Mathematics

By

Dr. R. R. Manza

Objectives

- Introduction
- Matrices
- Linear Equations
- Factorization
- Eigenvalues
- Polynomials
- Interpolation
- Data Analysis
- Polynomial Regression
- Basic Fitting
- Fourier approximation
- Integration and Differentiation
- Differentiation Equation
- Summary

Introduction

This chapter provides an assortment of functions for performing mathematical operations and analyzing data. It consists of matrices linear algebra polynomials, interpolation data analysis, statistics, function functions, differential equations and non-double data types. Matrices topic covers different operations supported in MATLAB.

Linear algebra deals with matrices arithmetic, leaner equations, eigenvalue, singular values and matrix factorization. Whereas polynomials and interpolation topic includes functions for standard polynomial operations, like polynomial roots, evaluation and differentiation. Some additional topics are also covered in this chapter like curve fitting and partial fraction expansion. Data analysis and statistics describes methods of array for data analysis, simple descriptive statistics functions, data preprocessing task. This topic covers regression, curve fitting, data filtering, and Fast Fourier Transforms (FFT).

Matrices

Two dimensional arrays of either real or complex numbers is called matrix.

Linear algebra describes many operations of matrix. Those operations can be easily handled in MATLAB, for example matrix arithmetic, linear equations eigenvalues, singular values, matrix factorization and so on.

Cont..

Identity Matrix:

Identity matrix is a matrix of various sizes having ones on the main diagonal and zeros on remaining positions. The identity matrix is denoted by letter I. The identity matrix having its property that AI=A and IA=A provided that the dimensions are compatible.

```
>> I=eye(4)

I =

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

>> A=magic(4)

A =

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

>> C=A*I

C =

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

> D=I*A

D =

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

In the above example elements of matrix A and elements of matrix C and D are the same. The function *eye* is used to create rectangular or square identity matrix.

Cont..

Kronecker Tensor product: The *kron* function is used to determine kronecker tensor product of two matrices in MATLAB. The resultant matrix of this function will be double in size than the matrix used as an input. This function is used to build repeated copies of small matrices of zeros and ones. For example,

```
>> A=magic(3)

A =

     8     1     6
     3     5     7
     4     9     2

>> I=eye(2)

I =

     1     0
     0     1

>> C=kron(A,I)

C =

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

>> C=kron(I,A)

C =

     8     1     6     0     0     0
     3     5     7     0     0     0
     4     9     2     0     0     0
     0     0     0     8     1     6
     0     0     0     3     5     7
     0     0     0     4     9     2
```

Cont..

Vector and Matrix norms: The *norm* function is used to compute the norm of a vector or matrix. For example *norm*(x,p) is used to compute the p-norm of vector x. which is defined by any value of p greater than one. The most common values of p are one, two and infinity. The default value of p is two, which corresponds to Euclidean length.

```
>> x=1:2:10
x =
    1    3    5    7    9
>> norm(x)
ans =
   12.8452
>> norm(x,1)
ans =
    25
>> norm(x,inf)

ans =

    9
```

Cont..

Inverse and Determinant of Matrix:

A matrix is singular if and only if its determinant is 0 and it does not have inverse matrix. Nonsingular matrices are sometimes also called regular matrices, and non-singular matrix has a matrix inverse. Matrix inverse is calculated by using Y=inv(X). Here Y is the inverse matrix of the Square matrix X. To determine inverse, X cannot be a scaled or singular matrix.

Example:

```
>> X=pascal(4)
X =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
>> Y=inv(X)
Y =
    4.0000   -6.0000    4.0000   -1.0000
   -6.0000   14.0000  -11.0000    3.0000
    4.0000  -11.0000   10.0000   -3.0000
   -1.0000    3.0000   -3.0000    1.0000
```

If the matrix is singular the MATLAB shows the warning "Matrix is singular to working precision". The *pinv* function of MATLAB is used to find Moore Penrose Pseudo inverse matrix.

Example:

```
>> A= [1 2 4; 1 3 5; 2 3 6]
A =
     1     2     4
     1     3     5
     2     3     6
>> B=pinv(A)

B =
   -3.0000   -0.0000    2.0000
   -4.0000    2.0000    1.0000
    3.0000   -1.0000   -1.0000
```

Pseudo inverse is the partial replacement for inverse of matrix.

Cont..

The *det* returns the determinant of square matrix. If matrix contains only integer entries the determinant will also be an integer.

Example:

```
>> X=rand(4)

X =

    0.9355    0.0579    0.1389    0.2722
    0.9169    0.3529    0.2028    0.1988
    0.4103    0.8132    0.1987    0.0153
    0.8936    0.0099    0.6038    0.7468

>> det(X)
ans =
   -0.0154
```

Linear Equations

Linear Equations in one variable:

An equation with equality involving one or more unknown quantities is known as variable. An equation is called a linear equation in one variable or an equation of degree one in one variable, if only a single variable with degree one occurs in the equation. For example

1) 2x + 3 = 4
2) x + 2 = 3x – 9
3) ½ y + 3/2 = 2 ^ 0.5 – 5
4) y + 2 = 0

these are the examples of linear equations in one variable. Whereas

ax^2 + bx +c = 0
x^2 – 1 = 9x^2 – 10
2x + 3 = 4x^2
x + y = 2

are not examples of linear equations in one variable.

A value of the variable that makes the two sides of an equation equal is called a solution of the equation. For example, when we substitute 2 for x in the equation 3x – 5 = x – 1, we get

LHS = 3x – 5
= 3 * 2 – 5
= 1
RHS = x – 1
= 2 –1
= 1.

Since LHS = RHS for x = 2, therefore '2' is a solution of the equation 3x – 5 = x – 1.

Cont..

Properties to solve linear equations:

We can add the same quantity to both sides of equality without changing the equality.

We can subtract the same quantity from both sides of equality without changing the equality.

We can multiply or divide both the sides of equality by the same non-zero number without changing the equality.

Example 1: Solve the equation $24x + 8 = 12x + 40$.

Solution: Dividing both the sides of the equation by 4, the equation becomes

$$6x + 2 = 3x + 10$$

Subtracting 2 from both sides, we get

$$6x = 3x + 8$$

Adding $-3x$ to both sides, we get

$$3x = 8$$

Multiplying both sides by 1/3, we get

$$x = 8/3$$

Because of the above properties 1 through 4, all the equalities of equation (1) to (4) are the same as the given equation. Hence 8/3 is the solution of the given equation.

We can verify the solution by substituting $x = 8/3$ in both sides of the given equation, getting 72 on both sides. Thus LHS = RHS for $x = 8/3$. Therefore, 8/3 is the solution of the given equation.

Note that when we added $-3x$ to both sides of equation (2), the effect was to make $6x$ on the RHS equal to $3x - 3x = 0$. Thus, $3x$ disappeared from the right and appeared on the left with its sign changed is called *transposition* or *transposing* a term from one side to the other.

Cont..

A suitable way of solving an equation is to transpose all the terms containing the variable to one side and the constant terms to the other as shown in the following example:

Example 2 Solve the equation $5x – 3 = 2x + 9$

Solution: Transposing $2x$ from RHS to the LHS, we get

$$5x – 3 –2x = 9$$

Transposing $–3$ from the LHS to the RHS,

$$5x – 2x = 9 + 3,$$

$3x = 12$, or $x = 4$ after dividing both the sides by 3.

Cont..

Linear Equation in Two Variables:

Linear equations in two variables differ from linear equation in one variable by having two unknowns instead of one. Thus like in linear equation of one variable is of the form ax + b = 0, where a ~= 0, a linear equation in two variables is of the form ax + by + c = 0, where a ~= 0, b ~= 0.

It is the usual way to denote the two variables by x and y, but other variable may also be used. For example,

$$x + 2y + 4 = 0,\ x + 2y = 4,\ -3x + 7y - 5 = 0,\ 5u + 6v = 11$$

these are all linear equation in two variables. Solution of linear equation in two variables means *a pair of values, one for x and the other for y, which when substituted in the given equation, make the two sides of the equation equal.* For example, x = 2, y = 3, is a solution of the equation 3x + 4y = 18, because when we substitute x = 2, y = 3, in above equation, we find that

$$LHS = 3x + 4y = 3 * 2 + 4 * 3 = 18 = RHS$$

On the other hand, x = 1, y = 2, is not a solution of the above equation, as x = 1, y = 2 makes the LHS equal to 11 which is different from the RHS which is 18. An interesting fact about a solution of a linear equation in two variables is that it is not unique. Verify that x = 6, y = 0, is also a solution of the above equation. Thus equation has at least two solutions x = 2, y = 3 and x = 6, y = 0. But in reality a linear equation in two variables has infinitely many solutions.

Cont..

Example 3: Find four different solutions of the equation x + 2y = 3.

Solution: By inspection, x = 1, y = 1, is a solution because for x = 1, y = 1.

LHS = x + 2y = 1 + 2 = 3 = RHS

Lets now take x = 0. With this value of x, the given equation reduces to 2y = 3 which has a unique solution y = 1.5. Hence x = 0, y = 1.5 is also a solution. Taking y = 0, the given equation reduces to x =3. Hence x = 3, y = 0, is a solution as well. Finally lets us take y = -1. The given equation now reduces to x – 2 = 3. i.e., x = 5. Hence x = 5, y = -1, is yet another solution. Thus, four of the infinitely many solutions of the given equation are

x =1, y = 1, x = 3, y = 0,

x = 5, y = -1, x = 0, y = 1.5

The two convenient solutions of an equation in two variables are of the form (0, y) and (x, 0).

The solution of Example 3 can be expressed in the form of a table as follows:

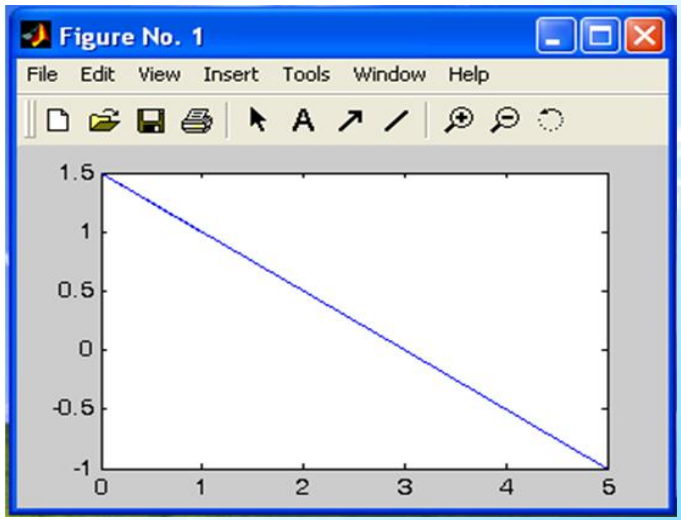
Table 1

x	**1**	**0**	**3**	**5**
y	**1**	**1.5**	**0**	**-1**

Cont..

Now plot the points (1, 1), (0, 1.5), (3, 0), (5, -1) on the graph in MATLAB as follows:

```
>> x = [1        0          3          5]
x =
     1     0     3     5
>> y = [1        1.5        0          -1]
y =
    1.0000    1.5000         0   -1.0000

>> plot(x,y)
```



Factorization

- In MATLAB linear equations are solved, by using following matrix factorization methods:

  1) Cholesky Factorization

  2) LU Factorization

  3) QR Factorization

Cont..

Cholesky factorization method is used for symmetric, positive definite matrices, whereas LU and QR are used for general square matrices and rectangular matrices respectively. To execute these factorization methods in MATLAB *chol*, *lu* and *qr* functions are used. In these methods of factorization triangular matrices are used, whose all elements of either above or below the diagonal are zero. The Cholesky factorization can solve the symmetric matrix as the product of triangular matrix and its transpose. A symmetric matrix is a matrix where $A_{ij}=A_{ji}$ for all i and j. In other words the symmetric matrix can also be defined as $[A]=[A]^T$ Symmetric matrix are computationally more beneficial because they need only half the storage space as well as half computational time to determine their solutions. The cholesky decomposition technique is based on the fact that a symmetric matrix can be decomposed as $[A]=[L]\ [L]^T$. i.e. the resulting triangular factors are the transpose of each other.

Cont..

Example 1:

```
>> D=[6 15 55; 15 55 225; 55 225 979]

D =

     6    15    55
    15    55   225
    55   225   979

>> R=chol(D)

R =

    2.4495    6.1237   22.4537
         0    4.1833   20.9165
         0         0    6.1101

>> R1=R'*R

R1 =

    6.0000   15.0000   55.0000
   15.0000   55.0000  225.0000
   55.0000  225.0000  979.0000
```

Example 2:

A= B B'

```
>> A=pascal(5)
A =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
>> R=chol(A)
R =
     1     1     1     1     1
     0     1     2     3     4
     0     0     1     3     6
     0     0     0     1     4
     0     0     0     0     1
>> R'
ans =
     1     0     0     0     0
     1     1     0     0     0
     1     2     1     0     0
     1     3     3     1     0
     1     4     6     4     1

>> B=R'*R
B =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
```

Cont..

LU Factorization: LU factorization is used for elimination of time-consuming steps by formulating values in such a way that it involves only operations of coefficients of matrix. This factorization method is well suited for those applications where right hand side vectors must be evaluated for a single value of matrix of left hand side. This technique also provides an efficient way to compute the inverse of the matrix. For example Gauss elimination method is used to solve systems of linear algebraic equations like [A] {X}={B}. Some time Gauss elimination becomes inefficient, when solving equations with the same coefficients [A], but with different right hand side constants. In such cases Gauss elimination requires two steps forward elimination and backward substitution. Out of these the forward elimination steps comprises the bulk of computational efforts. To overcome this problem and increase the speed of processing we use LU factorization for large systems of equations. LU Factorization separates the time consuming elimination of the matrix [A] from the manipulation of the right hand side {B}. Thus once [A] has been factorized multiple right hand side vectors can be evaluated in an efficient manner.

Cont..

Example:

```
>> A=[3 -0.1 -0.2;0.1 7 -0.3;0.3 -0.2
10]

A =

    3.0000   -0.1000   -0.2000
    0.1000    7.0000   -0.3000
    0.3000   -0.2000   10.0000

>> [L U]=lu(A)
L =
    1.0000         0         0
    0.0333    1.0000         0
    0.1000   -0.0271    1.0000
U =
    3.0000   -0.1000   -0.2000
         0    7.0033   -0.2933
         0         0   10.0120

>> B=L*U

B =

    3.0000   -0.1000   -0.2000
    0.1000    7.0000   -0.3000
    0.3000   -0.2000   10.0000
```

Hence B=A;

Cont..

QR Factorization:

An orthogonal matrix or a matrix with an orthogonal column is a real matrix whose columns have unit length and are perpendicular to each other. If Q is orthogonal then Q'*Q =1. Orthogonal and unitary matrices are desirable for numerical computations because they preserve length, preserve angle and do not magnify errors. The orthogonal or QR Factorization expresses any rectangular matrix as a product of an orthogonal or unitary matrix and an upper triangular matrix. A column permutation may also be involved like A=QR or AP=QR where, Q is orthogonal or unitary, R is upper triangular and P is permutation.

Cont..

```
>> A=magic(3)
A =

     8     1     6
     3     5     7
     4     9     2

>> [Q,R,P]=qr(A)
Q =
   -0.8480    0.5223    0.0901
   -0.3180   -0.3655   -0.8748
   -0.4240   -0.7705    0.4760

R =
   -9.4340   -6.2540    -8.1620
         0   -8.2394    -0.9655
         0         0    -4.6314
P =

     0     0     1
     1     0     0
     0     1     0
```

Eigenvalues

An eigenvalue and eigenvector of a square matrix A are a scalar λ and a vector v that satisfy Av=λv. With eigenvalue on the diagonal matrix Λ and the corresponding eigenvectors forming the columns of matrix V by AV=V Λ; if V is non-singular then this becomes the eigenvalue decomposition. Example:

```
>> X=[1 3 7 -2; 5 4 3 2; 3 0 2 4; 9 8 7 6]

X =

     1     3     7    -2
     5     4     3     2
     3     0     2     4
     9     8     7     6

>> eig_vector=eig(X)

eig_vector =

  13.4902
  -2.7739
   1.1418 + 2.3758i
   1.1418 - 2.3758i
```

With two output arguments *eig* function computes the eigenvectors and stores the eigenvalues in a diagonal matrix.

Cont..

```
>> [V D]=eig(X)

V =

    0.1363         -0.8259         -0.5615 + 0.1157i         -0.5615 - 0.1157i
    0.3590          0.3962          0.2362 + 0.3481i          0.2362 - 0.3481i
    0.3351          0.3374          0.0242 - 0.3374i          0.0242 + 0.3374i
    0.8604          0.2168          0.6163                    0.6163


D =

   13.4902         0                0                         0
         0        -2.7739           0                         0
         0         0                1.1418 + 2.3758i          0
         0         0                0                         1.1418 - 2.3758i
```

The first two eigenvectors are real and the other two vectors are complex conjugates of each other.

Polynomials

Polynomials are used in many applications of engineering and science like curve fitting, characterizing dynamic system and linear systems, mechanical devices, structures and electrical circuits. Therefore polynomials are widely applicable to solve many engineering and science problems. The general form of polynomial equation is

$$f_n(x)=a_0+a_1x+a_2x^2+\ldots+a_nx^n$$

where n is the order of the polynomial and a is the constant coefficients.

To determine a single root of an n^{th} order polynomial we have to repeat root location procedure; it may give the same root. Therefore, it would be nice to remove the found root before proceeding. This removal process is referred to as polynomial deflation. Polynomials are typically represented in their order format. For example a fifth order polynomial could be written as

$$f_5(x) = -120 - 46x + 79x^2 - 3x^3 - 7x^4 + x^5$$

MATLAB provides various functions for polynomial operations such as polynomial roots, evaluation, differentiation, curve fitting and partial fraction expansion. In MATLAB polynomials are represented by row vectors containing coefficients ordered by descending powers. For example

$$P(x)=x^4-3x^3-2x^2+x+10$$

To enter this polynomial into MATLAB the following statement is used

```
>>p=[1 –3 –2 1 10]
p =

    1    -3    -2     1    10
```

The *roots* function of MATLAB calculates the roots of polynomial and stores in column vector.

Cont..

```
>> r=roots(p)
r =

   3.2260
   1.6616
  -0.9438 + 0.9873i
  -0.9438 - 0.9873i
```

The *poly* function returns polynomial coefficients

```
>> P1=poly(r)

P1 =

 1.0000   -3.0000   -2.0000    1.0000
10.0000
```

The *poly* function also computes the coefficients of the characteristics polynomial of a matrix. The *polyval* function evaluates a polynomial at a specified value.

```
 >> polyval(p, 2)

ans =

    -4

>> polyval(p, 6)

ans =

   592
```

In these examples, value of x is considered as 2 and 6 respectively.

Cont..

MATLAB also supports to evaluate a polynomial in a matrix format. In the above example if x is a square matrix then the equation becomes

$$P(X) = X^4 - 3X^3 - 2X^2 + X + 10I$$

Where I is Identity matrix. This equation can be solved in MATLAB by using polyvalm function as follows.

```
>> X = [1 3 7 -2; 5 4 3 2; 3 0 2 4; 9 8
       7 6]
```

X =

1	3	7	-2
5	4	3	2
3	0	2	4
9	8	7	6

```
>> Y=polyvalm(p,X)
```

Y =

2756	2096	3014	1540
7018	5618	8168	3968
6580	5234	7722	3664
16866	13308	19592	9558

Cont..

The multiplication and division operations on polynomial are performed by convolution and deconvolution method in MATLAB using *conv* and *deconv* functions respectively. For example,

$$P(x) = 2x^3 - 2x + 1 \quad \text{and}$$

$$Q(x) = x^4 - 3x^3 - 2x^2 + x + 10$$

are the polynomials. To compute their product by using *conv* function

```
>> P=[2 0 -2 1]

P =

    2     0    -2     1

>> Q=[1 -3 -2 1 10]

Q =

    1    -3    -2     1    10

>> S=conv(P,Q)

S =

    2    -6    -6     9    21    -4   -19    10
```

Deconvolution of these polynomials is computed by *deconv* function

```
>> [q r]=deconv(P,Q)

q =

     0


r =

     2     0    -2     1
```

Cont..

The *polyder* function of MATLAB is used to compute the derivative of the polynomial equation; for example.

$$P(x) = x^3 + 2x^2 + 5x + 1$$

```
>> p=[1 2 5 1]

p =

    1    2    5    1

>> D=polyder(p)

D =

    3    4    5
```

polyder function is also used to compute the derivative of the product or quotient of two polynomials.

$$a(x) = -2x^2 + 5x + 1$$

$$b(x) = x^2 - 5x + 1$$

to calculate the derivative of the product

```
> a=[2 5 1]

a =

    2    5    1

>> b=[1 -5 1]

b =

    1   -5    1

>> c=polyder(a,b)

c =

    8  -15  -44    0
```

to calculate the derivative of quotient.

```
>> [q,d]=polyder(a,b)

q =

  -15    2   10

d =

    1  -10   27  -10    1
```

Cont..

Similarly, *polyint* function is used to calculate integral of a polynomial; it is easy to express.

```
>> I=polyint(a)
I =
   0.6667    2.5000    1.0000         0
```

polyfit finds the coefficients of polynomial that fits a set of data in a least squares format. Syntax:

p=polyfit(x,y, n)

Where, x and y are vectors containing the x and y data that to be fit and n is the order of the polynomial. For example, consider x and y data

```
>>x=[5 4 3 2 1 0];
>>y=[1.4 5.3 7 9.9 19.5 25.5];
```

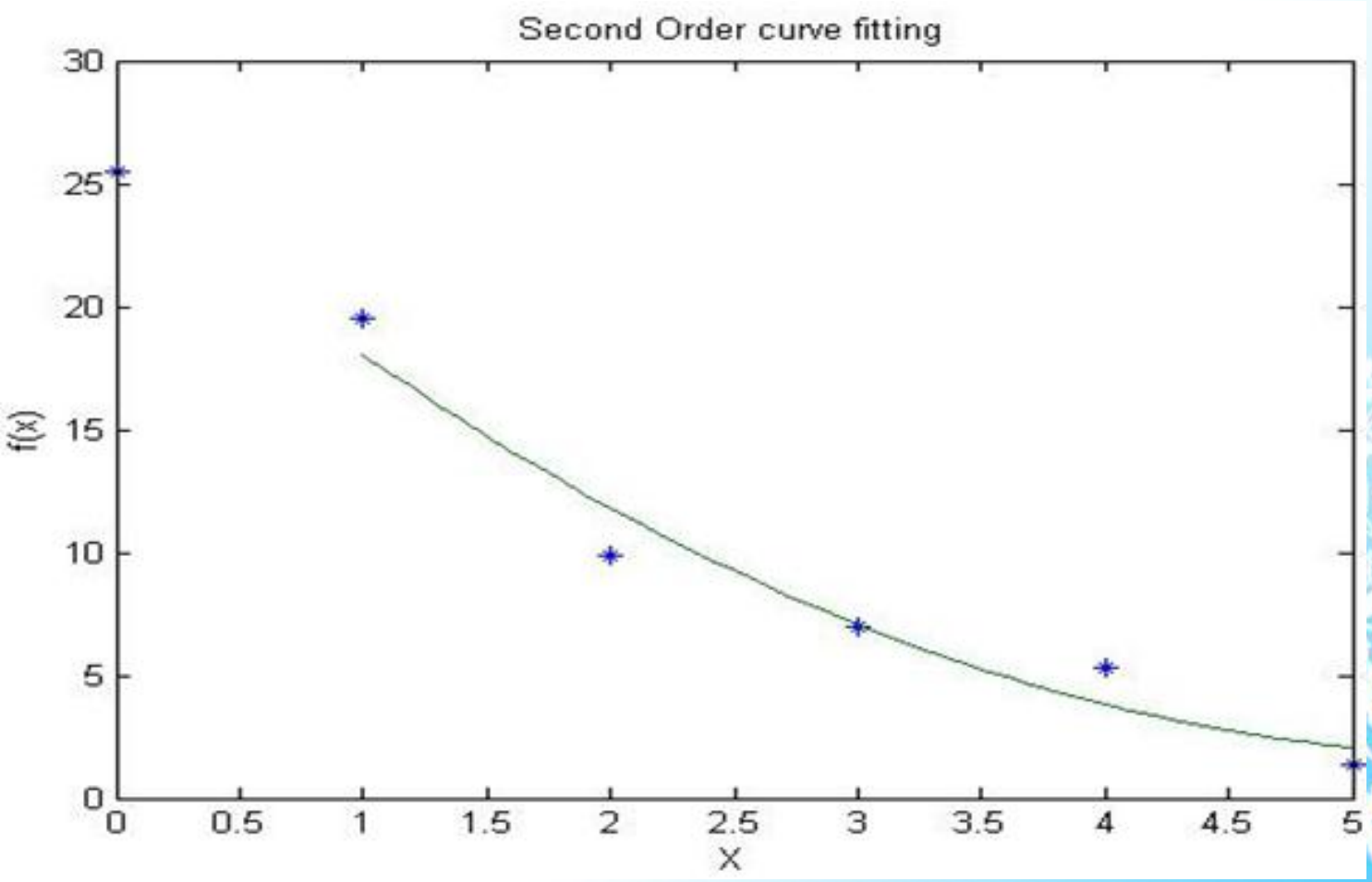
>> p=polyfit(x,y,2) % Second order polynomial to fit approximately the data values.

```
p =
    1.1000   -9.6600   23.6400
```

%compute the values of the polyfit estimate over a finer range and plot the estimate over %the real data values for comparison

```
>> x1=1:0.1:5;
>> y1=polyval(p,x1);
>> plot(x,y,'*',x1,y1)
```

Cont..

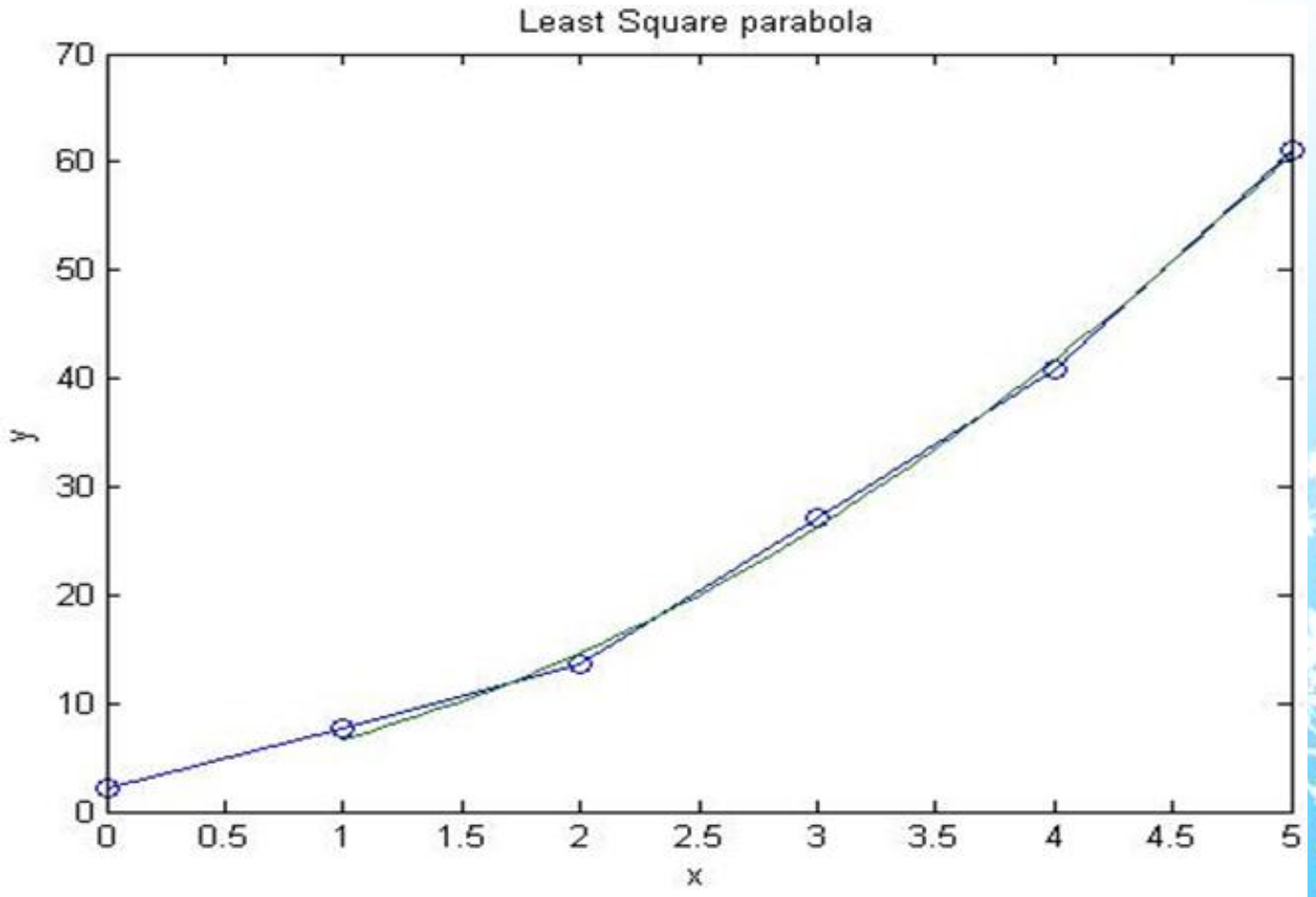

Second Order curve fitting
f(x)
X
0
5
10
15
20
25
30
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Cont..

Example: Computations for an error analysis of the quadratic least square fit

```
>> x=[0 1 2 3 4 5];
>> y=[2.1 7.7 13.6 27.2 40.9 61.1];
>> p=polyfit(x,y,2);
>> x1=1:0.1:5;
>> y1=polyval(p,x1);
>> plot(x,y,'--',x1,y1)
```

Cont..


Least Square parabola
y
x
0
10
20
30
40
50
60
70
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Cont..

Function	Description
Poly	Polynomial with specified roots
polyder	Polynomial derivative
conv	Convolution and polynomial multiplication
deconv	Deconvolution and polynomial division
polyfit	Polynomial curve fitting
polyint	Analytic polynomial integration
Polyval	Polynomial evaluation
polyvalm	Matrix polynomial evaluation
residue	Convert between partial fraction expansion and polynomial coefficients
Roots	Computes polynomial roots
polyeig	Polynomial eigenvalue problem

Interpolation

It is a process for estimating a value that lies in between known data points. Interpolation is an important part of signal and image processing. MATLAB provides a number of interpolation techniques that are used to balance the smoothness of the data fit with speed of execution and memory uses. There are two kinds of one-dimensional interpolation used in MATLAB: 1) Polynomial Interpolation and 2) FFT based interpolation. The function *interp1* performs one-dimensional interpolation, data analysis and curve fitting. This function uses polynomial techniques, fitting the supplied with polynomial functions. Between data points and evaluate an appropriate function at the desired interpolation point. Its most general form is

```
yi = interp1(x,y,xi,method)
```

Where, y is a vector containing the values of function, x is the vector of the same length containing the points for which the values in y are given. xi is a vector containing the points at which to interpolate. The method is an optional string specifying an interpolation method.

Cont..

The function *interpft* performs one-dimensional interpolation using FFT based method. This method calculates the Fourier Transform of a vector that contains the value of a periodic function. It then calculates the Inverse Fourier Transform using more points. General syntax is

Y=interpft(x,n),

Where x is vector of periodic function sampled at equally spaced points, n is the number of equally spaced points to return.

The function *interp2* performs 2-dimensional interpolation. This function is used for image processing and data visualization. Its general form is

ZI=interp2(X,Y,Z, XI, YI, Method)

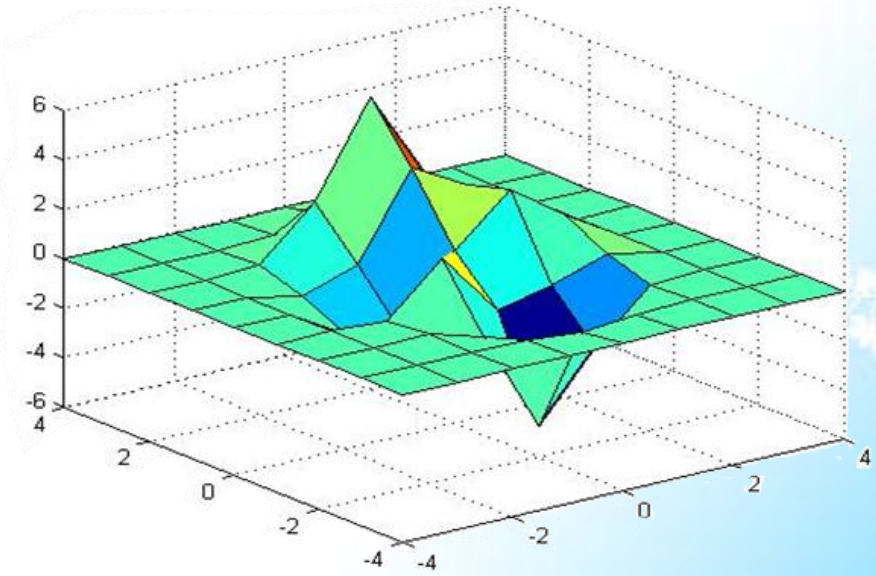
Where, Z is a rectangular array of two-dimensional function and X and Y are arrays of same size containing the points for which the values in Z are given. XI and YI are matrices containing the points to be interpolated.

Cont..

Example:

1. Generate the peaks function at low resolution

```
>> [x y]=meshgrid(-4:1:4);
>> z=peaks(x,y);
>> surf(x,y,z)
```



Cont..

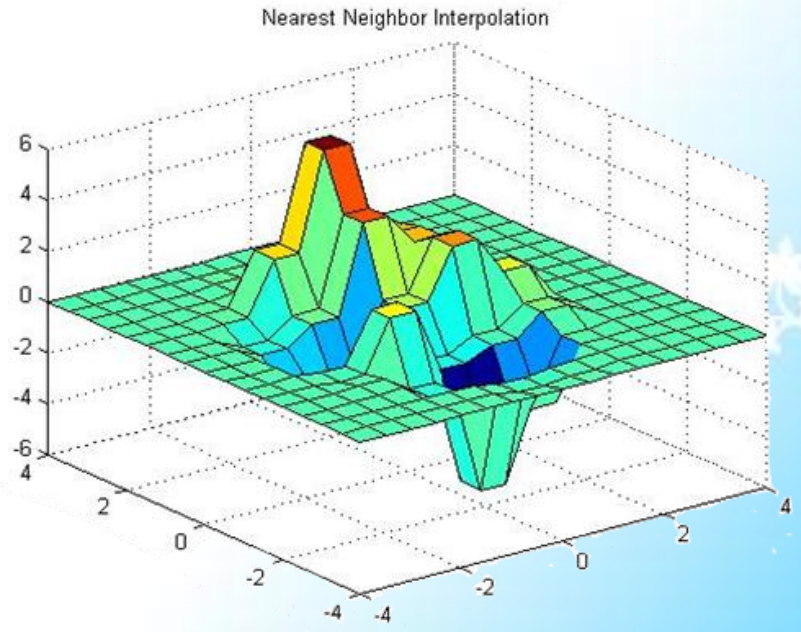
2. Generate a finer mesh for interpolation

>> [x1 y1]=meshgrid(-4:0.5:4);

3. Interpolate using nearest neighbor interpolation

>>z1=interp2(x,y,z,x1,y1,'nearest')

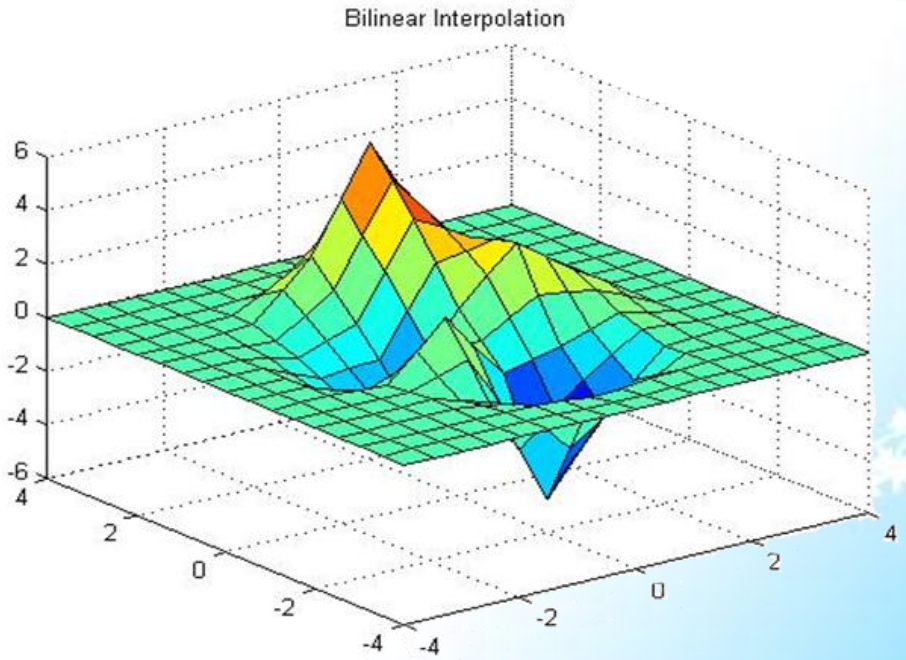
>> surf(x1,y1,z1)



Cont..

4. Interpolate using bilinear interpolation

```
>> z2=interp2(x,y,z,x1,y1,'bilinear');
>> surf(x1,y1,z2)
```

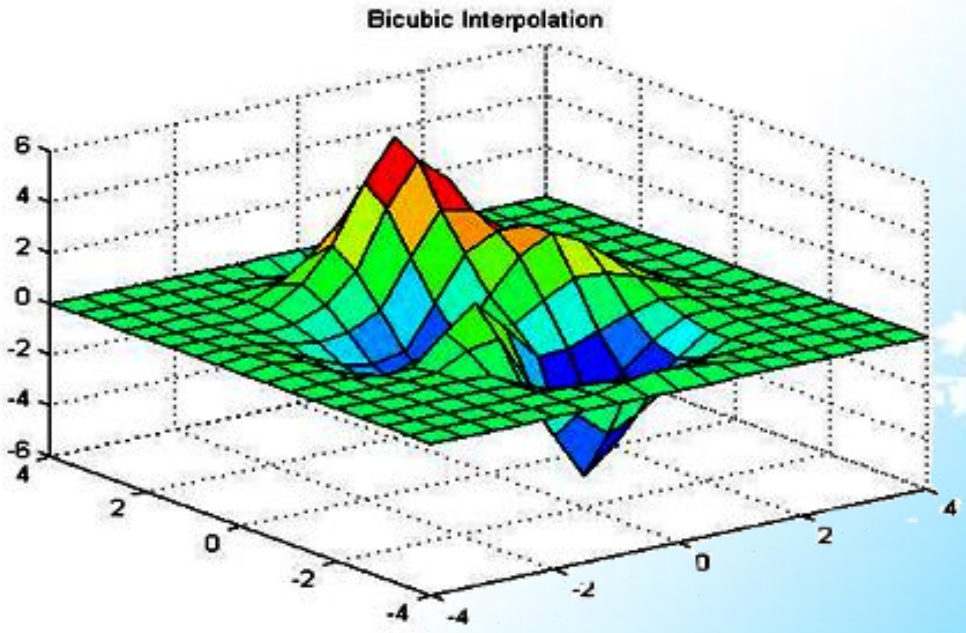


Cont..

5. Interpolate using bicubic interpolation

>> z3=interp2(x,y,z,x1,y1,'bicubic');

>> surf(x1,y1,z3)



Cont..

Function	Description
Dsearch	Search for nearest point
dsearchn	Multidimensional closest point search
griddata	Data gridding
griddata3	Data gridding and hyper surface fitting for three-dimensional data
griddatan	Data gridding and hypersurface fitting (dimension >= 2)
interp1	One-dimensional data interpolation
interp2	Two-dimensional data interpolation
interp3	Three-dimensional data interpolation
interpft	One-dimensional interpolation using fast Fourier transform method
interpn	Multidimensional data interpolation
ndgrid	Generate arrays for multidimensional functions and interpolation
pchip	Piecewise Cubic Hermite Interpolating Polynomial (PCHIP)
spline	Cubic spline data interpolation
tsearchn	Multidimensional closest simplex search
mkpp	Make piecewise polynomial
Ppval	Piecewise polynomial evaluation
Unmkpp	Piecewise polynomial details

Data Analysis

User's data can be stored in row format or column format, called as row-oriented or column-oriented respectively. In other way the vector can be either 1-by-n or n-by-1 for multivariate data. A matrix is the natural representation used by two possible ways. In the first way the different variables are put into columns allowing observations to vary down through the rows. Therefore a data set-consisting of 7 days for four cities is stored in a matrix of size 7-by-4. Consider a sample data set of 7 days of week of sales of P4 computers from four different cities.

Sample Data set of Computer Sale:

City / Days	Nagpur	Mumbai	Aurangabad	Pune
Monday	50	70	85	65
Tuesday	112	55	35	86
Wednesday	200	25	115	210
Thursday	45	28	80	95
Friday	70	89	93	62
Saturday	59	63	97	78
Sunday	41	73	211	142

Cont..

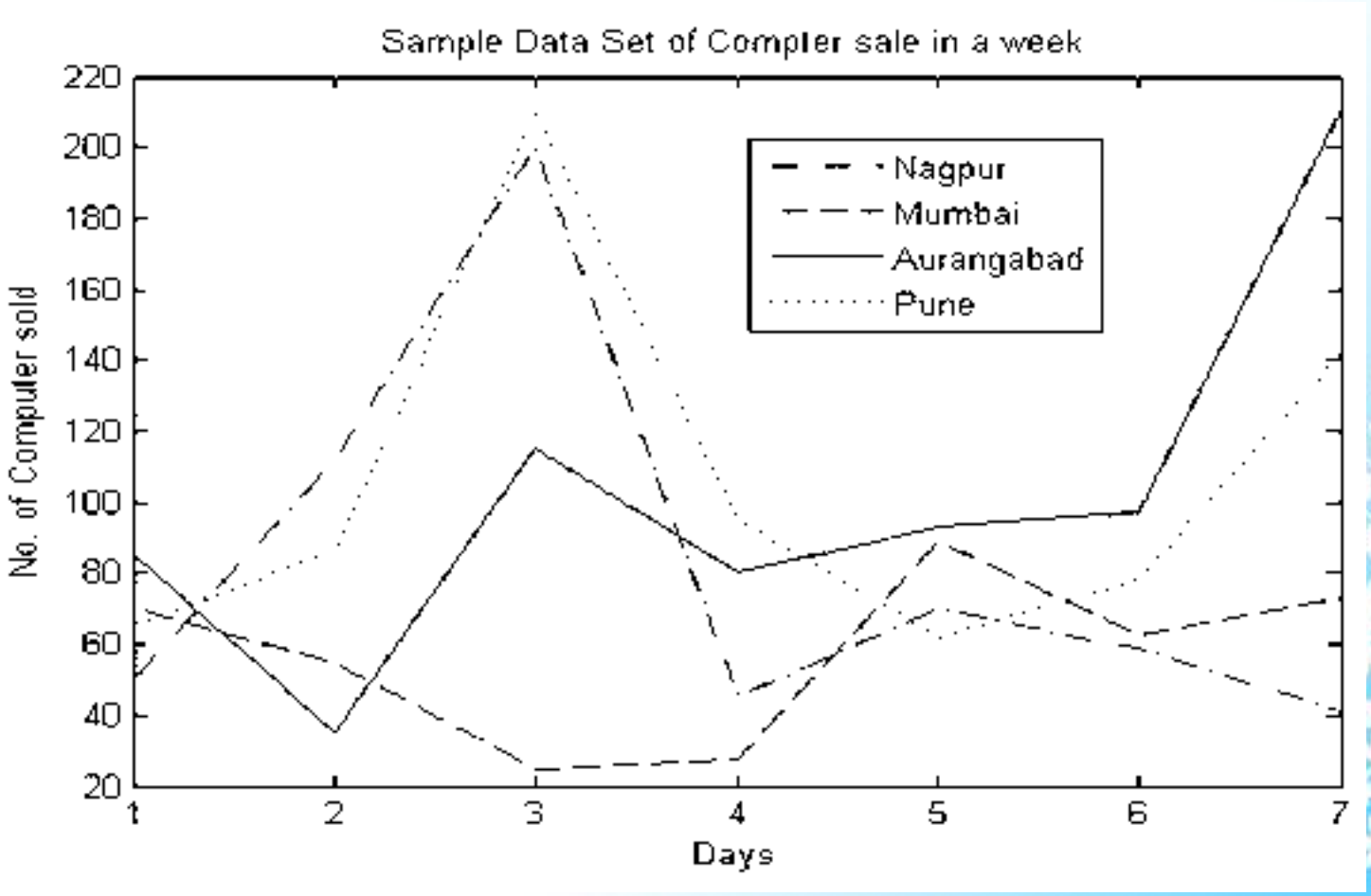
This row data is stored in the file saledata.dat. To view the contents of saledata.dat file use load command. The load command creates a matrix saledata in the workspace.

```
>> load saledata.dat
>> saledata
saledata =

    50    70    85    65
   112    55    35    86
   200    25   115   210
    45    28    80    95
    70    89    93    62
    59    63    97    78
    41    73   211   142
>> [r c]=size(saledata)
r =
     7
c =
     4
>> t=1:r;                    %Create a time vector t of
      integer from 1 to r.
```

Plot the saledata vs time

```
>> t=1:r;
>> set(0,'defaultaxeslinestyleorder','-.|--|-|:');
>> set(0, 'defaultaxescolororder',[0 0 0]);
>> plot(t, saledata)
```

Cont..

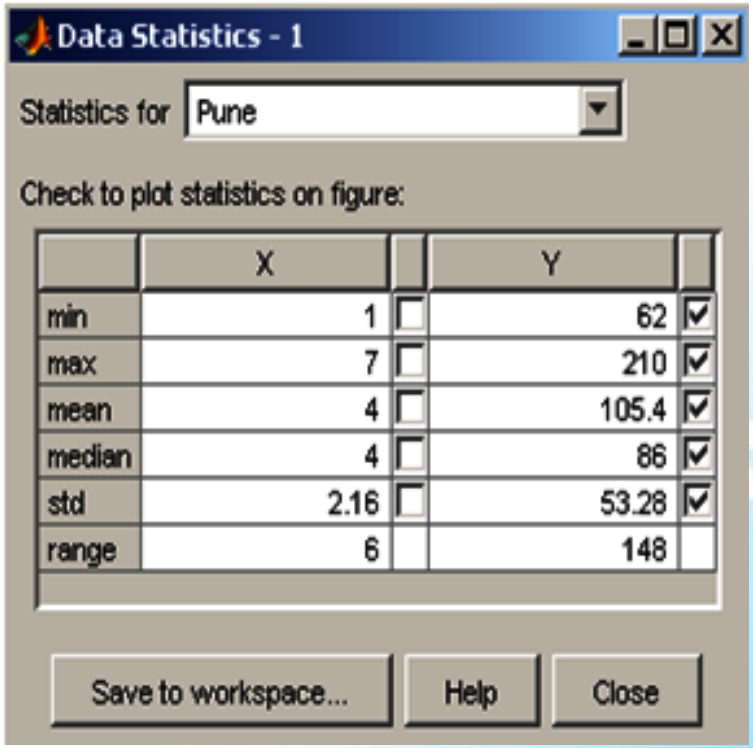

Sample Data Set of Compter sale in a week
No. of Computer sold
Days
Nagpur
Mumbai
Aurangabad
Pune
220
200
180
160
140
120
100
80
60
40
20
1
2
3
4
5
6
7

Cont..

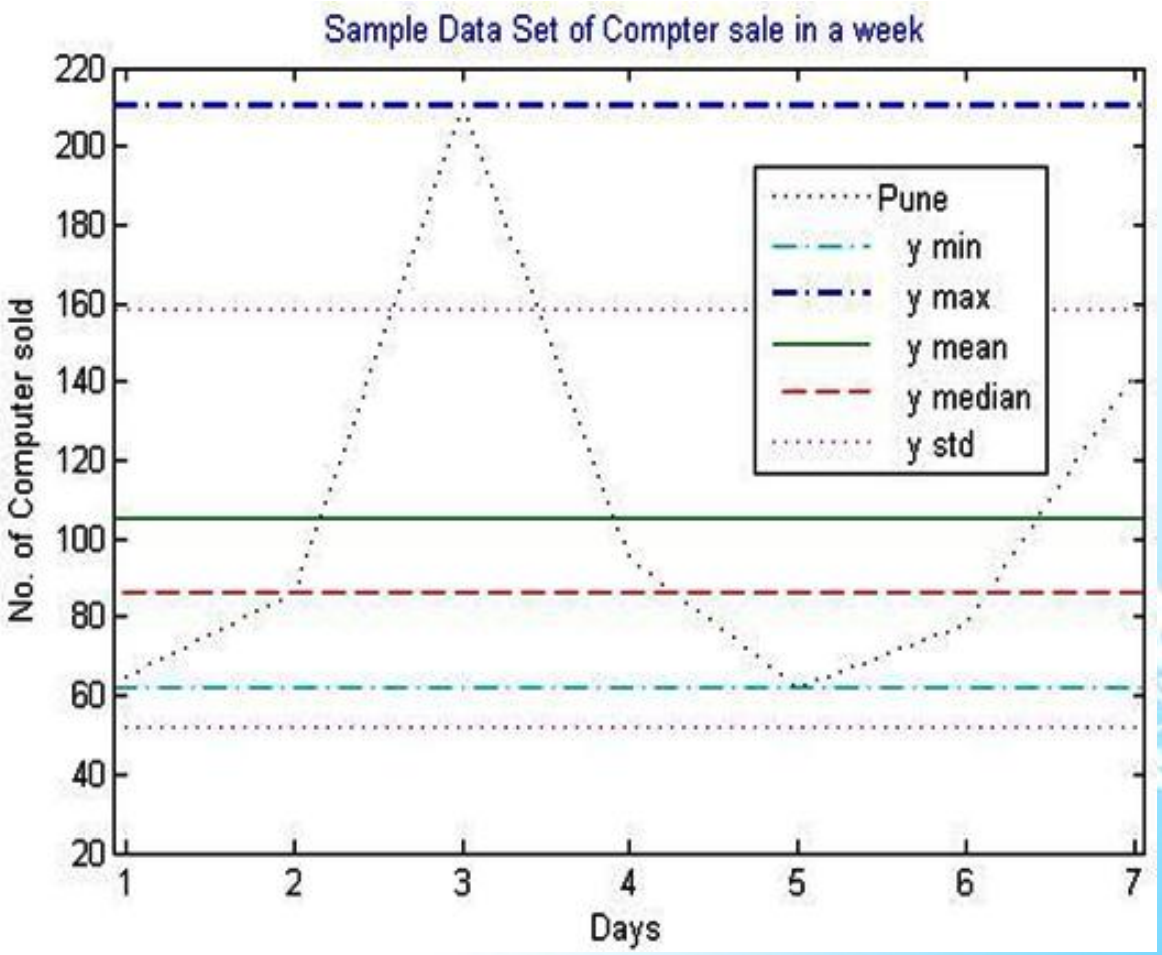
Function	Description
max	Largest component in Data
min	Smallest component of data
mean	Average or mean value
median	Median value
sort	Sort in ascending order
sortrows	Sort rows in ascending order
std	Standard deviation
prod	Product of element
diff	Difference functions and approximate derivatives
cumtrapz	Cumulative Trapezoidal Numerical Integration
cumsum	Cumulative sum of elements
comprod	Cumulative product of elements
sum	Sum of elements
trapz	Trapezoidal Numerical integration

Cont..

Out of these functions *min*, *max*, *mean*, *median*, and *std* are operated through the Data Statistics option of tool of the above figure window. The result of Pune city data is shown in the following figures.



Cont..


Sample Data Set of Compter sale in a week
No. of Computer sold
Days
Pune
y min
y max
y mean
y median
y std
220
200
180
160
140
120
100
80
60
40
20
1
2
3
4
5
6
7

Cont..

MATLAB statistical capabilities include *cov* and *corrcoef* functions for the computation of co-variance and correlation coefficients respectively. The *cov* function computes the co-variance of vector and matrix and returns the variance for the vector of data, whereas the *corrcoef* computes co-relations coefficients and normalizes the measure of linear relationship strength between variables.

Example: Covariance of saledata

```
>> cov(saledata(:,1))

ans =

  3261.6
```

Example: Correlation coefficients of saledata.

```
>> corrcoef(saledata)
ans =

    1.0000   -0.5382   -0.1823    0.7169
   -0.5382    1.0000    0.2042   -0.5956
   -0.1823    0.2042    1.0000    0.4698
    0.7169   -0.5956    0.4698    1.0000
```

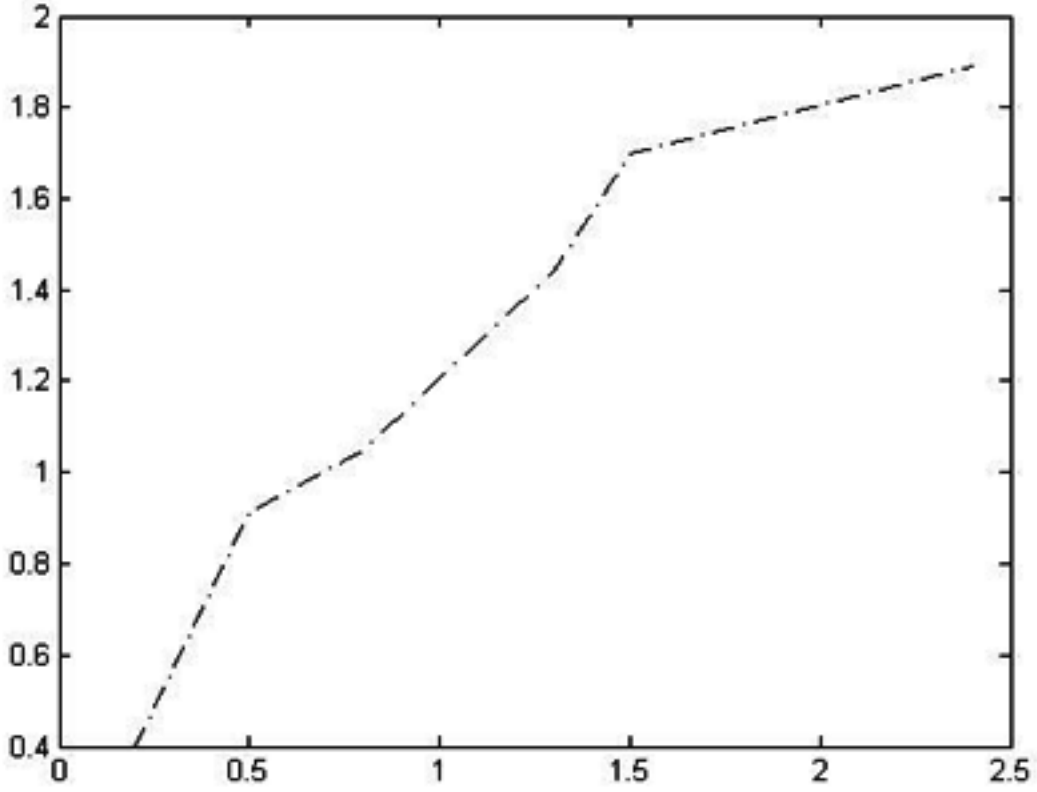
Polynomial Regression

In statistical property, user's data can be plotted using straight-line options. But some engineering data although exhibit a marked pattern obtained by poor representation because of straight line. To accomplish better curve by fitting the data one alternative is transformations. Another alternative is to fit polynomials to the data using polynomial regression. The least square procedure can be readily extended to fit the data toward higher order polynomial. Regression and curve fitting are useful to find functions that describe the relationship between some variables. Identifications of the coefficients of the function often leads to the formulation of an over-determine system of simultaneous linear equations. To find these coefficients MATLAB backslash operator is used.

Cont..

For example

```
>> t=[0.2 0.5 0.8 1.3 1.5 2.4];
>> y=[0.4 0.91 1.05 1.44 1.70 1.89];
>> plot(t,y)
```

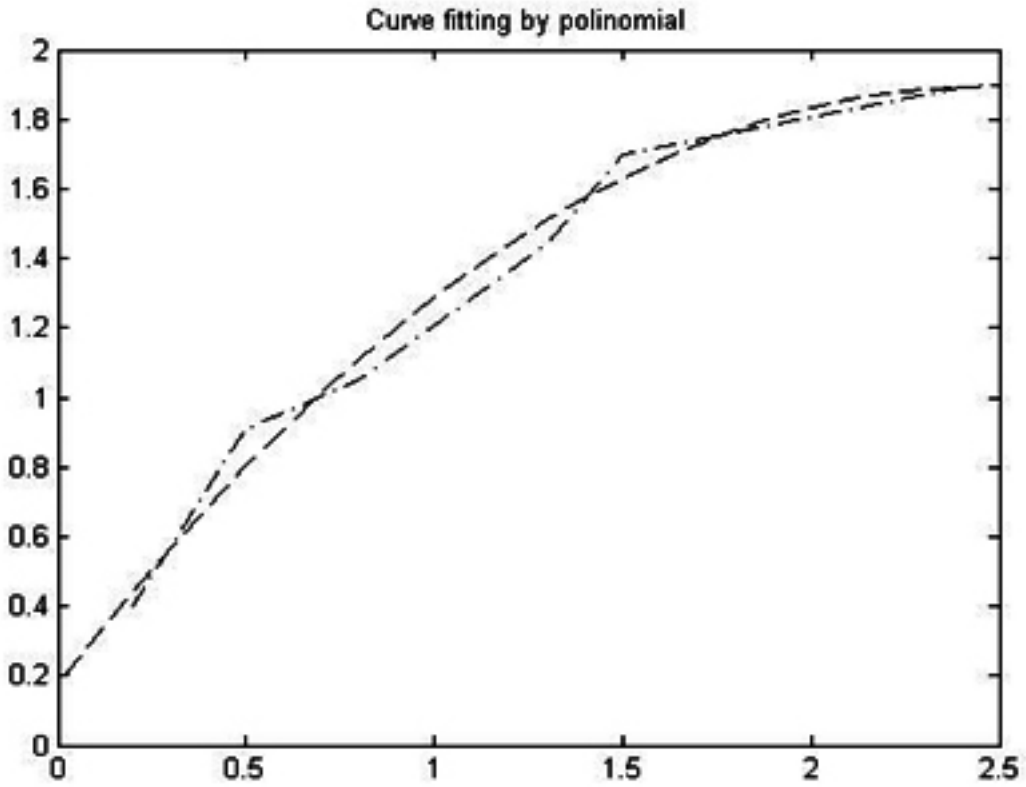


Cont..

On this above data, regression and curve fitting can be manipulated by polynomial regression, linear-in-the-parameter regression and multiple regression. In polynomial regression the data can be modeled by a polynomial function which consists of the unknown coefficients.

```
>> X=[ones(size(t)) t t.^2]

X =

    1.0000    0.2000    0.0400
    1.0000    0.5000    0.2500
    1.0000    0.8000    0.6400
    1.0000    1.3000    1.6900
    1.0000    1.5000    2.2500
    1.0000    2.4000    5.7600

>> a=X\y
a =
    0.1755
    1.3939
   -0.2824

>> T=(0:.1:2.5)';
>> Y=[ones(size(T)) T T.^2]*a;
>> plot(T,Y,'--',t,y,'-.')
```

Cont..


Curve fitting by polinomial
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2
2.5

Basic Fitting

For performing many curves fitting task we can also use the basic fitting option of tools menu of figure window within the same window. This option allows fitting data using spline interpolant, hermite interpolant, or polynomial up to 10th degree, plotting multiple fits simultaneously for a given data set, plot the fit residuals, examine the numerical results of a fit, evaluate i.e. interpolate or extrapolate a fit, annotate the plot with the numerical fit results and the norm of residuals and save the fit and evaluate its result to MATLAB workspace. Depending on the application we can use basic fitting options or command line functionality or both.

Fourier approximation

Interpolation method is emphasized by standard polynomials i.e. linear combinations are the monomials 1, x, x^2, …x^m The same interpolation can also be handled by the trigonometric functions 1, cos x, cos 2x,…, cos nx, sin x, sin 2x, …, sin nx. In engineering fields the oscillating and vibrating systems are widely used. To model such systems trigonometric functions play an important role. Fourier approximation represents a systematic framework of trigonometric series used to model the system. The hallmarks of a Fourier analysis is that it deals with both the time and frequency domains. Frequency domain tools such as fourier series, fourier transform and their discrete time counterparts form a cornerstone in signal processing. These transform decompose a signal into a frequency or continuum of sinusoidal components that identify the frequency domain content of the signal. MATLAB has provided various set of functions for fourier analysis. This set of functions performs the discrete fourier transforms and its inverse in one or more dimensions.

Cont..

Example: Periodic function f(t)

f(t)= f(t+T), where T is a constant called period.

Fourier analysis is useful for data analysis, as it breaks down a signal into constituent sinusoids of different frequencies. For sampled vector data Fourier analysis is performed using the Discrete Fourier Transform (DFT) whereas the Fast Fourier Transform (FFT) is an efficient algorithm for computing the DFT of a sequence. It is specially used for signal and image processing for filtering, convolution and frequency analysis.

Cont..

Function	Description
fft	Discrete Fourier Transform
fft2	Two-Dimensional Discrete Fourier Transform
fftn	N-Dimensional Discrete Fourier Transform
ifft	Inverse Discrete Fourier Transform
ifft2	Two-Dimensional Inverse Discrete Fourier Transform
ifftn	N-Dimensional Inverse Discrete Fourier Transform
angle	Phase angle
unwrap	Unwrap Phase angle in radiance.
fftshift	Move 0^{th} lag to center of spectrum.
cplzpair	Sort Numbers into complex conjugate pairs
nectpow2	Next higher power of 2.

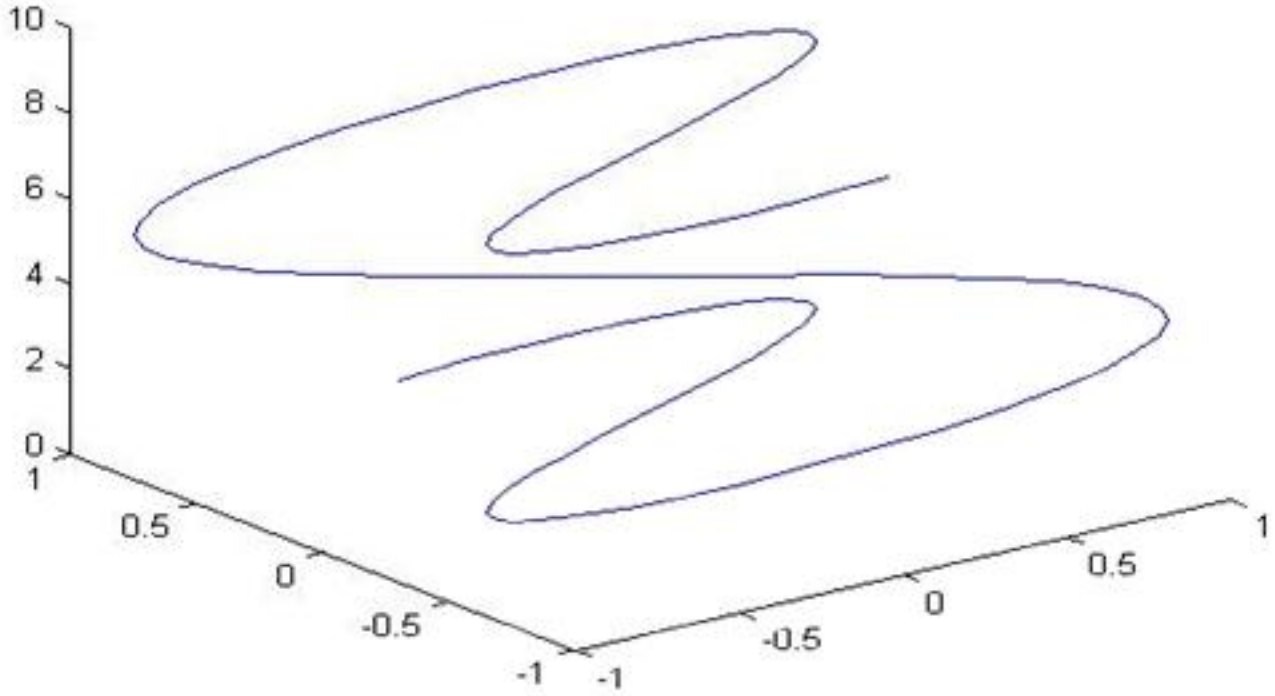
Integration and Differentiation

Integration and differentiation are fundamental techniques necessary in the field of calculus. Integration is used to compute the area under a function whereas differentiation describes a slope or gradient of a function. MATLAB has provided the functions for numerically approximating the integral and slope of the function. MATLAB has provided four functions for computing integrals of functions; they are *quad, quad1, dblquad* and *triplequad*.

Example:
Consider the curve parameterized by the equations x(t)=sign(2t), y(t)=cos(t) and z(t)=t; where, ; 3-D plot of this curve is

```
>> t=0:.1:3*pi;
>> plot3(sin(2*t),cos(t),t)
```

Cont..



Cont..

Differentiation: The Operation of finding the derivative of a function is called differentiation. Differentiation has many applications in engineering, science, social science, life science, information science and so on. Differentiation is useful to describe the slope of a function at a point, which is a microscopic property of a function. As a result differentiation is sensitive to minor changes in the shape of a function. Any small change in a function can easily create large changes in its slope in the neighborhood of the change. Because of this inherent sensitivity in differentiation, numerical differentiation is especially useful to perform a least squares curve fit to the data and then differentiate the resulting polynomial. We can also fit cubic splines to the data and then find the spline representation of the derivative.

Example:

Here the *polyder* function is used for the derivative

```
>> x=0:.1:1;
>> y=rand(1,11);
>> n=2;
>> p=polyfit(x,y,n)

p =

   -0.4511    0.8791    0.1304

>> pder=polyder(p)

pder =

   -0.9021    0.8791
```

Differentiation Equation

A differential equation is an equation that defines a relationship between a function and one or more derivatives of that function. Let *y* be some function of the independent variable *t*. Then the following are some differential equations relating *y* to one or more of its derivatives.

$$\frac{\partial}{\partial t} y(t) = t^2 y(t)$$

The equation states that the first derivative of the function *y* equals the product of t^2 and the function *y* itself. An additional, implicit statement in this differential equation is that the stated relationship holds only for all *t* for which both the function and its first derivative are defined.

A partial differential equation (PDE) is an equation involving functions and their partial derivatives; for example, the wave equation

Cont..

MATLAB PDE Solver is used to solve initial boundary value problems for systems of parabolic and elliptic partial differential equations in the one space variable x and time t. There must be at least one parabolic equation in the system. The basic syntax of PDE solver is

Solution=pdepe(m, pdefun, icfun, bcfun, xmesh, tspan)

Where,

- m is 0 for slab, 1 for cylindrical and 2 for spherical.
- pdefun is a component of the PDE and computes as [c,f,s]=pdefun(x,t,u,dudx), where x and t are scalars and u and dudx are vectors.
- icfn is of the form u=icfm(x) and used to evaluate the initial values of the solution components at x in the column vector u.
- bcfun is used to evaluate the terms p and q of the boundary conditions. [pl,ql,pr,qr]=bcfun(xl, ul, xr, ur, t)
- xmesh specifies the points at which a numerical solution is requested.
- tspan is used to specify the span at which the solution is requested for every value in x mesh.

The output of *pdepe* function is a three dimensional array, whose first dimension is to store approximate component of the solution u, second dimension is used to store solution at time t span and third dimension for solution at time t span and mesh points.

Summary

Two dimensional arrays of either real or complex numbers is called matrix. Identity matrix is a matrix of various sizes having ones on the main diagonal and zeros on remaining positions. Different manipulations on matrices are Kronecker Tensor product, Vector and Matrix norms, Inverse and Determinant of Matrix. An equation is called a linear equation in one variable or an equation of degree one in one variable. It is also possible to use linear equations for one variable and two variables. It is also possible to handle different factorization methods in MATLAB such as Cholesky factorization, LU Factorization and QR Factorization. An eigenvalue and eigenvector of a square matrix A are a scalar λ and a vector v that satisfy $Av=\lambda v$. Polynomials are used in many applications of engineering and science like curve fitting, characterizing dynamic system and linear systems, mechanical devices, structures and electrical circuits. Interpolation is a process for estimating a value that lies in between known data points. It is important part of signal and image processing. For data analysis user can use functions *min*, *max*, *mean*, *median*, and *std* which are operated through the Data Statistics. To accomplish better curve by fitting the data one alternative is transformations. Another alternative is to fit polynomials to the data using polynomial regression. Fourier approximation represents a systematic framework of trigonometric series used to model the system. Integration and differentiation are fundamental techniques necessary in the field of calculus. A differential equation is an equation that defines a relationship between a function and one or more derivatives of that function.

Thank You