
    y3+90*sin(phi3);
    phi1-2*pi*T-pi/2];
```

The equation of driving constraint is depended on current time T

The code of m.file (2)

```
% T vs displacement plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,X(i,:))
    set(gca,'xtick',[], 'FontSize', 5)
end
% Reset the bottom subplot to have xticks
set(gca,'xtickMode', 'auto')

% T vs velocity plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,V(i,:))
    set(gca,'xtick',[], 'FontSize', 5)
end
set(gca,'xtickMode', 'auto')

% T vs acceleration plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,A(i,:))
    AxeSup=max(A(i,:));
    AxeInf=min(A(i,:));
    if AxeSup-AxeInf<0.01
        axis([-inf,inf,(AxeSup+AxeSup)/2-0.1 (AxeSup+AxeSup)/2+0.1]);
    end
    set(gca,'xtick',[], 'FontSize', 5)
end
set(gca,'xtickMode', 'auto')
```

Determine the acceleration

```
20.    dphi1=Vtemp(3); dphi2=Vtemp(6); dphi3=Vtemp(9);
21.    Gamma=Gamma4bar(phi1,phi2,phi3,dphi1,dphi2,dphi3);
22.    Atemp=JacoMatrix\Gamma;
```

21. Call the function Gamma4bar to obtain the right-side of the velocity equations depended on current values of velocity.
22. Solve linear equation to obtain the current acceleration.

```
function Gamma=Gamma4bar(phi1,phi2,phi3,dphi1,dphi2,dphi3)

Gamma=[ 40*cos(phi1)*dphi1^2;
        40*sin(phi1)*dphi1^2;
        40*cos(phi1)*dphi1^2+130*cos(phi2)*dphi2^2;
        40*sin(phi1)*dphi1^2+130*sin(phi2)*dphi2^2;
        130*cos(phi2)*dphi2^2+90*cos(phi3)*dphi3^2;
        130*sin(phi2)*dphi2^2+90*sin(phi3)*dphi3^2;
        90*cos(phi3)*dphi3^2;
        90*sin(phi3)*dphi3^2;
        0];
```

Plot time response

```
37.figure
38.for i=1:9;
39.    subplot(9,1,i)
40.    plot (T_Int,X(i,:))
41.    set(gca,'xtick',[], 'FontSize', 5)
42.end
43.% Reset the bottom subplot to have xticks
44.set(gca,'xtickMode', 'auto')
45.
46.% T vs velocity plot for the nine coordinates
47.figure
48.for i=1:9;
37.…
```

37. Create a blank figure .

39. Locate the position of subplot in the figure.

40. Plot the nine subplots for the time responses of nine coordinates.

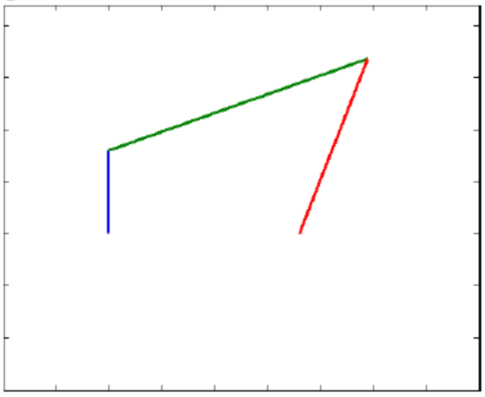
41. Eliminate x-label for time-axis and set the font size of y-label.

44. Resume x-label at bottom because the nine subplots share the same time-axis.

47.~ It is similar to above.

Animation

```
69.Ox=zeros(1,length(T_Int));
70.Oy=zeros(1,length(T_Int));
71.Ax=80*cos(X(3,:));
72.Ay=80*sin(X(3,:));
73.Bx=Ax+260*cos(X(6,:));
74.…

80.for t=1:length(T_Int);
81.    bar1x=[Ox(t) Ax(t)];
82.    bar1y=[Oy(t) Ay(t)];
83.    bar2x=[Ax(t) Bx(t)];
84.    bar2y=[Ay(t) By(t)];
85.    bar3x=[Bx(t) Cx(t)];
86.    bar3y=[By(t) Cy(t)];
87.
88.    plot (bar1x,bar1y,bar2x,bar2y,bar3x,bar3y)
89.    axis([-120,400,-120,200]);
90.    axis normal
91.
92.    M(:,t)=getframe;
93.end
```



69. Determine the displacement of revolute joint.

80. Repeat to plot the locations by continue time elapsing.

81. Determine the horizontal location of $\overline{OA}$.

88. Plot $\overline{OA}$, $\overline{AB}$, $\overline{BC}$, and $\overline{OC}$.

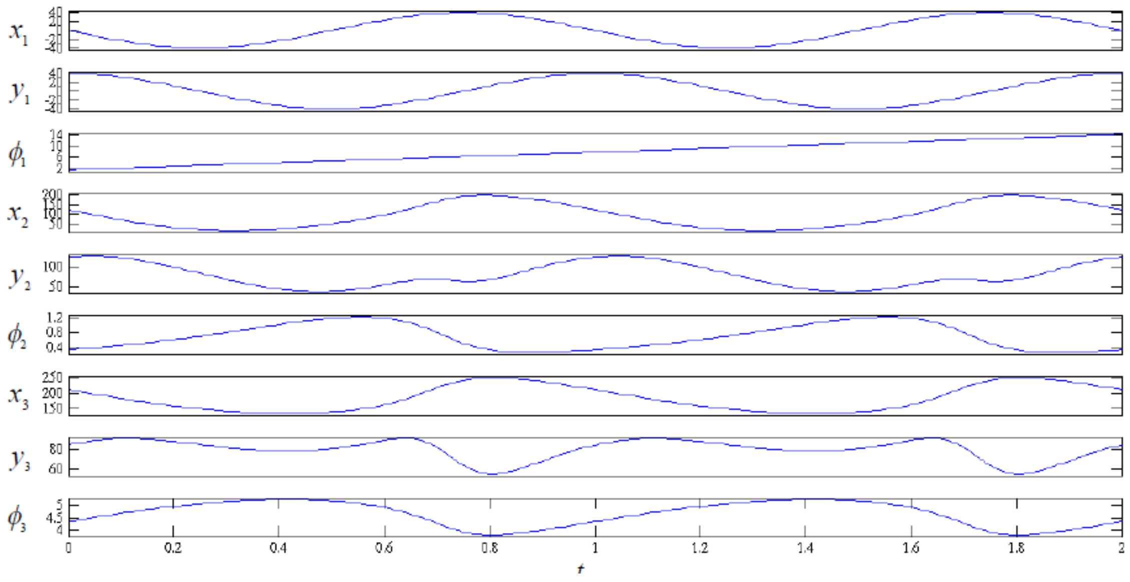
89. Set an appropriate range of axis.

Determine next initial positions

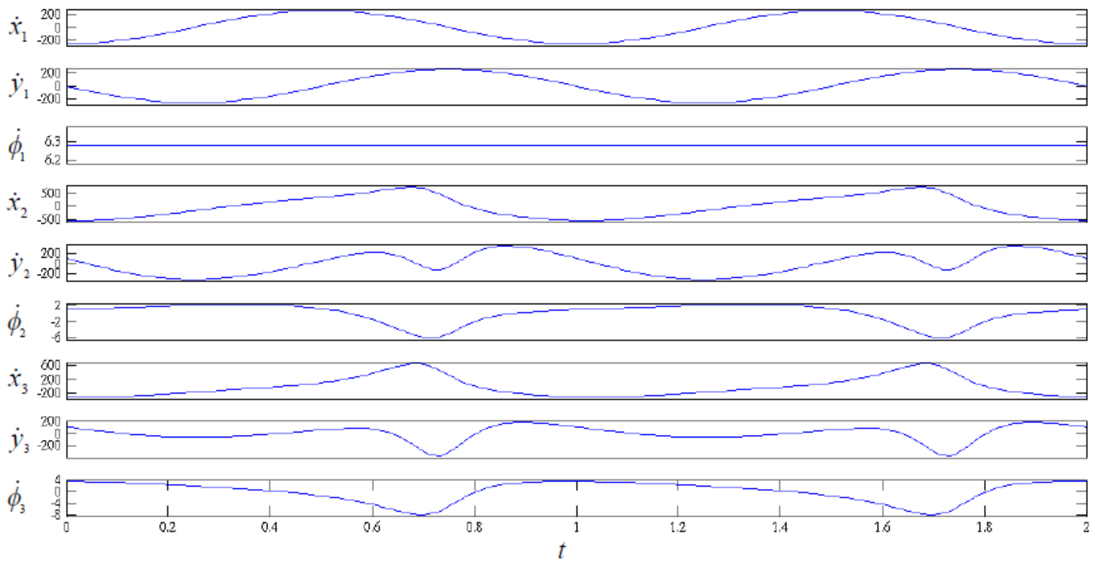
```
29.    if Iter==1
30.        Xinit=X(:,Iter);
31.    else
32.        Xinit=X(:,Iter)+(X(:,Iter)-X(:,Iter-1));
33.    end
```

29.~33. Predict the next initial positions with assumption of inertia except the first time of the loop.

Time response of displacement



Time response of velocity



EXPERIMENT-3

3.1 OBJECTIVE

Thermal stress analysis of Piston.

3.2 SOFTWARE REQUIRED

1. MATLAB R2013a.
2. Windows 7/XP SP2.

3.3 PROCEDURE

1. Open MATLAB
2. Open new M-file
3. Type the program
4. Save in current directory
5. Compile and Run the program
6. For the output see command window\ Figure window

3.4 PROGRAM

PISTON FUNCTION AND ASSOCIATED PROBLEM AREAS
The function of the piston is to absorb the energy by the gas / air which enters in the cylinder and then accelerates to produce useful mechanical energy. So to prevent the leakage & proper compressor of air /gas the piston must be sealed with the cylinder surface. This is accomplished by the piston rings which also help to prevent oil from entering in the cylinder from underneath the piston. Another function of the rings is to keep the piston from contacting the cylinder wall. Less contact area between the cylinder and piston reduces friction, thereby increasing efficiency. As a result of this process, heat is transferred from the combustion gases into the piston and other components that makeup the cylinder walls. This reduces the thermodynamic efficiency of the process, and therefore also diminishes power.

PISTON COOLING
The reduction of Temperature in the piston can bed one by heat pipe cooling method. This system allows for a channel inside the piston skirt that directs heat away from the piston itself. This will increases the heat transfer through the piston, which does not help the efficiency of the compressor, but we can use the special light alloys to form the piston. Magnesium and its alloys have much larger creep rates than other metals and therefore can usually not sustain the same load and temperatures as steel or aluminum. A heat pipe system can drastically reduce the temperature of the piston crown from about 700ºC to only 350ºC.Therefore, using heat-pipe technology makes it easier to employ magnesium alloys in pistons. Cooling gallery pistons are manufactured with water soluble salt cores or as ‘‘cooled ring carriers’’ with a sheet metal cooling gallery attached to the ring carrier. An adjusted jet with a fixed housing injects the cooling oil into the annular cooling gallery through an inlet orifice in the piston. Maintaining the correct quantity of oil decisively influences heat dissipation. The outlet is formed by one or more bores inside a piston, preferably positioned on the side of the cooling gallery approximately opposite the inlet.

PISTON TEMPERATURE DISTRIBUTION
Generally the heat flux is highest in the center of the cylinder head, in the exhaust valve seat region, and to the center of the piston. Cast-iron pistons run about40 to 800 C h otter than aluminum pistons. In the flat topped piston the center of the crown is hottest and the outer edge cooler by 20 to 500 C. The maximum temperatures occur where the heat flux is high and access for cooling is difficult. Such locations are the bridge between the valves and the region between the exhaust valves of adjacent cylinders. The heat generated by friction between the piston and the liner is a significant fraction of the liner thermal loading. Temperature distributions in the piston can be calculated from the knowledge of the heat fluxes across the component surface using finite element analysis techniques. For steady-state engine operation, the depth within a component to which the unsteady temperature fluctuations (caused by the variations in heat flux during the cycle) penetrate is
small, so a quasi-steady solution is satisfactory. In this finite element analysis, the actual piston shapewas approximated with a three-dimensional grid for one quadrant of the piston. A standard finite element analysis

of the heat flow through the piston yields the temperature distribution within the piston. The thermal stresses can therefore be calculated and added to the mechanical stress field to determine the total stress distribution. This can be used to define the potential fatigue regions in the actual piston design.

DESIGN DATA OF COMPRESSOR:
• Power Capacity : 5 H.P
• Speed : 1440 R.P.M
• Piston Displacement : 500 LPM
• Atmospheric Pressure : 1.01325 bar
• Working Pressure : 10 bar
• Temperature on top surface of Piston: 158.530 C
The above data is taken for the design of piston through which various geometries of the piston can be found out which are mentioned below. The material of the piston is Aluminum alloy 6061.
Design of the Piston can be done by general programme in MATLAB Software.

PROGRAMME IN MATLAB
```
clc
clear
k=0.04
n=1.25
p1=input('Enter the Value of atm. Pressure')
p3=input('Enter the Value of working Pressure')
p2=sqrt(p1*p3)
% VE=Volumetric Efficiency
VE=1-(k*((p2/p1)^(1/n)-1))
% ME=Mechanical Efficiency
% BP=Brake Power
% IP=Indicated Power
% WD=Work Done
ME=0.80
BP=input('Enter the Value of Brake Power')
IP=(ME)*(BP)
WD=(IP)*60
R=0.287
T1=273
T2=T1*((p2/p1)^((n-1)/n))
% T1=atm. Temperature
% T2=Intermediate Temperature
% m=Mass of Air Delivery
m=(WD*(n-1))/(R*n*((p2/p1)^((n-1)/n)-1)*(T1+T2))
m1=m/60
% FAD=Free Air Delivery
% N=Revolution per Minute
N=1440
FAD=(m1*R*T2*10)/(p2*1000*N)
% Vs=Swept Volume
Vs=FAD/VE
% L=Stroke of Piston
% D=Diameter of Piston
L=0.1
D=(sqrt((Vs*4)/(pi*L)))*10
D1=D*1000
% th=Thickness of head
% ts=Tensile Stress
% p=Working Pressure
p=1
ts=56.4
th=sqrt((3*p*(D1^2))/(16*ts))
ts=input('Enter the Value of Tensile Stress')
```

```
% t1=Radial Thickness of Piston Ring
Pw=0.035
t1=D1*(sqrt((3*Pw)/ts))
% mu=Coefficient Of Friction
% R=Maximum Side Thrust
mu=0.1
R=(mu*pi*(D1^2)*p)/4
% l=Length of Piston skirt
% Pb=Bearing Pressure
Pb=0.25
l=R/(Pb*D1)
```

The above programme is for Design of Piston of Reciprocating air Compressor. By entering various Input parameter we can get the Output of the Programme which shows the design of Piston.

OUTPUT

```
k = 0.0400
n = 1.2500
Enter the Value of atm. Pressure1.01325
p1 = 1.0133 bar
Enter the Value of working Pressure14
p3 = 14 bar
p2 = 3.7664 bar
VE = 0.9257
ME = 0.8000
Enter the Value of Brake Power 3.73 KW
BP =3.7300 KW
IP =2.9840 KW
WD =179.0400 KW
R = 0.2870 KJ/Kg K
T1 =273 K
T2 = 354.9799 K
m = 0.6616 kg/sec
m1 = 0.0110 kg/min
N =1440 R.P.M
FAD = 2.0714 x 10 -6 m3
Vs = 2.2377 x 10 -6 m3
L = 0.1000 m
D = 0.0534 m
D1 = 53.3777 mm
p = 1 N/mm2
ts = 56.4000 mm
th =3.0777 mm
Enter the Value of Tensile Stress19.6 N/mm2
ts =19.6000 mm
Pw = 0.0350 N/mm2
t1 = 3.9068 mm
mu = 0.1000
R = 223.7736 N
Pb = 0.2500 N/mm2
l = 16.7691 mm
```

EXPERIMENT-4

4.1 OBJECTIVE

Formulation of ideal and real gas equations.

SOFTWARE REQUIRED

1. MATLAB R2013a.
2. Windows 7/XP SP2.

4.3 PROCEDURE

1. Open MATLAB
2. Open new M-file
3. Type the program
4. Save in current directory
5. Compile and Run the program
6. For the output see command window\ Figure window

4.4 PROGRAM

The gas law, for example,
P = f(n,T,V) [= nRT/V]
if you give the function values for n,T,V, the function returns a value for P. The formula in brackets is what the ideal gas law would use to compute the value of the function, but other formula could also be used.

In Matlab, we would write such a function for the ideal gas law like this:

```
function P = idealP(V)

% computes pressure using the ideal gas law
% for 1 mole
% assumes the temperature is 293

R = .08206;
T = 293.0;
n = 1.0;
P = n*R*T./V;
```

Important things to note:

- The first word in the file is *function.*
- The variable to be computed, P, is on the left
- The name of the function appears to the right of = and is followed by parameters or arguments to be supplied when the function is called.
- Somewhere in the function body, the variable P must be given a value.
- All variables used in the function are *local*> to the function.
- If you expect to send in a vector as an argument, make certain any formulas in the function can handle vectors.

A more general version, as a function of n,V,T:

```
function P = idealP(n,V,T)

% computes pressure using the ideal gas law
% for given values of n,V,T
% can compute P for a range of V

R = .08206;
P = n*R*T./V;
```

EXPERIMENT-5

5.1 OBJECTIVE

Dynamics and vibration analysis

5.2 SOFTWARE REQUIRED

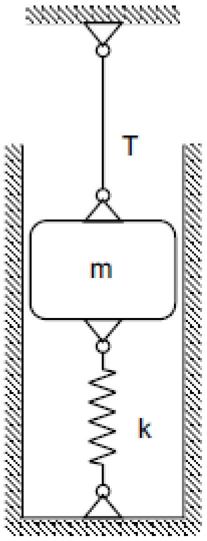
1. MATLAB R2013a.
2. Windows 7/XP SP2.

5.3 PROCEDURE

1. Open MATLAB
2. Open new M-file
3. Type the program
4. Save in current directory
5. Compile and Run the program
6. For the output see command window\ Figure window

5.4 PROGRAM

A 5 kg block is attached to a cable cable and to a spring as shown in Fig.



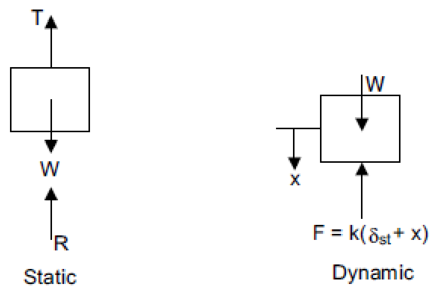
The constant of the spring is k = 3 kN/m and the tension in the cable is 30 N.
When the cable is cut,
(a) derive an expression for the velocity of the block as a function of its displacement x, (b) determine
the maximum displacement x_m and the maximum speed v_m, (c) plot the speed of the block as a function of
x for 0 δ x δ x_m.
Solution: Free-body diagram of the block before and after the cable is cut is shown in Fig.
For the static case we have entire forces are in equilibrium.

4 $T + R - W = 0$ with T =
30 N, $W = mg$ = 50 N
from which R = 20 N

But,

$$R = k\delta_{st} \quad \text{or} \quad \delta_{st} = \frac{R}{k} \text{ which is initial tension in spring.}$$

$$\delta_{st} = \frac{20}{3000} = 0.006667 \text{ m}$$



Using the principle of work and energy we have

$$T_1 + U_{1\to 2} = T_2$$

or
$$\int_0^x [W - k(\delta_{st} + x)]\, dx = \frac{1}{2}mv^2$$

Substituting yields

$$\int_0^x [50 - 3000(0.006667 + x)]dx = 2.5v^2$$

$$\Rightarrow \quad 50x - 20x - 1500x^2 = 2.5v^2$$

$$v^2 = 12x - 600x^2$$

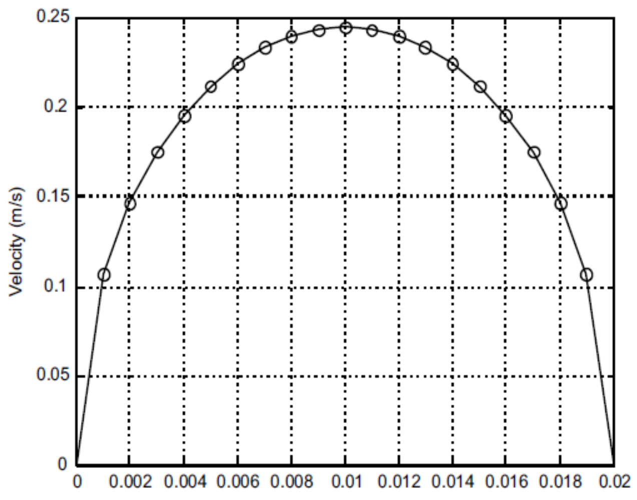
At maximum displacement, the velocity is zero.

$$\therefore \quad 12x - 600x^2 = 0 \quad \text{or} \quad x = 0.02 \text{ m.}$$

The MATLAB program for plotting can be written as follows:

```
xmax=0.02;
x=[0:.001:xmax];
v=sqrt(12.*x-600.*x.^2);% Expression for velocity for given values
[vmax,i]=max(v); % finding minimum value of velocity and corresponding
index
fprintf('The maximum velocity is %5.4fm/s and the maximum displacement is
%5.4fm\n',vmax,xmax);
figure(1)
plot(x,v,'-o')
xlabel('x,(m)')
ylabel('Velocity(m/s)')
grid on
```

Output is shown in the figure



```
The maximum velocity is 0.2449m/s and the maximum displacement is 0.0200m.
```

EXPERIMENT-6

6.1 OBJECTIVE

Pipe flow analysis.

6.2 SOFTWARE REQUIRED

1. MATLAB R2013a.
2. Windows 7/XP SP2.

6.3 PROCEDURE

1. Open MATLAB
2. Open new M-file
3. Type the program
4. Save in current directory
5. Compile and Run the program
6. For the output see command window\ Figure window

6.4 PROGRAM

Laminar flow in a pipe that is started from rest

Unsteady laminar flow is obtained by starting the flow from rest and imposing a constant pressure gradient thereafter. The effect of non slip condition on axial velocity diffuses inward from the pipe walls and reaches the centre of pipe with a time scale of $T = R^2 /v$. In the long time limit the solution for the steady flow is obtained. We assume that R = 5.0 mm, n = 0.00038 m2/s, r = 1,000 kg/m3, $dP/dz = 1.0 * 10^6$ Pa/m.

We use pdepe with m = 1 which signifies a cylindrical coordinate system We shall determine the solution for

$0 \leq t \leq 0.07$s. The program is as follows.

```
function
nu = 0.00038; rho = 1000;
dPdz = -1e6; nr = 100; rmax = 0.005;
nt = 15; tmax = 0.07;
r = linspace(0, rmax, nr);
t = linspace(0, tmax, nt);
u = pdepe(1,@pdPipe, @pdPipeIC, @pdPipeBC, r, t, [], nu, rho, dPdz);
hold on
for ijd = [2, 3, 4, 6, nt]
plot(u(ijd,:), r*1000, 'k')
if ijd == nt
text(u(ijd,20), rmax*0.25*1000, [num2str(tmax*1000, 4) ' ms'])
elseif ijd == 2
text(u(ijd,20), rmax*0.25*1000, ['t = ' num2str(t(ijd)*1000, 4) ' ms'])
else
text(u(ijd,20), rmax*0.25*1000, [num2str(t(ijd)*1000, 4) ' ms'])
end

end
xlabel('Axial Velocity, u_z (m/s)')
ylabel('r (mm)')
text(0.5*u(nt,1), 0.8*rmax*1000, ['u_z(0,' num2str(t(nt)) ') = '
num2str(u(nt,1),5)' m/s'])
function [c,f,s] = pdPipe(r, t, u, DuDr, nu, rho, dPdz)
c = 1.0/nu;
f = DuDr;
s = -dPdz/(rho*nu);
function u0 = pdPipeIC(r, nu, rho, dPdz)
u0 = 0;
function [pl, ql, pr, qr] = pdPipeBC(rl, ul, rr, ur, t, nu, rho, dPdz)
pl = 1; ql = 0;
pr = 0; qr = ur;
```