# IIIT Vadodara
# WINTER 2020-21
# MA202 Numerical Techniques
# LAB#8 Interpolation[1]

Interpolation is to connect discrete data points in a plausible way so that one can get reasonable estimates of data points among the given points. The interpolation curve passes through all the data points.

On the other hand, Curve fitting is to find a curve that could best indicate the trend of a given set of data points. In this case, the resultant curve does not necessarily have to pass through the data points. In some scenario, the data may have different accuracy/reliability/uncertainty and we need weighted least-squares curve fitting to process such data.

For a given set of $N+1$ data points $(x_0, y_0), (x_1, y_1), ..., (x_N, y_N)$, we want to find the coefficients of an $N^{th}$ degree polynomial function to match them:

$$P_N(x) = a_0 + a_1 x + a_2 x^2 + ... + a_N x^N \tag{1}$$

**Interpolation by Newton polynomial** The $N^{th}$-degree Newton polynomial matching the $N+1$ data points $(x_0, y_0), (x_1, y_1), ..., (x_N, y_N)$ can be recursively obtained as the sum of the $(N-1)^{th}$ degree Newton polynomial matching the $N$ data points $(x_0, y_0), (x_1, y_1), ..., (x_N, y_N)$ and one additional term.

$$n_N(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + ... = n_{N-1}(x) + a_N(x - x_0)(x - x_1)...(x - x_{N-1}) \tag{2}$$

with $n_0(x) = a_0$

In order to derive a formula to find the successive coefficients $a_0, a_1, ..., a_N$ that make this equation accommodate the data points, we will determine $a_0$ and $a_1$ so that

$$n_1(x) = n_0(x) + a_1(x - x_0) \tag{3}$$

matches the first two data points $(x_0, y_0)$ and $(x_1, y_1)$. We need to solve the two following equations

$$n_1(x_0) = a_0 + a_1(x_0 - x_0) = y_0 \tag{4}$$

and

$$n_1(x_1) = a_0 + a_1(x_1 - x_0) = y_1 \tag{5}$$

to get

$$a_0 = y_0, a_1 = \frac{y_1 - a_0}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \equiv Df_0 \tag{6}$$

Starting from this first-degree Newton polynomial, we can proceed to the second degree Newton polynomial

$$n_2(x) = n_1(x) + a_2(x - x_0)(x - x_1) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) \tag{7}$$

which, with the same coefficients $a_0$ and $a_1$ as (6), still matches the first two data points $(x_0, y_0)$ and $(x_1, y_1)$, since the additional (third) term is zero at $(x_0, y_0)$ and $(x_1, y_1)$. This is to say that the additional polynomial term does not disturb the matching of previously existing data. Therefore, given the additional matching condition for the third data point $(x_2, y_2)$, we only have to solve

$$n_2(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \equiv y_2 \tag{8}$$

[1] submission deadline : $28^{th}$ March 11 PM

for only one more coefficient $a_2$ to get

$$\frac{Df_1 - Df_0}{x_2 - x_0} \equiv D^2 f_0 \tag{9}$$

Generalizing these results (6) and (9) yields the formula to get the $N^{th}$ coefficient $a_N$ of the Newton polynomial function (2) as

$$a_N = \frac{D^{N-1} f_1 - D^{N-1} f_0}{x_N - x_0} \equiv D^N f_0 \tag{10}$$

This is the divided difference, which can be obtained successively from the second row of Table 1.

Table 1: Divided Difference Table

| $x_k$ | $y_k$ | $Df_k$ | $D^2 f_k$ | $D^3 f_k$ | - |
|---|---|---|---|---|---|
| $x_0$ | $y_0$ | $Df_0 = \frac{y_1 - y_0}{x_1 - x_0}$ | $D^2 f_0 = \frac{Df_1 - Df_0}{x_2 - x_0}$ | $D^3 f_0 \frac{D^2 f_1 - D^2 f_0}{x_3 - x_0}$ | - |
| $x_1$ | $y_1$ | $Df_1 = \frac{y_2 - y_1}{x_2 - x_1}$ | $D^2 f_1 = \frac{Df_2 - Df_1}{x_3 - x_1}$ | - | |
| $x_2$ | $y_2$ | $Df_2 = \frac{y_3 - y_2}{x_3 - x_2}$ | | - | |
| $x_3$ | $y_3$ | - | | | |

Q. 1: Find and draw a Newton polynomial matching the following data points
x: [2 -1 1 2 4]
y: [6 0 0 6 60]

Q. 2: Given the following data:
x: [0.8 1.4 2.7 3.8 4.8 4.9]
y: [0.69 1.00 2.00 2.39 2.34 2.83]
Find a Newton polynomial curve passing through the data.

Q. 3: Explore the MATLAB commands related to all interpolation techniques like interp1, spline, pchip, makima, ppval. Use at least two different MATLAB fuctions to interpolate the data in Q.1 and Q.2

**Divided difference algorithm**

A pseudocode for computing divided difference is

real array $(a_{ij})_{0:n \times 0:n}$
integer i,j,n
for i = 0 to n do
$(a_{i0}) \leftarrow$f(x$_i$) = $y_0$
end for
for j = 1 to n do
for i = 0 to n-j do
$(a_{ij}) \leftarrow ((a_{i+1,j-1}) - a_{i,j-1})/(x_{i+j} - x_i)$
end for
end for

This algorithm computes and stores all components of the divided difference. The coefficients

of the Newton interpolating polynomial are stored in the first row of the array $(a_{ij})_{0:n\times 0:n}$ i.e., in $a(0:n,0)$.

From the given data points, construct the divided difference table as Table 1 using the pseudocode mentioned above. Then use the Table together with equation (2) to get the Newton polynomial. Plot the polynomial over the given points. This process should be casted as a MATLAB function. Note that the Newton polynomial does not depend on the order of the data points; that is, changing the order of the data points does not make any difference.