MATLAB
LECTURE NOTES

Dr. ADİL YÜCEL

Istanbul Technical University

Department of Mechanical Engineering

MATLAB
LECTURE NOTES

Student Name	Student ID

Dr. ADİL YÜCEL

Istanbul Technical University

Department of Mechanical Engineering

MATLAB
LECTURE NOTES

LESSON 1

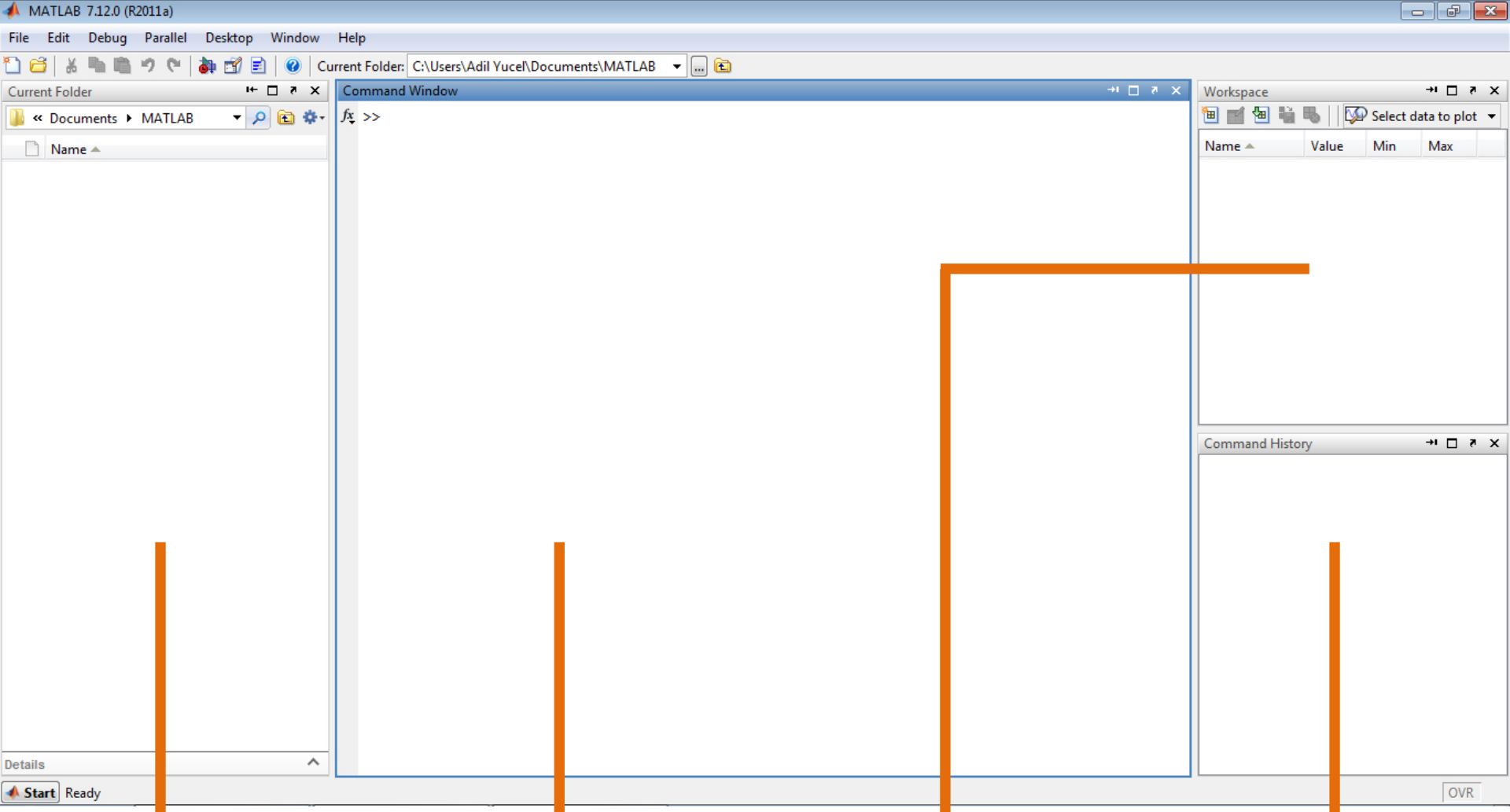
INTRODUCTION

Dr. ADİL YÜCEL

Istanbul Technical University

Department of Mechanical Engineering

- MATLAB stands for MATrix LABoratory.
- MATLAB was invented by Cleve Moler in the late 1970s.
- MATLAB is written in C.
- MATLAB works as an interpreter.
- MATLAB works slower than C and Fortran.
- MATLAB has many toolboxes for different disciplines.
- MATLAB has many visualisation tools.



This section lists the script and function files in your working folder

Command Window is the main window that you write your commands

Workspace shows the names and values of the variables defined by the user

Command History lists the copy of the commands you have run on the command window

clc

Clears the command window.

Does not delete any variable.

home

Move cursor to the upper left corner of the command window.

clear

Deletes all the variables.

clear <variable name>

Deletes a certain variable.

who

Lists the names of the variables defined by the user.

whos

Lists detailed information about the variables defined by the user.

beep

Produce a beep sound.

computer

Identify the computer on which MATLAB is running.

realmin

Smallest positive real number.

realmax

Largest positive real number.

date

Shows the current system date.

disp (<variable name>)

Displays the value of the variable.

disp (<string constant>)

Displays the string constant.

ans

Automatically created variable when the result of an operation is not assigned to a variable.

pi

Mathematical π constant.

%

Used for writing a comment.

;

Used for suppressing a variable or the result of an operation.

+ : Plus

– : Minus

* : Matrix multiplication

.* : Array multiplication (Element wise)

^ : Matrix power

.^ : Array power (Element wise)

\ : Backslash or left division

/ : Slash or right division

.\ : Left array division (Element wise)

./ : Right array division (Element wise)

: : Colon

‘ : Transpose

= : Assignment

== : Equal to

~= : Not equal to

> : Greater than

< : Less than

>= : Greater than or equal to

<= : Less than or equal to

& : Logical AND

| : Logical OR

~ : Logical NOT

true : Logical TRUE

false : Logical FALSE

abs (x)

Calculates absolute value of x

ceil (x)

Rounds x to the next integer

floor (x)

Rounds x to the previous integer

round (x)

Rounds x to the nearest integer

exp (x)

Calculates exponential of x (e^x)

log (x)

Calculates natural logarithm of x (lnx)

log2 (x)

Calculates base 2 logarithm of x ($\log_2 x$)

log10 (x)

Calculates base 10 logarithm of x ($\log_{10} x$)

mod (x,a)

Calculates modulus of x (x mod a)

sqrt (x)

Calculates square root of x

nthroot (x,n)

Calculates n^{th} root of x

factorial (x)

Calculates factorial of x (x!)

sin (x)

Calculates sine of x

sinh (x)

Calculates hyperbolic sine of x

asin (x)

Calculates inverse sine of x

asinh (x)

Calculates inverse hyperbolic sine of x

cos (x)

Calculates cosine of x

cosh (x)

Calculates hyperbolic cosine of x

acos (x)

Calculates inverse cosine of x

acosh (x)

Calculates inverse hyperbolic cosine of x

tan (x)

Calculates tangent of x

tanh (x)

Calculates hyperbolic tangent of x

atan (x)

Calculates inverse tangent of x

atanh (x)

Calculates inverse hyperbolic tangent of x

sind (x)
Calculates sine of x (x in degrees)

cosd (x)
Calculates cosine of x (x in degrees)

tand (x)
Calculates tangent of x (x in degrees)

cotd (x)
Calculates cotangent of x (x in degrees)

secd (x)
Calculates secant of x (x in degrees)

asind (x)
Calculates inverse sine of x (result in degrees)

acosd (x)
Calculates inverse cosine of x (result in degrees)

atand (x)
Calculates inverse tangent of x (result in degrees)

acotd (x)
Calculates inverse cotangent of x (result in degrees)

asecd (x)
Calculates inverse secant of x (result in degrees)

format short
4 decimal places (3.1416)

format short e
4 decimal places with exponent (3.1416e+000)

format long
Many decimal places (3.141592653589793)

format long e
Many decimal places with exponent
(3.141592653589793e+000)

format bank
2 decimal places (3.14)

- A scalar is a matrix with only one row and one column. It is simply a single value.
- A vector is a special form of matrix which contains only one row or one column. A matrix with only one row is called a “row vector” and a matrix with only one column is called a “column vector”. They can be named as one-dimensional arrays.
- A matrix is a rectangular multidimensional array of numbers.

A scalar can be created as follows :

```
>> x = 25

x =

    25

>> y = 128

y =

   128

>> z = 12.45

z =

   12.4500
```

A row vector can be created as follows :

```
>> A = [12 28 32 69 123 456]

A =

    12    28    32    69   123   456
```

A column vector can be created as follows :

```
>> B = [12;028;32;69;123;456]

B =

    12
    28
    32
    69
   123
   456
```

A matrix can be created as follows :

```
>> C = [23 45 67;45 56 89;25 89 76]

C =

    23    45    67
    45    56    89
    25    89    76

>> D = [28 32 68;12 15 46]

D =

    28    32    68
    12    15    46
```

- Variable and function names are case sensitive.
- They can not start with a number.
- They can not include language specific characters.
- They can not include punctuation characters.
- They can not include spaces.

MATLAB
LECTURE NOTES

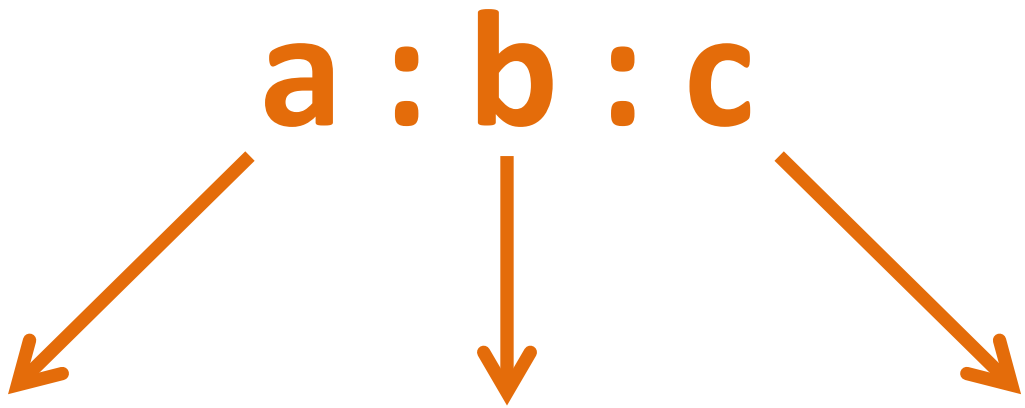
LESSON 2

MATRIX OPERATIONS

Dr. ADİL YÜCEL

Istanbul Technical University

Department of Mechanical Engineering



This method creates an array with values starting from **a** and ending at **c** with an increment of **b**.

```
>> 0:2:20

ans =

     0     2     4     6     8    10    12    14    16    18    20
```

An array can be created as follows :

```
>> A = [0 5 10 15 20 25 30 35 40]

A =

     0     5    10    15    20    25    30    35    40
```

Same array can also be created as follows :

```
>> A = 0:5:40

A =

     0     5    10    15    20    25    30    35    40
```

linspace (a,b,n)

This method creates an array with n number of equally spaced values starting from a and ending at b.

```
>> linspace (10,20,5)

ans =

   10.0000   12.5000   15.0000   17.5000   20.0000
```

logspace (a,b,n)

This method creates an array with n number of equally spaced logarithmic (10^x) values starting from a and ending at b.

```
>> logspace (1,4,4)

ans =

          10         100        1000       10000
```

Addition and subtraction of two vectors :

```
>> A = [12 8 5 15];
>> B = [10 4 8 12];
>> A + B

ans =

    22    12    13    27

>> A - B

ans =

     2     4    -3     3
```

Multiplication and division of two vectors :

```
>> A = [12 8 5 15];
>> B = [4 4 1 3];
>> A .* B

ans =

    48    32     5    45

>> A ./ B

ans =

     3     2     5     5
```

Multiplication of a scalar and a vector :

```
>> A = [12 8 5 15];
>> k = 5;
>> k * A

ans =

    60    40    25    75

>> k .* A

ans =

    60    40    25    75
```

Addition and subtraction of two matrices :

```
>> A = [6 8 12;9 15 18;32 22 16];
>> B = [4 9 15;3 12 9;16 20 14];
>> A + B

ans =

    10    17    27
    12    27    27
    48    42    30

>> A - B

ans =

     2    -1    -3
     6     3     9
    16     2     2
```

Matrix multiplication of two matrices :

```
>> A = [2 4 5;1 2 3;5 4 8];
>> B = [2 5 3;4 5 2;1 4 7];
>> A * B

ans =

    25    50    49
    13    27    28
    34    77    79
```

Element wise multiplication of two matrices :

```
>> A = [2 4 5;1 2 3;5 4 8];
>> B = [2 5 3;4 5 2;1 4 7];
>> A .* B

ans =

     4    20    15
     4    10     6
     5    16    56
```

Element wise division of two matrices :

```
>> A = [2 4 5;1 2 3;5 4 8];
>> B = [2 5 3;4 5 2;1 4 7];
>> A ./ B

ans =

    1.0000    0.8000    1.6667
    0.2500    0.4000    1.5000
    5.0000    1.0000    1.1429
```

Multiplication of a scalar and a matrix :

```
>> A = [2 4 5;1 2 3;5 4 8];
>> k = 3;
>> k * A

ans =

     6    12    15
     3     6     9
    15    12    24

>> k .* A

ans =

     6    12    15
     3     6     9
    15    12    24
```

dot (A,B)

Calculates the dot product of two vectors.

```
>> A = [2 4 8 3];
>> B = [4 5 8 9];
>> dot (A,B)

ans =

   119
```

cross (A,B)

Calculates the cross product of two vectors.

```
>> A = [2 3 4];
>> B = [4 5 6];
>> cross (A,B)

ans =

    -2     4    -2
```

sort (A)

Sorts the elements of the given vector.

```
>> A = [12 8 5 15 12 16 23]

A =

    12     8     5    15    12    16    23

>> sort (A)

ans =

     5     8    12    12    15    16    23
```

mean (A)

Calculates the average value of the elements of the given vector.

```
>> A = [3 4 5]

A =

     3     4     5

>> mean (A)

ans =

     4
```

sum (A)

Calculates the sum of the elements of the given vector.

```
>> A = [12 8 5 15 12 16 23]

A =

    12     8     5    15    12    16    23

>> sum (A)

ans =

    91
```

prod (A)

Calculates the product of the elements of the given vector.

```
>> A = [2 3 4 5]

A =

     2     3     4     5

>> prod (A)

ans =

   120
```

min (A)

Results the minimum value of the given vector.

```
>> A = [12 8 5 15 12 16 23]

A =

    12     8     5    15    12    16    23

>> min (A)

ans =

     5
```

max (A)

Results the maximum value of the given vector.

```
>> A = [12 8 5 15 12 16 23]

A =

    12     8     5    15    12    16    23

>> max (A)

ans =

    23
```

zeros (m,n)

Creates an mxn matrix of all zeros.

```
>> zeros (5,3)

ans =

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

ones (m,n)

Creates an mxn matrix of all ones.

```
>> ones (5,3)

ans =

     1     1     1
     1     1     1
     1     1     1
     1     1     1
     1     1     1
```

eye (m)

Creates an mxm identity matrix.

```
>> eye (5)

ans =

     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
```

rand (m,n)

Creates an mxn matrix full of uniformly distributed random numbers between 0 and 1.

```
>> rand (4,5)

ans =

    0.9649    0.4854    0.9157    0.0357    0.7577
    0.1576    0.8003    0.7922    0.8491    0.7431
    0.9706    0.1419    0.9595    0.9340    0.3922
    0.9572    0.4218    0.6557    0.6787    0.6555
```

diag (A)

Creates a diagonal matrix with the elements of the given vector as the diagonal.

```
>> A = [2 4 7 5 9];
>> diag (A)

ans =

     2     0     0     0     0
     0     4     0     0     0
     0     0     7     0     0
     0     0     0     5     0
     0     0     0     0     9
```

size (A)

Returns the dimensions of the given matrix.

```
>> A = rand (4,5)

A =

    0.1712    0.0462    0.3171    0.3816    0.4898
    0.7060    0.0971    0.9502    0.7655    0.4456
    0.0318    0.8235    0.0344    0.7952    0.6463
    0.2769    0.6948    0.4387    0.1869    0.7094

>> size (A)

ans =

     4     5
```

length (A)

Returns the length of the given vector.

```
>> A = [12 8 5 15 12 16 23]

A =

    12     8     5    15    12    16    23

>> length (A)

ans =

     7
```

det (A)

Calculates the determinant of the given matrix.

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

A =

     2     4     5
     1     2     3
     5     4     8

>> det (A)

ans =

     6
```

inv (A)

Calculates the inverse of the given matrix.

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

A =

     2     4     5
     1     2     3
     5     4     8

>> inv (A)

ans =

    0.6667   -2.0000    0.3333
    1.1667   -1.5000   -0.1667
   -1.0000    2.0000         0
```

A’

Calculates the transpose of the given matrix.

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

A =

     2     4     5
     1     2     3
     5     4     8

>> A'

ans =

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

A (m , n)

Results the element of the given matrix in row m and column n.

```
>> A = [12 22 15;28 16 8;32 18 24]

A =

    12    22    15
    28    16     8
    32    18    24

>> A (3,2)

ans =

    18
```

A(:,n)

Results the elements of the given matrix in column n.

```
>> A = [12 22 15;28 16 8;32 18 24]

A =

    12    22    15
    28    16     8
    32    18    24

>> A (:,3)

ans =

    15
     8
    24
```

A (m , :)

Results the elements of the given matrix in row m.

```
>> A = [12 22 15;28 16 8;32 18 24]

A =

    12    22    15
    28    16     8
    32    18    24

>> A (2,:)

ans =

    28    16     8
```

Solving a system of linear equations :

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} * \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

K X S

K.X = S

$K^{-1}.K.X = K^{-1}.S$

$X = K^{-1}.S$ → **X = inv (K)*S**

$$3a + 6b + 8c = 101.5$$

$$5a - 4b - 7c = -56.5$$

$$4a + 5b + 9c = 108.5$$

```
>> K = [3 6 8;5 -4 -7;4 5 9]

K =

     3     6     8
     5    -4    -7
     4     5     9

>> S = [101.5;-56.5;108.5]

S =

  101.5000
  -56.5000
  108.5000

>> inv (K) * S

ans =

    3.5000
    4.5000
    8.0000
```

MATLAB
LECTURE NOTES

LESSON 3

DATA VISUALIZATION

Dr. ADİL YÜCEL

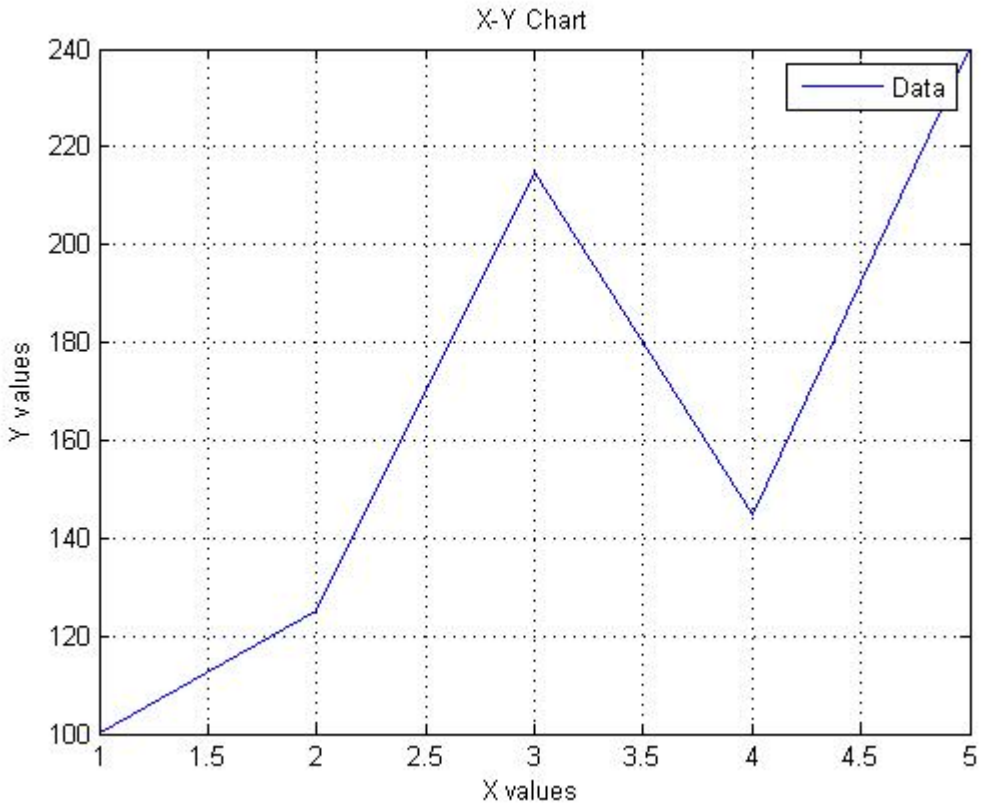
Istanbul Technical University

Department of Mechanical Engineering

plot (x,y)

Plots the given vectors on a normal chart.

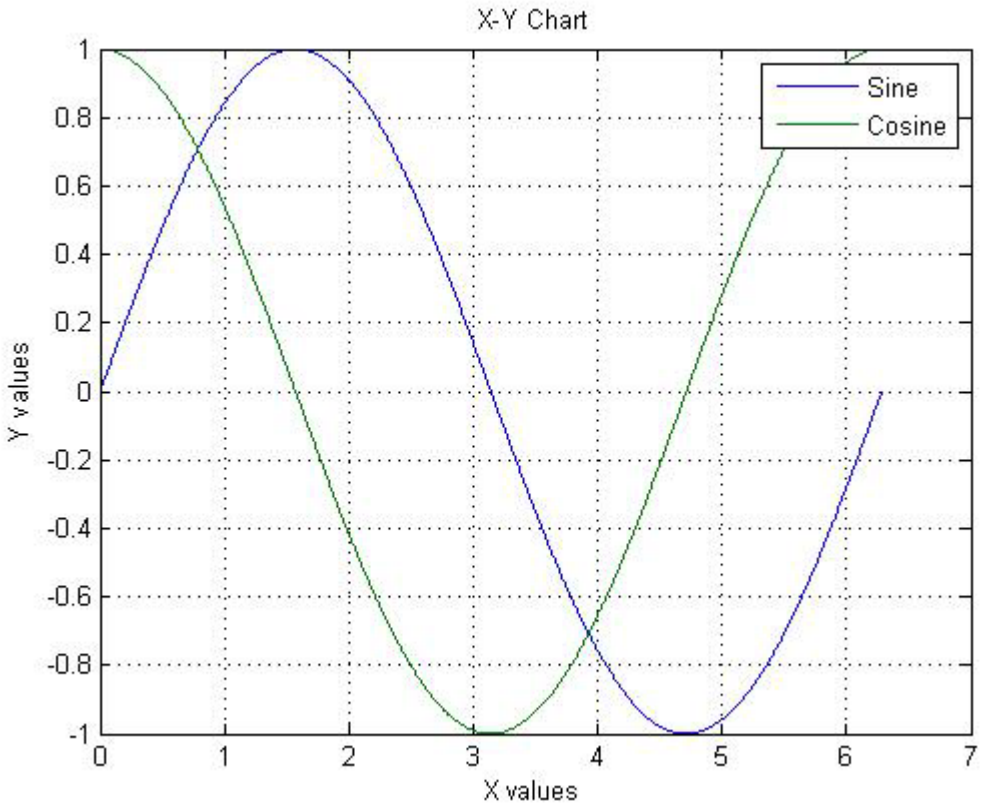
```
>> x = [1 2 3 4 5];
>> y = [100 125 215 145 240];
>> plot (x,y)
>> xlabel ('X values')
>> ylabel ('Y values')
>> title ('X-Y Chart')
>> legend ('Data')
>> grid
```



plot (x1,y1,x2,y2)

Plots the two given vectors on a normal chart.

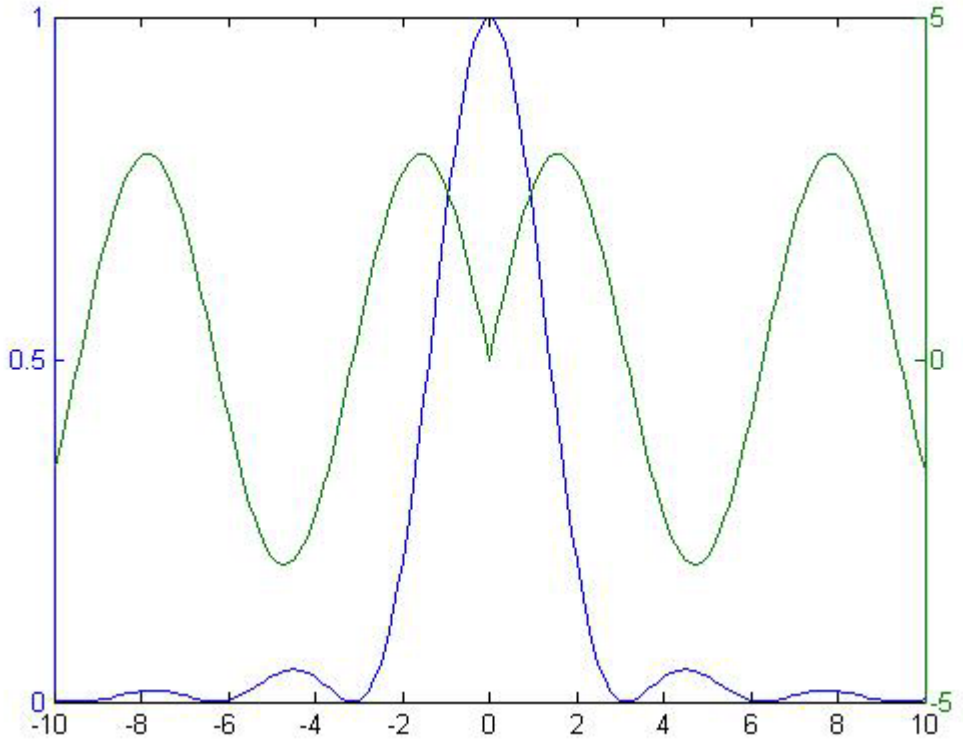
```
>> x = linspace(0,2*pi,1000);
>> y1 = sin(x);
>> y2 = cos(x);
>> plot (x,y1,x,y2)
>> xlabel ('X values')
>> ylabel ('Y values')
>> title ('X-Y Chart')
>> legend ('Sine','Cosine')
>> grid
```



plotyy (x,y1,x,y2)

Plots the two given vectors with two different y-axis.

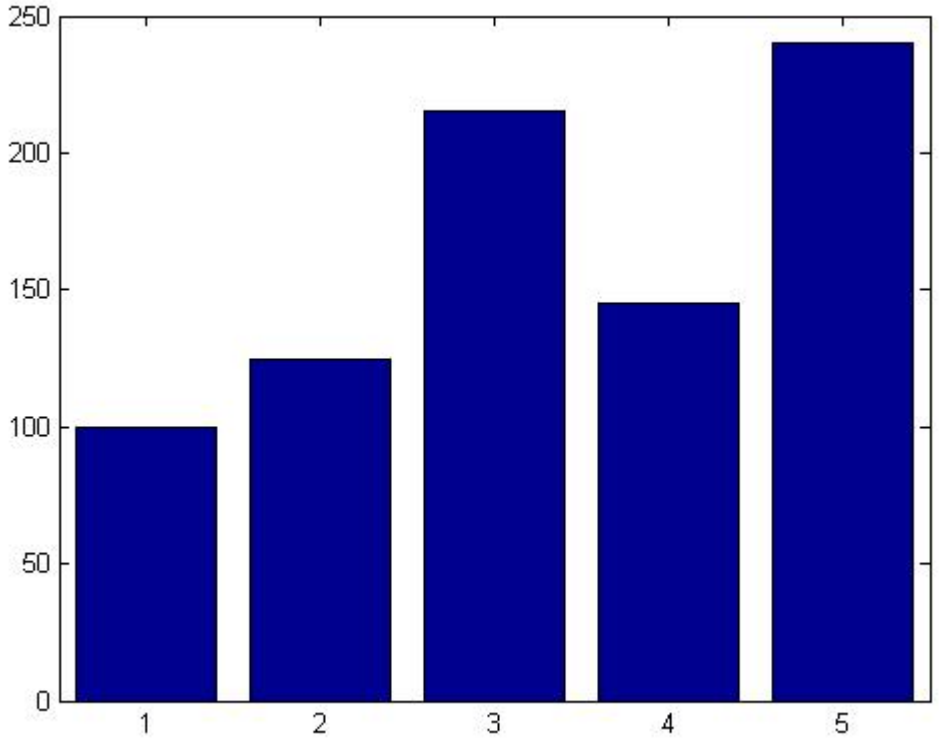
```
>> x = -10:0.1:10;
>> y1 = (sin(x)./x).^2;
>> y2 = 3*sin(abs(x));
>> plotyy (x,y1,x,y2)
```



bar (x,y)

Plots the given vectors on a vertical bar chart.

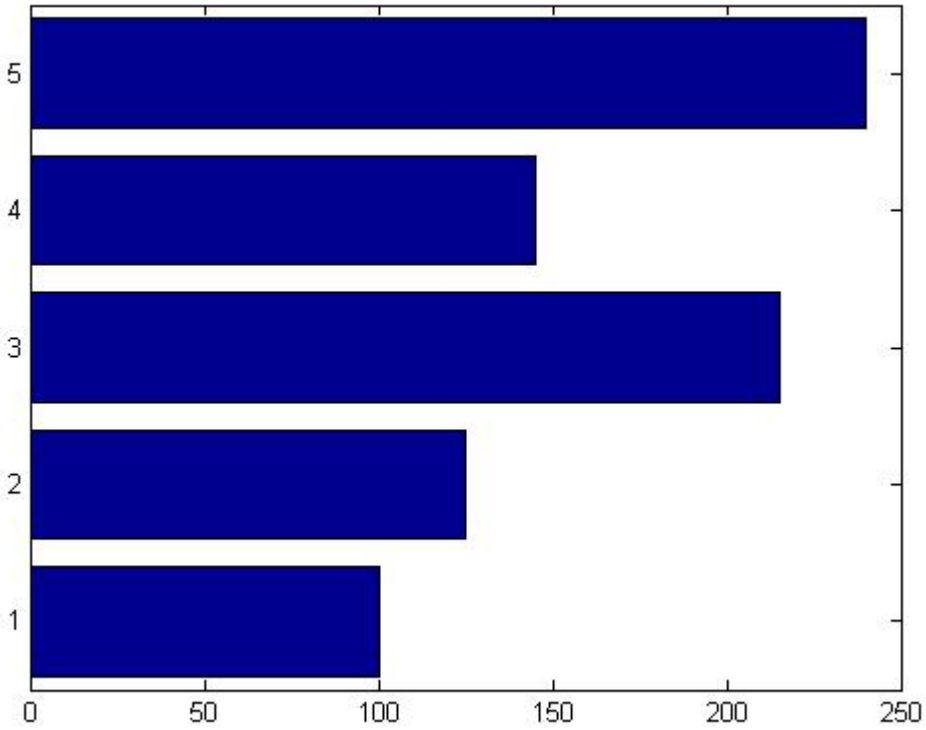
```
>> x = [1 2 3 4 5];
>> y = [100 125 215 145 240];
>> bar (x,y)
```



barh (x,y)

Plots the given vectors on a horizontal bar chart.

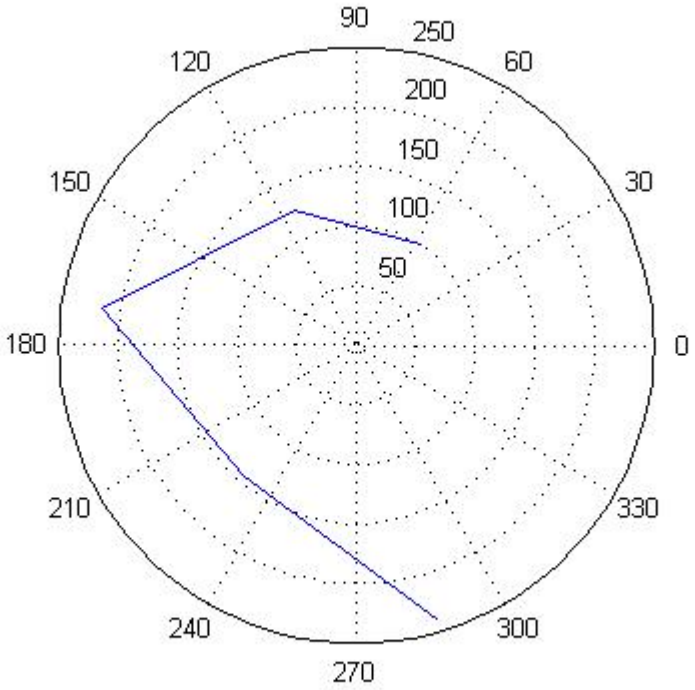
```
>> x = [1 2 3 4 5];
>> y = [100 125 215 145 240];
>> barh (x,y)
```



polar (x,y)

Plots the given vectors on a polar coordinate chart.

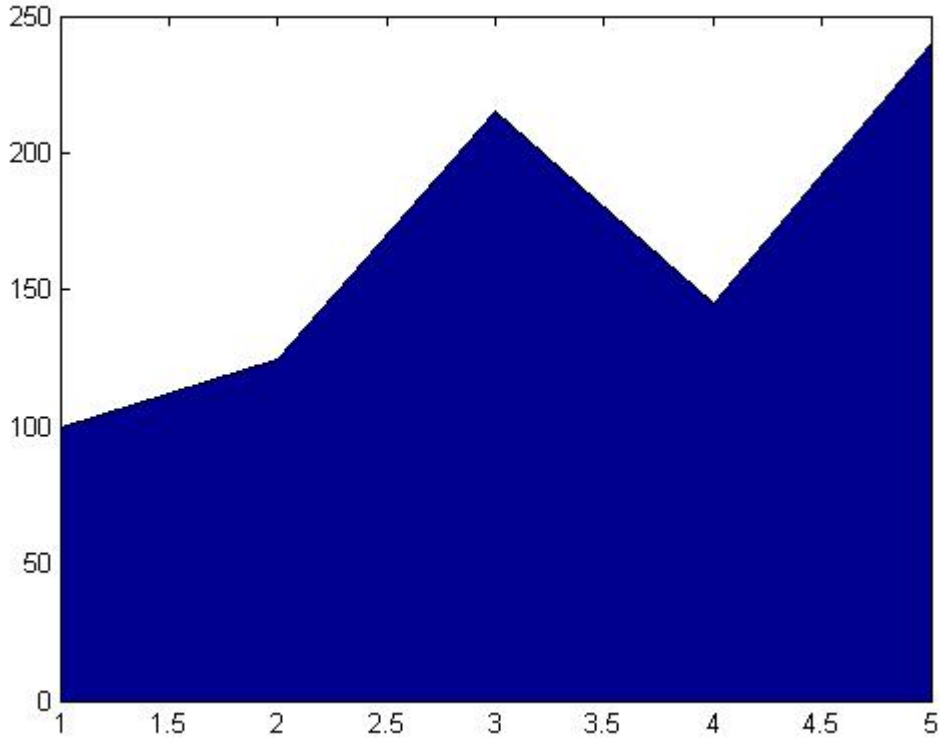
```
>> x = [1 2 3 4 5];
>> y = [100 125 215 145 240];
>> polar (x,y)
```



area (x,y)

Plots the given vectors on an area chart.

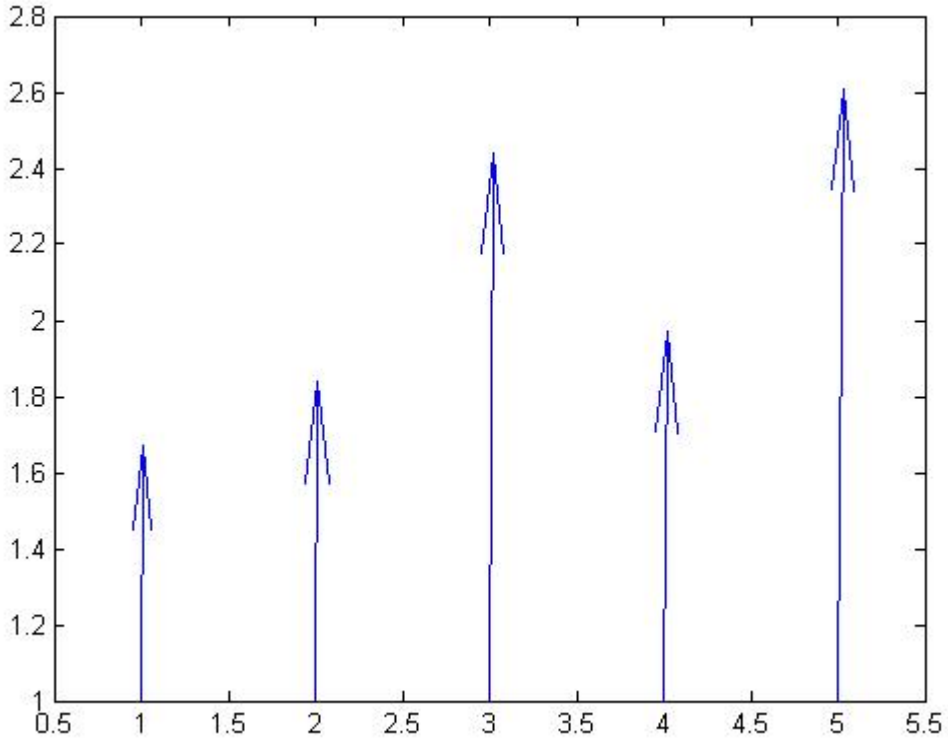
```
>> x = [1 2 3 4 5];
>> y = [100 125 215 145 240];
>> area (x,y)
```



quiver (x,y)

Plots the given vectors on a velocity chart.

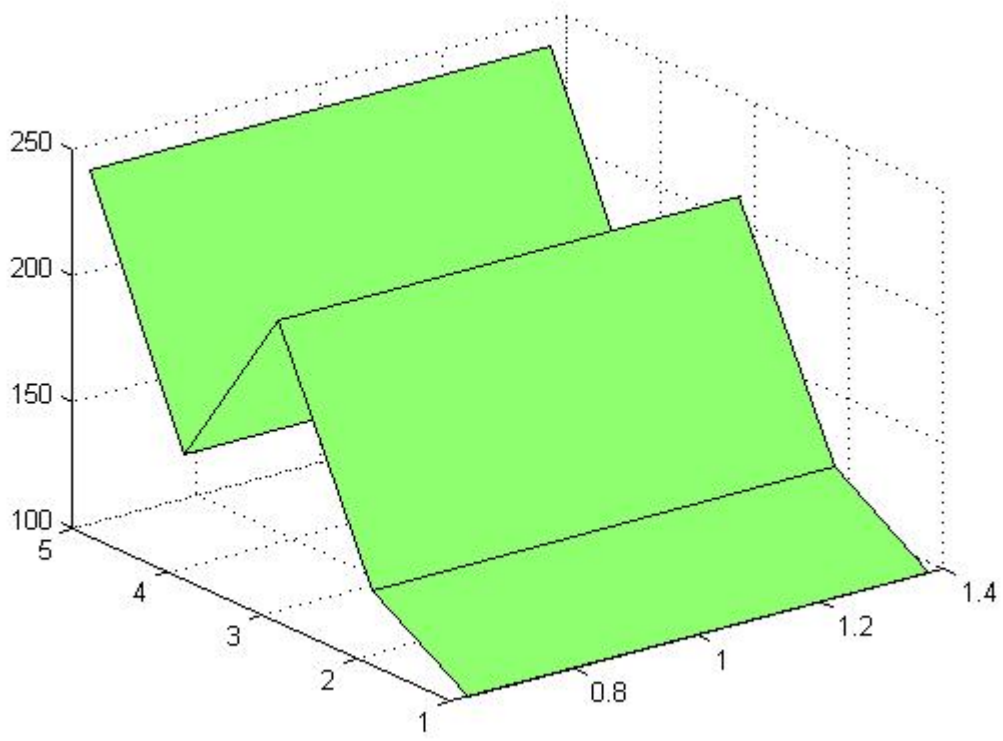
```
>> x = [1 2 3 4 5];
>> y = [100 125 215 145 240];
>> quiver (x,y)
```



ribbon (x,y)

Plots the given vectors on a ribbon chart.

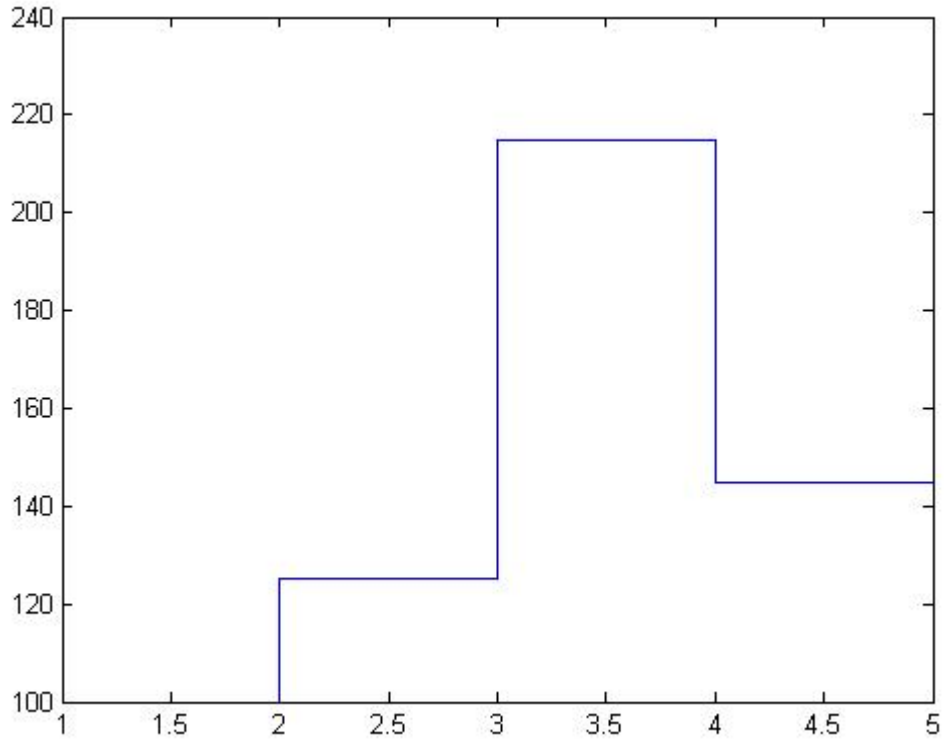
```
>> x = [1 2 3 4 5];
>> y = [100 125 215 145 240];
>> ribbon (x,y)
```



stairs (x,y)

Plots the given vectors on a stairstep chart.

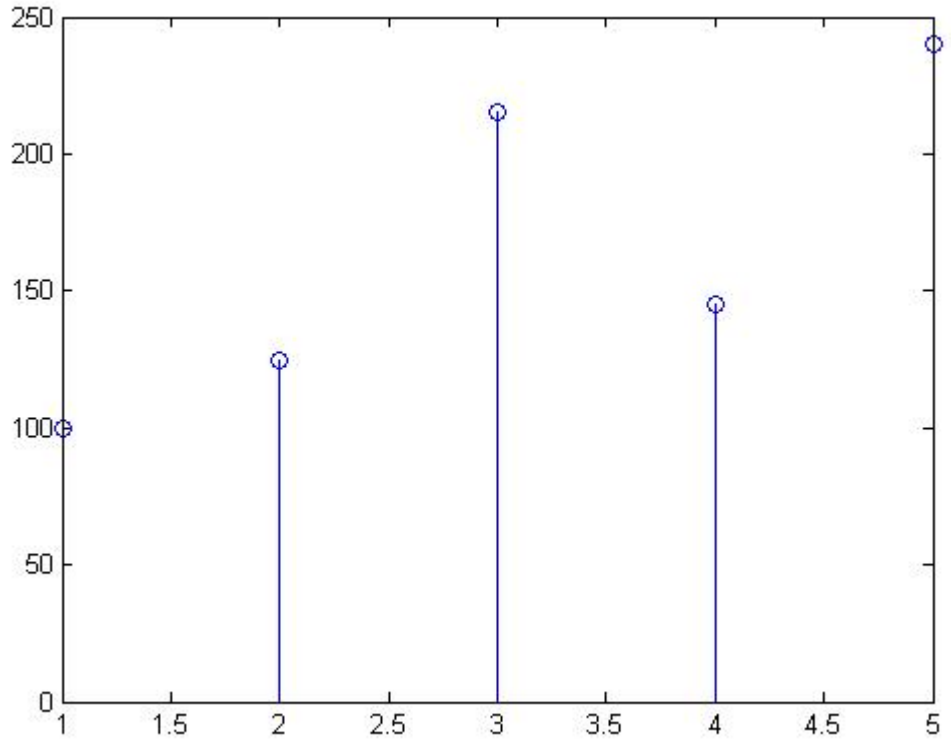
```
>> x = [1 2 3 4 5];
>> y = [100 125 215 145 240];
>> stairs (x,y)
```



stem (x,y)

Plots the given vectors on a discrete data chart.

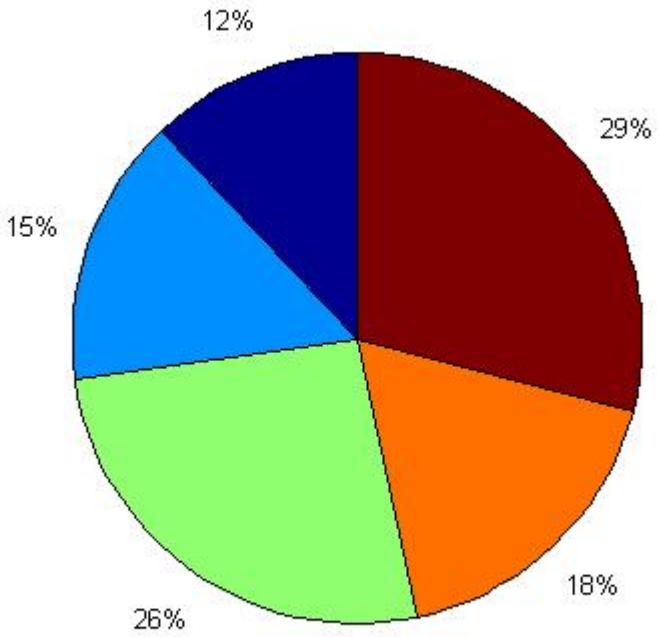
```
>> x = [1 2 3 4 5];
>> y = [100 125 215 145 240];
>> stem (x,y)
```



pie (x)

Plots the pie chart for the values given in the vector.

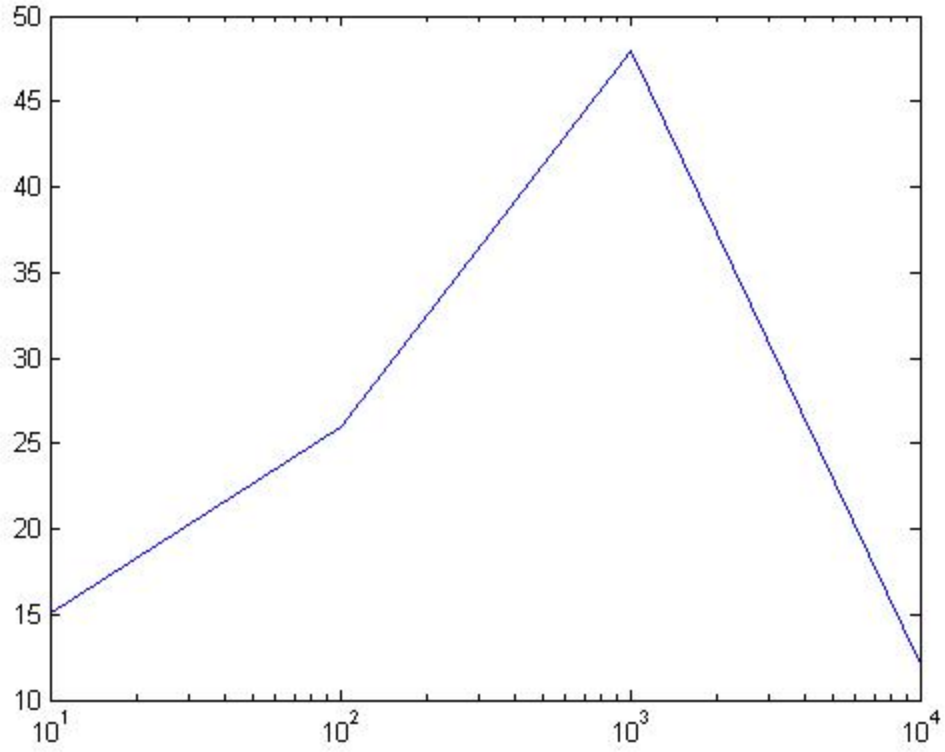
```
>> x = [100 125 215 145 240];
>> pie (x)
```



semilogx (x,y)

Plots the given vectors on a normal chart with logarithmic x axis and linear y axis.

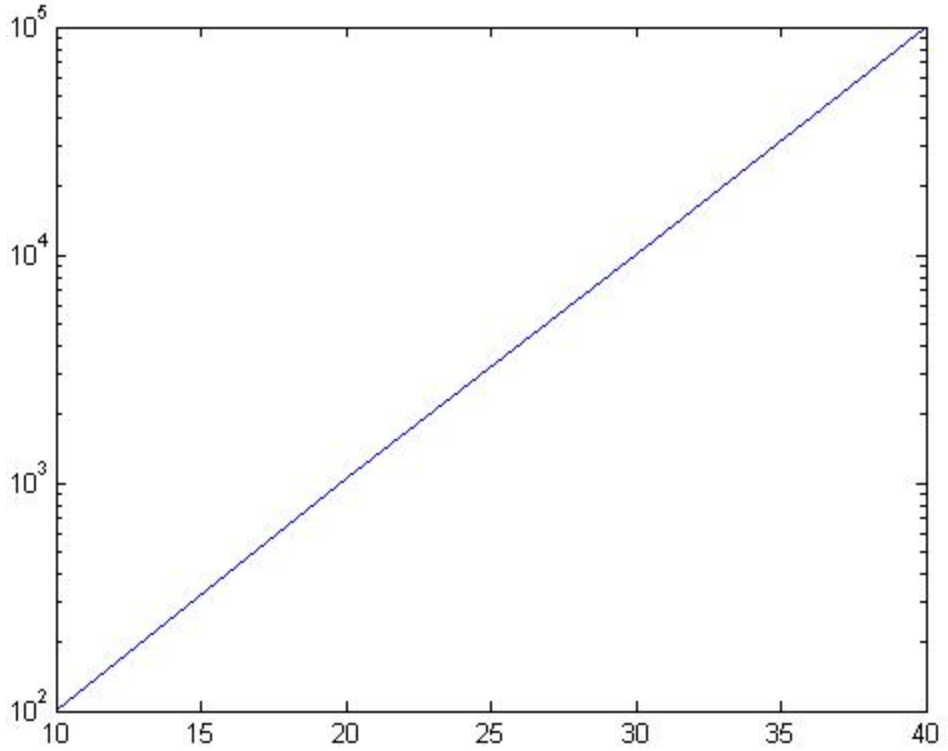
```
>> x = [10 100 1000 10000];
>> y = [15 26 48 12];
>> semilogx (x,y)
```



semilogy (x,y)

Plots the given vectors on a normal chart with logarithmic y axis and linear x axis.

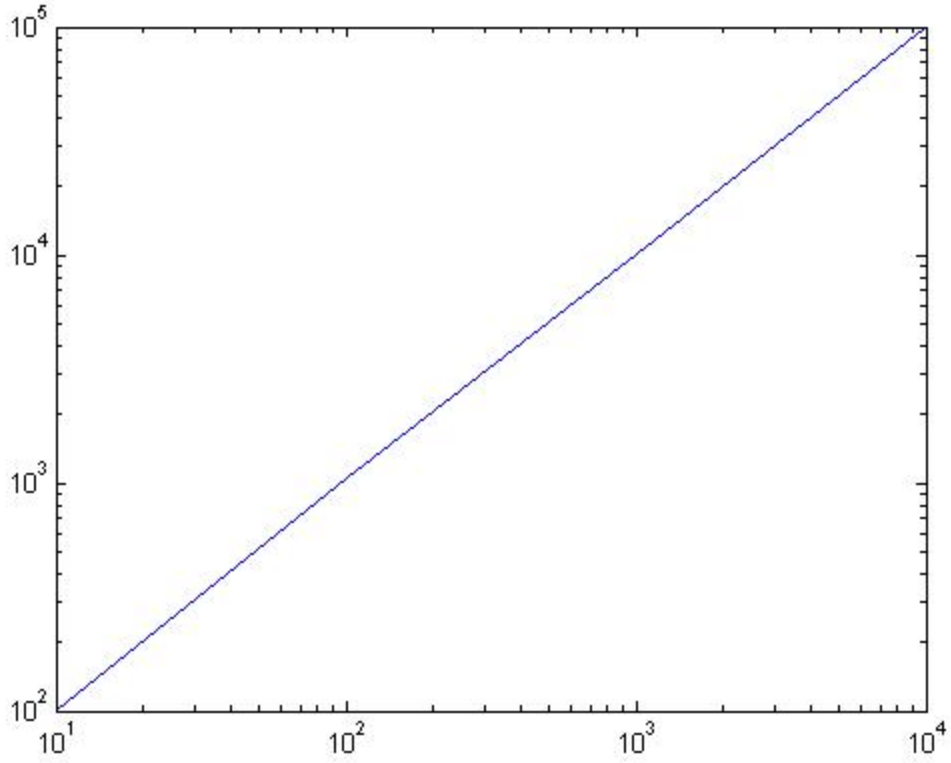
```
>> x = [10 20 30 40];
>> y = [100 1050 10200 100500];
>> semilogy (x,y)
```



loglog (x,y)

Plots the given vectors on a normal chart with logarithmic x axis and logarithmic y axis .

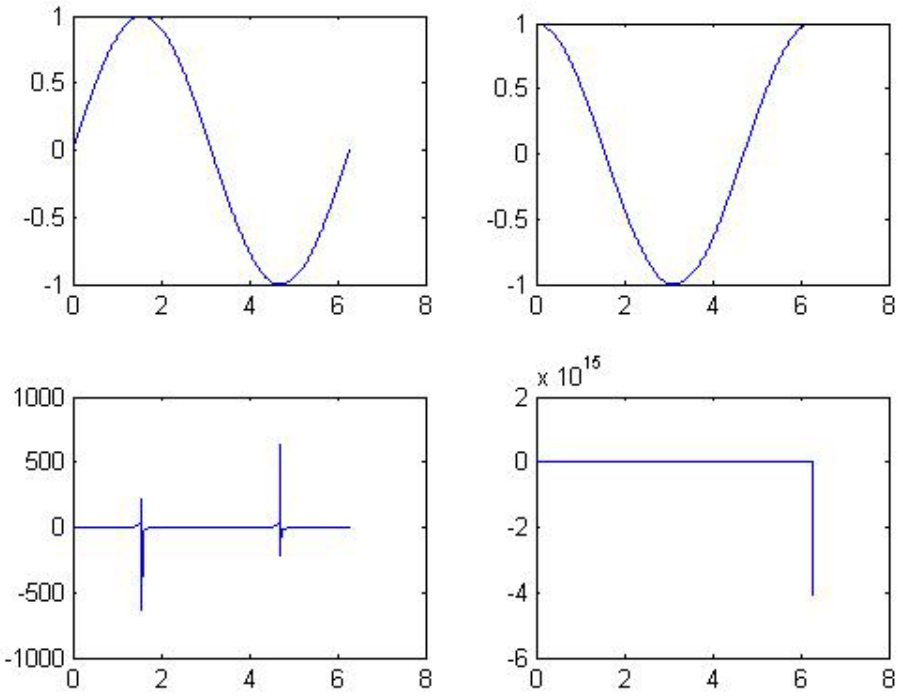
```
>> x = [10 100 1000 10000];
>> y = [100 1050 10200 100500];
>> loglog (x,y)
```



subplot (m,n,k)

Creates a chart area of mxn and locates the next plot to the position of k.

```
>> x = linspace(0,2*pi,1000);
>> y1 = sin(x);
>> y2 = cos(x);
>> y3 = tan(x);
>> y4 = cot(x);
>> subplot (2,2,1),plot (x,y1)
>> subplot (2,2,2),plot (x,y2)
>> subplot (2,2,3),plot (x,y3)
>> subplot (2,2,4),plot (x,y4)
```



MATLAB
LECTURE NOTES

LESSON 4

POLYNOMIALS

Dr. ADİL YÜCEL

Istanbul Technical University

Department of Mechanical Engineering

General form of an n^{th} degree polynomial function is defined as follows :

$$a_n x^n + \ldots\ldots\ldots\ldots + a_3 x^3 + a_2 x^2 + a_1 x + a_0$$

The vector representing the polynomial function is defined as follows :

$$[\ a_n \ \ldots\ldots\ldots\ldots \ a_3 \ a_2 \ a_1 \ a_0\ ]$$

Examples of polynomial vectors :

$p(x) = 6x^3 + 2x^2 + 4x + 5 \rightarrow p = [\ 6\ \ 2\ \ 4\ \ 5\ ]$

$p(x) = 5x^4 + 8x^2 + 5 \rightarrow p = [\ 5\ \ 0\ \ 8\ \ 0\ \ 5\ ]$

$p(x) = 7x^5 - 3x^2 + 8x - 9 \rightarrow p = [\ 7\ \ 0\ \ 0\ \ -3\ \ \ 8\ \ -9\ ]$

$p(x) = 5x^3 - 4x^2 \rightarrow p = [\ 5\ \ -4\ \ 0\ \ 0\ ]$

$p(x) = 6x^5 + 8 \rightarrow p = [\ 6\ \ 0\ \ 0\ \ 0\ \ 0\ \ 8\ ]$

$p(x) = \sqrt{7}x^2 - 4.8x + 7.5 \rightarrow p = [\ sqrt(7)\ \ -4.8\ \ 7.5\ ]$

$p(x) = x + 3x^2 + 5x^3 \rightarrow p = [\ 5\ \ 3\ \ 1\ \ 0\ ]$

polyval (p,x)

This method calculates the value of the polynomial p for the given value of x.

```
>> p = [ 6 2 4 5 ];
>> polyval (p,2)

ans =

        69.00
```

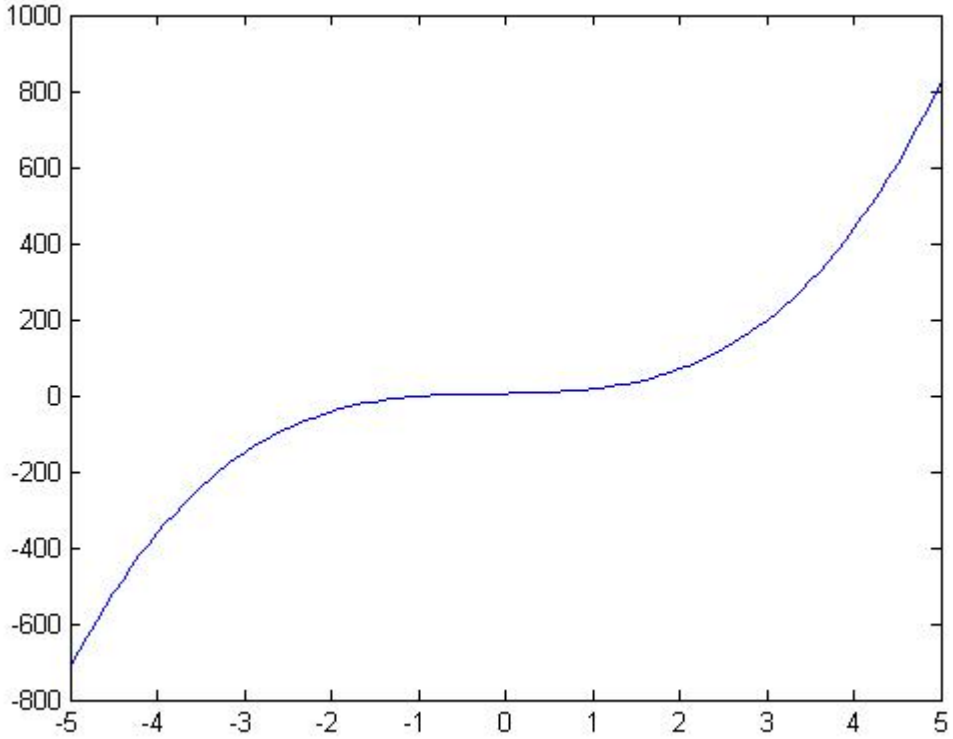
$$p(x) = 6x^3 + 2x^2 + 4x + 5$$

```
>> p = [ 6 2 4 5 ];
>> x = [ 3 4 5 ];
>> polyval (p,x)

ans =

       197.00       437.00       825.00
```

$$p(x) = 6x^3 + 2x^2 + 4x + 5$$

Plotting polynomial vectors :

```
>> p = [ 6 2 4 5 ];
>> x = -5:0.1:5;
>> y = polyval (p,x);
>> plot (x,y)
```



roots (p)

Calculates the roots of the given polynomial vector.

```
>> p = [ 6 2 4 5 ]

p =

          6.00          2.00          4.00          5.00

>> roots (p)

ans =

          0.23
          0.23
         -0.80
```

$$p(x) = 6x^3 + 2x^2 + 4x + 5$$

poly (r)

Calculates the polynomial vector for the given roots.

```
>> r = [3.5 4.5 6]

r =

         3.50          4.50          6.00

>> poly (r)

ans =

         1.00        -14.00         63.75        -94.50
```

$$p(x) = x^3 - 14x^2 + 63.75x - 94.50$$

polyder (p)

Calculates the derivative of the given polynomial vector.

```
>> p = [ 2 4 6 5 ]

p =

        2.00        4.00        6.00        5.00

>> polyder (p)

ans =

        6.00        8.00        6.00
```

$p(x) = 2x^3 + 4x^2 + 6x + 5$

$p'(x) = 6x^2 + 8x + 6$

polyint (p)

Calculates the analytic integral of the given polynomial vector.

```
>> p = [ 3 4 6 ]

p =

        3.00        4.00        6.00

>> polyint (p)

ans =

        1.00        2.00        6.00           0
```

$$p(x) = 3x^2 + 4x + 6$$

$$\int p(x).dx = x^3 + 2x^2 + 6x$$

conv (p1,p2)

Multiplies the given two polynomial vectors.

```
>> p1 = [ 3 4 ];
>> p2 = [ 2 4 6 ];
>> conv (p1,p2)

ans =

        6.00        20.00        34.00        24.00
```

$p_1(x) = 3x + 4$

$p_2(x) = 2x^2 + 4x + 6$

$p_1(x).p_2(x) = 6x^3 + 20x^2 + 34x + 24$

deconv (p1,p2)

Divides the given two polynomial vectors.

```
>> p1 = [ 2 9 7 -12 ];
>> p2 = [ 1 3 ];
>> [a b] = deconv (p1,p2)

a =

        2.00        3.00       -2.00

b =

           0           0           0       -6.00
```

$p_1(x) = 2x^3 + 9x^2 + 7x - 12$

$p_2(x) = x + 3$

$2x^2 + 3x - 2$

polyfit (x,y,degree)

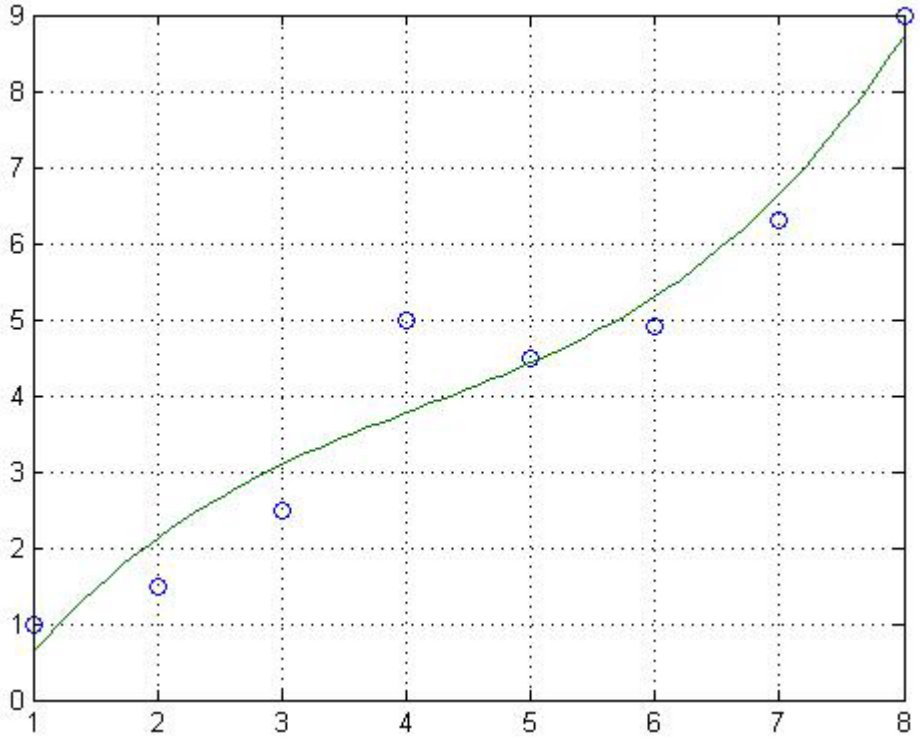
Calculates the polynomial vector of n^{th} degree that represents the data given by x and y. In other words, it is a polynomial curve fitting using the least squares method.

```
>> x = [1 2 3 4 5 6 7 8];
>> y = [1 1.5 2.5 5 4.5 4.9 6.3 9];
>> p = polyfit (x,y,3)

p =

    0.04   -0.52    2.77   -1.67
```

$$p(x) = 0.04x^3 - 0.52x^2 + 2.77x - 1.67$$

Example of curve fitting :

```
>> x = [1 2 3 4 5 6 7 8];
>> y = [1 1.5 2.5 5 4.5 4.9 6.3 9];
>> p = polyfit (x,y,3);
>> xnew = 1:0.1:8;
>> ynew = polyval (p,xnew);
>> plot (x,y,'o',xnew,ynew);
>> grid
```



Curve fitting with different functions :

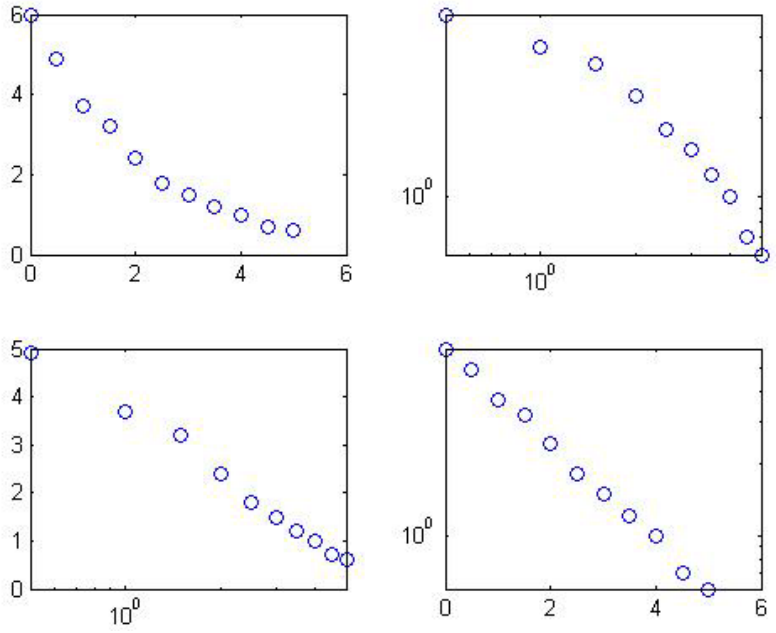
Plotting Style	Function	Method
plot (x,y) x-axis : linear y-axis : linear	Linear Function $y = mx + b$	p = polyfit (x , y , 1)
loglog (x,y) x-axis : logarithmic y-axis : logarithmic	Power Function $y = bx^m$	p = polyfit (log(x) , log(y) , 1)
semilogx (x,y) x-axis : logarithmic y-axis : linear	Logarithmic Function $y = mln(x) + b$ $y = mlog(x) + b$	p = polyfit (log(x) , y , 1) p = polyfit (log10(x) , y , 1)
semilogy (x,y) x-axis : linear y-axis : logarithmic	Exponential Function $y = be^{mx}$ $y = b10^{mx}$	p = polyfit (x , log(y) , 1) p = polyfit (x , log10(y) , 1)

Curve fitting with different functions :

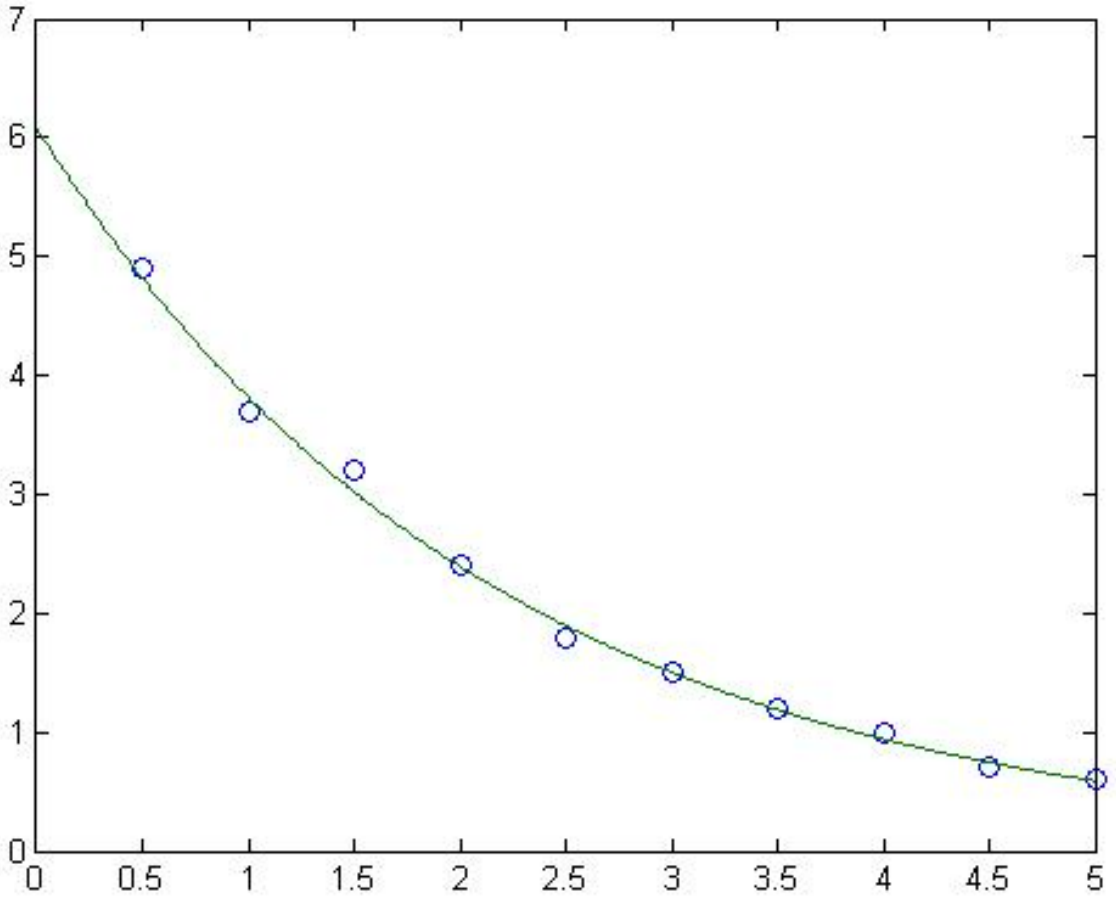
Plotting Style	Function	Coefficients
plot (x,y) x-axis : linear y-axis : linear	Linear Function $y = mx + b$	m = p(1) & b = p(2)
loglog (x,y) x-axis : logarithmic y-axis : logarithmic	Power Function $y = bx^m$	m = p(1) & b = exp(p(2))
semilogx (x,y) x-axis : logarithmic y-axis : linear	Logarithmic Function $y = mln(x) + b$ $y = mlog(x) + b$	m = p(1) & b = p(2) m = p(1) & b = p(2)
semilogy (x,y) x-axis : linear y-axis : logarithmic	Exponential Function $y = be^{mx}$ $y = b10^{mx}$	m = p(1) & b = exp(p(2)) m = p(1) & b = 10^(p(2))

Example of curve fitting with different functions :

```
>> x = [0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5];
>> y = [6 4.9 3.7 3.2 2.4 1.8 1.5 1.2 1 0.7 0.6];
>> subplot (2,2,1),plot (x,y,'o');
>> subplot (2,2,2),loglog (x,y,'o');
>> subplot (2,2,3),semilogx (x,y,'o');
>> subplot (2,2,4),semilogy (x,y,'o');
```



```
>> p = polyfit (x,log(y),1);
>> xnew = 0:0.01:5;
>> ynew = polyval (p,xnew);
>> plot (x,y,'o',xnew,exp(ynew))
```



interp1 (x,y,xnew,method)

Performs a one-dimensional interpolation over the data given by x and y for the values of xnew using the specified method. Methods are ;

nearest : This method accepts the nearest neighbor point.

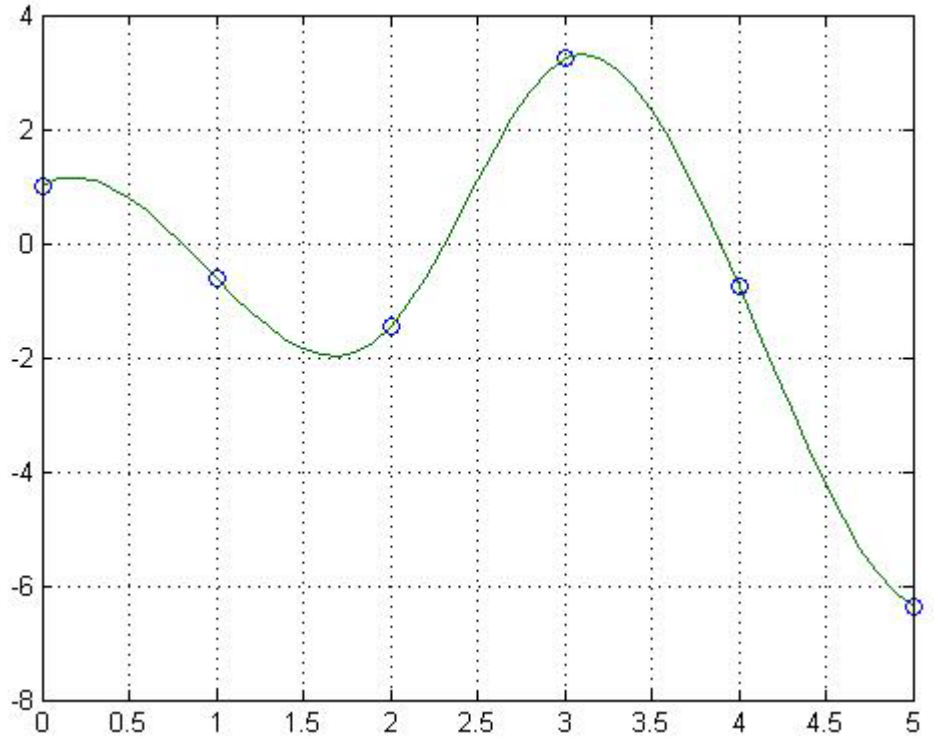
linear : This method performs a linear interpolation.

spline : This method performs a cubic spline polynomial interpolation.

cubic : This method performs a cubic Hermite interpolation.

Example of spline interpolation :

```
>> x = [0 1 2 3 4 5];
>> y = [1 -0.62 -1.47 3.24 -0.74 -6.37];
>> xnew = 0:0.1:5;
>> ynew = interp1 (x,y,xnew,'spline');
>> plot (x,y,'o',xnew,ynew)
>> grid
```



MATLAB
LECTURE NOTES

LESSON 5

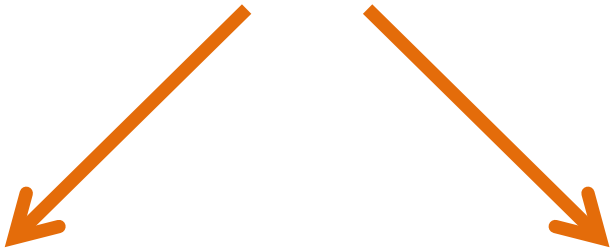
PROGRAMMING

Dr. ADİL YÜCEL

Istanbul Technical University

Department of Mechanical Engineering

m-file



Script File

A script file is a sequence of commands or statements.

Function File

A function file is a sub-program to execute a certain process.

Examples of function files :

```
function result = add (x,y,z)
result = x + y + z;
```

```
function [s c] = SquareAndCube (x)
s = x.^2;
c = x.^3;
```

```
function result = FahToCel (f)
result = (f – 32)*5/9;
```

inline ('...')

Lets you define simple functions without writing external m-files.

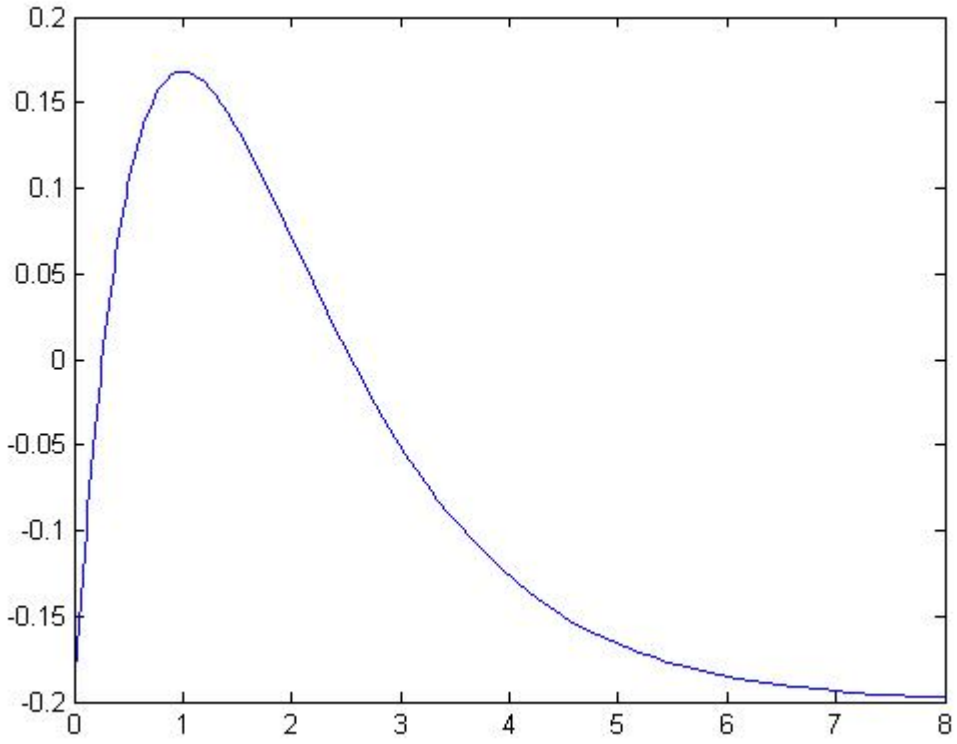
```
>> f = inline ('x^2 + 5*x + 4')

f =

     Inline function:
     f(x) = x^2 + 5*x + 4

>> f (10)

ans =

        154.00
```

fplot ('...' , [a b])

Plots simple functions within the range of a and b.

```
>> fplot ('x*exp(-x) - 0.2', [0 8])
```

$$f(x) = xe^{-x} - 0.2$$



input ('...')

Lets you input data into a variable from command window.

```
>> age = input ('Please enter your age: ')
Please enter your age: 24

age =

        24.00
```

fprintf ('...')

Displays constant data or the values of variables in a certain format. Use the formatting strings given below to format the output.

- Use **%s** to format as a string.
- Use **%d** to format a number as an integer.
- Use **%f** to format a number as a floating-point value.
- Use **%e** to format a number in a scientific notation.
- Use **\n** to insert a newline.
- Use **\t** to insert a tab.

Examples of fprintf :

```
>> grade = 75;
>> fprintf ('My grade is %.2f \n',grade);
My grade is 75.00
```

```
>> grade = 75;
>> fprintf ('Last grade of %s is %d \n','Adil',grade);
Last grade of Adil is 75
```

```
>> fprintf ('Total amount of force is %e Newton \n',1250.45)
Total amount of force is 1.250450e+003 Newton
```

```
>> fprintf ('Prices are %d, %d and %d USD \n',25,40,65);
Prices are 25, 40 and 65 USD
```

if ... elseif ... else ... end

This statement executes the program block which satisfies the corresponding condition.

```
if  <condition>
...
...
elseif  <condition>
...
...
else
...
...
end
```

Program Block

Program Block

Program Block

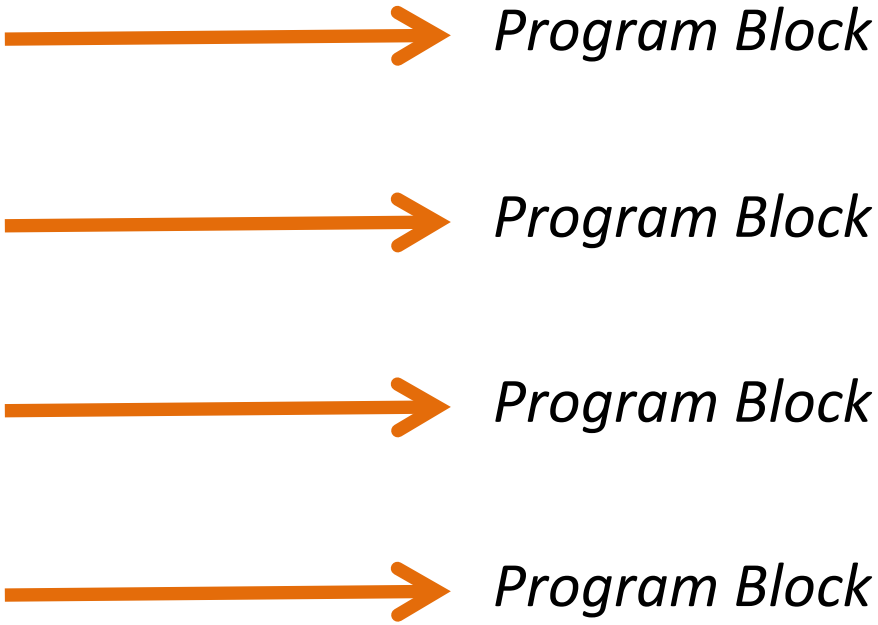
```
if  grade < 50
    disp ('You failed.')
else
    disp ('You passed.')
end
```

```
if  mach < 1
    disp ('Flow is subsonic.')
elseif  mach > 1
    disp ('Flow is supersonic.')
else
    disp ('Flow is sonic.')
end
```

switch ... end

This statement executes the program block which the variable satisfies the corresponding case value.

```
switch  <variable>
    case  <value1>
    ...
    case  <value2>
    ...
    case  <value3>
    ...
    otherwise
    ...
end
```



```
mach = input ('Please enter mach number: ');
switch mach
   case 1
      disp ('Flow is sonic.');
   case 0.5
      disp ('Flow is subsonic.');
   case 1.2
      disp ('Flow is transonic.');
   case 3
      disp ('Flow is supersonic.');
   case 5
      disp ('Flow is hypersonic.');
   otherwise
      disp ('Flow is undefined.');
end
```

for ... end

This statement repeats the program block within the range of the loop statement.

```
for   <expression>
    ...
    ...
    ...
end
```

Program Block

```
for k = 1:1:10
   s = k^2;
   fprintf ('Square of %2d = %d \n',k,s);
end
```

```
Square of  1 = 1
Square of  2 = 4
Square of  3 = 9
Square of  4 = 16
Square of  5 = 25
Square of  6 = 36
Square of  7 = 49
Square of  8 = 64
Square of  9 = 81
Square of 10 = 100
```

continue

This command jumps to the next iteration of the loop.

```
for k = 1:1:10
   if k == 5
      continue
   end
   s = k^2;
   fprintf ('Square of %2d = %d \n',k,s);
end
```

```
Square of  1 = 1
Square of  2 = 4
Square of  3 = 9
Square of  4 = 16
Square of  6 = 36
Square of  7 = 49
Square of  8 = 64
Square of  9 = 81
Square of 10 = 100
```

break

This command stops the loop.

```
for k = 1:1:10
   if k == 5
      break
   end
   s = k^2;
   fprintf ('Square of %2d = %d \n',k,s);
end
```

```
Square of  1 = 1
Square of  2 = 4
Square of  3 = 9
Square of  4 = 16
```

while ... end

This statement repeats the program block as long as the loop expression is true.

```
while   <expression>
    ...
    ...
    ...
end
```

Program Block

```
k = 5;
while k <= 10
   c = k^3;
   fprintf ('Cube of %2d = %d \n',k,c);
   k = k + 1;
end
```

```
Cube of  5 = 125
Cube of  6 = 216
Cube of  7 = 343
Cube of  8 = 512
Cube of  9 = 729
Cube of 10 = 1000
```

MATLAB
LECTURE NOTES

LESSON 6

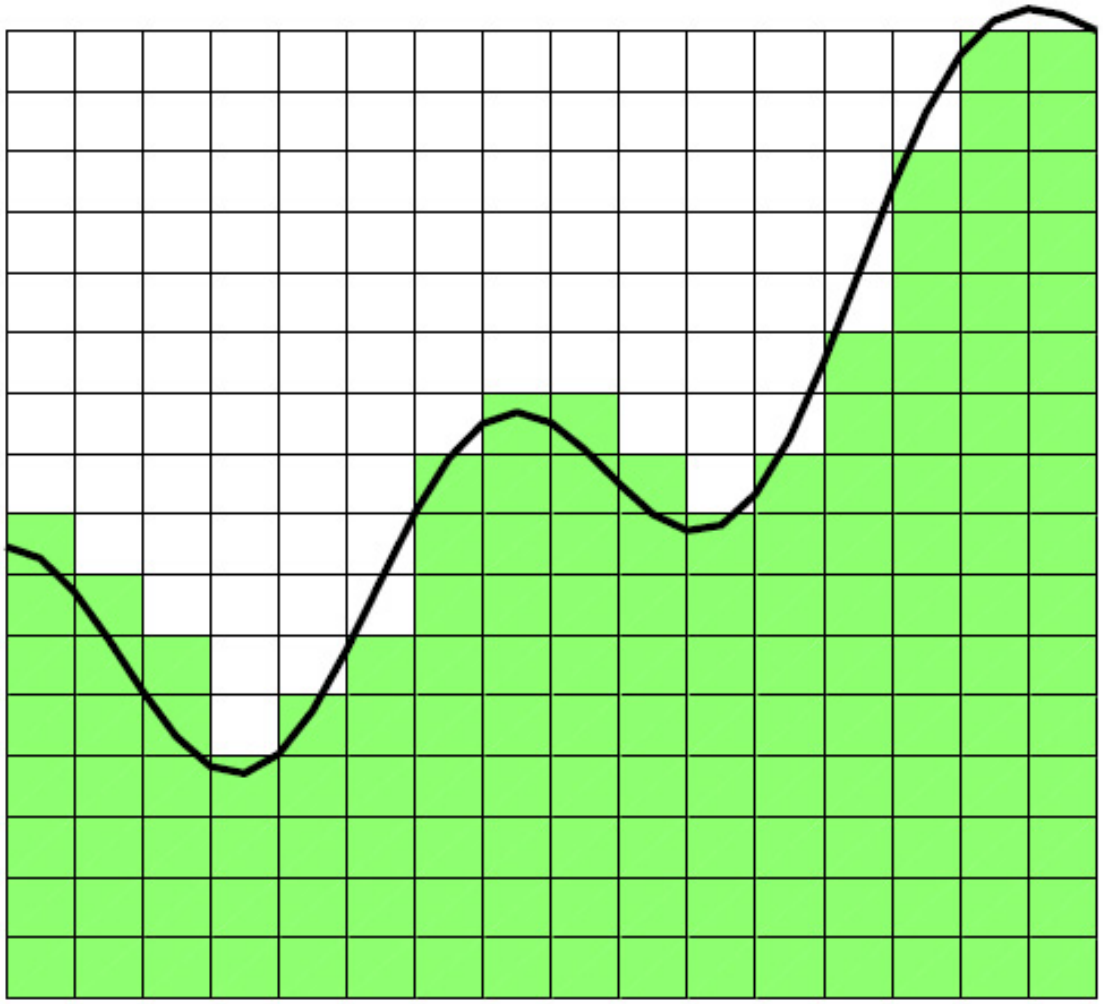
NUMERICAL INTEGRATION

Dr. ADİL YÜCEL

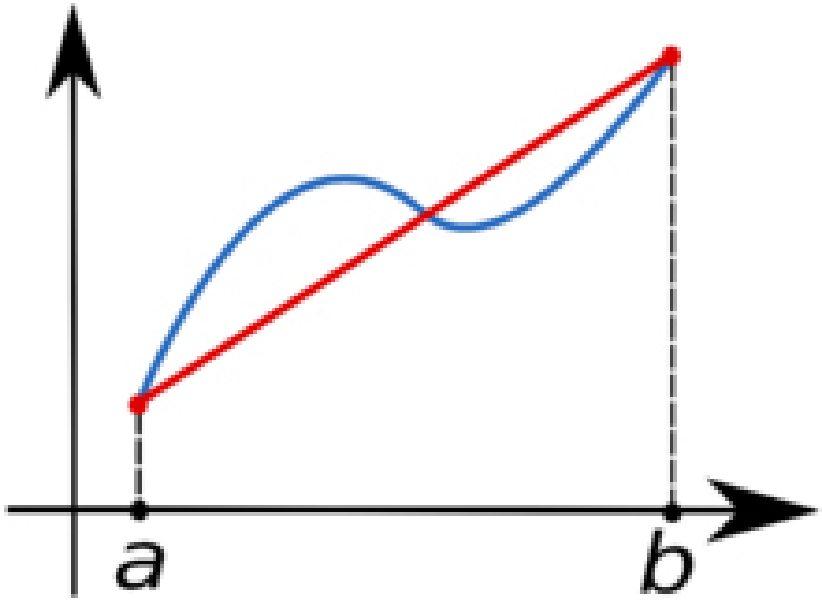
Istanbul Technical University

Department of Mechanical Engineering

Numerical integration (quadrature) is simply the area under the curve of a function.

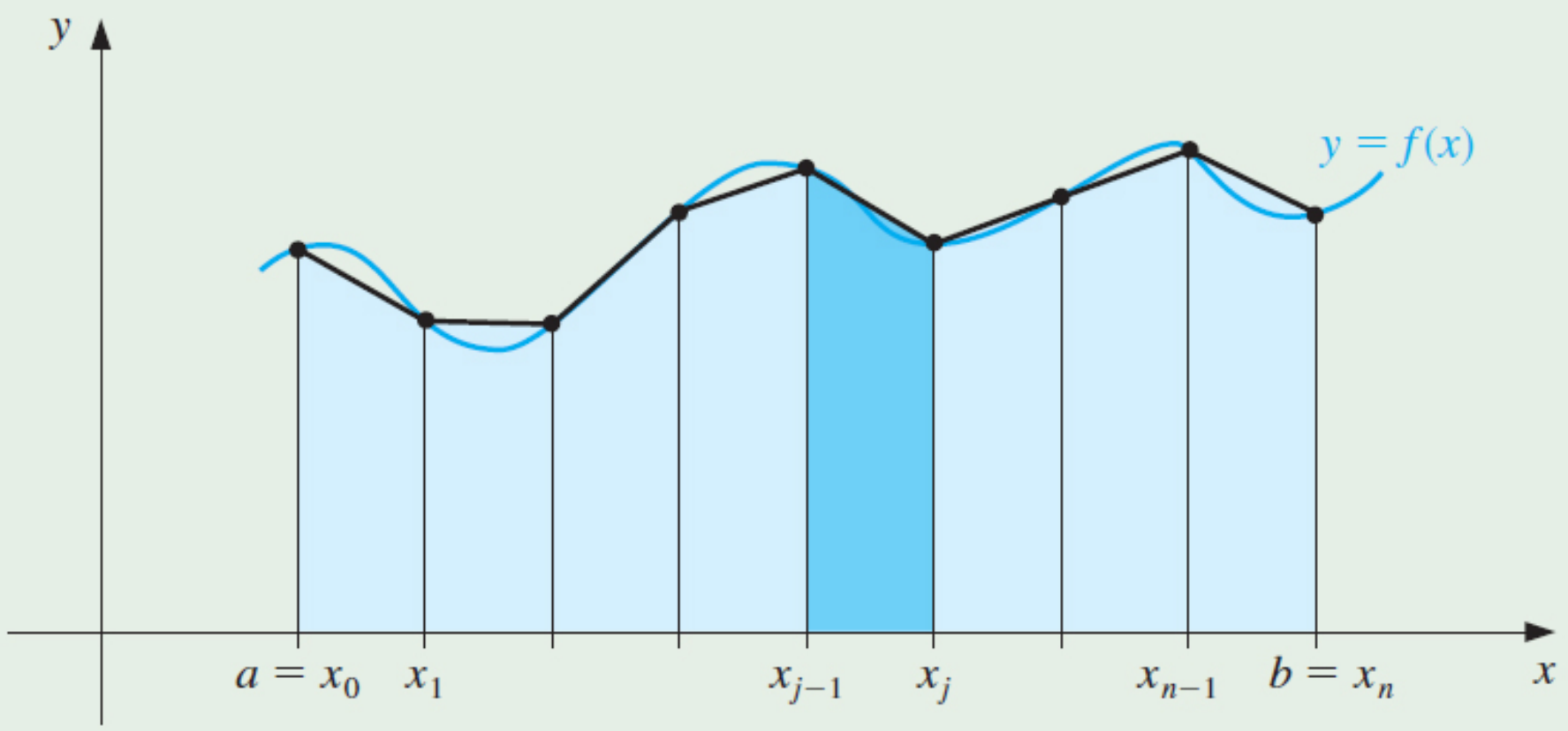


Trapezoidal Rule :



$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\,[\,f(a)+f(b)\,]$$

Trapezoidal Rule :



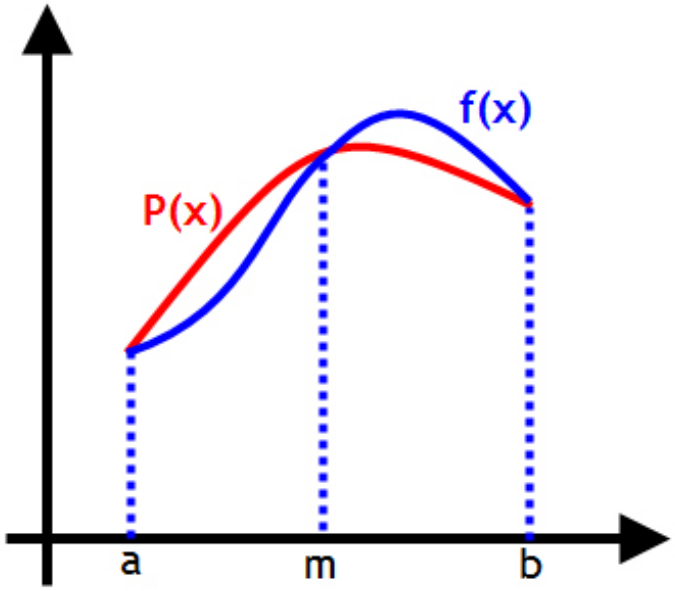
$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left[\, f(x_0) + 2f(x_1) + \ \ldots\ + 2f(x_{n-1}) + f(x_n) \,\right]$$

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left[f(x_0) + 2\sum_{j=1}^{n-1} f(x_j) + f(x_n) \right] \qquad h = \frac{b-a}{n}$$

Trapezoidal Rule Algorithm :

```
segment = ... ;
point = segment + 1;
a = ... ;
b = ... ;
h = (b - a) / (point – 1);
x = a:h:b;
y = ... ;
I = (h/2) * ( y(1) + 2*sum( y(2:point–1) ) + y(point) )
```

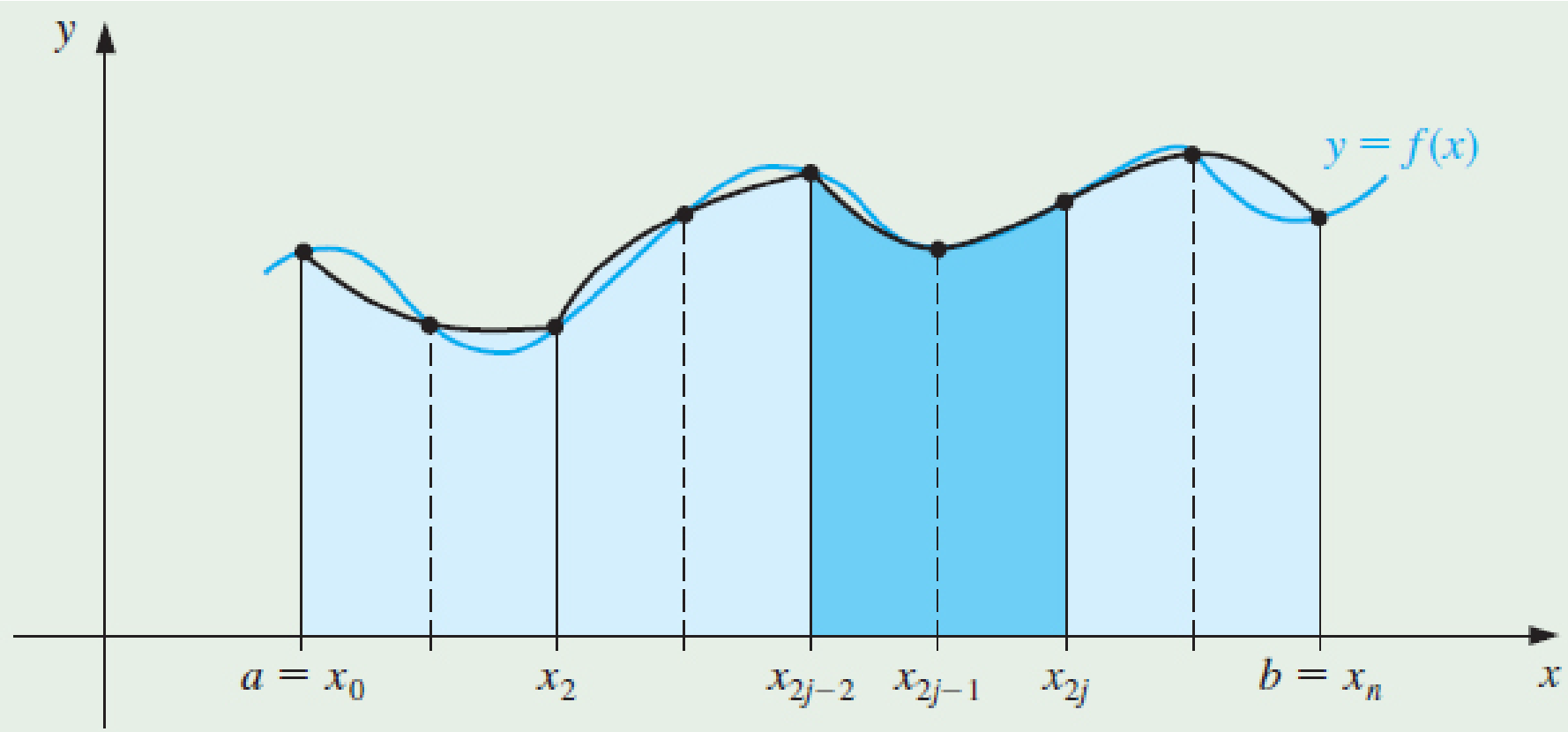
Simpson Rule :



$$m = \frac{a+b}{2}$$

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\,[f(a) + 4f(m) + f(b)]$$

Simpson Rule :



$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right]$$

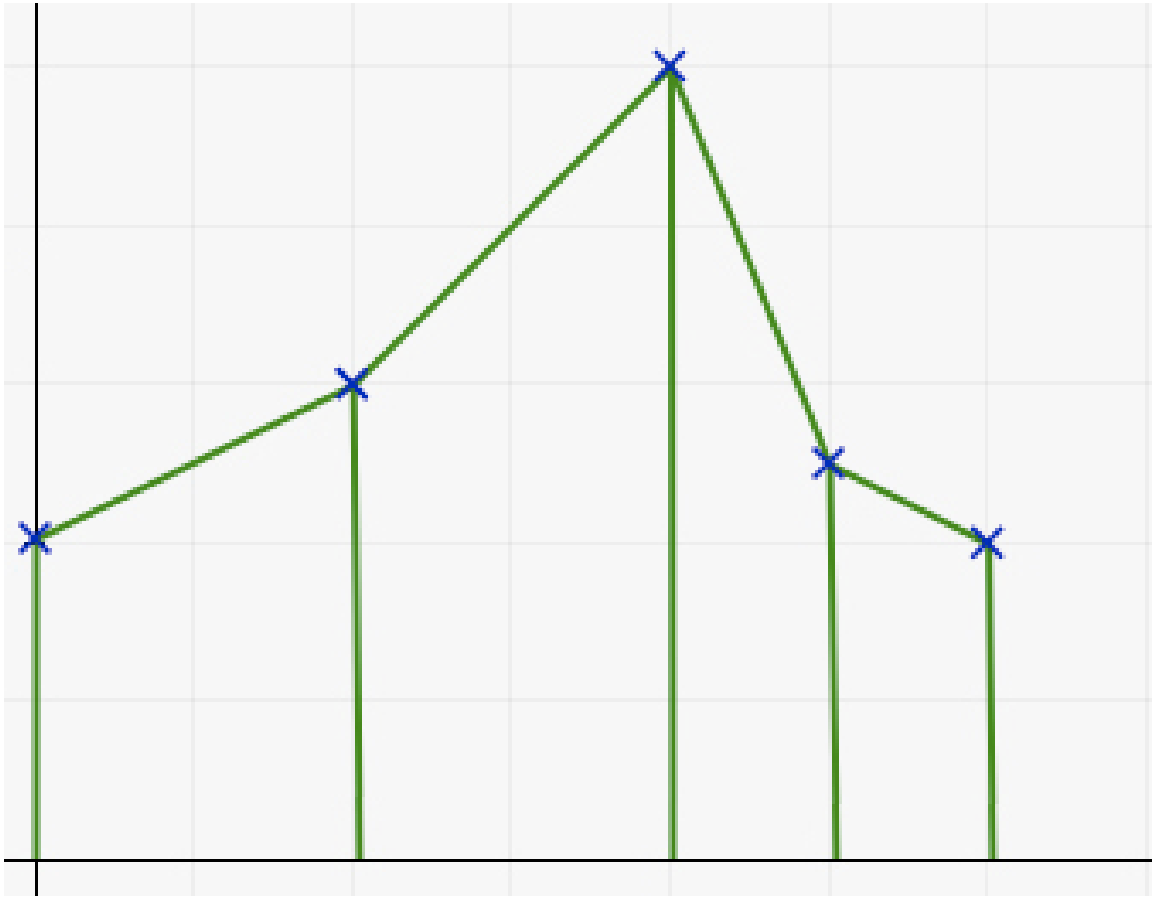
$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + 4\sum_{j=1,3,5\ldots}^{n-1} f(x_j) + 2\sum_{j=2,4,6\ldots}^{n-2} f(x_j) + f(x_n)\right] \qquad h = \frac{b-a}{n}$$

Simpson Rule Algorithm :

```
segment = ... ;
point = 2*segment + 1;
a = ... ;
b = ... ;
h = (b - a) / (point – 1);
x = a:h:b;
y = ... ;
I = (h/3) * ( y(1) + 4*sum( y(2:2:point–1) )
                   + 2*sum( y(3:2:point–2) )
                   + y(point) )
```

trapz (x,y)

Calculates the area under the trapezoids created by x-y pairs.



quad ('...',a,b)

Numerically integrates the given function in the range of a and b using the adaptive simpson method.

```
>> quad ('x.^2',0,8)

ans =

        170.67

>> quad ('cos(x).^2 + x',0,pi/2)

ans =

          2.02
```

MATLAB
LECTURE NOTES

LESSON 7

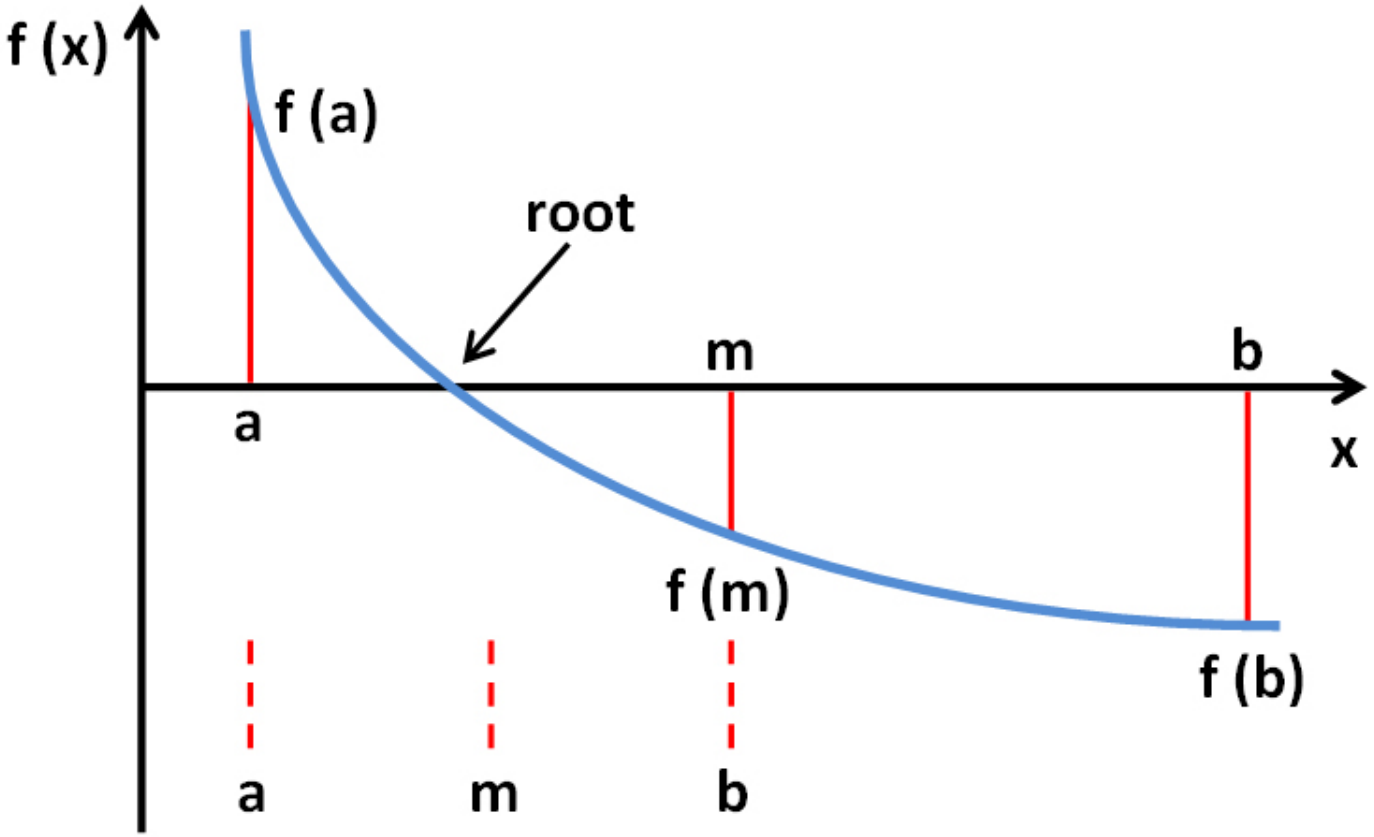
ROOT FINDING ALGORITHMS

Dr. ADİL YÜCEL

Istanbul Technical University

Department of Mechanical Engineering

Bisection Method :



$$x_j = m = \frac{a+b}{2}$$

if f(a) . f(m) > 0 → a = m

if f(b) . f(m) > 0 → b = m

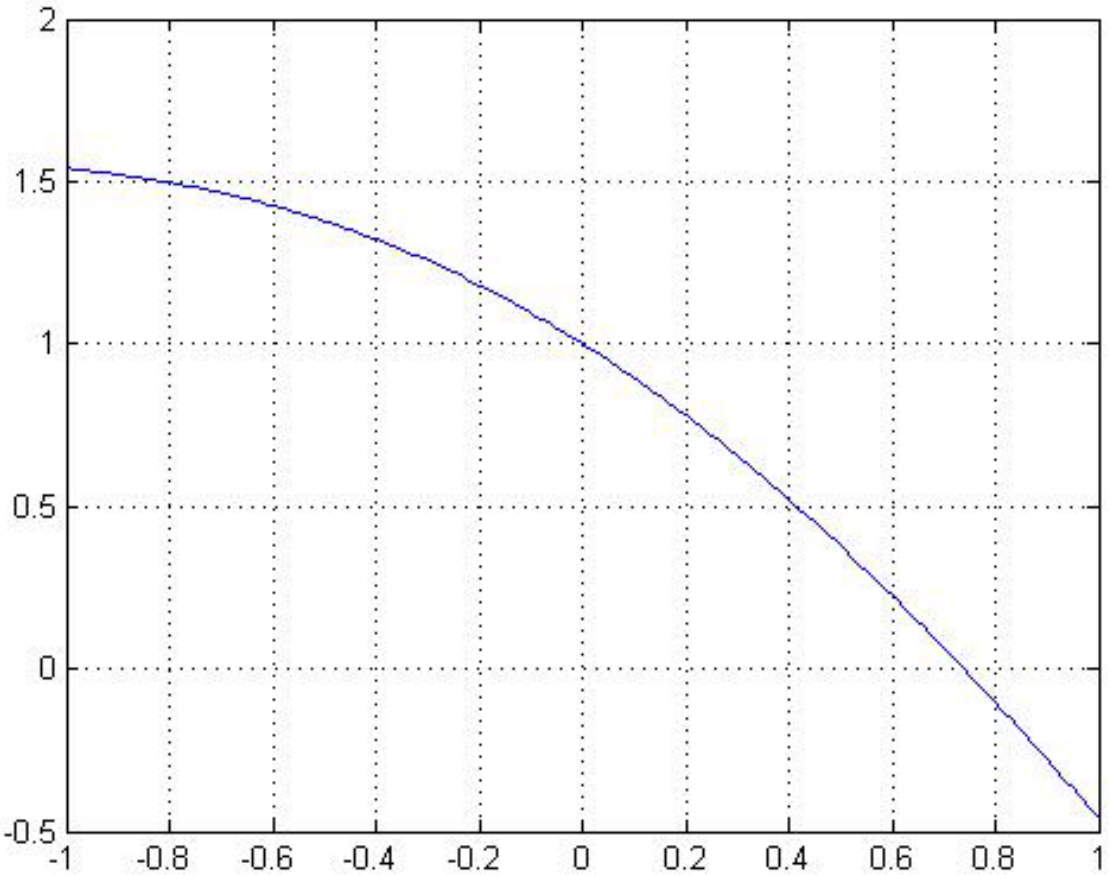
Bisection Method Algorithm :

```
a = ... ;
b = ... ;
m = (a + b) / 2;
while  abs ( f(m) ) > 1e-6
        if  f(a) * f(m) > 0
                a = m;
        else
                b = m;
        end
        m = (a + b) / 2;
end
m
```

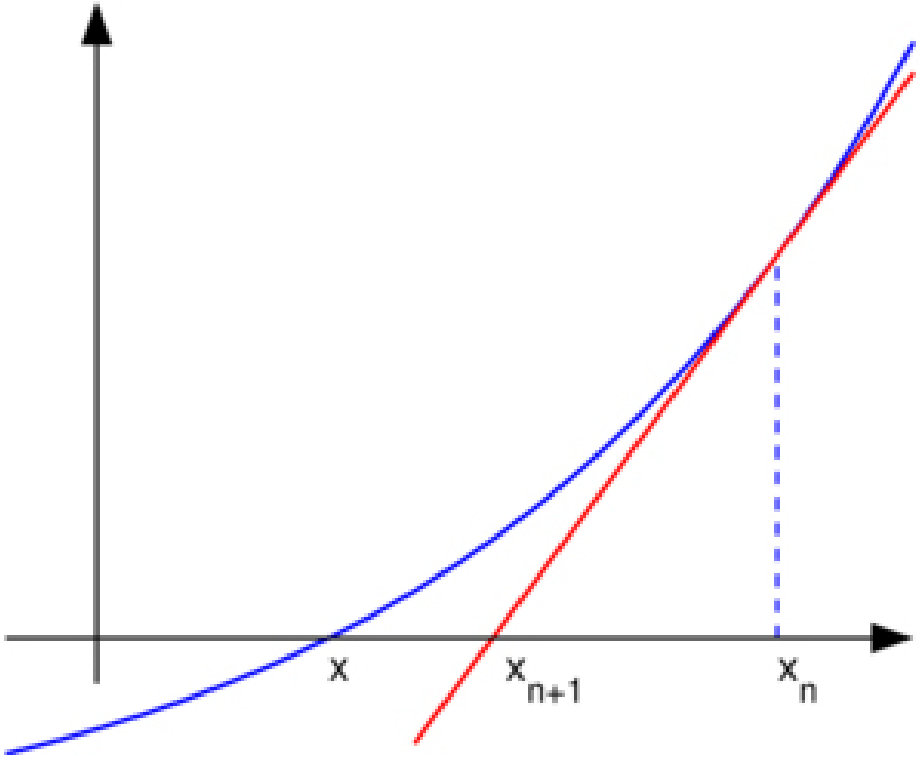
Bisection Method Example:

```
f = inline ('cos(x) – x');
x = –1:0.01:1;
y = f(x);
plot (x,y)
grid

a = –1 ;
b = 1 ;
m = (a + b) / 2;
while  abs ( f(m) ) > 1e-6
        if  f(a) * f(m) > 0
                a = m;
        else
                b = m;
        end
        m = (a + b) / 2;
end
m
```

$$f(x) = cosx - x = 0$$



Newton's Method :



$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Newton's Method Algorithm :

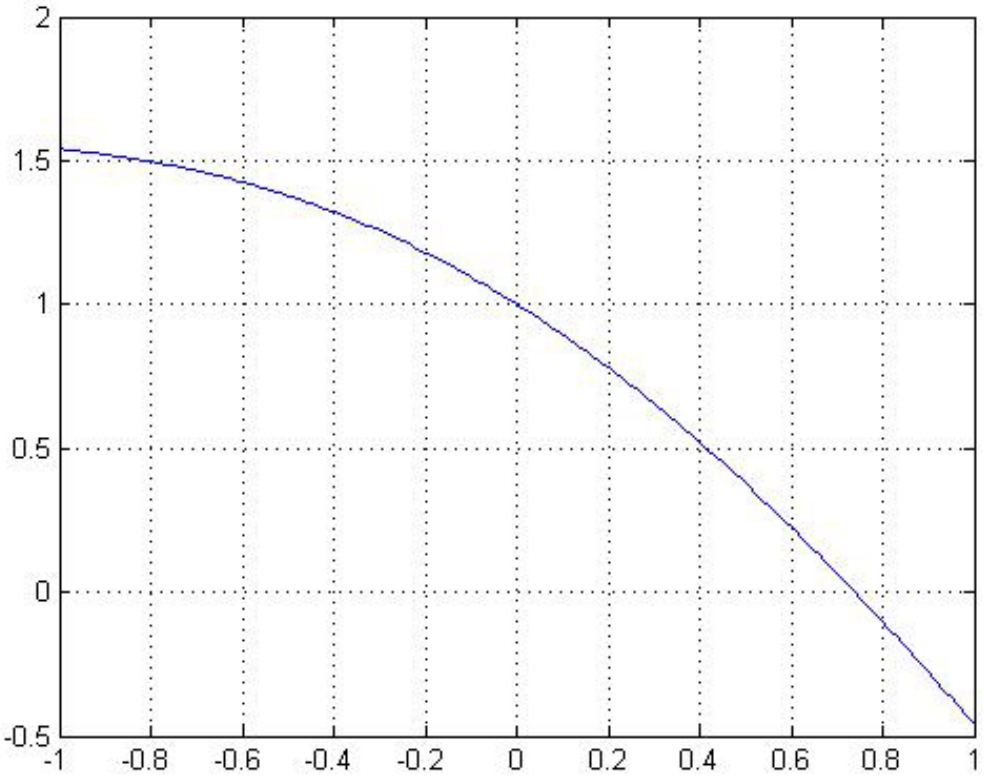
```
x0 = ... ;
while  abs ( f(x0) ) > 1e-6
        x1 = x0 – ( f(x0) / fd(x0) );
        x0 = x1;
end
x0
```

Newton's Method Example:

```
f = inline ('cos(x) – x');
fd = inline ('– sin(x) – 1');
x = –1:0.01:1;
y = f(x);
plot (x,y)
grid

x0 = 0.2;
while  abs ( f(x0) ) > 1e-6
        x1 = x0 – ( f(x0) / fd(x0) );
        x0 = x1;
end
x0
```

$$f(x) = cosx - x$$

$$fd(x) = -sinx - 1$$



Fixed Point Iteration Method :

$$f(x) = x^2 - 2x - 3$$

$$x = \sqrt{2x + 3} = g(x)$$

$$x_{n+1} = \sqrt{2x_n + 3} = g(x_n)$$

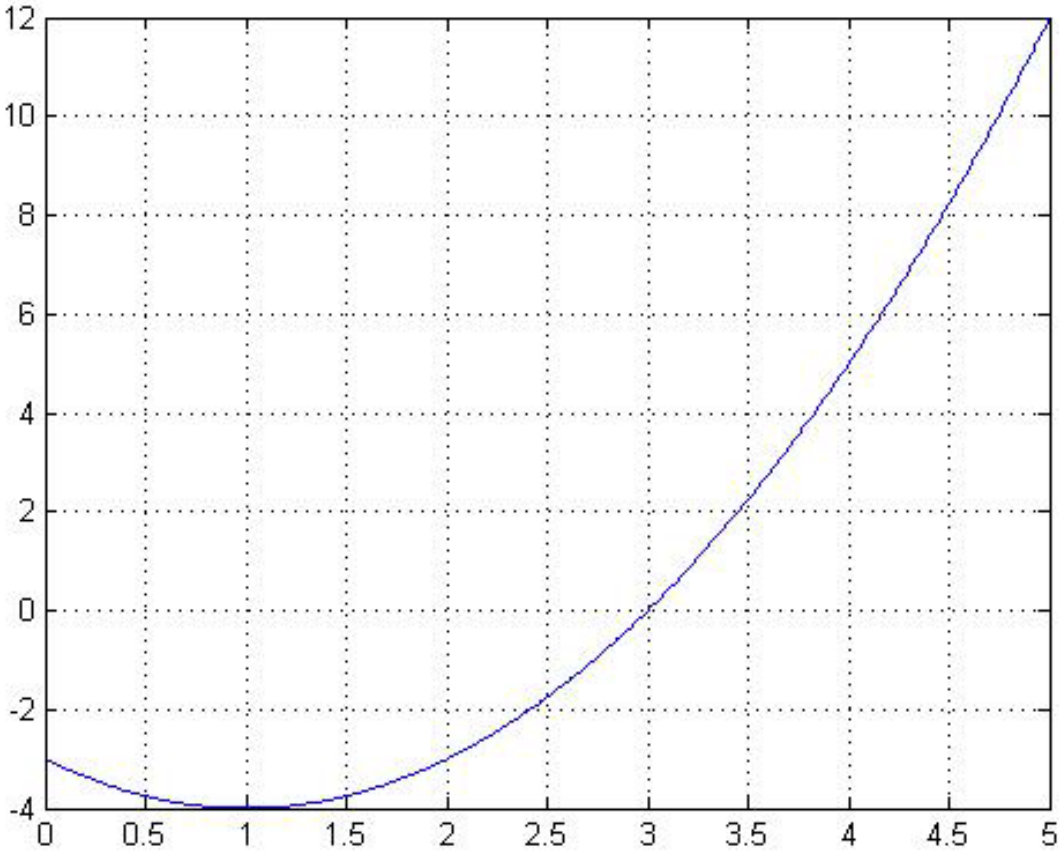
x_{n+1} converges to a root.

Fixed Point Iteration Method Algorithm :

```
x0 = ... ;
tol = 1;
while  abs ( tol ) > 1e-6
        x1 = g(x0);
        tol = x0 – x1;
        x0 = x1;
end
x0
```

Fixed Point Iteration Method Example:

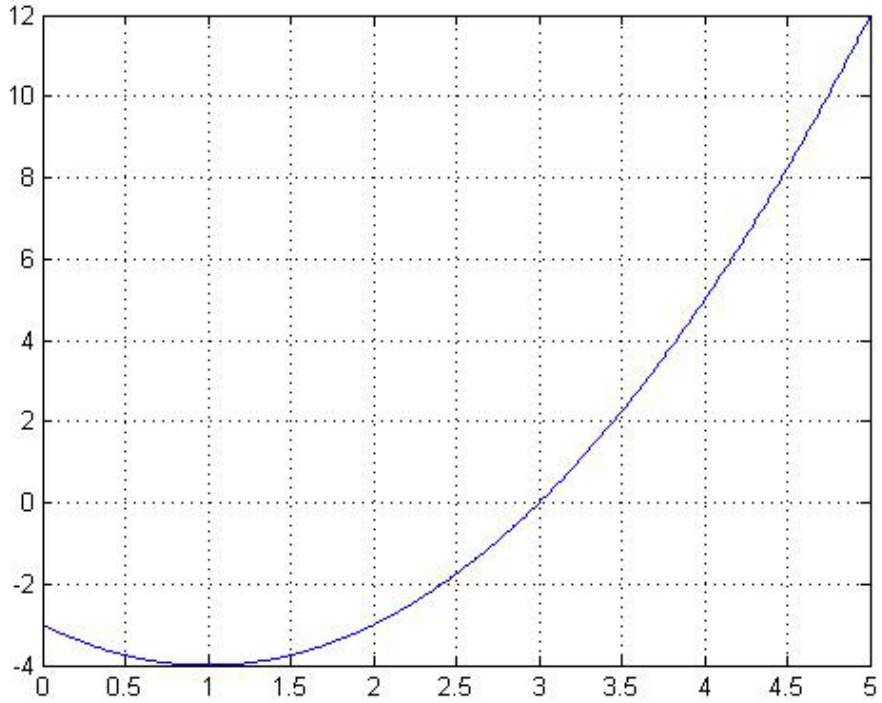
```
f = inline ('x.^2 – 2*x – 3');
g = inline ('sqrt (2*x + 3)');
x = 0:0.01:5;
y = f(x);
plot (x,y)
grid

x0 = 0;
tol = 1;
while  abs ( tol ) > 1e-6
        x1 = g(x0);
        tol = x0 – x1;
        x0 = x1;
end
x0
```



fzero (f,x0)

Finds the root of the function f starting from an initial guess of x0.

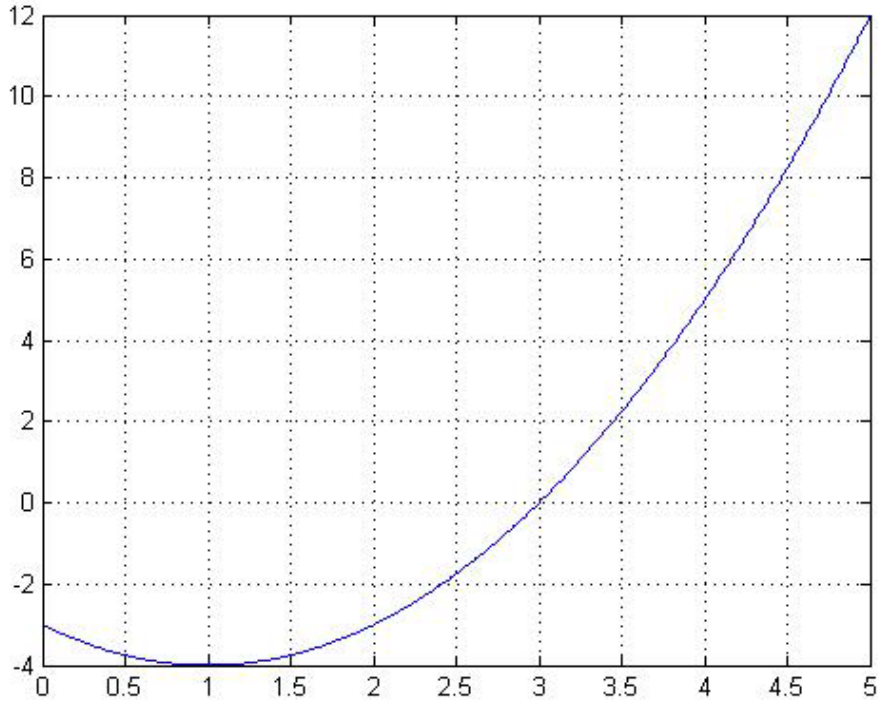
```
>> f = inline ('x.^2 - 2*x - 3');
>> r = fzero (f,2)

r =

     3
```



fminbnd (f,a,b)

Finds the local minimizer of function f in the interval a and b.

```
>> f = inline ('x.^2 - 2*x - 3');
>> m = fminbnd (f,0,5)

m =

     1
```



MATLAB
LECTURE NOTES

LESSON 8

SYMBOLIC ALGEBRA

Dr. ADİL YÜCEL

Istanbul Technical University

Department of Mechanical Engineering

syms ...

Used to create symbolic variables.

```
>> syms x y z
>> f = x^2 + 3*y^3 + z^4

f =

x^2 + 3*y^3 + z^4
```

sym ('...')

Used to create symbolic variables, expressions and equations.

```
>> f = sym('x^3 - 2*y^2 + 3*a')

f =

x^3 - 2*y^2 + 3*a

>> E = sym('m*c^2')

E =

c^2*m
```

findsym (f)

Determines the symbolic variables present in an expression.

```
>> f = sym('x^3 - 2*y^2 + 3*a')

f =

x^3 - 2*y^2 + 3*a

>> findsym (f)

ans =

a,x,y
```

subs (S,old,new)

Substitutes a variable with a new variable or a new value in a symbolic expression.

```
>> S = sym ('x^2 + 3*y^3 + z^4')

S =

x^2 + 3*y^3 + z^4

>> subs (S,x,3)

ans =

3*y^3 + z^4 + 9
```

collect (S)

Collects the coefficients in a symbolic expression.

```
>> syms x
>> S = x^2*y + y*x - x^2 - 2*x

S =

x*y - 2*x + x^2*y - x^2

>> collect (S)

ans =

(y - 1)*x^2 + (y - 2)*x
```

factor (S)

Gives the prime factors of a symbolic expression.

```
>> syms x
>> S = (x^9) - 1

S =

x^9 - 1

>> factor (S)

ans =

(x - 1)*(x^2 + x + 1)*(x^6 + x^3 + 1)
```

expand (S)

Expands a symbolic expression.

```
>> syms x
>> S = (x+1)^3

S =

(x + 1)^3

>> expand (S)

ans =

x^3 + 3*x^2 + 3*x + 1
```

simplify (S)

Simplifies a symbolic expression.

```
>> syms x
>> S = (2*((x+3)^2)) / (x^2 + 6*x + 9)

S =

(2*(x + 3)^2)/(x^2 + 6*x + 9)

>> simplify (S)

ans =

2
```

$$\frac{2\,(x+3)^2}{x^2+6x+9} = \frac{2(x^2+6x+9)}{x^2+6x+9} = 2$$

poly2sym (p)

Converts a polynomial vector to a symbolic expression.

```
>> p = [2 3 5]

p =

     2     3     5

>> S = poly2sym (p)

S =

2*x^2 + 3*x + 5
```

sym2poly (S)

Converts a symbolic expression to a polynomial vector.

```
>> S = sym ('x^3 + 5*x^2 + 10')

S =

x^3 + 5*x^2 + 10

>> p = sym2poly (S)

p =

     1     5     0    10
```

diff (S)

Calculates the derivative of the symbolic expression.

```
>> syms x
>> S = x^3 - cos(x)

S =

x^3 - cos(x)

>> D = diff (S)

D =

sin(x) + 3*x^2
```

diff (S,u)

Calculates the derivative of the symbolic expression with respect to a variable.

```
>> syms x y
>> S = x^3 - y^2

S =

x^3 - y^2

>> D = diff (S,y)

D =

-2*y
```

int (S)

Integrates the symbolic expression.

```
>> syms x
>> S = x^3 - cos(x)

S =

x^3 - cos(x)

>> I = int (S)

I =

x^4/4 - sin(x)
```

int (S,u)

Integrates the symbolic expression with respect to a variable.

```
>> syms x y
>> S = x^3 - y^2

S =

x^3 - y^2

>> I = int (S,y)

I =

x^3*y - y^3/3
```

int (S,u,a,b)

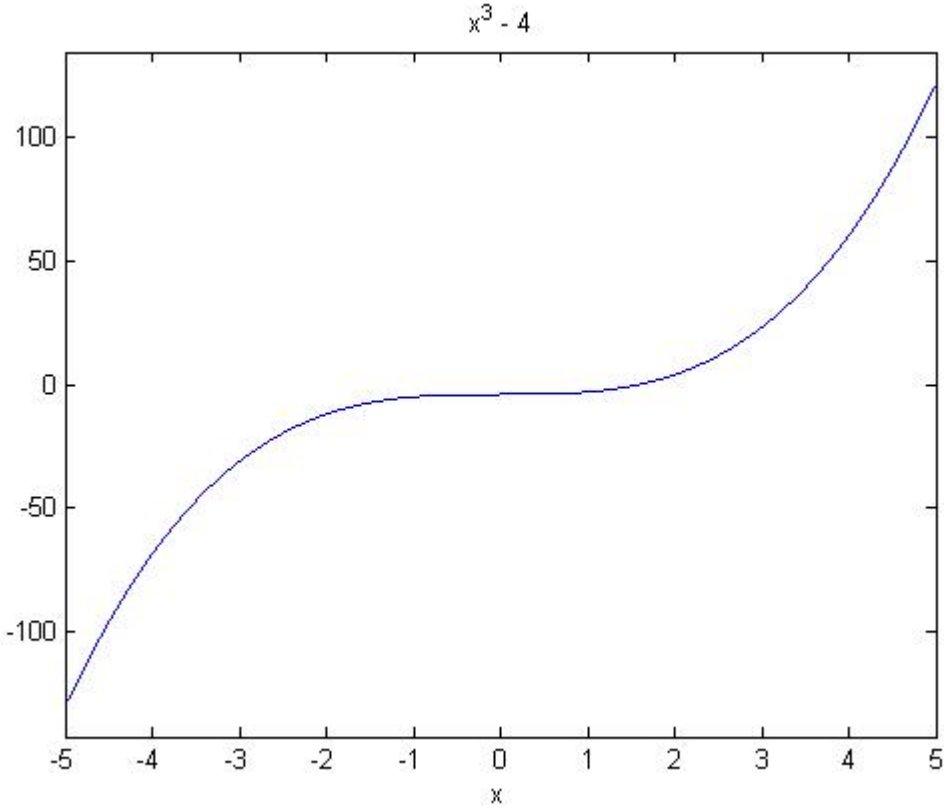
Integrates the symbolic expression with respect to a variable within the range of a and b.

```
>> syms x y
>> S = x^3 - y^2

S =

x^3 - y^2

>> I = int (S,x,0,2)

I =

4 - 2*y^2
```

ezplot (S, [xmin xmax])

Plots a symbolic expression in the given range. If the range is not defined, the default range is -2π to +2π.

```
>> S = sym ('x^3 - 4')

S =

x^3 - 4

>> ezplot (S,[-5 +5])
```



solve ('...','...','...')

Solves the given equation or the system of equations.

```
>> [x y] = solve ('x^2 + y^2 = 25','x + y = 7')

x =

  4
  3


y =

  3
  4
```

$x^2 + y^2 = 25$ $x + y = 7$

dsolve ('...','...','...')

Solves the given differential equation or the system of differential equations.

```
>> dsolve ('Dy = x + y','x')
```

→ $\frac{dy}{dx} = x + y$

```
ans =

C15*exp(x) - x - 1
```

```
>> dsolve ('D2y - 2*y - x = 0','x')
```

→ $\frac{d^2y}{dx^2} - 2y - x = 0$

```
ans =

C11*exp(2^(1/2)*x) - x/2 + C12/exp(2^(1/2)*x)
```

MATLAB
LECTURE NOTES

LESSON 9

THREE DIMENSIONAL PLOTS

Dr. ADİL YÜCEL

Istanbul Technical University

Department of Mechanical Engineering

meshgrid (rangeX, rangeY)

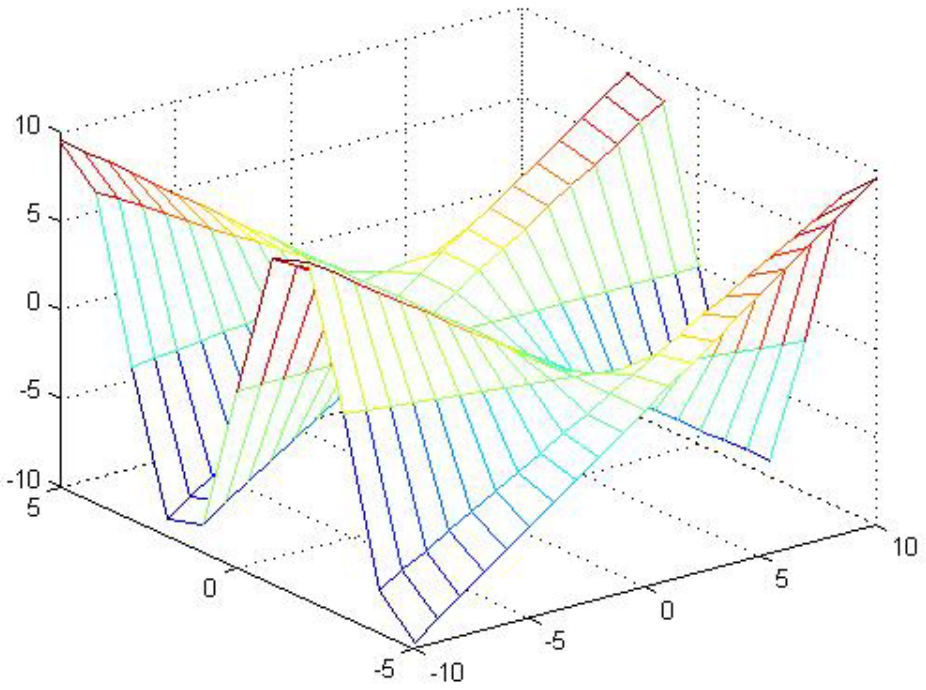
Used to create the grid within the range of rangeX and rangeY.

```
rangeX = -10:1:10;
rangeY = -5:1:5;
[X, Y] = meshgrid (rangeX, rangeY);
Z = X .* sin(Y);
```

mesh (X, Y, Z)

Used to plot X, Y and Z values using a mesh view.

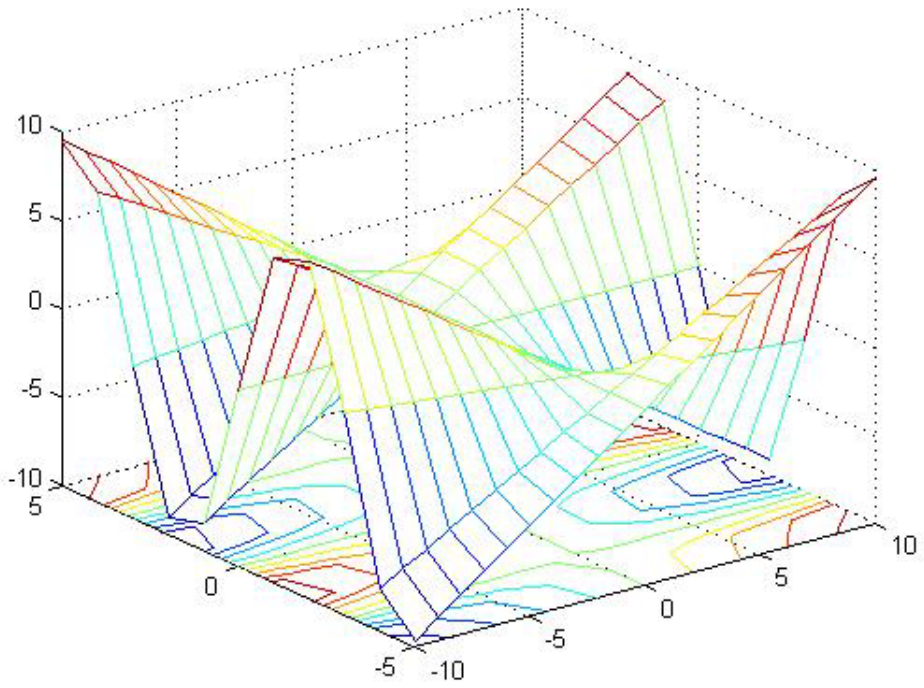
```
rangeX = -10:1:10;
rangeY = -5:1:5;
[X, Y] = meshgrid (rangeX, rangeY);
Z = X .* sin(Y);
mesh (X, Y, Z);
```



meshc (X, Y, Z)

Used to plot X, Y and Z values using a mesh view with projections.

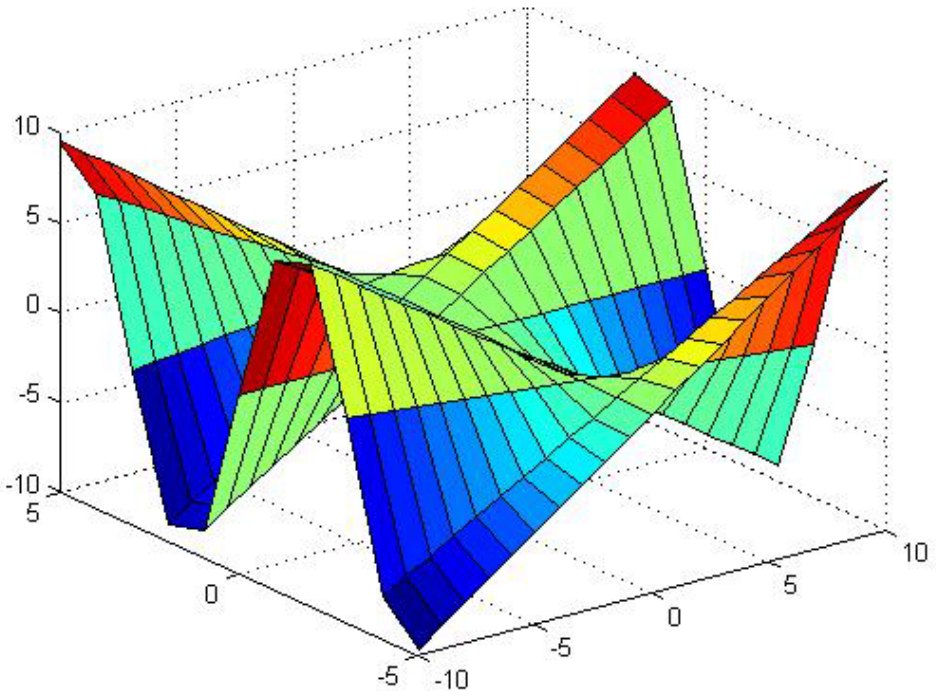
```
rangeX = -10:1:10;
rangeY = -5:1:5;
[X, Y] = meshgrid (rangeX, rangeY);
Z = X .* sin(Y);
meshc (X, Y, Z);
```



surf (X, Y, Z)

Used to plot X, Y and Z values using a surface mesh view.

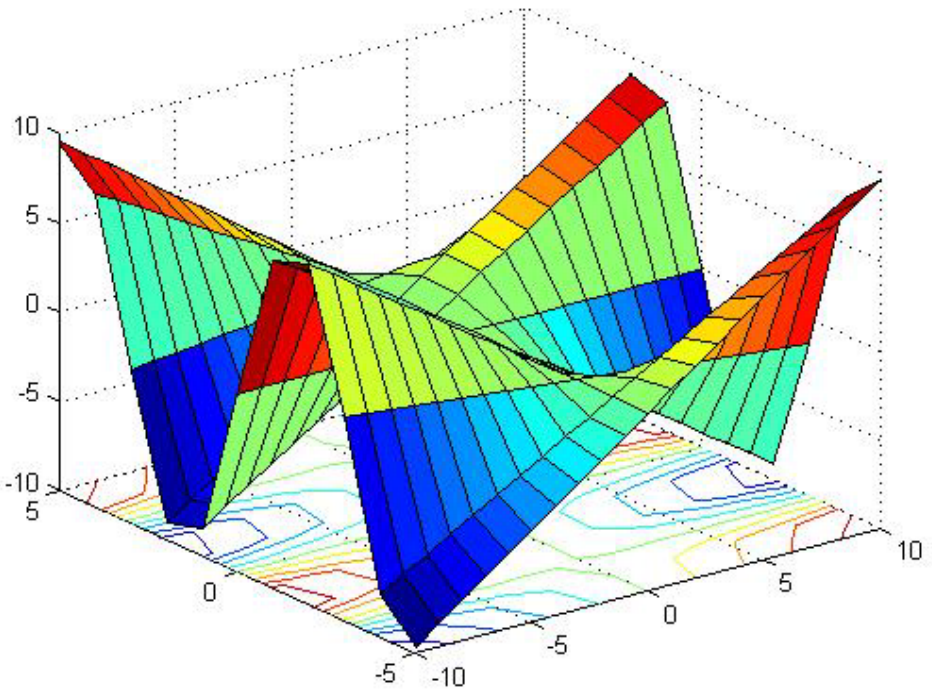
```
rangeX = -10:1:10;
rangeY = -5:1:5;
[X, Y] = meshgrid (rangeX, rangeY);
Z = X .* sin(Y);
surf (X, Y, Z);
```



surfc (X, Y, Z)

Used to plot X, Y and Z values using a surface mesh view with projections.

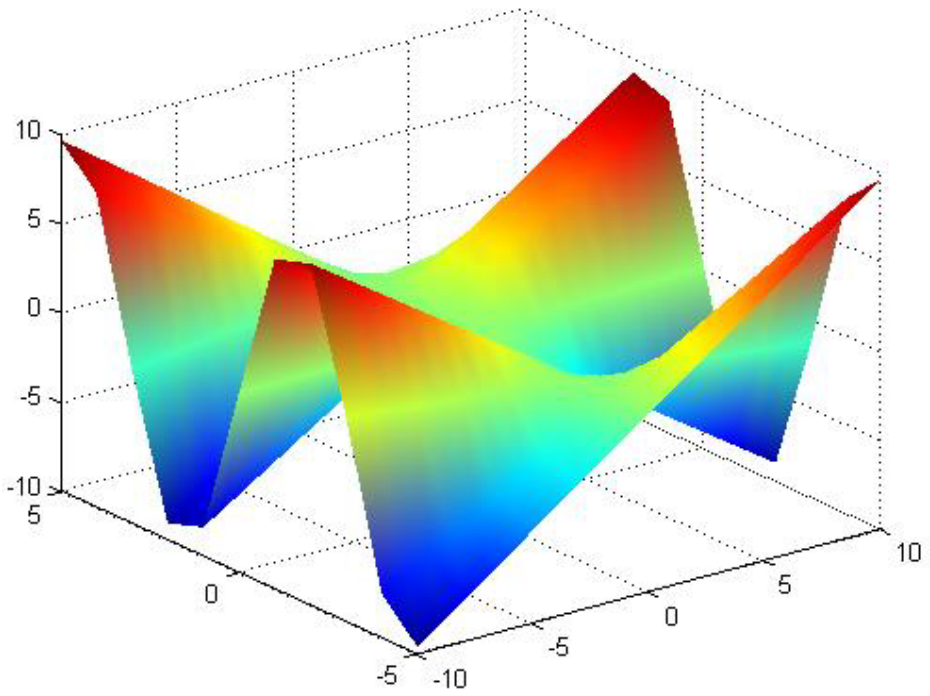
```
rangeX = -10:1:10;
rangeY = -5:1:5;
[X, Y] = meshgrid (rangeX, rangeY);
Z = X .* sin(Y);
surfc (X, Y, Z);
```



shading interp

Plots the graphic with interpolated colors.

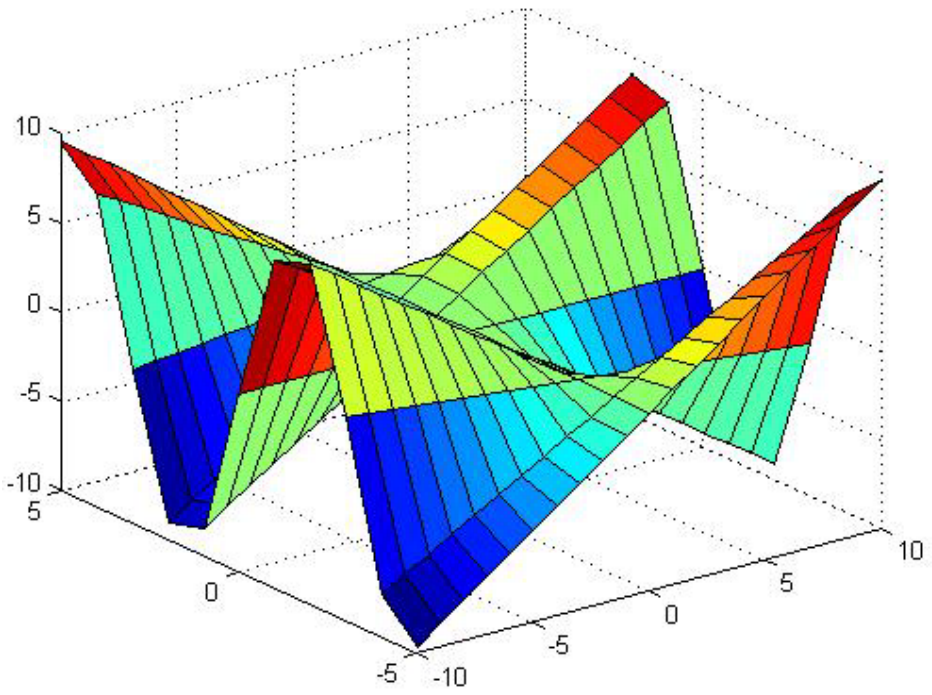
```
rangeX = -10:1:10;
rangeY = -5:1:5;
[X, Y] = meshgrid (rangeX, rangeY);
Z = X .* sin(Y);
surf (X, Y, Z);
shading interp
```



shading faceted

Plots the graphic with meshed colors.

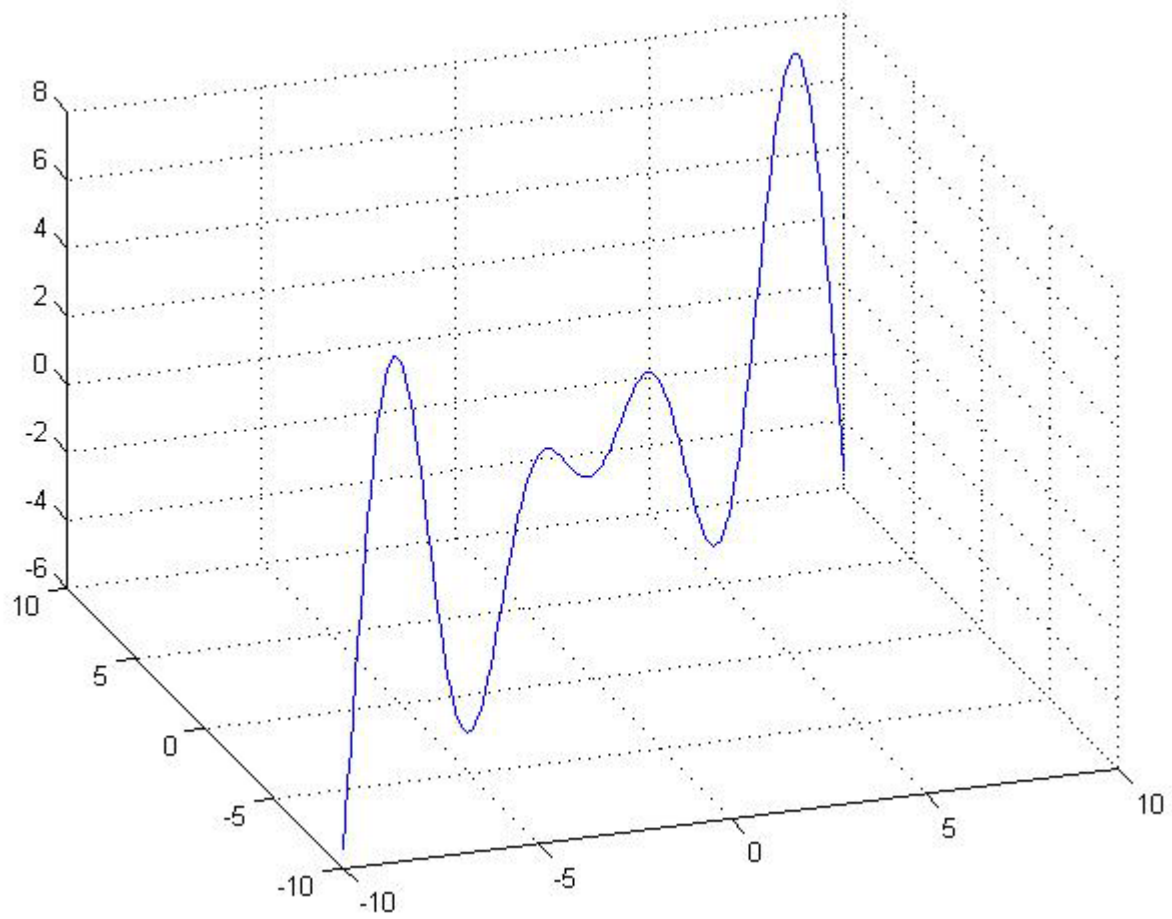
```
rangeX = -10:1:10;
rangeY = -5:1:5;
[X, Y] = meshgrid (rangeX, rangeY);
Z = X .* sin(Y);
surf (X, Y, Z);
shading faceted
```



plot3 (X, Y, Z)

Used to create a simple plot in three dimensions.

```
X = -10:0.1:10;
Y = -10:0.1:10;
Z = X .* sin(Y);
plot3 (X, Y, Z);
grid
```



Dr. ADİL YÜCEL

Istanbul Technical University

Department of Mechanical Engineering

adil.yucel@itu.edu.tr – adil.yucel@gmail.com

www.adilyucel.com