Brac University
Department of Electrical and Electronic Engineering
EEE282/ECE282 (V3)
Numerical Techniques
Experiment-06: Curve Fitting: Linear & Polynomial Interpolation

---

Introduction:

Forming a polynomial:

A polynomial, p(x) of degree n in MATLAB is stored as a row vector, **p**, of length n+1. The components represent the coefficients of the polynomial and are given in the descending order of the power of x, that is

p = [a_n a_{n-1} a_1 a_0]

is interpreted as

$p(x) = a_n x^n + a_{n-1} x^{n-1} + \quad + a_1 x + a_0$

In MATLAB the following commands are used to evaluate a polynomial:
polyval, **poly**, **roots**, **conv** etc.

Lab Task 1: Construct a polynomial such that C(x)= A(x)*B(x) Where A(x)= $3x^2+2x-4$ and B(x)= $2x^3-2$ Also find the roots of A(x), B(x) and C(x).

Interpolation:

In the mathematical subfield of numerical analysis, **interpolation** is a method of constructing new data points from a discrete set of known data points.

In engineering and science one often has a number of data points, as obtained by sampling or some experiment, and tries to construct a function which closely fits those data points. This is called curve fitting. Interpolation is a specific case of curve fitting, in which the function must go exactly through the data points.

Definition:

Given a sequence of *n distinct* numbers x_k called **nodes** and for each x_k a second number y_k, we are looking for a function *f* so that

$$f(x_k) = y_k \ , \ k = 1, \ldots, n$$

A pair x_k,y_k is called a **data point** and *f* is called the **interpolant** for the data points.

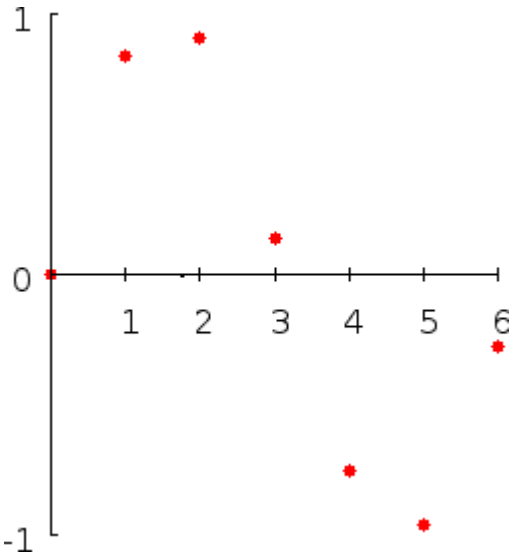
For example, suppose we have a table like this, which gives some values of an unknown function *f*. The data are given in the table:

Table 1

x	*f(x)*
0	0
1	0.8415
2	0.9093
3	0.1411
4	-0.7568
5	-0.9589
6	-0.2794

The plot can be shown as:

.

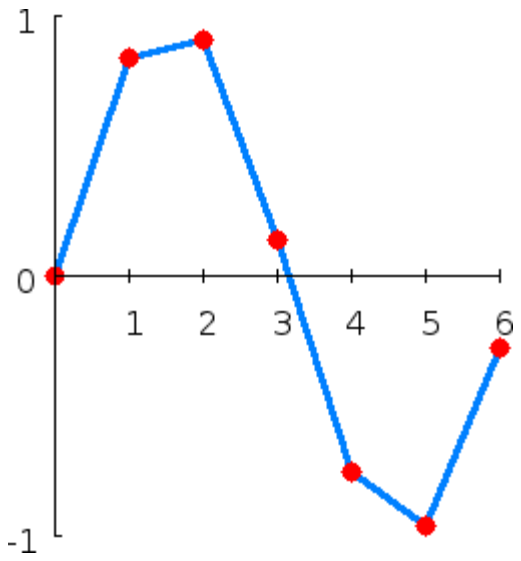


What value does the function have at, say, $x = 2.5$? Interpolation answers questions like this.

Types of interpolation:

A. Linear interpolation

One of the simplest methods is linear interpolation. Consider the above example of determining $f(2.5)$. We join the data points by linear interpolation and get the following plot:



Noe we can get $f(2.5)$. Since 2.5 is midway between 2 and 3, it is reasonable to take $f(2.5)$ midway between $f(2) = 0.9093$ and $f(3) = 0.1411$, which yields 0.5252.

Generally, linear interpolation takes two data points, say (x_a,y_a) and (x_b,y_b), and the interpolant is given by

$$f(x) = \frac{x - x_b}{x_a - x_b} y_a - \frac{x - x_a}{x_a - x_b} y_b$$

This formula can be interpreted as a weighted mean.

Linear interpolation is quick and easy, but it is not very precise.

Lab Task 2. Write a MATLAB code to implement Linear Interpolation and Plot the curve corresponding to table1 using Linear Interpolation.

Code-

```
% Data points from Table 1
x = [0, 1, 2, 3, 4, 5, 6];
fx = [0, 0.8415, 0.9093, 0.1411, -0.7568, -0.9589, -0.2794];

% Value of x for which you want to interpolate
x_interpolation = 2.5;

% Performing linear interpolation using Matlab's builtin function interp1
f_interpolated = interp1(x, fx, x_interpolation, 'linear');

% Displaying the resulted values
fprintf('Resulting Interpolated value at x = %.2f: %.4f\n', x_interpolation, f_interpolated);
% For .2f, the values will be showed after two decimal points, and for .4f,
% four decimal points will be displayed
```

Output-

```
Resulting Interpolated value at x = 2.50: 0.5252
>>
```

B. Polynomial interpolation

Polynomial interpolation is a generalization of linear interpolation. Note that the linear interpolant is a linear function. We now replace this interpolant by a polynomial of higher degree.

Consider again the problem given above. The following sixth-degree polynomial goes through all the seven points:

$$f(x) = -0.0001521x^6 - 0.003130x^5 + 0.07321x^4 - 0.3577x^3 + 0.2255x^2 + 0.9038x$$

Substituting $x = 2.5$, we find that $f(2.5) = 0.5965$.

Generally, if we have n data points, there is exactly one polynomial of degree $n-1$ going through all the data points. The interpolation error is proportional to the distance between the data points to the power n.

However, polynomial interpolation also has some disadvantages. Calculating the interpolating polynomial is relatively very computationally expensive Furthermore, polynomial interpolation may not be so exact after all, especially at the endpoints.

a. Lagrange Polynomial:

The Lagrange interpolating polynomial is the polynomial $P(x)$ of degree $(n-1)$ that passes through the n points $(x_1, y_1 = f(x_1))$, $(x_2, y_2 = f(x_2))$, ..., $(x_n, y_n = f(x_n))$, and is given by

$$P(x) = \sum_{j=1}^{n} P_j(x),$$

where

$$P_j(x) = y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}.$$

Written explicitly,

$$P(x) = \frac{(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)} y_1 + \frac{(x-x_1)(x-x_3)\cdots(x-x_n)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_n)} y_2 + \cdots + \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})} y_n.$$

When constructing interpolating polynomials, there is a tradeoff between having a better fit and having a smooth well-behaved fitting function. The more data points that are used in the interpolation, the higher the degree of the resulting polynomial, and therefore the greater oscillation it will exhibit between the data points. Therefore, a high-degree interpolation may be a poor predictor of the function between points, although the accuracy at the data points will be "perfect."

For $n = 3$ points,

$$P(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} y_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} y_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} y_3$$

Note that the function $P(x)$ passes through the points (x_i, y_i), as can be seen for the case $n = 3$,

$$\begin{aligned}
\frac{(x_1 - x_2)(x_1 - x_3)}{(x_1 - x_2)(x_1 - x_3)} y_1 + \frac{(x_1 - x_1)(x_1 - x_3)}{(x_2 - x_1)(x_2 - x_3)} y_2 + \frac{(x_1 - x_1)(x_1 - x_2)}{(x_3 - x_1)(x_3 - x_2)} y_3 &= y_1 \\
\frac{(x_2 - x_2)(x_2 - x_3)}{(x_1 - x_2)(x_1 - x_3)} y_1 + \frac{(x_2 - x_1)(x_2 - x_3)}{(x_2 - x_1)(x_2 - x_3)} y_2 + \frac{(x_2 - x_1)(x_2 - x_2)}{(x_3 - x_1)(x_3 - x_2)} y_3 &= y_2 \\
\frac{(x_3 - x_2)(x_3 - x_3)}{(x_1 - x_2)(x_1 - x_3)} y_1 + \frac{(x_3 - x_1)(x_3 - x_3)}{(x_2 - x_1)(x_2 - x_3)} y_2 + \frac{(x_3 - x_1)(x_3 - x_2)}{(x_3 - x_1)(x_3 - x_2)} y_3 &= y_3.
\end{aligned}$$

Algorithm for the Lagrange Polynomial: To construct the Lagrange polynomial

$$P(x) = \sum_{k=0}^{n} y_k L_{n,k}(x)$$

of degree n, based on the n+1 points (x_k, y_k) for $k = 0, 1, \cdots, n$. The Lagrange coefficient polynomials for degree *n* are:

$$L_{n,k}(x) = \frac{(x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)}$$

for $k = 0, 1, \cdots, n$.

So, for a given *x* and a set of *(N+1)* data pairs, (x_i, f_i), *i= 0, 1, N:*

Set SUM=0

DO FOR *i*=0 to *N*

Set *P*=1

DO FOR *j*=0 to *N*

IF *j~=i*

Set *P=P*(x-x(j))/(x(i)-x(j))*

End DO(*j*)

Lab Task 3. Construct Lagrange interpolating Polynomials for the data points given

in table -1 and Plot the curve.

Function generating code for Lagrange Polynomial-

```
function sum = lagrange_polynomial(x,y)

   n = length(x);
   sum = 0;

   for i = 1:n
      num = 1;
      den = 1;
      for j = 1:n
         if (i~= j)
%              num = conv(num,poly(x(j)));
            num = conv(num,[1 -x(j)]);
            den = den * (x(i) - x(j));
         end
      end
         c = num / den;
         c1 = c* y(i);
         sum = sum + c1;
   end
end
```

Code for Plotting Lagrange Polynomial-

```
clear all
close all
clc

%x = input('X:');
%y = input('Y:');
%Define Data Points
x = [0,1,2,3,4,5,6];
y = [0,0.8415,0.9093,0.1411,-0.7568,-0.9589,-0.2794];

P = lagrange_polynomial(x,y);%lagrange polynomial coefficients
% Y = [];

%Get the approximated values using lagrange polynomial
% for i = x(1):0.1:x(end)
```

```
%     y1 = polyval(P,i);
%     Y = [Y y1];
% end

s = x(1):0.1:x(end);
Y = polyval(P,s);

%Plot the results

plot(s,Y,'linewidth',3),hold on
plot(x,y,'*r','linewidth',3),hold off
```

Note: Run the code for generating the function in a separate tab, and keeping it aside the scripted plotting execution code will be needing to execute in Matlab.

Practice Problems (Experiment - 06)

Problem-1: The following data come from a table that was measured with high precision. Use the best numerical method (for this type of problem) to determine y at x = 3.5 using MATLAB. Note that a polynomial will yield an exact value.

x	0	1.8	5	6	8.2	9.2	12
y	26	16.415	5.375	3.5	2.015	2.54	8

Problem-2: The following data define the sea-level concentration of dissolved Oxygen for fresh water as a function of temperature:

T (℃)	0	8	16	24	32	40
Oxygen (mg/L)	14.621	11.843	9.870	8.418	7.305	6.413

Estimate Oxygen concentration using MATLAB for T = 27℃, considering:

(a) linear interpolation,

(b) Lagrange's interpolating polynomial.

(c) Note that the exact result is 7.986 mg/L. Calculate the percentage error for both of the cases.

Problem-3: Generate eight equally-spaced points from the function using MATLAB:

$$f(t) = sin^2(t)$$

from t = 0 to 2π. Fit these data with a seventh-order interpolating polynomial.