# Computer Aided Design (CAD)

## Lecture 4

- Operations & Plotting Vectors.
- Arrays.

Dr.Eng. Basem ElHalawany

# Schedule (Draft)

| Topics | Estimated Duration (# Lectures) |
|---|---|
| Introduction | 1 |
| Introduction to Matlab Environment | 1 |
| Matlab Programing (m-files) | 5 **(2/5)** |
| Modeling using Matlab Simulink Tool | 4 |
| Communication Systems Simulation (Applications) | 3 |
| Midterm | 8th Week |
| Introduction to FPGA + Review on Digital Logic/Circuits | 2 |
| VHDL Modeling Language | 4 |
| VHDL Application | 2 |
| Introduction to OPNET Network Simulator | 3 |
| Course Closeout / Feedback/ project (s) Delivery | 1 |


## The Lecture is based on :

A. Matlab by Example: Programming Basics, Munther Gdeisat

# Accessing Elements in Vectors (Cont.)

## 3.3.4 Accessing Elements in a Vector Using the Relational and Logical Operators

- Matlab has an interesting way of using the relational and logical operations to access elements in vectors.

```
x = [0,4,7,0,-1,2];
y = [1,3,8,0,-4,6];
x > 3
```

Matlab responds with

```
ans =
    0   1   1   0   0   0
```

```
>> r = y(x > 3)
```

Matlab responds with

```
r =
    3   8
```

i.e. outputs the elements of the vector y that are in the same elemental positions as those elements of the vector x which have a value that is greater than 3;

# Lesson 3.4 Arithmetical Operations on Vectors

## 3.4.1 Vector Addition and Subtraction

```
x = [1,2,3];
y = [4,5,6];
```

- The addition and subtraction of two vectors is performed on an element-by-element basis.
- The dimensions of the two vectors must be equal. If this is not the case, then Matlab will give you an error message.

```
>> z = x + y

   z =
         5   7   9
```

```
>> s = x - y

   s =
        -3   -3   -3
```

### 3.4.1.3 *Adding a Number to a Vector*

```
x = [1,2,3];
s = x + 10


s =
    11   12   13
```

add 10 to all of the elements in the vector x

## 3.4.2 Matrix and Element-by-Element Arithmetical Operations for Vectors

- Arithmetical operations that are performed on vectors using Matlab can be divided into **element-by-element** and **matrix operations**.

## 3.4.3 Vector Multiplication

## 3.4.4 Vector Division

### 3.4.3.1 *Element-by-Element Multiplication for Vectors*

- To multiply the two vectors x and y on an element-by-element basis type:

```
>> z = x.*y
```

```
x = [1,2,3];
y = [4,5,6];
```

```
z =
    4   10   18
```

## 3.4.3.2 Matrix Multiplication for Vectors

>> z = x*y

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 & \dots & x_n \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_n \end{bmatrix} = \boxed{z = x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4 + \dots + x_ny_n}$$

- The number of columns in x must be similar to the number of rows in y
- The multiplication process here produces a single value scalar number

>> R = y*x

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_n \end{bmatrix} \times \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & \dots & x_n \end{bmatrix} =$$

| | | | | | |
|---|---|---|---|---|---|
| $y_1x_1$ | $y_1x_2$ | $y_1x_3$ | $y_1x_4$ | ..... | $y_1x_n$ |
| $y_2x_1$ | $y_2x_2$ | $y_2x_3$ | $y_2x_4$ | ..... | $y_2x_n$ |
| $y_3x_1$ | $y_3x_2$ | $y_3x_3$ | $y_3x_4$ | ..... | $y_3x_n$ |
| $y_4x_1$ | $y_4x_2$ | $y_4x_3$ | $y_4x_4$ | ..... | $y_4x_n$ |
| ..... | ..... | ..... | ..... | ..... | ..... |
| $y_nx_1$ | $y_nx_2$ | $y_nx_3$ | $y_nx_4$ | ..... | $y_nx_n$ |

- The number of columns in y must be equal to the number of rows in x.
- The multiplication process is very different and this time produces a square matrix

It is important to realize that in matrix multiplication $\mathbf{xy} \neq \mathbf{yx}$.

## 3.4.4 Vector Division

### *3.4.4.1 Element-by-Element Division for Vectors*

```
>> z = x./y
```

Remember that the two vectors x and y here must have the same dimensions.

### *3.4.4.2 Matrix Division for Vectors*

Matrix division for vectors **will not be discussed** here because **it does not have any mathematical meaning.**

# Lesson 3.5 Plotting Vectors

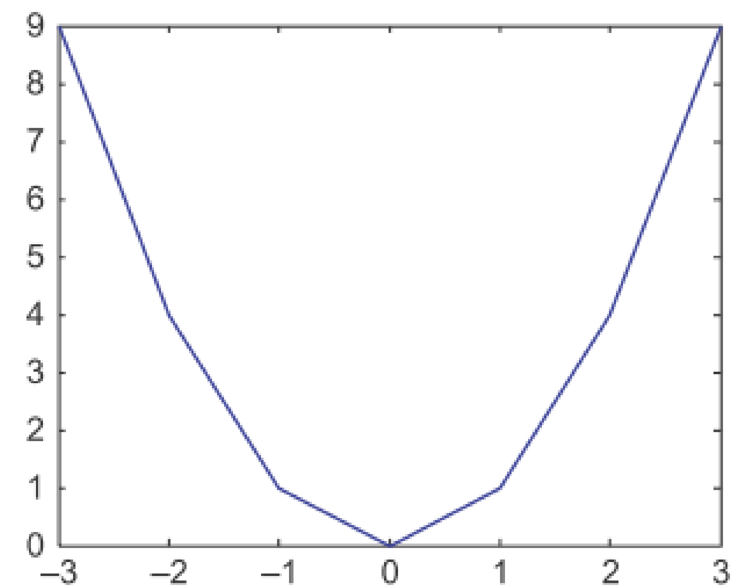
- Matlab is an excellent software package for performing **2D graphics.**
- For example, let us plot the function y = x^2, where x is in the range [ 2 3, 3].

```
>> x = -3:1:3
>> y = x.^2
>> plot(x,y)
```

```
x =
    -3   -2   -1   0   1   2   3

y =
    9   4   1   0   1   4   9
```

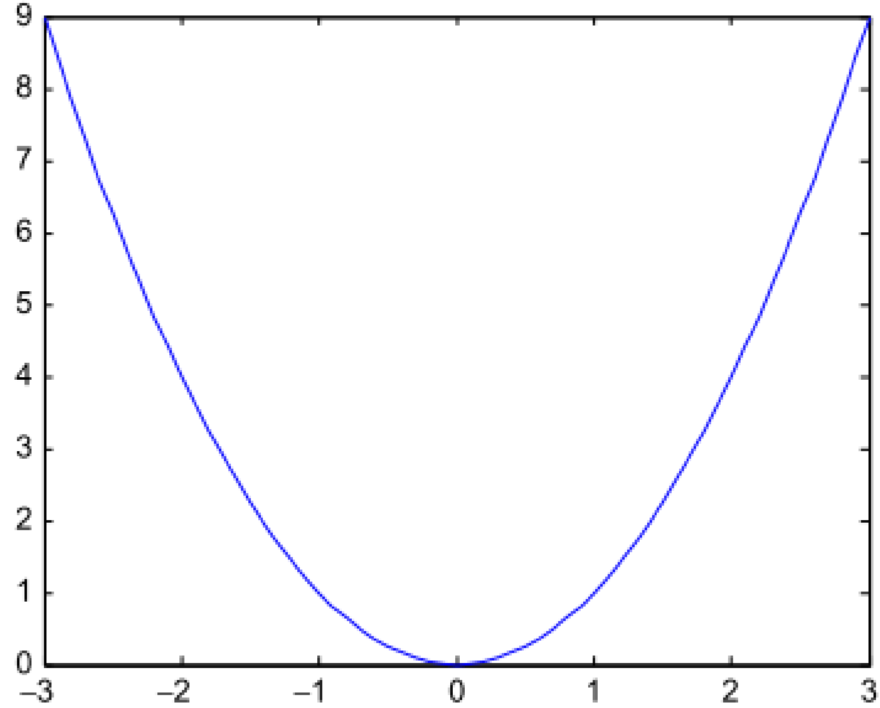
- The plot function draws seven points, which are the number of elements in the vector x, and then connects the points together using straight lines.
- This is the reason why the function appears in the form of connected line segments



## 3.5.2 Increasing the Resolution of a Plot

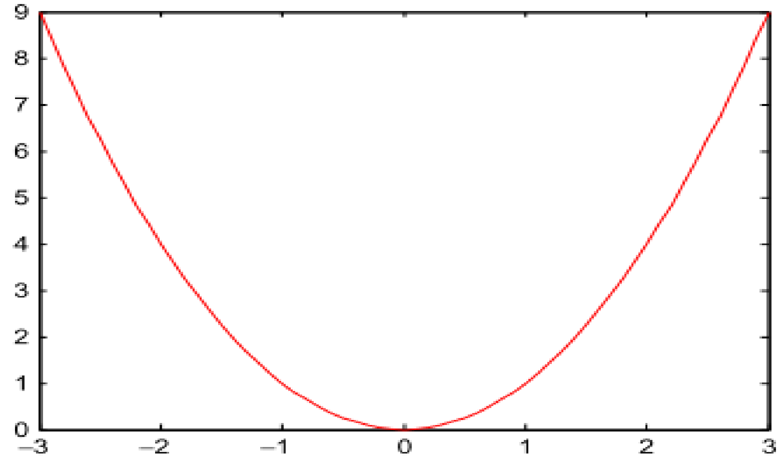
- To improve the resolution of the plot, you need to increase the number of points for the x vector

```
x = -3:0.1:3;
y = x.^2;
plot(x,y)
```



### 3.5.3 Changing the Color of a Plot

```
>>plot(x,y,'r')
```



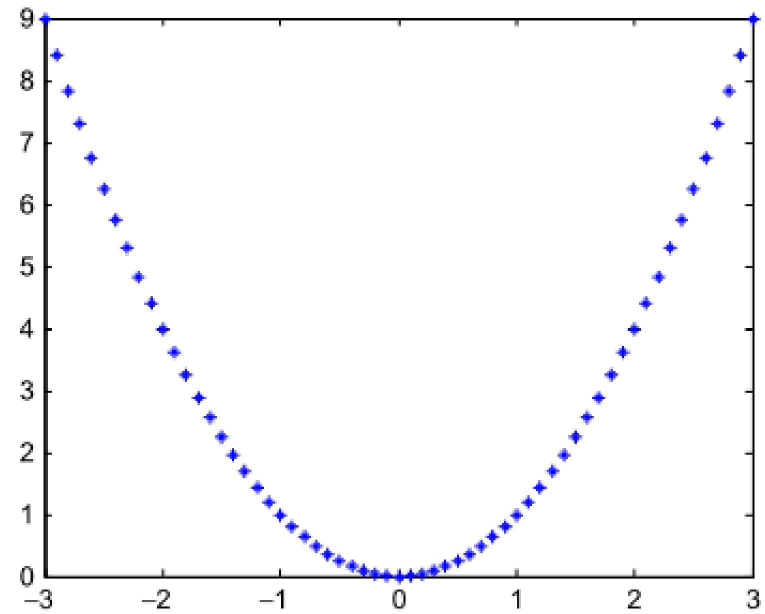

Matlab support the colors:

- red “r”,
- green “g”,
- blue “b”,
- cyan “c”,
- magenta “m”,
- yellow “y”,
- white “w”
- black “b”.

# 3.5.4 Draw a Function as Points

- In the form of unconnected points only, as shown below.
- Here we represent each point as an asterisk “*”.

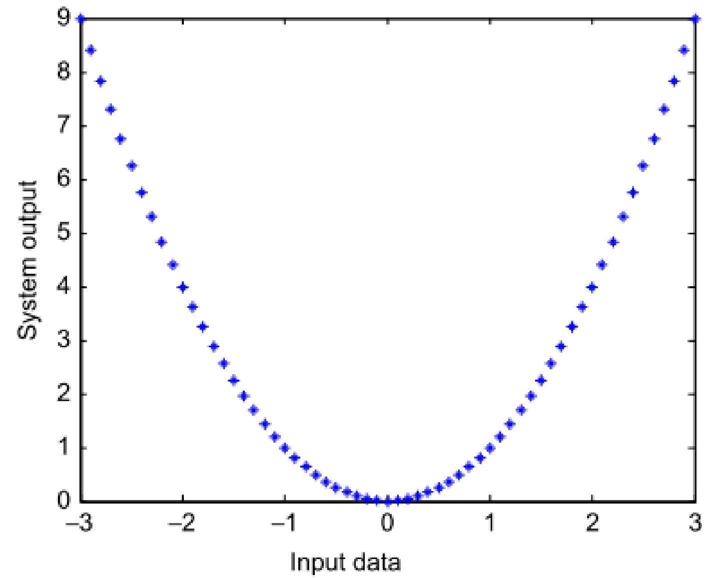
```
>>plot(x,y,'*')
```




- Matlab supports a long list of different symbols that can be used to represent points in a curve.
- For example, “1 ”, “o” or “x”.

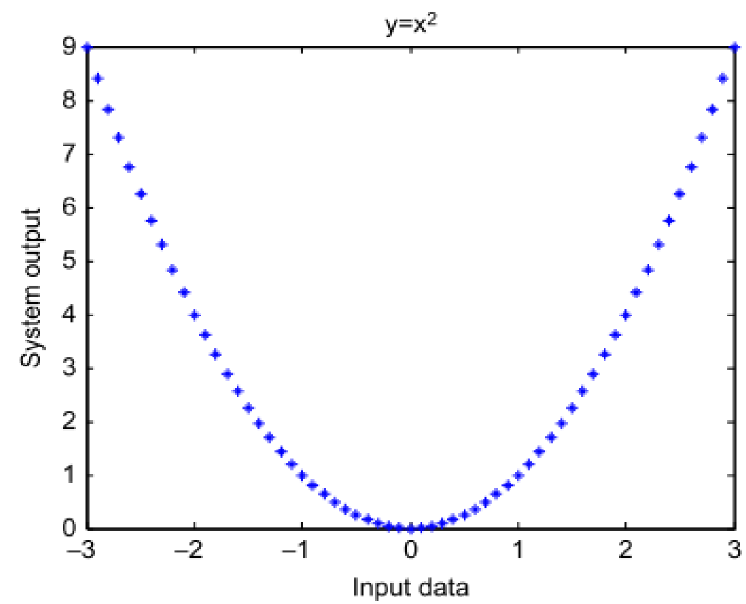
## 3.5.5 Labeling the x and y Axes

```
xlabel('Input data')
ylabel('System output')
```



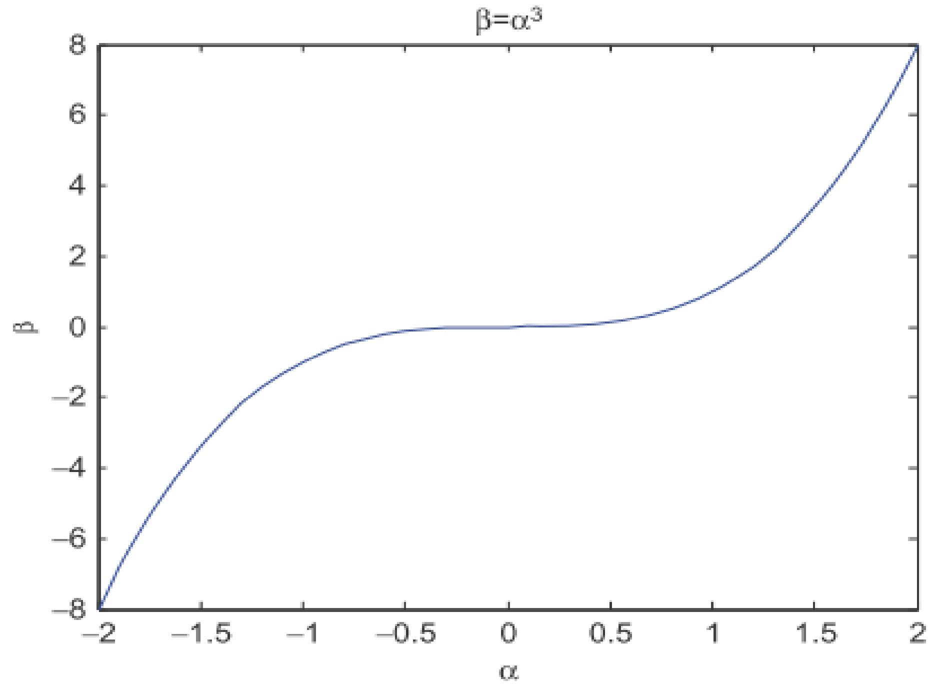

## 3.5.6 Adding a Title to a Figure

```
>>title('y=x^2')
```



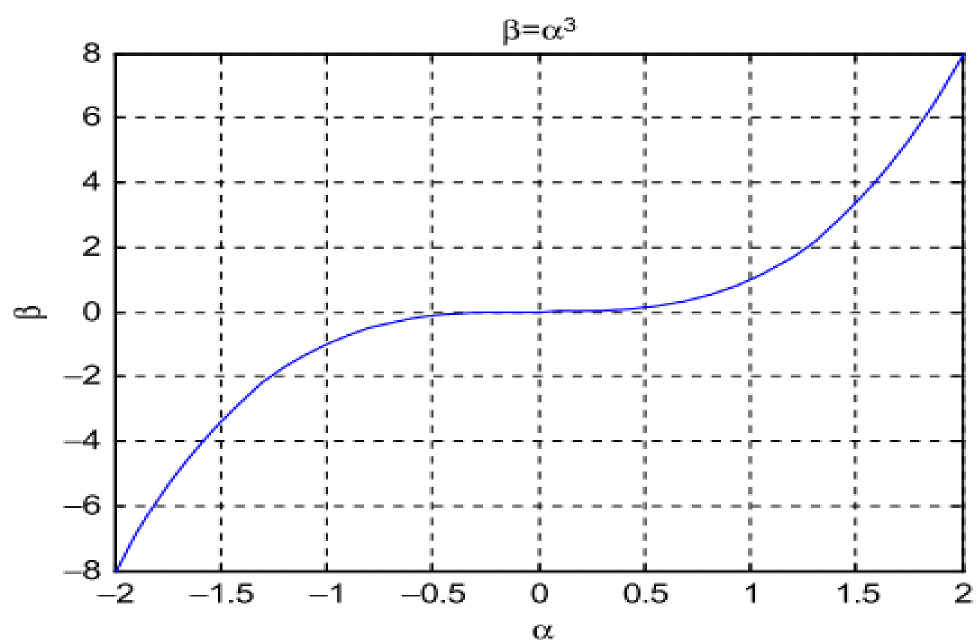
### 3.5.7 Using Greek Letters

```
alpha = -2:0.1:2;
beta = alpha.^3;
plot(alpha, beta)
xlabel('\alpha')
ylabel('\beta')
title('\beta = \alpha^3')
```




```
>> grid on
```

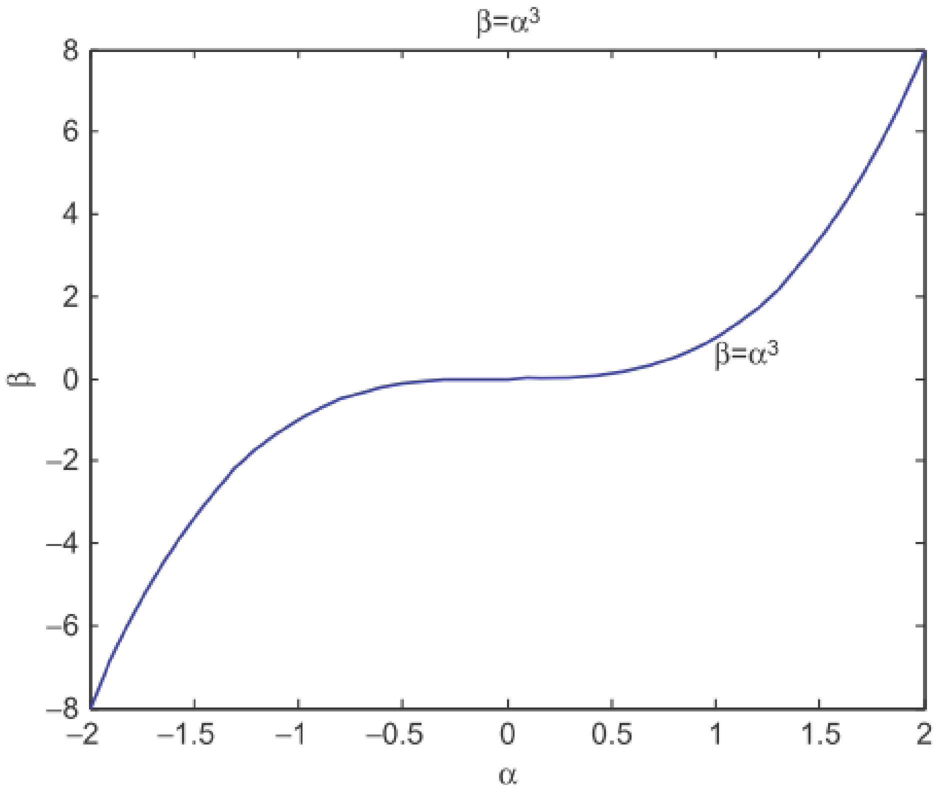
```
>> grid off
```



# 3.5.9 Adding a Text to a Figure

- You can add a text anywhere to a figure using the text command

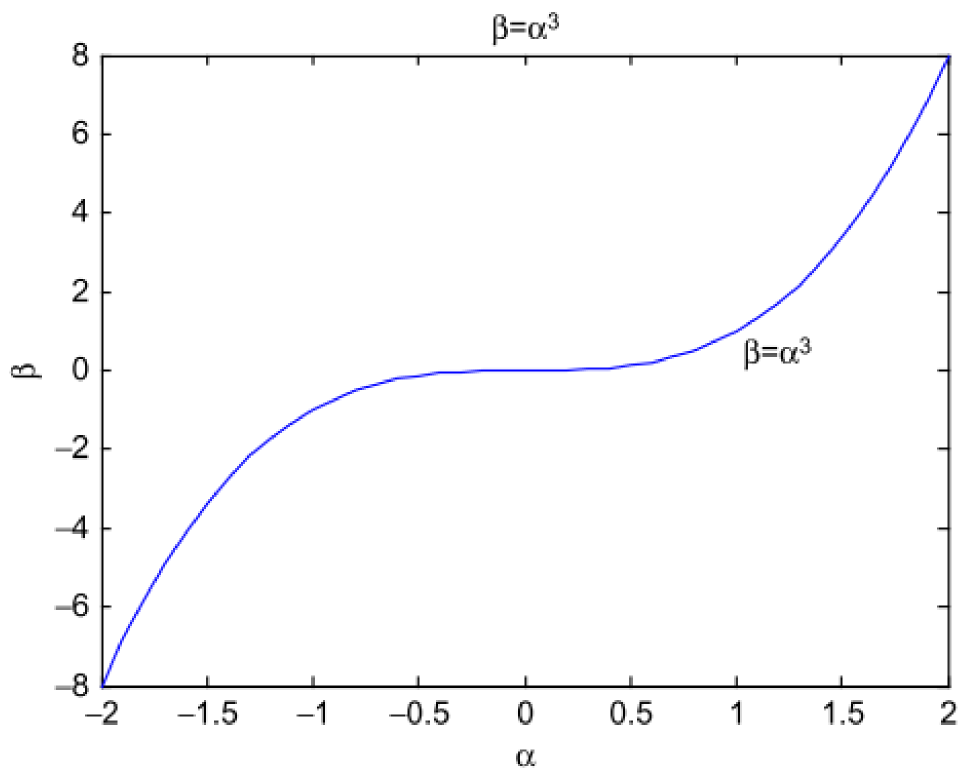
```
>> text(1,0.75,'\beta=\alpha^3')
```



## 3.5.10 Changing the Font Size

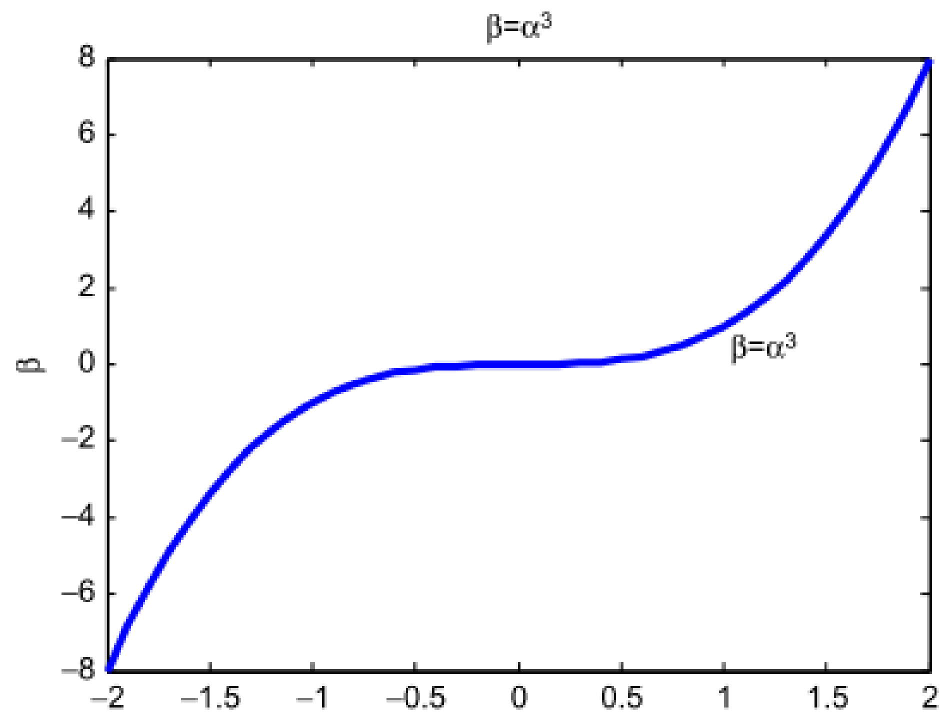
```
alpha = -2:0.1:2;
beta = alpha.^3;
plot(alpha, beta)
xlabel('\alpha','FontSize',24)
ylabel('\beta','FontSize',24)
title('\beta = \alpha^3','FontSize',17)
text(1,0.75, '\beta = \alpha^3','FontSize',18)
```

**You can change the font size for the axes labels, the figure title, and any text that you add to a figure.**



## 3.5.11 Changing the Line Width

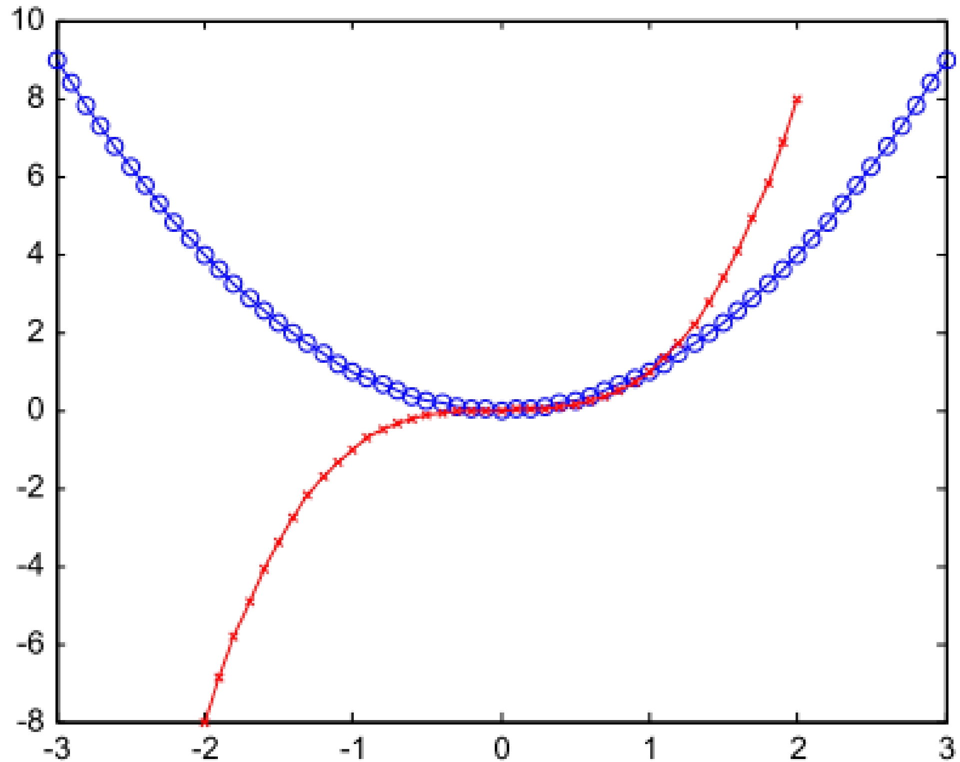
```
alpha =-2:0.1:2;
beta = alpha.^3;
plot(alpha, beta ,'LineWidth',3)
xlabel('\alpha','FontSize',24)
ylabel('\beta','FontSize',24)
title('\beta=\alpha^3','FontSize',17)
text(1,0.75, '\beta=\alpha^3','FontSize',18)
```



# 3.5.12 Multiple Plots

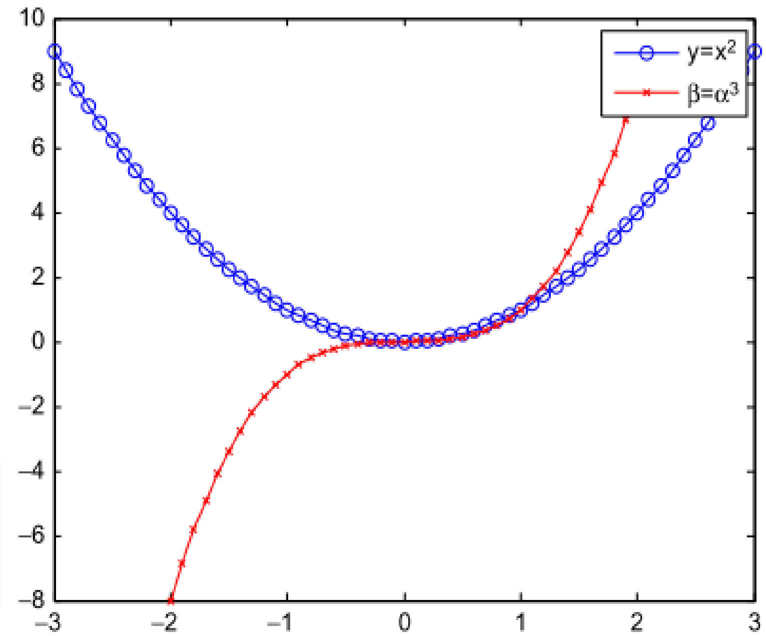
- Matlab enables you to plot more than one function on the same figure.
- For example, let us plot two functions on the same figure.

```
x =-3:0.1:3;
y = x.^2;
plot(x,y,'bo-')
hold on
alpha =-2:0.1:2;
beta = alpha.^3;
plot(alpha,beta,'rx-')
hold off
```



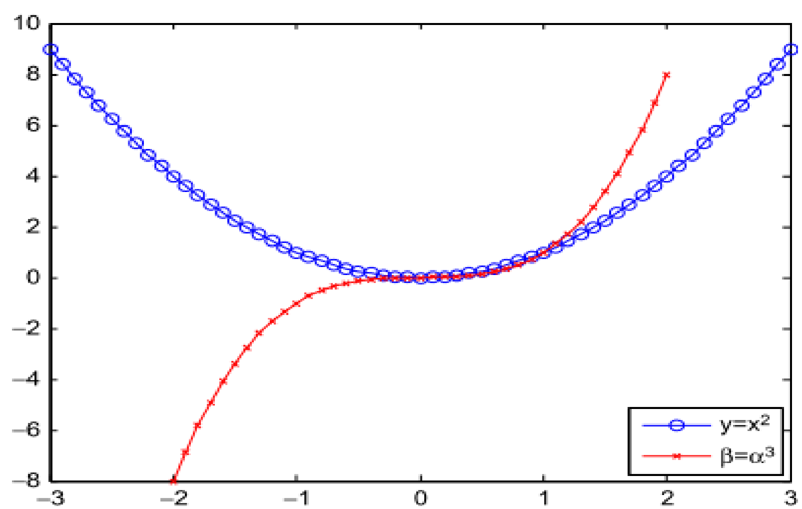
## 3.5.13 Adding a Legend to a Plot

```
>>legend('y=x^2','\beta=\alpha^3')
```




- **we can change the location of the legend as follows:**

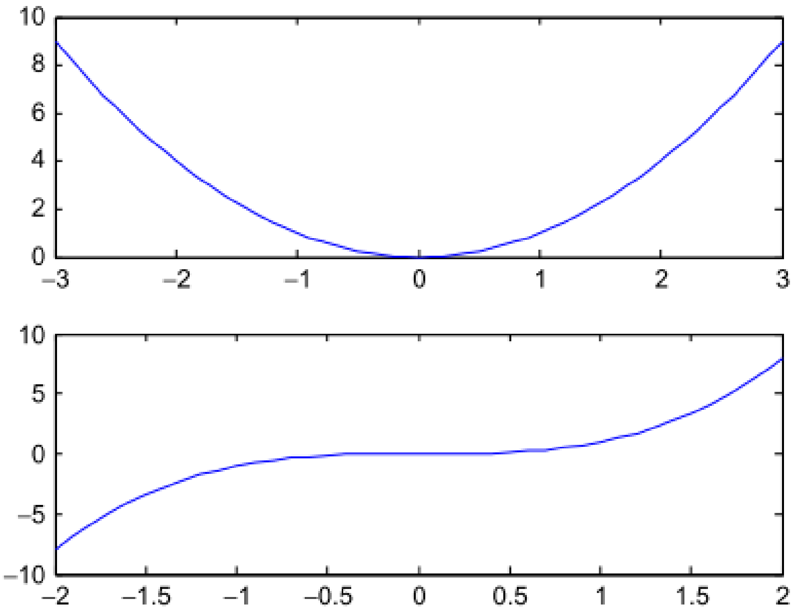
```
>>legend('y=x^2','\beta=\alpha^3','Location','SouthEast')
```



# 3.5.14 Multiple Subplots

- You can have more than one subplot in your figure.

```
x1 =-3:0.1:3;
y1 = x1.^2;
subplot(2,1,1)
plot(x1,y1)
x2 =-2:0.1:2;
y2 = x2.^3;
subplot(2,1,2)
plot(x2,y2)
```

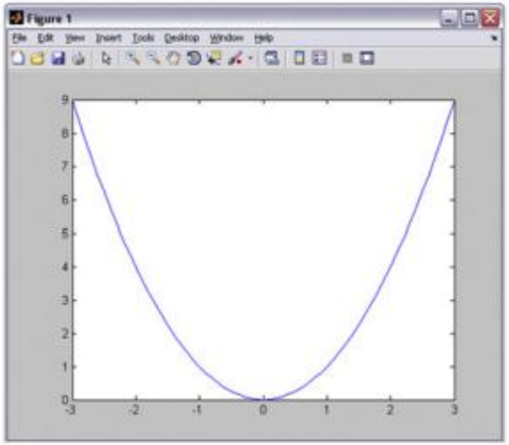


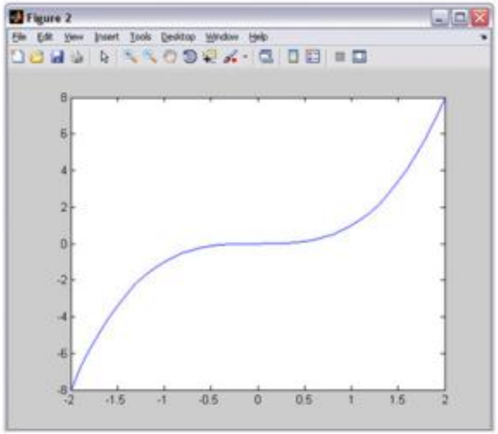

- The subplot(m,n,p) command breaks the figure window down into an mXn matrix of smaller axes and selects the pth axis to display the current plot.

# 3.5.15 Multiple Figures

- Matlab enables you to produce more than one figure.

```
x1 = -3:0.1:3;
y1 = x1.^2;
figure(1)
plot(x1,y1)
x2 = -2:0.1:2;
y2 = x2.^3;
figure(2)
plot(x2,y2)
```

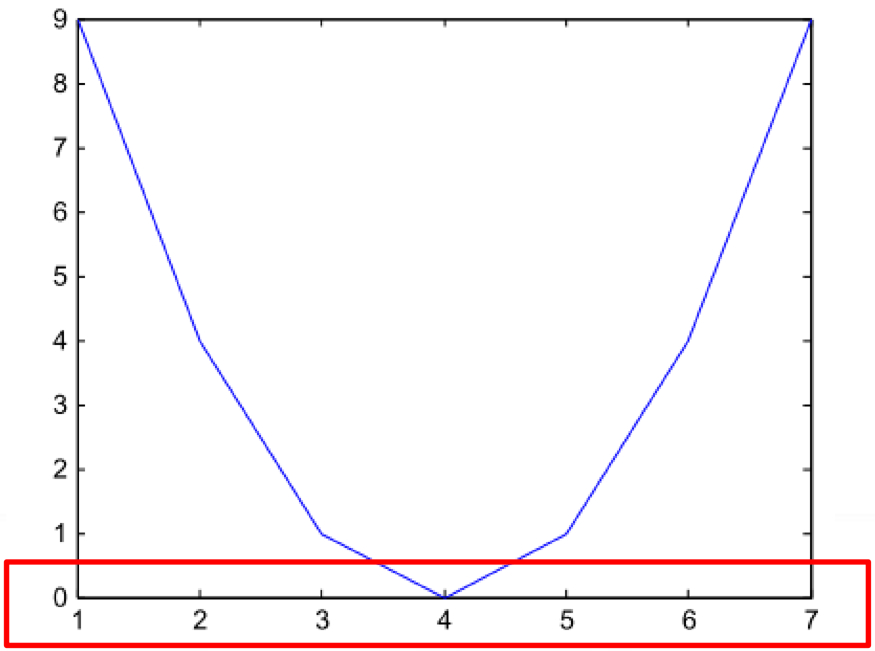


### 3.5.16 Plotting a Vector Using its Indices

```
>> x = -3:1:3
>> y = x.^2
>> plot(y)
```

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| indices | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| y | 9 | 4 | 1 | 0 | 1 | 4 | 9 |



# 4 Arrays in Matlab

## 4.1.2 Creating Arrays Manually

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 4 \\ 7 & 3 & 5 \end{bmatrix}$$

```
>>X=[1,2,4;7,3,5];

>>whos X
```

| Name | Size | Bytes | Class | Attributes |
|---|---|---|---|---|
| X | 2×3 | 48 | double | |

- The array has a data class of double (8 bytes) and
- Matlab has therefore used 48 bytes (6 3 8) of your computer memory to save the array X.

### *4.1.2.2 Creating Arrays Manually: Column-By-Column*

```
>>X=[[1;7],[2;3],[4;5]];
```

### 4.1.3 Creating Arrays Using the `repmat` Function

- The repmat function is an abbreviation of "repeat matrix," and
- It has the syntax B = repmat(A,M,N).
- This function creates a large matrix B consisting of an M X N tiling of copies of A.
- This function has the following three arguments:

1. A is the source array.
2. M is the number of times A is repeated in the vertical direction.
3. N is the number of times A is repeated in the horizontal direction.

**Example 1**

Suppose that you have the following array **A**:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

```
>>a=[1,2,3,4,5,6];
>>A=repmat(a,5,1)
```

### 4.1.3 Creating Arrays Using the `repmat` Function

**Example 3**

Create the array below using the `repmat` function.

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

```
>>C1=[1,2;3,4];
>>C=repmat(C1,2,2);
```

## 4.1.4 Transpose an Array

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 4 \\ 7 & 3 & 5 \end{bmatrix}, \quad \mathbf{X}^{\mathrm{T}} = \begin{bmatrix} 1 & 7 \\ 2 & 3 \\ 4 & 5 \end{bmatrix}$$

```
>> X = [1,2,4;7,3,5]

>> XT = X'
```

## 4.1.5 Changing Array Dimensions Using the `reshape` Function

- The reshape function has the syntax B = reshape(X,M,N).
- This function changes the dimensions of the array X to the new size of M x N.
- The elements are taken from the source array X in a column-by-column fashion
- The arguments for the reshape function are

1. X is the source array.
2. M is the number of rows in the destination array B.
3. N is the number of columns in the destination array B.

**Example 4**

Using Matlab, change the dimensions of the 3 X 2 array

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 4 \\ 7 & 3 & 5 \end{bmatrix}$$

```
>>X=[1,2,4;7,3,5]
>>B=reshape(X,3,2)
```

```
B =
     1   3
     7   4
     2   5
```

## 4.1.5 Changing Array Dimensions Using the `reshape` Function

**Example 5**

Create the array **B** using the `reshape` function.

$$\mathbf{B} = \begin{bmatrix} 1 & 6 & 11 & 16 & 21 & 26 \\ 2 & 7 & 12 & 17 & 22 & 27 \\ 3 & 8 & 13 & 18 & 23 & 28 \\ 4 & 9 & 14 & 19 & 24 & 29 \\ 5 & 10 & 15 & 20 & 25 & 30 \end{bmatrix}$$

**Answer**

```
>>b=1:1:30;
>>B=reshape(b,5,6)
```

**Example 6**

Create the following array **S** using the `reshape` function.

$$\mathbf{S} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$

**Answer**

```
>>s=1:1:25;
>>S=reshape(s,5,5);
>>S=S';
```

## 4.1.6 Finding the Size of an Array

- Matlab enables you to determine the number of rows and columns in an array.

✓ To find the number of rows in X, type

$$X = \begin{bmatrix} 1 & 2 & 4 \\ 7 & 3 & 5 \end{bmatrix}$$

```
>>m=size(X,1)
```

```
m =
     2
```

- Here the "1" keyword in the size function indicates that we wish to know the **first dimension** of the array X, that is, the number of rows.

✓ To find the number of columns in X, type

```
>>n=size(X,2)
```

```
n =
     3
```

- Here the "2" keyword in the size function indicates that we wish to know the **second dimension** of the array X, that is, the number of columns.

✓ To find the total number of elements in the array X, type

```
>>r=numel(X)
```

```
r =
     6
```

✓ To find the number of dimension of the array X, type

```
>> length(x)
```

```
B =
     2
```