Mechanical Engineering Applications using GNU Octave

Prof. Rajanarasimha sangam
KJSCE Somaiya Vidyavihar University Mumbai

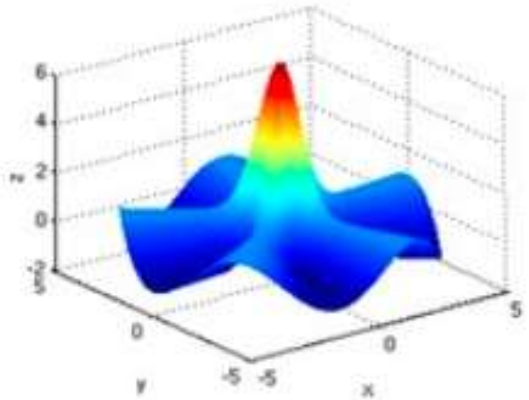
What is Octave?

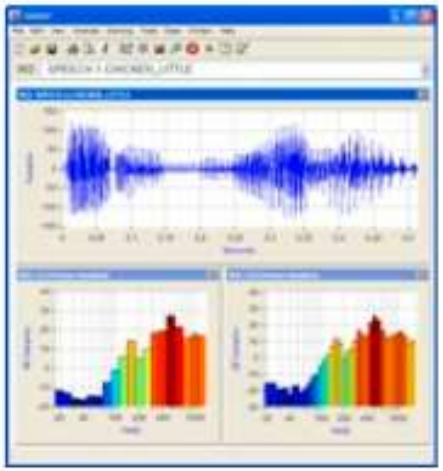
- Octave is a high-level language, intended primarily for numerical calculations. This language provides capabilities for numerical resolution of linear and nonlinear problems, and for performing other numerical tests. It also provides extensive graphical capabilities for data visualization and manipulation.
- Octave is normally used through its interactive command line interface, but it can also be used to write non-interactive programs. The Octave programming language is quite similar to Matlab, so most programs are reusable in this language.
- In this course we are going to see different aspects of this programming language that will help us strengthen certain concepts about Octave.

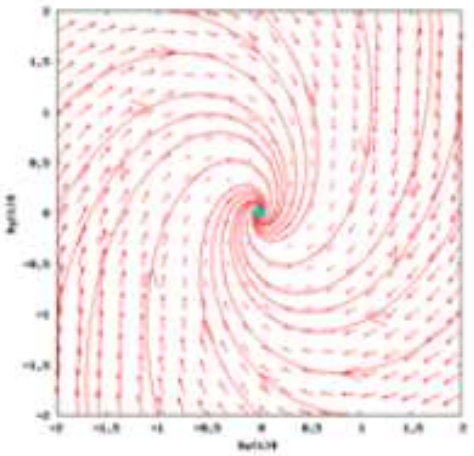
Overview

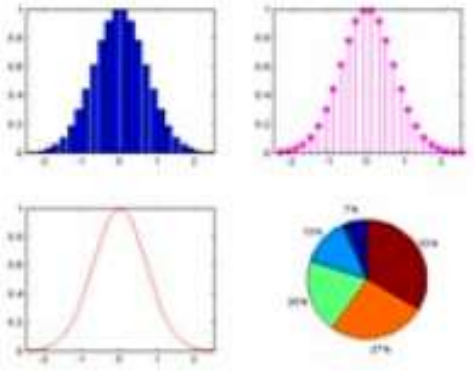
- OCTAVE is an open-source interactive software system for numerical computations and graphics.
  - Matrix computations
  - Display data in a variety of different ways
  - Programming language
  - Very powerful, programmable, graphical calculator
  - Compatible with MATLAB
- Widely used by engineers and scientists, in both industry and academia
  - NASA use it to develop spacecraft docking systems

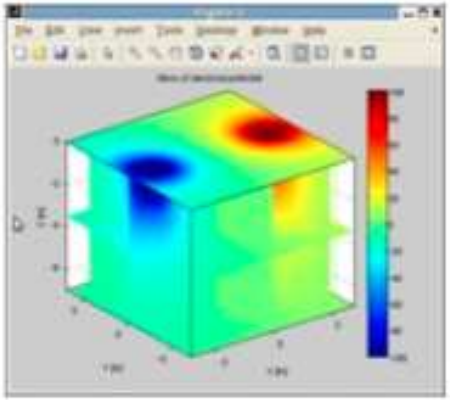
Powerful Graphics

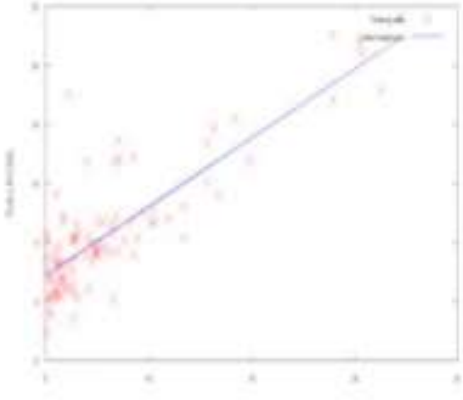












GNU Octave

- It is Mathematical language…..
  - Hence MATRIX is basic Variable
- In this matrix…
  - Individual element can be

    Number (integer,float…etc)

    Text (Character,string etc)

Whenever we create any variable in Octave

By Default it creates a MATRIX having rows and columns.

Octave Environment

- It is a powerful software tool for
  - Performing mathematical computations and signal processing
  - Analysing and Visualising data
  - Modelling Physical systems and phenomena
  - Testing Engineering Designs

Industry Applications

- Aircraft / Defence
- Robotics
- Automotive
- Communications
- Biotech, Pharmaceutical and Medical

Octave Desktop

- Command Window: Where you type OCTAVE Commands following the prompt: >>
- Workspace window: It shows all the variable you have defined in your current session.
- Command History: Window displays all the OCTAVE commands you have used recently-even includes some past sessions.
- Current Folder: Window displays all the files in whatever folder you select to be cuurent.

Requirements for Successful Coding in Octave

- One must learn the exact rules for writing Octave statements
- Need to develop a logical plan of attack for solving particular problems
- Garbage in, garbage out.
- If you give Octave a garbage instruction, you will get a garbage result.
- With experience you will be able to design, develop and implement computational and graphical tools for complex engineering and science problems.

Arithmetic operators and its order

Precedence	Operator
1	Parenthesis (round brackets)
2	Power, left to right
3	Multiplication and Division, left to right
4	Addition and Subtraction, left to right

Arithmetic Operations

- Assign values to variables to do arithmetic operations

  >> a= 3 >> b= 10 >> c= (a+b)/a etc

- Octave has all of the usual mathematical functions that are on scientific calculator like sin , cos, tan, log etc

  Carry out simple trigonometric operations

- Octave has numerous general functions such as “date”, “calendar”, ” clc”, “clear”, “who”,”whos” etc

Variables and the workspace

- Variables are fundamental to coding. In a sense, the art of programming is
  - getting the right variables at the right time

  A variable name must comply with following rules
  - It may consist only of the letters a-z, the digits 0-9 and underscore (_).
  - It must start with a letter.
  - Octave is case sensitive
  - Give name to variable which is easily distinguishable.

Few Math functions used in octave

Function	MATLAB®	Function	MATLAB®
cosine	cos or cosd	square root	sqrt
sine	sin or sind	exponential	exp
tangent	tan or tand	logarithm (base 10)	log10
cotangent	cot or cotd	natural log (base e)	log
arc cosine	acos or acosd	round to nearest integer	round
arc sine	asin or asind	round down to integer	floor
arc tangent	atan or atand	round up to integer	ceil
arc cotangent	acot or acotd		

Note: cos(α) assumes α in radians; whereas, cosd(α) assumes α in degrees. acos(x) returns the angle in radians; whereas, acosd(x) returns the angle in degrees.

π radians = 180 degrees

Arrays: Vectors and Matrices

- A Vector is a special type of matrix, having only one row, or column. Vectors are also referred to as lists or arrays.

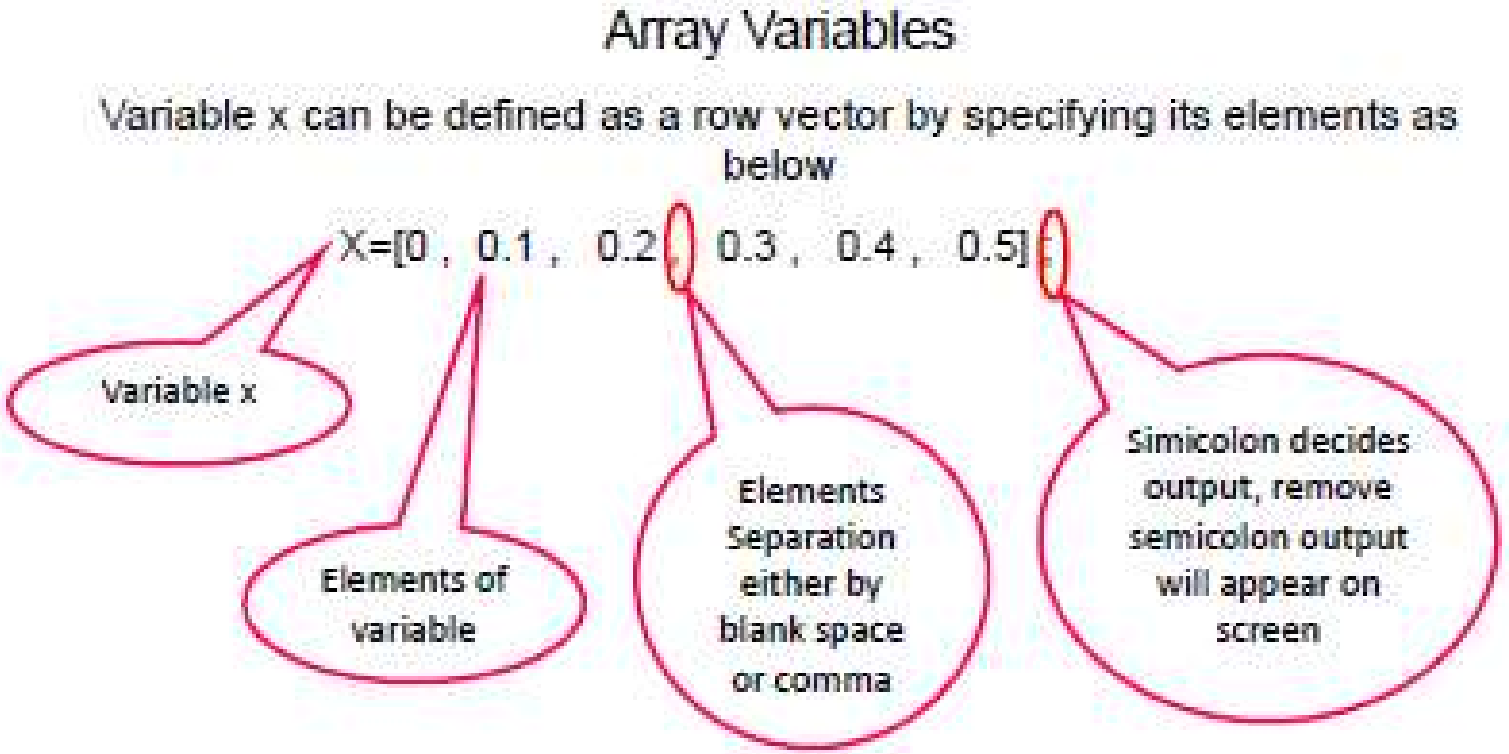
- A matrix is a rectangular object (e.g a table) consisting of rows and columns.

Entering Matrix in Octave

- 4 Ways of entering matrices in Octave
  - Enter an explicit list of elements
  - Load matrices from external data files
  - Generate matrices using built-in functions
  - Create matrices with your own functions in M-files
- Rules of entering matrices
  - Separate the elements of row with blanks or commas
  - Use a semicolon”;” to indicate the end of each row
  - Surround the entire list of elements with square brackets,[]
- To enter matrix, simply type:

  >>B =[1 2 3; 3 5 6;5 7 8]

Defining a Array Explicit list of Elements



Defining Array: Initializing vectors: linspace, rand etc functions

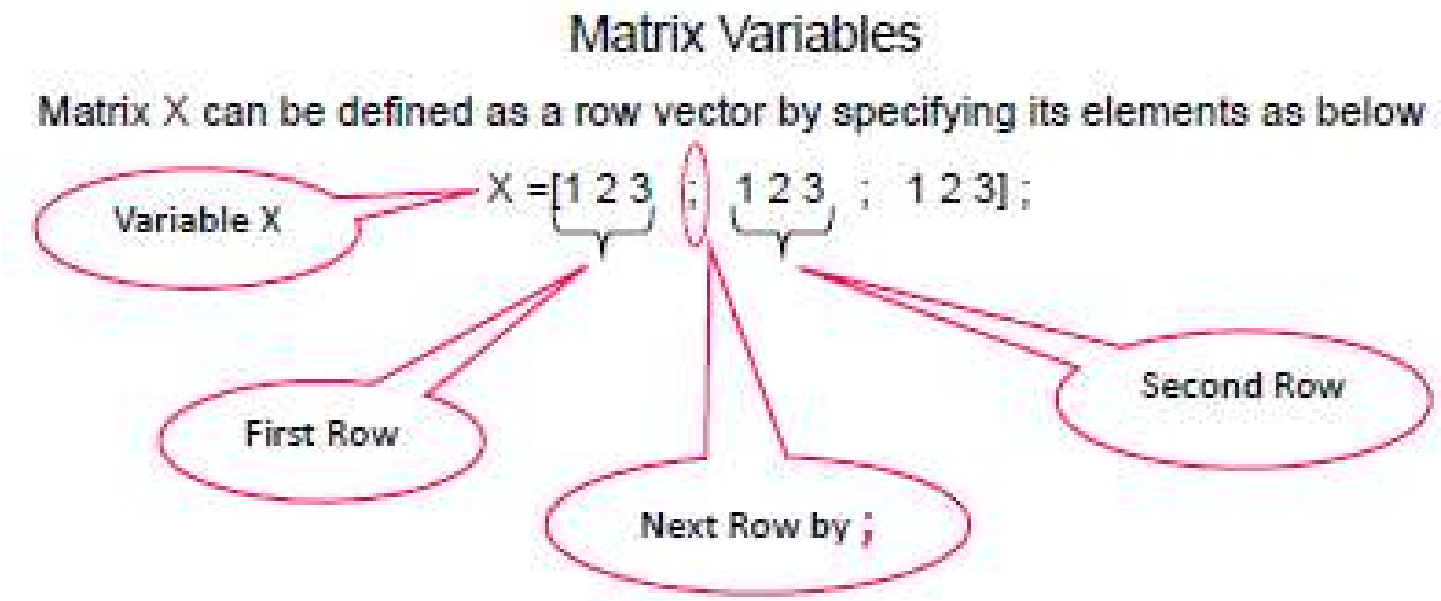
linspace Function : Linearly spaced vector

```
>> x = linspace( start_no, end_no, no_of_divisions)
>> x = linspace(0, pi/2, 10)
```

rand function: Uniformly distributed pseudorandom numbers

```
>> x = rand( row_nos, col_nos)
>> x = rand(1,7)
```

Defining Array: Explicit list of elements



Indexing of matrices

- Indexing using parentheses

A =

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

```
>> B=A(2,3)
B =  6
```

Indexing of matrices ….

A =

1 2 3
4 5 6
7 8 9

>> C =A(1,end)

C = 3

Indexing submatrices using vectors of row and column indices

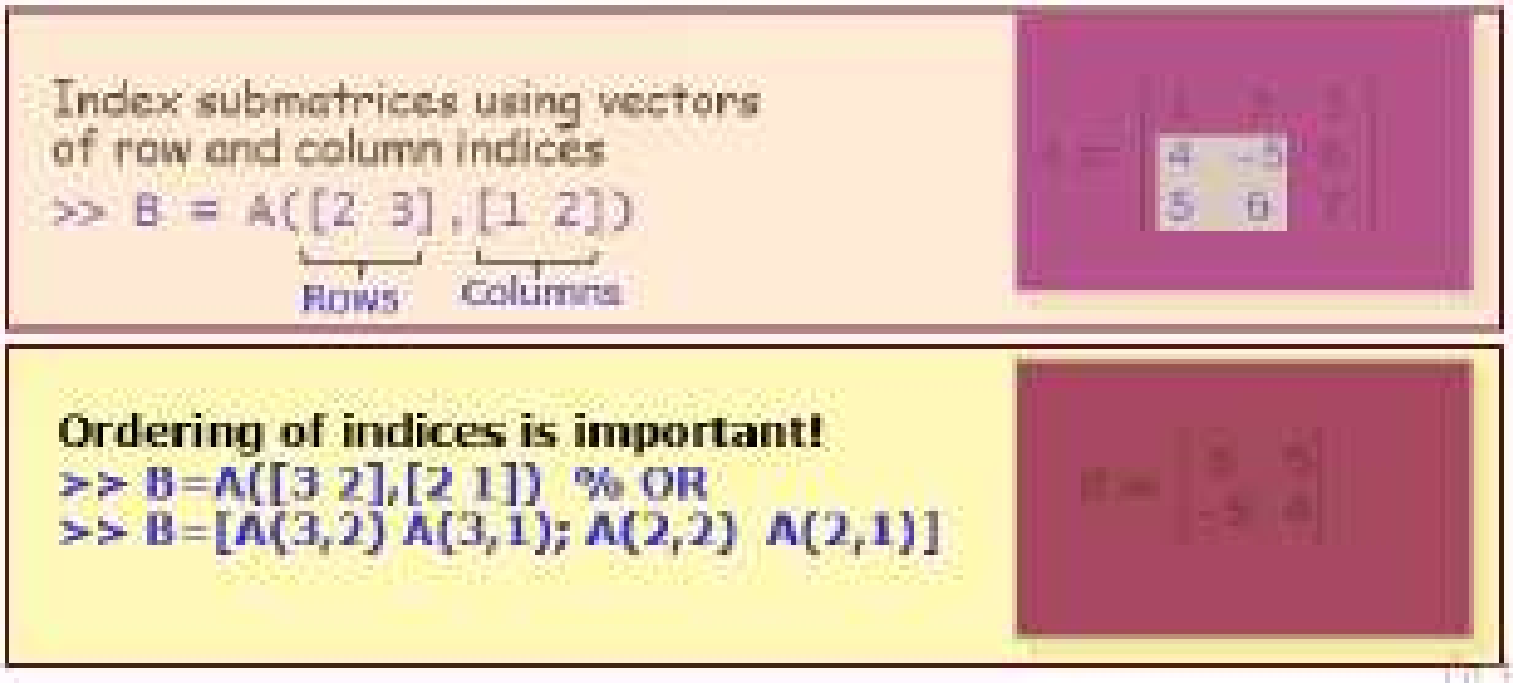
```
A =
   1   2   3
   4  -5   6
   5   6   7

>> B=A([2 3],[2 1])
B =

  -5   4
   6   5
```

Indexing submatrices using vectors of row and column indices

Ordering of indices is important



Defining matrix with inbuilt functions eye function

- eye function: identity matrix

X=eye(row_nos,col_nos)

>> x=eye(5,5)

x =

Diagonal Matrix

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Defining matrix with inbuilt functions zero function

- zero function: zeros array (M –by-N matrix of zeros)

X=zeros(rows_nos,col_nos)

>> X=zeros(5,5)

X =

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

Defining matrix with inbuilt functions ones function

- ones function: ones array(M-by-N matrix of ones)

X=ones(rows_nos,col_nos)

>> X=ones(5,5)

X =

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Matrix manipulations

- >> a=[1 2 3;4 5 6;9 3 6]
- a =

- 1 2 3
- 4 5 6
- 9 3 6

- >> b=a‘ (transpose)
- b =

- 1 4 9
- 2 5 3
- 3 6 6

- >> c=inv(a)
- c =

- -0.44444 0.11111 0.11111
- -1.11111 0.77778 -0.22222
- 1.22222 -0.55556 0.11111

Plotting
2-Dimensional plot

Command for basic 2-D plot is:

```
plot(x, y)
```

Where `x` **is one dimensional array**
`y` **is one dimensional array**

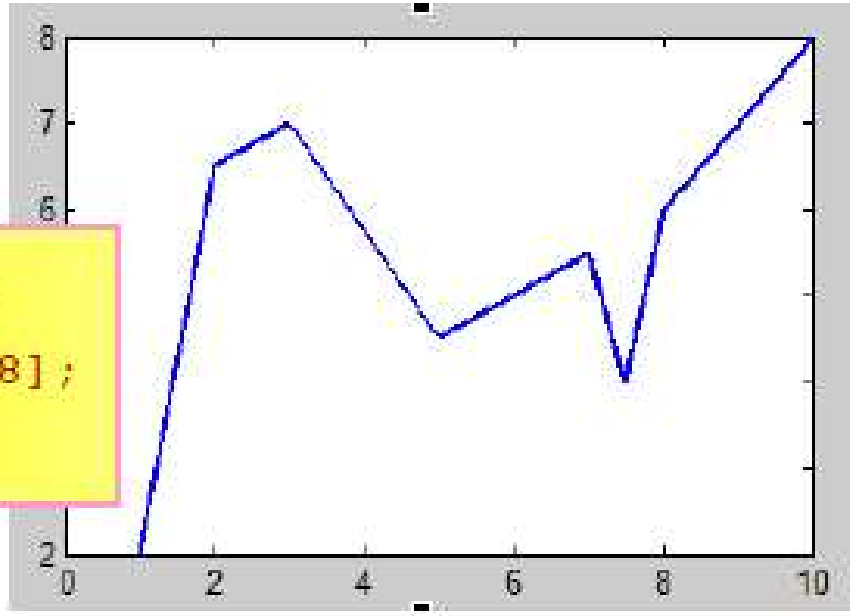
Both vectors `x` **and** `y` **must have the same lengths**

Plotting of given data

x	1	2	3	5	7	7.5	8	10
y	2	6.5	7	4.5	5.5	4	6	8

Create plot using commands below.

```
>> x=[1 2 3 5 7 7.5 8 10];
>> y=[2 6.5 7 4.5 5.5 4 6 8];
>> plot(x,y)
```



Line specifiers in the plot command

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

Line Style

Specifier	Line Style
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

Markers

Specifier	Marker Type
+	Plus sign
o	Circle
*	Asterisk
.	Point
x	Cross
'square' or s	Square
'diamond' or d	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle

Line Colors

Specifier	Color
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

Line specifies in the plot() command

- Line specifiers are typed as strings (in single quotes)
- Specifiers can be typed in any sequence
- The specifiers are optional, meaning that none, one, two, or all the three can be included in a command.

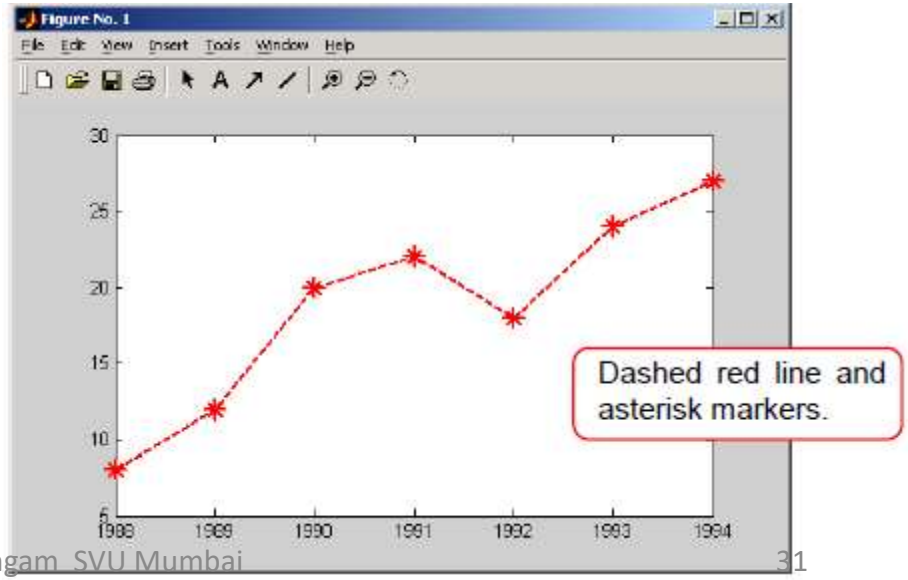
plot(x,y)	A solid blue line connects the points with no markers.
plot(x,y,'r')	A solid red line connects the points with no markers.
plot(x,y,'--y')	A yellow dashed line connects the points.
plot(x,y,'*')	The points are marked with * (But no line between the points.)
plot(x,y,'g:d')	A green dotted line connects the points which are marked with diamond markers.

Plot of given data using specifiers in the plot() command

Year	1988	1989	1990	1991	1992	1993	1994
Sales	127	130	136	145	158	178	211

```
>> year = [1988:1:1994];
>> sales = [127, 130, 136, 145, 158, 178, 211];
>> plot(year,sales,'--r*')
```

Line Specifiers: dashed red line and asterisk markers.



Creating a plot of a function

Consider: $y = 3.5^{(-0.5t)} \cos(6t) \quad for \; -2 \le t \le 4$

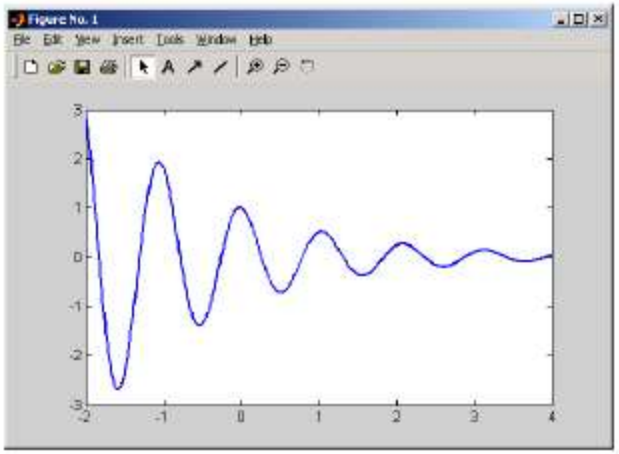
A script file for plotting the function is:

```
% A script file that creates a plot of
t = [-2 : 0.01 : 4];
y = 3.5.^(-0.5*t).*cos(6*t);
plot(t,y, '-o')
```

Creating a vector with spacing of 0.01

Calculating a value of y for each t.

Once the plot command is executed, the figure window opens with the following plot.



Plotting with linspace (Linnearly spaced Vectors)

```
>> t = linspace(-2, 4, 50) ;      % Create vector t, with 50
                                  % equally spaced entries between
                                  % -2 to 4

>> y = 3.5*exp(-0.5*t).*cos(6*t); % create a vector y comprising
                                  % of same number of entries

>> plot(t, y, '+')                % plot y as a function of t, using +
                                  % symbols
```

Adding Text to Figures

- Basic axis labels and title can be added via convenient functions:

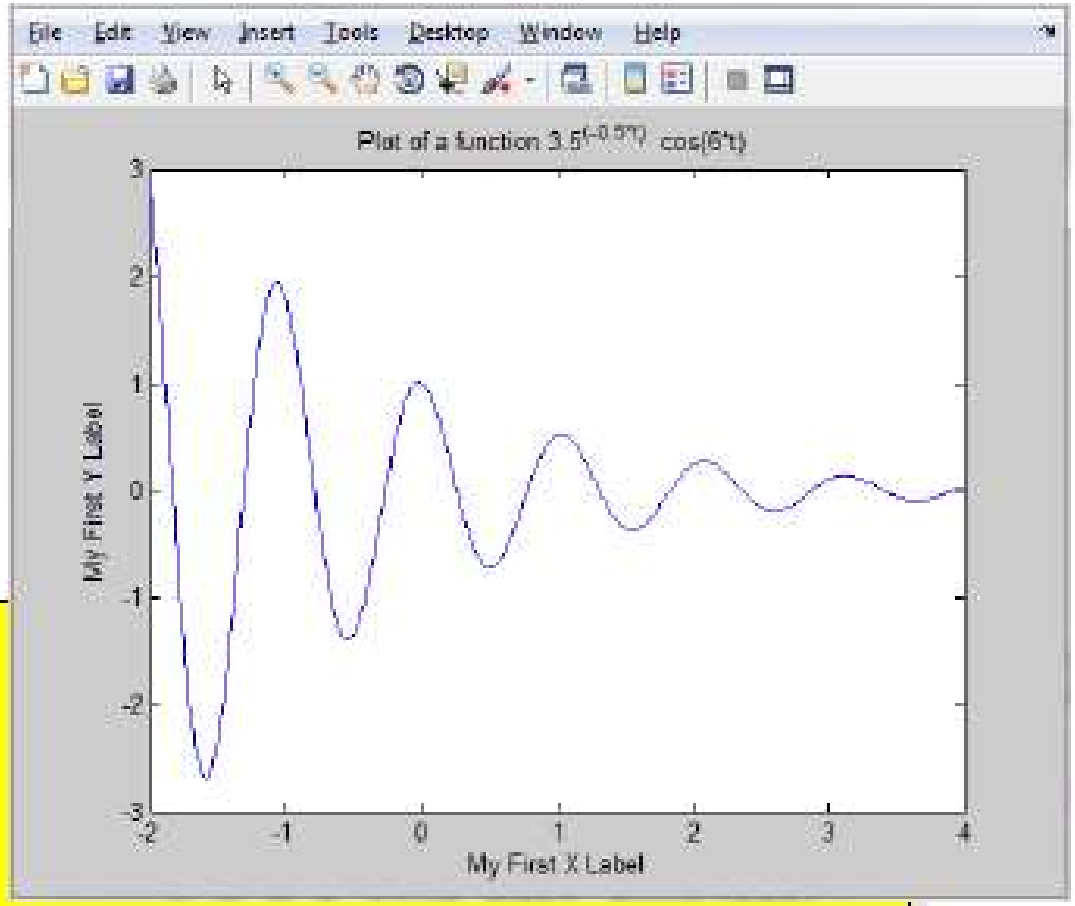
```
>> xlabel('x-axis label text')
>> ylabel('y-axis label text')
>> title('title text')
```

- Legends for line or symbol types are added via the **legend** function:

```
>> legend('Curve 1 caption', ' Curve 2 caption',…)
>> legend('Curve 1 caption', ' Curve 2 caption',2)
```

Use of Xlabel, ylabel and Title

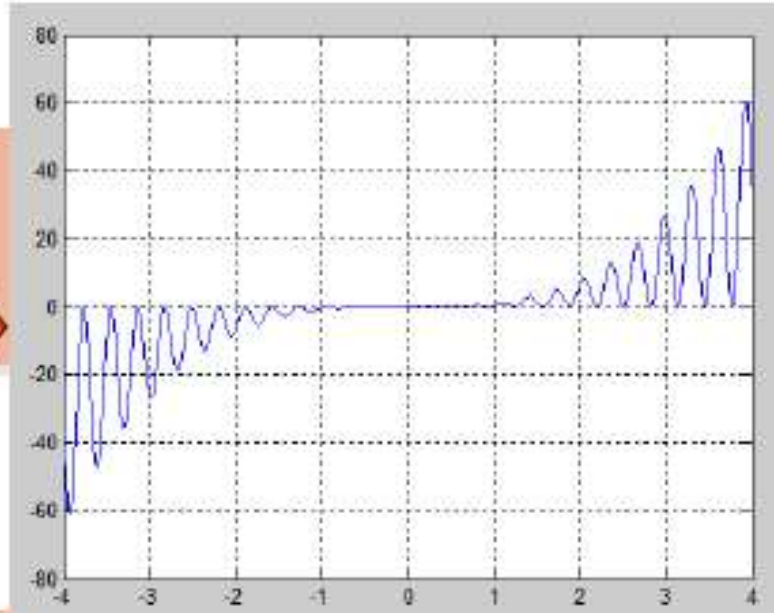
```
t = [-2 : 0.01 : 4];
y = 3.5.^(-0.5*t).*cos(6*t);
plot(t, y)
xlabel('My First X Label')
ylabel('My First Y Label')
title('Plot of a function 3.5^{(-0.5*t)}  cos(6*t) ')
```



Use of axis command

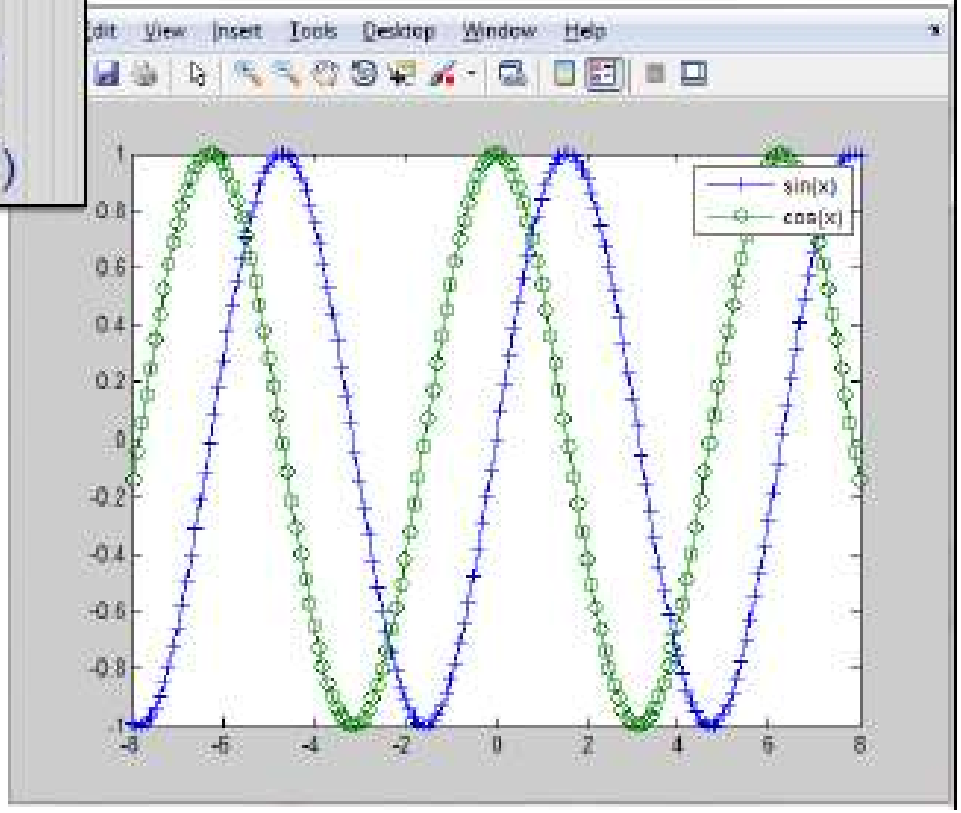
- Axis([xmin xmax ymin ymax])

```
x = [-4 : 0.001 : 4];
y = x.^3 .* (sin(10*x)).^2
plot(x,y), grid
```



Plotting multiple graphs on the same axis

```
>> x = [-8 : 0.1 : 8];
>> y = sin(x);
>> z = cos(x);
>> plot(x,y, '+-', x, z, 'o-')
>> legend('sin(x)', 'cos(x)')
```



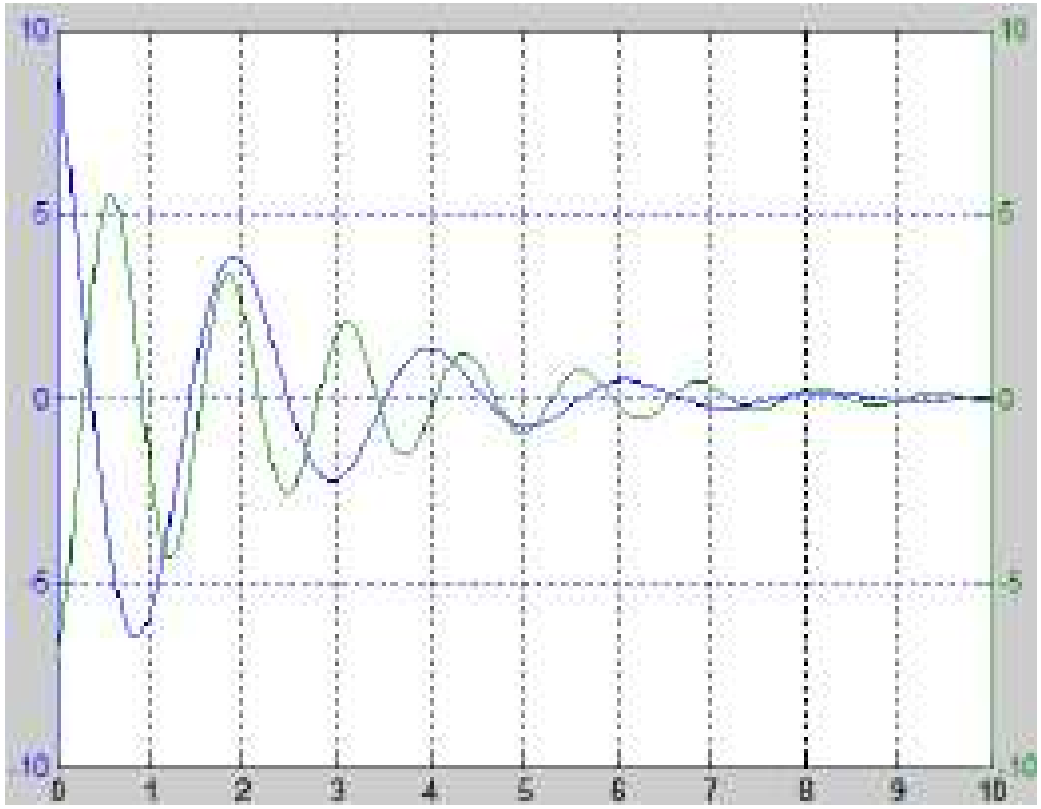
Using plotyy commond

- Examples of vibrations in machines and structures

$$x = 10\,e^{(-0.5\,t)}\,\sin(3\,t+2)$$

$$y = 70\,e^{(-0.4\,t)}\,\cos(5t-3)$$

```
t = 0:0.01:10;
x = 10*exp(-0.5*t) .* sin(3*t+2);
y = 70*exp(-0.4*t) .*  cos(5*t-3);
plotyy(t,x, t,y),  grid
```



Use of hold on and hold off command

`hold on`	Holds the current plot and so that subsequent plots are added to the existing plot.
`hold off`	Returns to the default mode whereby plot commands erase the previous plots and reset all axis properties before drawing new plots.
`ishold`	Logical command that returns 1 (True) if hold is on and 0 (False) if hold is off

This method is useful when all the information (vectors) used for the plotting is not available at the same time.

Plotting multiple graphs in the same plot

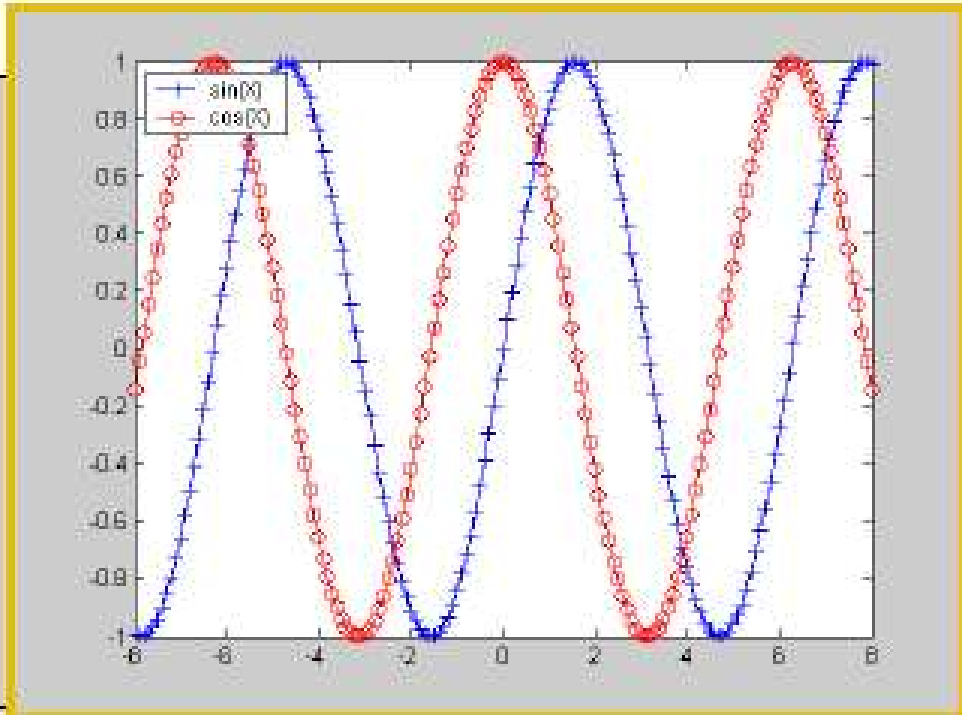
Plotting two (or more) graphs in one plot:

1. Using the **`plot`** command.
2. Using the **`hold on`**, **`hold off`** commands.

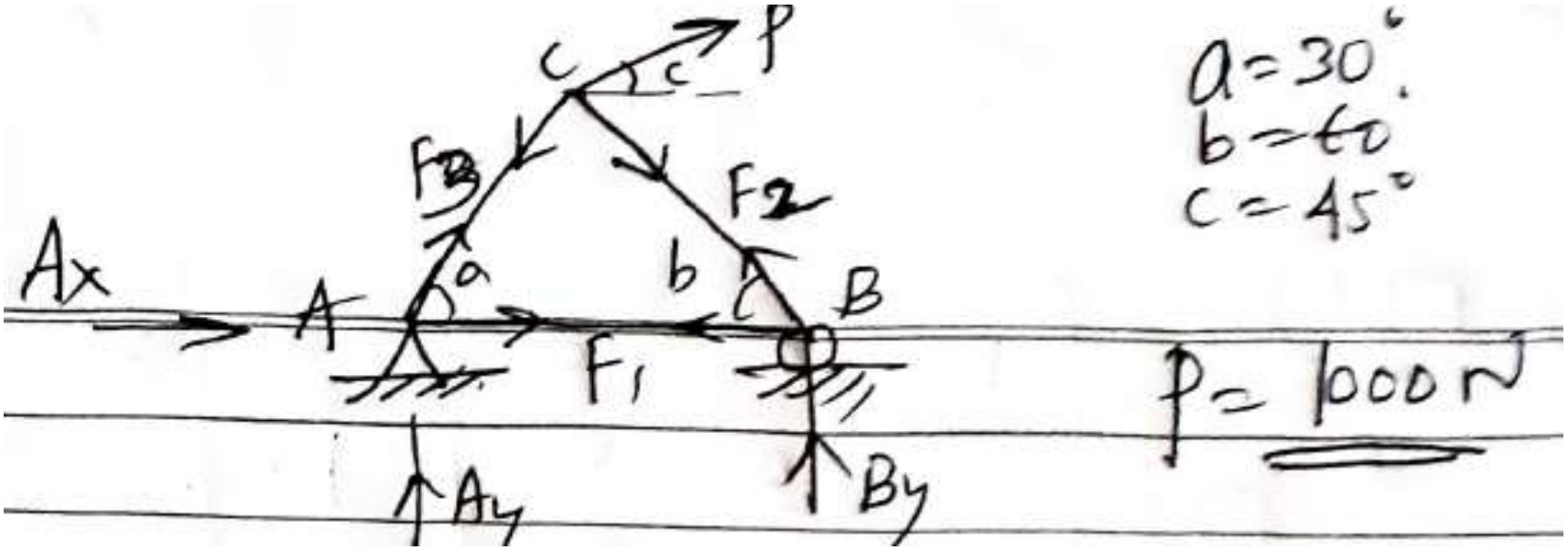
```
x = -8:0.1:8;
y = sin(x);
z = cos(x);

plot(x,y,'+-'),
hold on,
plot(x,z,'o-r'),
hold off

legend('sin(X)','cos(X)', 2)
```



Analysis of Truss



At A

$\Sigma F_x = 0 \quad F_1 + F_3\cos\alpha + A_x = 0$

$(1)F_1 + (0)F_2 + (\cos\alpha)F_3 + (1)A_x + (0)A_y + (0)B_y = 0$ — (1)

$\Sigma F_y = 0 \quad F_3\sin(\alpha) + A_y = 0$

$(0)F_1 + (0)F_2 + (\sin\alpha)F_3 + (0)A_x + (1)A_y + (0)B_y = 0$ — (2)

At B $\quad \Sigma F_x = 0 \quad -F_1 - F_2\cos b = 0$

$(-1)F_1 + (-\cos b)F_2 + (0)F_3 + (0)A_x + 0(A_y) + 0(B_y) = 0$ — (3)

$\Sigma F_y = 0 \quad F_2\sin b + B_y = 0$

$(0)F_1 + (\sin b)F_2 + (0)F_3 + (0)A_x + (0)A_y + (1)B_y = 0$ — (4)

At C $\quad \Sigma F_x = 0 \quad F_2\cos b - F_3\cos\alpha = -p\cos\epsilon$

$(0)F_1 + (\cos b)F_2 + (-\cos\alpha)F_3 + (0)(A_x) + (0)A_y + (0)B_y$

$\Sigma F_y = 0 \quad -F_2\sin b - F_3\sin\alpha = -p\sin\epsilon \quad = -p\cos\epsilon$ — (5)

$(0)F_1 + (-\sin b)F_2 + (-\sin\alpha)F_3 + (0)A_x + (0)A_y + (0)B_y = -p\sin\epsilon$ — (6)

Writing all (6) Equations in matrix form

$$\begin{bmatrix} 1 & 0 & \cos\alpha & 1 & 0 & 0 \\ 0 & 0 & \sin\alpha & 0 & 1 & 0 \\ -1 & -\cos\beta & 0 & 0 & 0 & 0 \\ 0 & \sin\beta & 0 & 0 & 0 & 1 \\ 0 & \cos\beta & -\cos\alpha & 0 & 0 & 0 \\ 0 & -\sin\beta & -\sin\alpha & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ A_X \\ A_Y \\ B_Y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -P\cos\alpha \\ -P\sin\alpha \end{bmatrix}$$

$$[A] \qquad [X] = [B]$$

$$X = [A]^{-1} B$$

- >> a=30*pi/180; % angle a= 30 degree
- >> b=60*pi/180; % angle b =60 degree
- >> c=45*pi/180; % angle c = 45 degree
- >> p=200; % force P at joint 3 =200 Newton
- >> A=[1,0,cos(a),1,0,0;0,0,sin(a),0,1,0;-1,-cos(b),0,0,0,0;0,sin(b),0,0,0,1;0,cos(b),-cos(a),0,0,0;0,-sin(b),-sin(a),0,0,0]
- A =
-
- 1.00000 0.00000 0.86603 1.00000 0.00000 0.00000
- 0.00000 0.00000 0.50000 0.00000 1.00000 0.00000
- -1.00000 -0.50000 0.00000 0.00000 0.00000 0.00000
- 0.00000 0.86603 0.00000 0.00000 0.00000 1.00000
- 0.00000 0.50000 -0.86603 0.00000 0.00000 0.00000
- 0.00000 -0.86603 -0.50000 0.00000 0.00000 0.00000

```
>> B=[0,0,0,0,-p*cos(c),-p*sin(c)]
B =

    0.00000     0.00000     0.00000     0.00000  -141.42136  -141.42136
>> C=B'
C =

    0.00000
    0.00000
    0.00000
    0.00000
  -141.42136
-141.42136
```

- >> X=inv(A)*C
- X =
-
- -25.882
- 51.764
- 193.185
- -141.421
- -96.593
- -44.829
- Answers;
- F1
- F2
- F3
- Ax
- Ay
- By

Projectile problem

- Angle of projections
- a1=30 deg,a2=45 deg,a3=60 deg

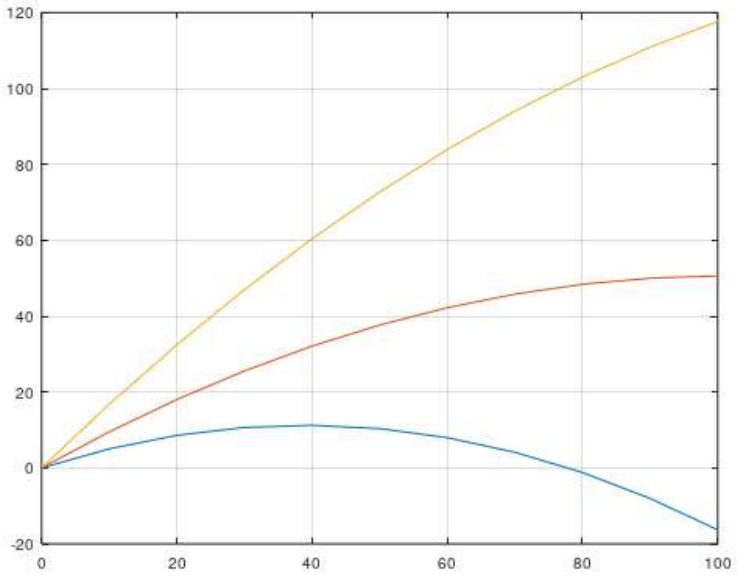
Velocity of projections

u1=30 m/s, u2 =45 m/s, u3= 60 m/s

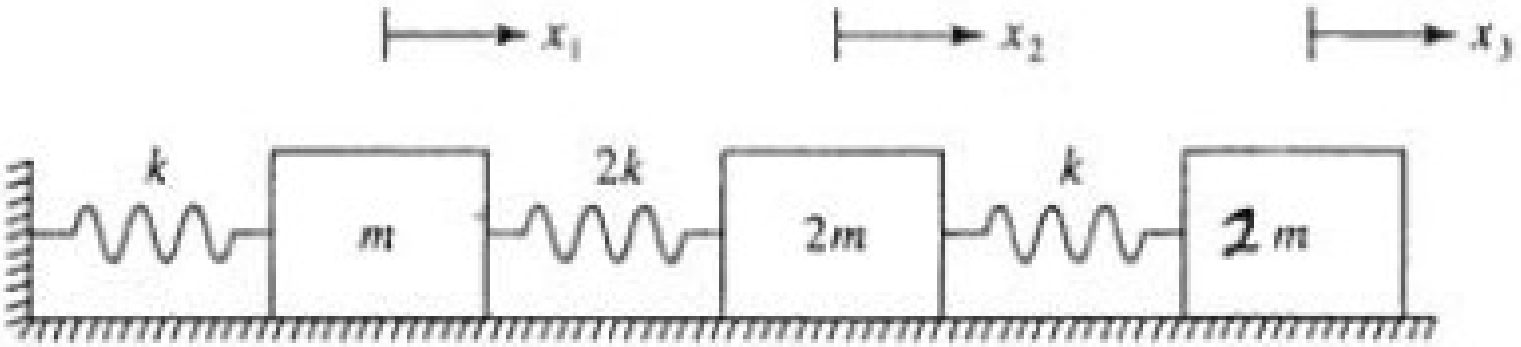
g=acceleration due to gravity = 10 m/s2.

$$y = x\,tan\,\theta - \frac{gx^2}{2v^2\,cos^2\,\theta}$$

- >> a1=30;
- >> a2=45;
- >> a3=60;
- >> u1=30; % in m/s
- >> u2=45; % in m/s
- >> u3=60; % in m/s
- >> x=0:10:100;
- >> y1=x*tand(a1)-10*x.^2/(2*u1.^2*(cosd(a1).^2));
- >> y2=x*tand(a2)-10*x.^2/(2*u2.^2*(cosd(a2).^2));
- >> y3=x*tand(a3)-10*x.^2/(2*u3.^2*(cosd(a3).^2));
- >> plot(x,y1,x,y2,x,y3), grid on



MDOF problems



$$\begin{bmatrix} m & 0 & 0 \\ 0 & 2m & 0 \\ 0 & 0 & 2m \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \end{bmatrix} + \begin{bmatrix} 3k & -2k & 0 \\ -2k & 3k & -k \\ 0 & -k & k \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

M=10kg, k=100 N/m

```
>> m=[10 0 0;0 20 0;0 0 20]
m =

   10    0    0
    0   20    0
    0    0   20
>> k=[300 -200 0;-200 300 -100;0 -100 100]
k =

   300  -200     0
  -200   300  -100
     0  -100   100
```

```
>> D=inv(m)*k   % Dynamic matrix
D =

   30  -20    0
  -10   15   -5
    0   -5    5

>> [v,d]=eig(D)
v =

   -0.915255  -0.577350   0.383307
    0.398515  -0.577350   0.550204
   -0.059112   0.577350   0.741857

d =

Diagonal Matrix

   38.7083         0         0
         0   10.0000         0
         0         0    1.2917
```

Plot of sine function

- >> theta =0:0.01:2*pi;
- >> f=sin(theta); % sin function works on an array
- >> plot(theta,f,'b') % 2D plot
- >> xlabel('theta');
- >> xlabel('theta'); % lebel for X axis
- >> ylabel('sin(theta)'); % label for Y axis
- >> title('plot of sine function')
- >> grid on % adding grid line

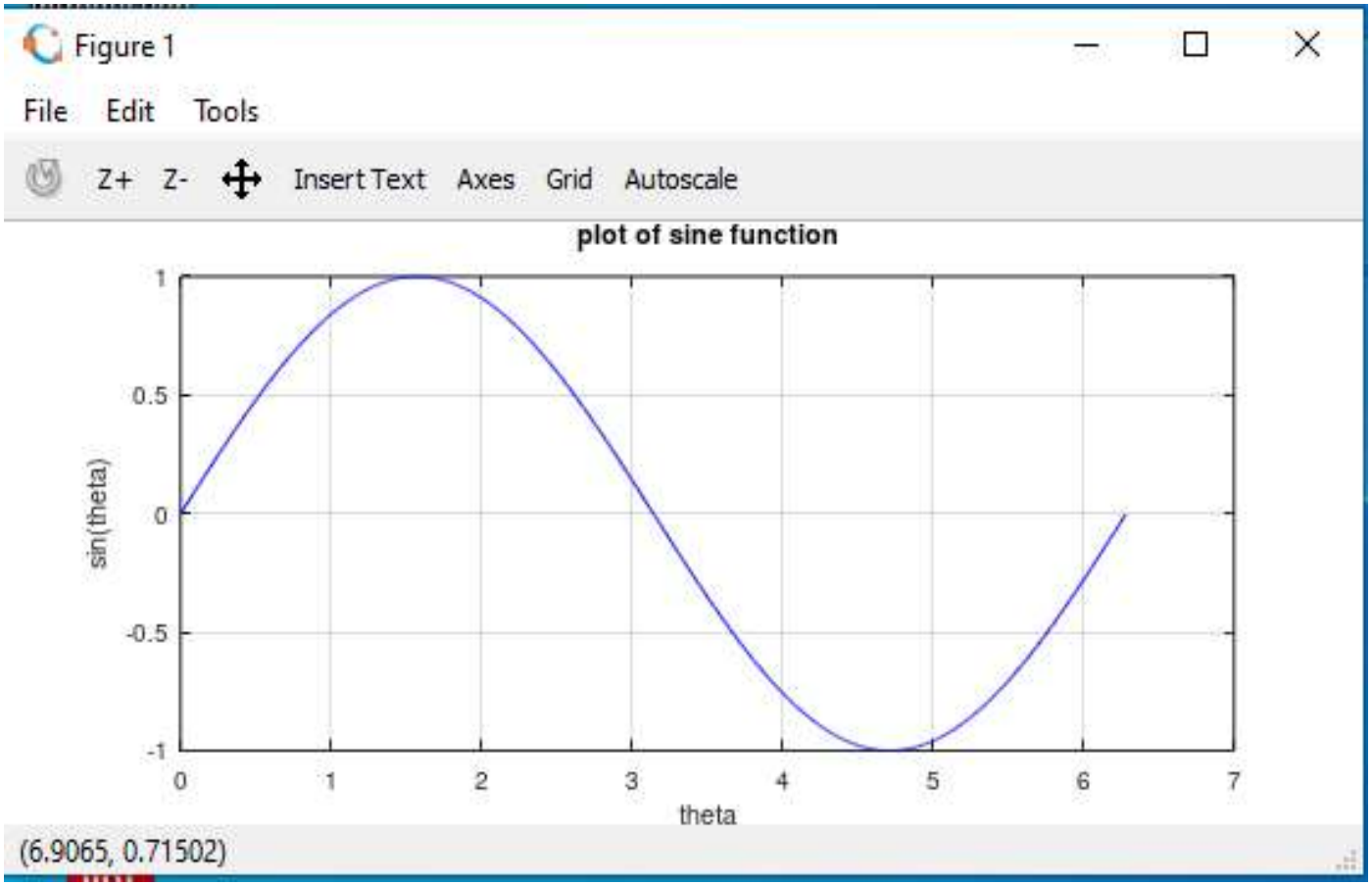


Figure 1
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
plot of sine function
sin(theta)
theta
1
0.5
0
-0.5
-1
0 1 2 3 4 5 6 7
(6.9065, 0.71502)

Another 2 D plot

- >> x=0:0.1:15;
- >> w=15; % omega is 10 rad/sec
- >> y=exp(-0.7*x).*sin(w*x);
- >> plot(x,y), grid on

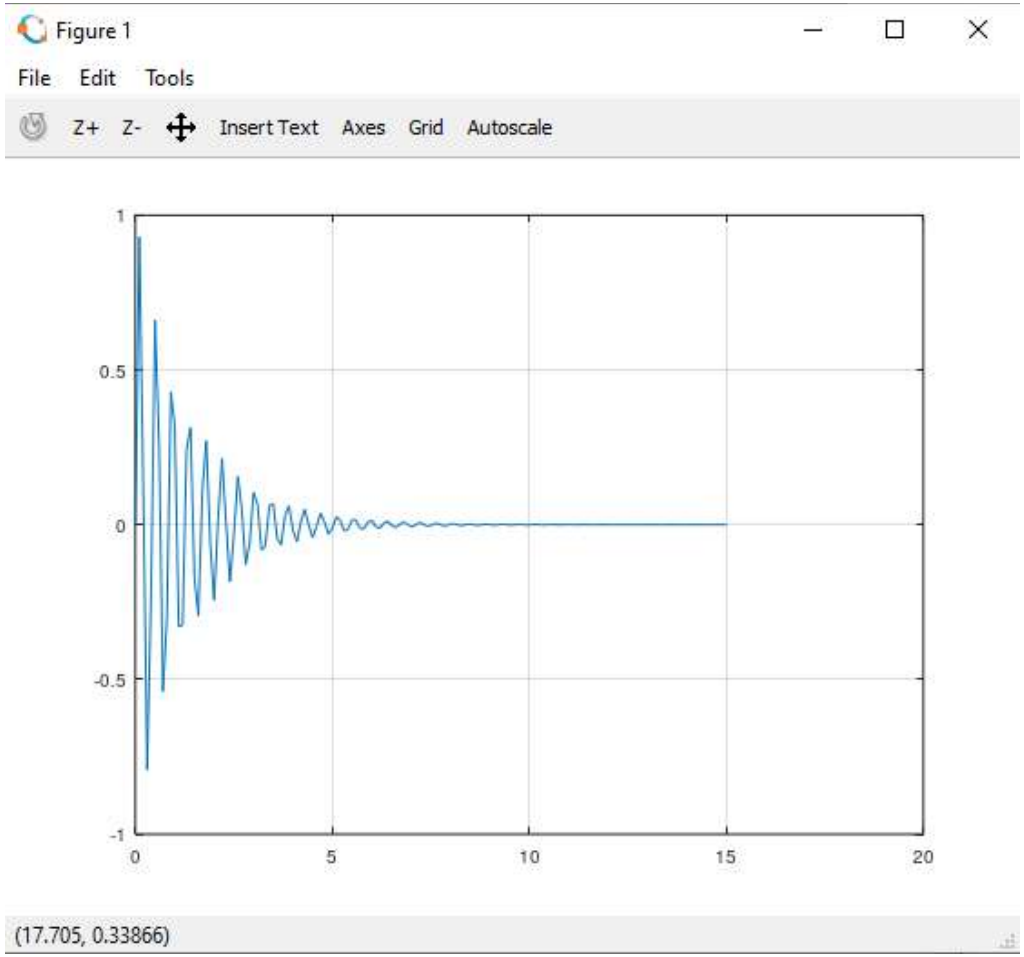


Figure 1
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
1
0.5
0
-0.5
-1
0
5
10
15
20
(17.705, 0.33866)

3 D plot

- >> t=linespace(0,2,100);
- error: 'linespace' undefined near line 1 column 3
- >> t=linspace(0,2,100);
- >> x=t;
- >> y=t.^2;
- >> z=t.^3;
- >> plot3(x,y,z), grid on

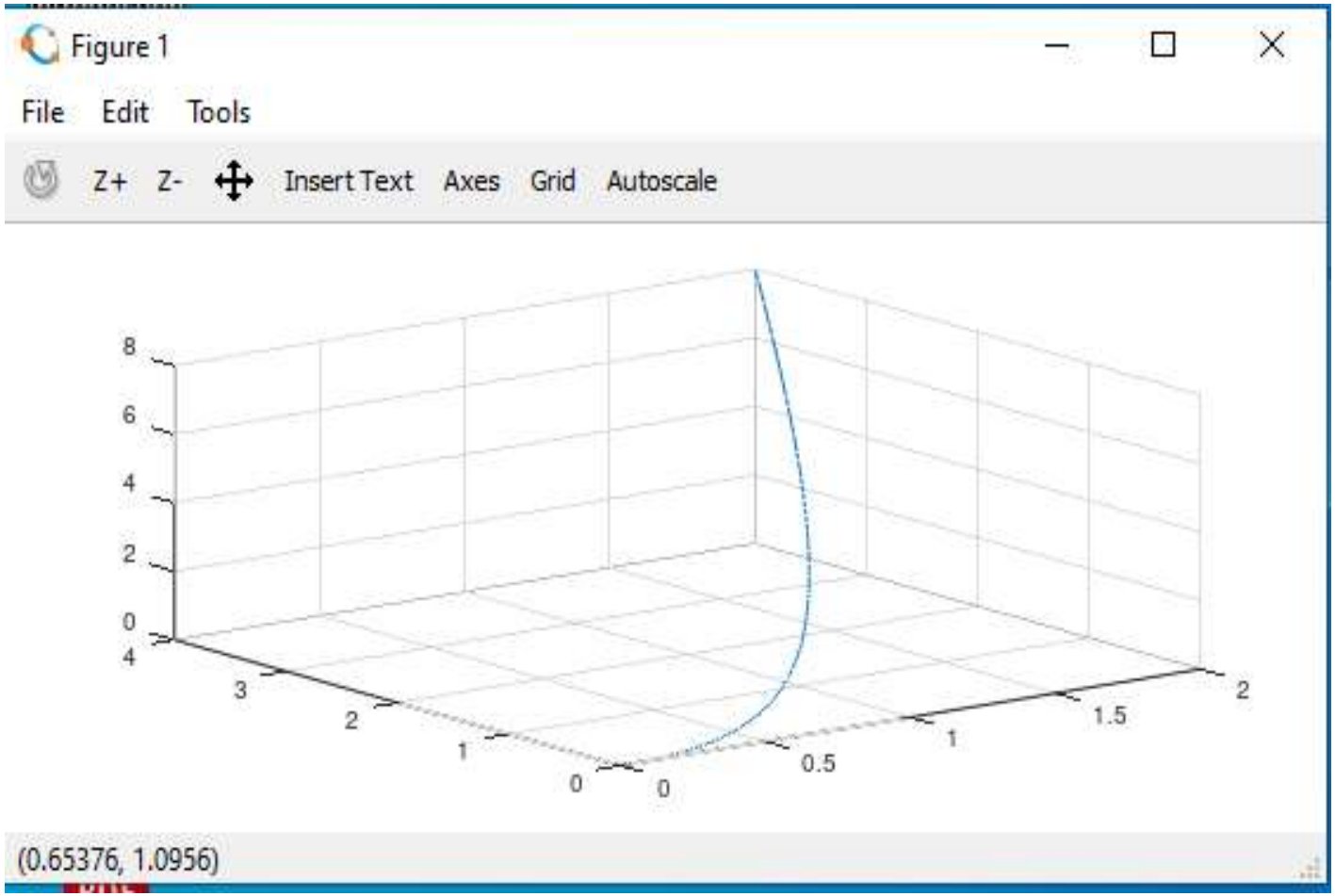


Figure 1
File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
8
6
4
2
0
4
3
2
1
0
0
0.5
1
1.5
2
(0.65376, 1.0956)