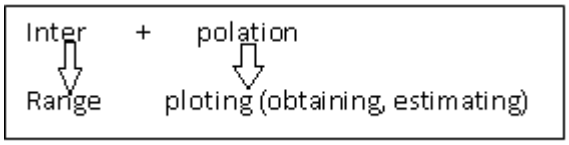
**Unit 2: Chapter 3:**
**Interpolation:** Introduction, Lagrange interpolation, Difference Tables- Newton-Gregory Forward and Backward interpolation, Truncation error in interpolation.

**Interpolation and extrapolation:**

Introduction:



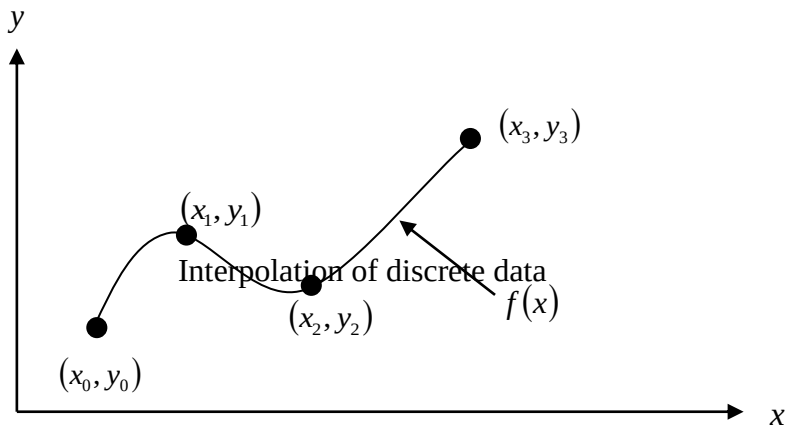

A function may be expressed as a table of values instead of as a closed form expression such Bessel Function. In these cases the value of the function *f(x)* is tabulated at a discrete set of values of the argument x as shown in the table below:

| X | $X_1$ | $X_2$ | $X_3$ | $X_4$ | …….. | $X_n$ |
|---|---|---|---|---|---|---|
| *f(x)* | $f(X_1)$ | $f(X_2)$ | $f(X_3)$ | $f(X_4)$ | …….. | $f(X_n)$ |

If the value of *f(x)* is to be found at some point y in the interval (x1, xn) and y is not one of the tabulated points then the value is estimated by using the known values of *f(x)* at the surrounding points. This is called Interpolation.
If the point is outside the interval (x1, xn) then the estimation of *f(y)* is called extrapolation.
So what kind of function $f(x)$ should one choose? A polynomial is a common choice for an interpolating function because polynomials are easy to

(A) evaluate,
(B) differentiate, and
(C) integrate,

relative to other choices such as a trigonometric and exponential series.




Definition:

Extrapolation: Extrapolation is an estimation of a value based on extending a known sequence of values or facts beyond the area that is certainly known. The estimation of the value of a function is extrapolation. In a general sense, to extrapolate is to infer something that is not explicitly stated from existing information.

Interpolation : Interpolation is an estimation of a value within two known values in a sequence of values. When graphical data contains a gap, but data is available on either side of the gap or at a few specific points within the gap, interpolation allows us to estimate the values within the gap. In other words, In the **mathematical** field of numerical analysis, **interpolation** is a method of constructing new data points within the range of a discrete set of known data points.

Types of interpolation:

1. Linear interpolation : **linear interpolation** is a method of curve fitting using linear polynomials to construct new data points within the range of a discrete set of known data points.
2. Polynomial interpolation : Polynomial interpolation is a generalization of linear interpolation. Note that the linear interpolant is a linear function. We replace this interpolant with a polynomial of higher degree. Polynomial interpolation is a method of estimating values between known data points. Polynomial interpolation involves finding a polynomial of order $n$ that passes through the $n+1$ data points. One of the methods used to find this polynomial is called the Lagrangian method of interpolation. Other methods include Newton's divided difference polynomial method and the direct method.
3. Spline interpolation: **spline interpolation** is a form of interpolation where the interpolant is a special type of piecewise polynomial called a spline.
4. Multivariate interpolation : In numerical analysis, **multivariate interpolation** or **spatial interpolation** is interpolation on functions of more than one variable.

Interpolation techniques:

- Newton-Gregory forward difference formulae
- Newton-Gregory backward difference formulae
- Gauss's forward
- Gauss's backward
- Bessel's formula etc.

Application of interpolation:

- Business
- Statistics
- Scatter data analysis
- Correlation and Regression
- Engineering etc

Finite difference: A **finite difference** is a mathematical expression of the form f(x + b) − f(x + a). If a **finite difference** is divided by b − a, one gets a **difference** quotient. **Finite differences** play a key role in the solution of differential equations and in the formulation of **interpolating** polynomials.

| X : X0 | X1=X0+h, | X2=X0+2h | X3=X0+3h | ……. | Xr=X0+rh | …Xn=X0+nh |
|---|---|---|---|---|---|---|
| *f(x):y0* | Y1 | Y2 | Y3 | …. | Yr | …..yn |

Where h is constant.

**Forward, backward, and central differences**

Three forms are commonly considered: forward, backward, and central differences.

A **forward difference** is an expression of the form

$$\Delta_h[f](x) = f(x+h) - f(x).$$

Depending on the application, the spacing $h$ may be variable or constant. When omitted, $h$ is taken to be 1: $\Delta[f](x) = \Delta_1[f](x)$.

A **backward difference** uses the function values at $x$ and $x - h$, instead of the values at $x + h$ and $x$:

$$\nabla_h[f](x) = f(x) - f(x-h).$$

Finally, the **central difference** is given by

$$\delta_h[f](x) = f(x + \tfrac{1}{2}h) - f(x - \tfrac{1}{2}h)\,.$$

**Truncation error in interpolation:**

In numerical analysis and scientific computing, truncation error is the error made by truncating an infinite sum and approximating it by a finite sum. For example, in numerical methods for ordinary differential equations, the continuously varying function that is the solution of the differential equation is approximated by a process that progresses step by step, and the error that this entails is a discretization or truncation error.

Truncation errors in numerical integration are of two kinds:

- *local truncation errors* – the error caused by one iteration, and
- *global truncation errors* – the cumulative error caused by many iterations.

Suppose we have a continuous differential equation

$$y' = f(t,y), \qquad y(t_0) = y_0, \qquad t \geq t_0$$

and we wish to compute an approximation $y_n$ of the true solution $y(t_n)$ at discrete time steps $t_1, t_2, \ldots, t_N$. For simplicity, assume the time steps are equally spaced:

$$h = t_n - t_{n-1}, \qquad n = 1, 2, \ldots, N.$$

Suppose we compute the sequence $y_n$ with a one-step method of the form

$$y_n = y_{n-1} + hA(t_{n-1}, y_{n-1}, h, f).$$

The function $A$ is called the *increment function*, and can be interpreted as an estimate of the slope of $y_n$.

The **local truncation error** $\mathbf{T_n}$, is the error that our increment function, **A,** causes during a single iteration, assuming perfect knowledge of the true solution at the previous iteration.

The **global truncation error** is the accumulation of the *local truncation error* over all of the iterations, assuming perfect knowledge of the true solution at the initial time step.

**Lagrange interpolation**: In numerical analysis, Lagrange polynomials are used for polynomial interpolation. For a given set of distinct points and numbers, the Lagrange polynomial is the polynomial of lowest degree that assumes at each point the corresponding value (i.e. the functions coincide at each point).
Lagrange polynomials are used for polynomial interpolation. For a given set of distinct points $x_j$ and numbers $y_j$. Lagrange's interpolation is also an $N_{th}$ degree polynomial approximation to $f(x)$.

**Lagrange Interpolation Formula** :

$$y = \frac{(x-x_1)(x-x_2).....(x-x_n)}{(x_0-x_1)(x_0-x_2).....(x_0-x_n)}y_0 + \frac{(x-x_0)(x-x_2).....(x-x_n)}{(x_1-x_0)(x_1-x_2).....(x_1-x_n)}y_1 + .. + \frac{(x-x_1)(x-x_1).....(x-x_{n-1})}{(x_n-x_0)(x_0-x_1).....(x_n-x_{n-1})}y_n$$

- **ADVANTAGES**
- The formula is simple and easy to remember.
- There is no need to construct the divided difference table.
- The application of the formula is not speedy.
- **DISADVANTAGE**
- There is always a chance to committing some error.
- The calculation provide no check whether the functional values used the taken correctly or not.

**Lagrange interpolation algorithm:**

1. Read x, n
2. For i = 1 to (n+1) in steps of 1 do Read $x_i$, $f_i$ end for
    a. Remarks: the above statement reads $x_i$s and the corresponding values of $f_i$s.
3. Sum <- 0
4. For i = 1 to (n+1) in steps of 1 do
5. Prodfunc <- 1
6. For j=1 to (n+1) in steps of 1 do
    a. If($j \neq i$) then
    b. Prodfun <- prodfunc X $(x-x_j)/(x_i-x_j)$
    c. End for
7. Sum <- sum + $f_i$ X prodfun
    a. remarks: sum is the value of f at x end for
8. write x, sum
9. stop

**Difference tables:**

A difference table is made by listing the terms of a sequence and its differences. It includes the first differences, which is a sequence that lists the differences of two consecutive terms of the original sequence. For instance, the first term of the first differences is the difference between the first and second term of the original sentence, the second term of the first

difference is the difference between the second and third term of the original sentence, and so forth.

Second-order differences, which is a sequence of differences of the first differences, and other higher-order differences can also be included in the difference table.

An important thing to notice is that we find the differences by subtracting the earlier term from the later term of the sequence and not by subtracting the term with the smaller value from the term with the larger value. Thus, it is possible to have a negative number in the difference table.

For example, let's create a difference table for the sequence of perfect squares :

$$0, 1, 4, 9, 16, 25, 36, \ldots$$

First, list the terms of the sequence on the top row. Then write each difference of two consecutive terms underneath and in between the terms it is the difference of. For instance, the terms $1, 3, 5, 7, \cdots$ in the first differences are found by $1 - 0 = 1, 4 - 1 = 3, 9 - 4 = 5, 16 - 9 = 7, \ldots$. We can also create another row of second-order differences that lists the differences of two consecutive terms from the first sequence of differences.

| **Original Sequence** | **0** | | **1** | | **4** | | **9** | | **16** | | **25** | | **36** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **First Difference** | | **1** | | **3** | | **5** | | **7** | | **9** | | **11** | |
| **Second Difference** | | | **2** | | **2** | | **2** | | **2** | | **2** | | |
| **Third Difference** | | | | **0** | | **0** | | **0** | | **0** | | ... | |

**Newton Forward interpolation**:
Lagrange Interpolation has a number of disadvantages:

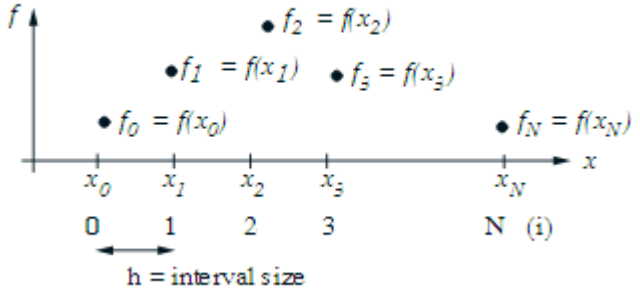
- The amount of computation required is large
- Interpolation for additional values of requires the same amount of effort as the first value (i.e. no part of the previous calculation can be used)
- When the numbers of interpolation points are changed (increased/decreased), the results of the previous computations cannot be used
- Error estimation is difficult (at least may not be convenience).

Newton Interpolation which is based on developing difference tables for a given set of data points.

- The degree interpolating polynomial obtained by fitting data points will be identical to that obtained using Lagrange formulae.
- Newton interpolation is simply another technique for obtaining the same interpolating polynomial as was obtained using the Lagrange formulae.

**Forward Difference Tables**

- We assume equi-spaced points (not necessary)




$\Delta^0 f_i \equiv f_i$ (Zero$^{th}$ order forward difference)

$\Delta f_i \equiv f_{i+1} - f_i$ (First order forward difference)

$\Delta^2 f_i \equiv \Delta f_{i+1} - \Delta f_i$ (Second order forward difference)

$\Delta^2 f_i = (f_{i+2} - f_{i+1}) - (f_{i+1} - f_i)$

$\Delta^2 f_i = f_{i+2} - 2f_{i+1} + f_i$

$\Delta^3 f_i \equiv \Delta^2 f_{i+1} - \Delta^2 f_i$ (Third order forward difference)

$\Delta^3 f_i = (f_{i+3} - 2f_{i+2} + f_{i+1}) - (f_{i+2} - 2f_{i+1} + f_i)$

$\Delta^3 f_i = f_{i+3} - 3f_{i+2} + 3f_{i+1} - f_i$

$\Delta^k f_i = \Delta^{k-1} f_{i+1} - \Delta^{k-1} f_i$ ($k^{th}$ order forward difference)

Typically we set up a difference table

| $i$ | $f_i$ | $\Delta f_i$ | $\Delta^2 f_i$ | $\Delta^3 f_i$ | $\Delta^4 f_i$ |
|---|---|---|---|---|---|
| 0 | $f_o$ | $\Delta f_o = f_1 - f_o$ | $\Delta^2 f_o = \Delta f_1 - \Delta f_o$ | $\Delta^3 f_o = \Delta^2 f_1 - \Delta^2 f_o$ | $\Delta^4 f_o = \Delta^3 f_1 - \Delta^3 f_o$ |
| 1 | $f_1$ | $\Delta f_1 = f_2 - f_1$ | $\Delta^2 f_1 = \Delta f_2 - \Delta f_1$ | $\Delta^3 f_1 = \Delta^2 f_2 - \Delta^2 f_1$ | |
| 2 | $f_2$ | $\Delta f_2 = f_3 - f_2$ | $\Delta^2 f_2 = \Delta f_3 - \Delta f_2$ | | |
| 3 | $f_3$ | $\Delta f_3 = f_4 - f_3$ | | | |
| 4 | $f_4$ | | | | |

- Note that to compute higher order differences in the tables, we take forward differences of previous order differences instead of using expanded formulae.
- The order of the differences that can be computed depends on how many total data points, $x_o, \ldots, x_N$, are available

- *$N+1$ data points can develop up to $N^{th}$ order forward differences*

**Algorithm to implement the Newton Gregory forward interpolation**

Step1: START

Step2: Read n

Step3: for i=0 to (n-1) do read xi,yi

Step4: read x

Step5: h←xi-x0

Step6: p←(x-xo)/n

Step7: for j=0 to n-2 do Δ1yj←yj+1-Δi-1

Step8: k←n-2

Step9: for i=2 to (n-1)do
Step9.1: k←k-1
Step9.2:for j=0 to k do Δiyj←Δi-1 yj+1-Δi-1yj

Step10: Sumy←y0

Step11: Pvalue←1

Step12: Fact value←1

Step13: for l=1 to (n-1) do
Step13.1: Pvalue←pvalue x (p-(l-1))
Step13.2: factvalue←factvaluex1
Step13.3: term←(pvalue x Δly) / factvalue
Step13.4: Sumy←Sumy+term

Step14: Print x,SUMY

Step15: STOP

## Newton Backward Interpolation

- Newton backward interpolation is essentially the same as Newton forward interpolation except that backward differences are used

- Backward differences are defined as:

$$\nabla^0 f_i \equiv f_i \quad \text{Zero}^{th} \text{ order backward difference}$$

$$\nabla f_i = f_i - f_{i-1} \quad \text{First order backward difference}$$

$$\nabla^2 f_i = \nabla f_i - \nabla f_{i-1} \quad \text{Second order backward difference}$$

$$\nabla^k f_i = \nabla^{k-1} f_i - \nabla^{k-1} f_{i-1} \quad k^{th} \text{ order backward difference}$$

- For $N+1$ data point which are fitted with an $N^{th}$ degree polynomial

$$g(x) = f_N + (x - x_N)\frac{\nabla f_N}{h} + \frac{1}{2!}(x - x_N)(x - x_{N-1})\frac{\nabla^2 f_N}{h^2}$$

$$+ \frac{1}{3!}(x - x_N)(x - x_{N-1})(x - x_{N-2})\frac{\nabla^3 f_N}{h^3}$$

$$+ \frac{1}{N!}(x - x_N)(x - x_{N-1})\ldots(x - x_1)\frac{\nabla^N f_N}{h^N}$$

- Note that we are really expanding about the right most point to the left. Therefore we must develop $f_N$, $\nabla f_N$ etc. in the difference table