INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

APPLICATIONS OF POLYNOMIAL INTERPOLATION

REPORT BY: Suman Mitra

ROLL NO.: 2020202018

DEPARTMENT: CSIS (PG1-2020)

SUBJECT: LINEAR ALGEBRA

PROJECT GROUP: 8

DATE: 21st Nov, 2020

ABSTRACT:

In this report, I will be focusing on one of the computation methods for polynomial interpolation: **Lagrange Polynomials.** Other than this, I will also briefly explain the **Vandermonde Method** and **Newton Polynomials**, though they will be covered in details by other team members.

The applications of **Polynomial Interpolation** like **Karatsuba** multiplication**, Tom-Cook** multiplication, Secure Multi Party Computation **(SMPC)**, **Secret Sharing** algorithm etc. will be again covered by other team members.

1. INTRODUCTION

Polynomial interpolation is a method of estimating values between known data points. 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, an estimate of values within the gap can be made by interpolation.

The simplest method of interpolation is to draw straight lines between the known data points and consider the function as the combination of those straight lines. This method, called **linear interpolation**, usually introduces considerable error. Therefore, a more precise approach uses a **polynomial** function to connect the points. Polynomials are used to approximate complicated curves.

There are mainly three techniques for finding interpolating polynomials:

- The Vandermonde method. (easy to calculate and generalize)
- Lagrange polynomials. (easy to find manually)
- Newton polynomials. (efficient to implement when using Horner's rule)

In this report, I will mainly focus on the **Lagrange polynomials**. I will also briefly touch upon the **Vandermonde method** and **Newton polynomials**, though these topics will be covered by other team members in details.

2. POLYNOMIAL INTERPOLATION AS A LINEAR PROBLEM:

Definition: Given points $\boldsymbol{X = (x_0, x_1, \dots, x_n)}$ which are pair-wise disjoint (i.e. no two points are equal) and a function $\boldsymbol{f}$ defined at these points. We can find a polynomial $\boldsymbol{p} \in \boldsymbol{\Pi}_n$ (where $\boldsymbol{\Pi}_n$ = Set of all polynomials with degree equal to or less than $\boldsymbol{n}$) such that:

$$\boldsymbol{p(x_i) = f(x_i);\ \forall\ 0 \le i \le n}$$

Though the polynomial $\boldsymbol{p}$ may not fit the function $\boldsymbol{f}$ very well outside the given points If we choose to represent $\boldsymbol{p}$ as a linear combination of monomials, then the problem of finding $\boldsymbol{p}$ is equivalent to solving a linear problem with $\boldsymbol{n+1}$ variables (the coefficients of the monomials) and $\boldsymbol{n+1}$ constraints:

$$\boldsymbol{p(t) = a(1)t^n + a(2)t^{n-1} + \cdots + a(n)t + a(n+\ 1)}$$

We can rewrite this system of equations as:

$$\boldsymbol{V\vec{a} = \vec{F}} \qquad \dots \boldsymbol{(1)}$$

where, $\boldsymbol{\vec{a} = \ (a(1), a(2), \dots, a(n+1))^T}$; $\boldsymbol{\vec{F} = \left(f(x_0), \dots, f(x_n)\right)^T}$ and

$$V = \begin{bmatrix} (x_0)^n & (x_0)^{n-1} & \dots & (x_0)^1 & (x_0)^0 \\ (x_1)^n & (x_1)^{n-1} & \dots & (x_1)^1 & (x_1)^0 \\ \vdots & \vdots & & \vdots & \vdots \\ (x_{n-1})^n & (x_{n-1})^{n-1} & \dots & (x_{n-1})^1 & (x_{n-1})^0 \\ (x_n)^n & (x_n)^{n-1} & \dots & (x_n)^1 & (x_n)^0 \end{bmatrix}$$

The matrix V formed here is known as **Vandermonde Matrix.**

2.1. THE VANDERMONDE METHOD

The Vandermonde method is the most straight-forward method which may be used to find an interpolating polynomial. The substitution of the points into the desired polynomial yields a system of linear equations in the coefficients of the polynomial which may then be solved using Gaussian elimination or PLU decomposition. This technique also generalizes for interpolating non-polynomial functions or points in more than two dimensions.

Suppose we have 3 points (2, 5), (3, 6), (7, 4) and we want to fit a quadratic polynomial through these points. The general form of a quadratic polynomial is:

$p(x) = c_1 x^2 + c_2 x + c_3$.

Thus, if we were to simply evaluate $p(x)$ at these three points, we get three equations:

$$p(2) = c_1 4 + c_2 2 + c_3 = 5$$

$$p(3) = c_1 9 + c_2 3 + c_3 = 6$$

$$p(7) = c_1 49 + c_2 7 + c_3 = 4$$

Therefore, we may draw the obvious generalization that given n points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, we can find a polynomial of degree $n - 1$ which passes through those points by doing the following:

- Writing down the general polynomial of degree $n - 1$.
- Evaluating the polynomial at the points $x_1, \dots, x_n$.
- Solving the resulting system of linear equations.

Rather than performing all of these operations, we will simply write down the problem in the form $Vc = y$ where y is the vector of y values, **c** is the vector of coefficients, and V is th**e Vandermonde matrix**.

2.2. DRAWBACKS OF THIS APPROACH

Transforming the problem of finding a polynomial p into an equivalent linear problem has helped us to understand polynomial interpolation. However, it has some significant drawbacks as a method for solving polynomial interpolation problems. Those drawbacks are:

- Solving a linear system of equations takes a significant amount of time.
- This system of linear equations is often ill-conditioned and prone to round-off errors. In other words, although the **Vandermonde** matrix $\boldsymbol{V}$ is **non-singular**, it may be close to **singular**, which makes it hard to solve accurately on a computer.
- It is not well-suited to a common kind of exploratory analysis. Suppose we are given $(\boldsymbol{x_0}, \boldsymbol{y_0}), \dots, (\boldsymbol{x_n}, \boldsymbol{y_n})$. We solve $\boldsymbol{Va} = \boldsymbol{y}$, and we get a solution $\boldsymbol{a}$. Suppose that we do not like the solution for some reason, so we add a new point $(\boldsymbol{x_{n+1}}, \boldsymbol{y_{n+1}})$. Now we make a new system $\boldsymbol{V_2 a_2} = \boldsymbol{y_2}$, and solve for $\boldsymbol{a_2}$. We have to do this from scratch. We cannot reuse all the computation we already invested to solve $\boldsymbol{Va} = \boldsymbol{y}$. If $\boldsymbol{n}$ is large, this could be very bad.

This equivalence between solving a linear problem and finding a polynomial $\boldsymbol{p}$ rests on our representation of $\boldsymbol{p}$ as a linear combination of monomials. To find a better method, we should consider alternate representations. We shall mainly talk about one such class of representations for polynomial interpolation which are called **Lagrange Polynomials**.

3. INTERPOLATION BY LAGRANGE POLYNOMIALS

Given the $\boldsymbol{n+1}$ interpolation points $\boldsymbol{x_0}, \boldsymbol{x_1}, \dots, \boldsymbol{x_n}$, suppose we can find $\boldsymbol{n+1}$ polynomials $\boldsymbol{l_0(t)}, \boldsymbol{l_1(t)}, \dots, \boldsymbol{l_n(t)}$ of degree $\leq \boldsymbol{n}$. These polynomials are fixed and independent of the function values $\boldsymbol{f(x_0)}, \boldsymbol{f(x_1)}, \dots, \boldsymbol{f(x_n)}$, and satisfy the following condition:

$$\boldsymbol{l_i(x_j)} = \begin{cases} 1; \; if \; i = j \\ 0; \; if \; i \neq j \end{cases} \qquad \dots \boldsymbol{(2)}$$

For example, given $\boldsymbol{x_0}$ and $\boldsymbol{x_1}$, two such polynomials are shown in Figure 1.

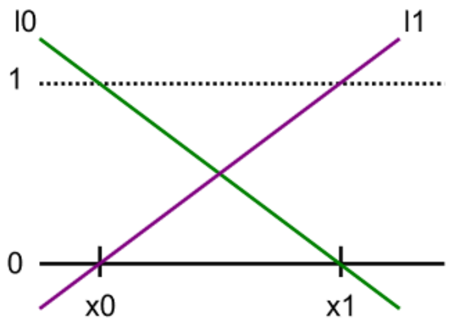

Figure 1: Lagrange Polynomials ℓ_0, ℓ_1 for two points x_0, x_1

3.1. USING LAGRANGE POLYNOMIAL FOR INTERPOLATION

Any polynomial $\boldsymbol{p} \in \boldsymbol{\Pi_n}$ can be represented as a linear combination of $\boldsymbol{n+1}$ Lagrange Polynomials of degree $\leq \boldsymbol{n}$. To illustrate the use of Lagrange Polynomials, consider the following example:

Example-1: Given the points $\vec{x} = (2\ \ 6\ \ 7)^T$, and the corresponding function values $\vec{F} = (-1\ \ 8\ \ -3)^T$, we have to find an interpolating polynomial $p \in \Pi_2$.

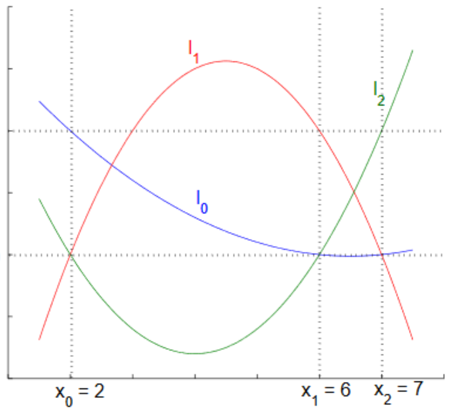

Figure 2: Quadratic Lagrange Polynomials ℓ_0, ℓ_1, ℓ_2 for $\vec{x} = (2\ 6\ 7)^T$

Consider that, if we had three quadratic Lagrange polynomials $l_0(t), l_1(t), l_2(t)$ (as shown in **Figure 2**) with:

$$l_0\left(\vec{x}\right) = (1\ \ 0\ \ 0)^T$$

$$l_1\left(\vec{x}\right) = (0\ \ 1\ \ 0)^T$$

$$l_2\left(\vec{x}\right) = (0\ \ 0\ \ 1)^T$$

If we multiply each polynomial l_i by the corresponding function value $f(x_i)$ and then add them together, the resulting polynomial will match f at each of the input points. So, we can define p as:

$$p(t) = -1.\,l_0(t) + 8.\,l_1(t) - 3.\,l_2(t)$$

Evaluating p at the input point $x_0 = 2$, we get:

$$\begin{aligned} p(2) &= -1(l_0(2)) + 8(l_1(2)) - 3(l_2(2)) \\ &= -1(1) + 8(0) - 3(0) \\ &= -1 \end{aligned}$$

In general, given a set of input points $x_0, x_1, \ldots, x_n$, the corresponding points on the graph of the function $f, f(x_0), f(x_1), \ldots, f(x_n)$, and the Lagrange polynomials for our input points $l_0(t), l_1(t), \ldots, l_n(t)$, we can define the following polynomial $p \in \Pi_n$ that interpolates f:

$$p(t) = \sum_{i=0}^{n} f(x_i) l_i(t) \qquad \ldots(3)$$

What we have actually done is, initially, when we first formulated the problem of polynomial interpolation, we did not specify any particular representation of polynomials. When we started to do computation, we needed a computationally accessible definition, so we choose the most natural basis that we all are familiar with: **monomials**. For example, we used the monomials t^0, t^1, t^2 as a basis for Π_2. But this is only one basis. ***The Lagrange polynomials also form a basis for the space of polynomials. The difference is that the Lagrange basis is chosen so as to make our problem easier.***

3.2. FINDING LAGRANGE POLYNOMIALS

So, we can find a polynomial l_i given its roots and a point x_i with $l_i(x_i)$ = 1. Consider that, if a polynomial has roots r_1, r_2, then it must have factors $(t - r_1)$ and $(t - r_2)$. We can get our polynomial by first multiplying the factors together, obtaining a polynomial with the correct roots. We can then multiply by a constant scaling factor so that $l_i(x_i)$ = 1.

To illustrate, consider **Example-1**. Since l_0 is **0** at **6** and **7**, it must have factors (t **- 6**) and (t **- 7**). If we consider (t **- 6**). (t **- 7**) evaluated at x_0 = 2, we see that its value will be (2−6).(2−7). So, if we divide by this value, we will have the correct polynomial:

$$l_0(t) = \frac{(t-6).(t-7)}{(2-6).(2-7)}$$

We can define l_1 and l_2 similarly:

$$l_1(t) = \frac{(t-2).(t-7)}{(6-2).(6-7)}$$

$$l_2(t) = \frac{(t-2).(t-6)}{(7-2).(7-6)}$$

In general, we can calculate each Lagrange polynomial l_i for a set of input points $x_0, x_1, \dots, x_\mathrm{n}$ as:

$$l_\mathrm{i}(t) = \prod_{j=0, j\neq i}^{n} \left(\frac{(t - x_j)}{(x_i - x_j)} \right)$$

3.3. ANALYSIS OF INTERPOLATION BY LAGRANGE POLYNOMIALS

So we have seen that we can solve a polynomial interpolation problem quite easily using a linear combination of Lagrange polynomials. We will now analyze how easy it is to use the resulting polynomial for numerical computation. For example:

We might argue that evaluating the polynomial is fairly straight forward. However, it is easy to see that evaluating the derivative could be very complicated (consider using the product rule on n terms with n factors each).

Thus, with Lagrange polynomials, we have struck on a polynomial representation that makes it easy to solve polynomial interpolation problems, but potentially difficult to use once the problem has been solved.

Now we will see that how important it is that the polynomial representation we end up with is easy to use. Consider that we have already used polynomial interpolation without knowing it, when we approximated the derivative of a function.

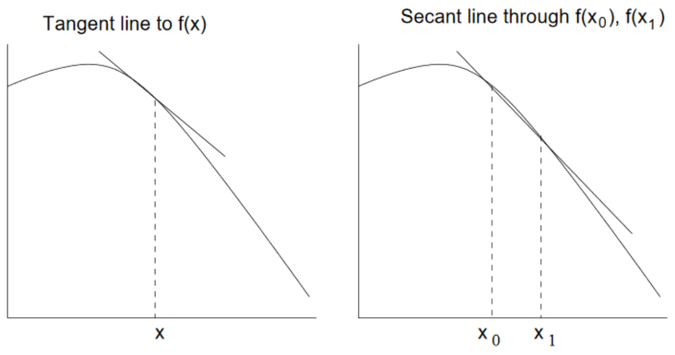

Figure 3: Approximating the Derivative Using a Secant Line

Recalling that the derivative of a function $\boldsymbol{f}$ at a point $(\boldsymbol{x}, \boldsymbol{f}(\boldsymbol{x}))$ is simply the slope of the tangent line at that point. We can approximate the tangent line by taking two points $(\boldsymbol{x_0}, \boldsymbol{f}(\boldsymbol{x_0})), (\boldsymbol{x_1}, \boldsymbol{f}(\boldsymbol{x_1}))$, and interpolating a degree $\leq \mathbf{1}$ polynomial (a secant line) that approximates the function between these two points (as shown in **Figure-3**). We can then approximate the slope at $\boldsymbol{x}$ by taking the derivative of the secant line. If we look at this problem as a polynomial interpolation problem, we might get a better estimate by interpolating over a larger number of points. In cases like this, clearly it is important that the representation of the resulting polynomial is easy to use.

3.4. UNIQUENESS OF SOLUTIONS

The process of interpolation by Lagrange polynomials shows that we can always find a solution to a polynomial interpolation problem.

Recalling that polynomial interpolation is equivalent to solving the linear problem:

$$\boldsymbol{V}\vec{\mathbf{a}} = \vec{\mathbf{F}} \qquad \dots (\mathbf{5})$$

From linear algebra, we know that the solution to this problem hinges on whether or not the matrix $\boldsymbol{V}$ is singular. If $\boldsymbol{V}$ is singular, then whether or not the equation has a solution depends on $\vec{\boldsymbol{F}}$:

$\boldsymbol{V}$ Is non-singular $\Rightarrow \forall \vec{\boldsymbol{F}}$ equation 5 has a unique solution.

$$\boldsymbol{V} \text{ is singular} \Rightarrow \begin{cases} \exists \vec{\mathbf{F}}\ such\ that\ eq.5\ has\ no\ solution \\ \exists \vec{\mathbf{F}}\ such\ that\ eq.5\ has\ infinitely\ many\ colutions \end{cases}$$

Now, since we know that we can always solve a polynomial interpolation problem using Lagrange polynomials, we know that equation 5 always has a solution, regardless of $\vec{\boldsymbol{F}}$. Since this is true, $\boldsymbol{V}$ cannot be singular, and therefore, we know that not only does equation 5 always have a solution, but that solution is always unique.

4. NEWTON POLYNOMIALS

With Lagrange polynomials, we found the interpolating polynomial by taking each point $\boldsymbol{x}_{\mathrm{i}}$ and finding a polynomial $\boldsymbol{y}_{\mathrm{i}}\boldsymbol{L}_{\mathrm{i}}(\boldsymbol{x})$ with the property that:

- $\boldsymbol{y}_{\mathrm{i}}\boldsymbol{L}_{\mathrm{i}}(\boldsymbol{x}_{\mathrm{i}}) = \boldsymbol{y}_{\mathrm{i}}$, and
- $\boldsymbol{y}_{\mathrm{i}}\boldsymbol{L}_{\mathrm{i}}(\boldsymbol{x}_{\mathrm{j}}) = \mathbf{0}$ for $\boldsymbol{j} \neq \boldsymbol{i}$

This technique is the most easily implemented by humans (at least for linear and quadratic interpolating polynomials), but is also the least useful programmatically. Also, if we found the Lagrange polynomial passing through the points $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_{\mathrm{n}}, \boldsymbol{y}_{\mathrm{n}})$, this gives us very little information for finding the Lagrange polynomial which passes through these $\boldsymbol{n}$ points together with a new point $(\boldsymbol{x}_{\mathrm{n+1}}, \boldsymbol{y}_{\mathrm{n+1}})$.

Let us, however, use this idea: given $\boldsymbol{n}$ points, how do we find the polynomial which passes through the given $\boldsymbol{n}$ points and another point $(\boldsymbol{x}_{\mathrm{n+1}}, \boldsymbol{y}_{\mathrm{n+1}})$?

To demonstrate this, we will find the interpolating quadratic polynomial which passes through the three points in Figure 1.

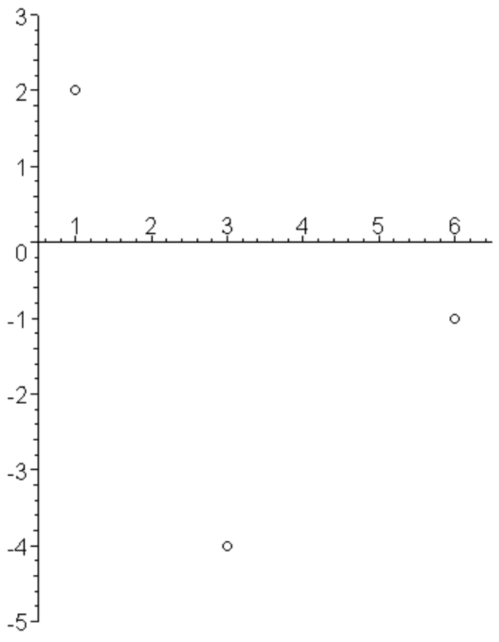

Figure 1. Three points to be interpolate.

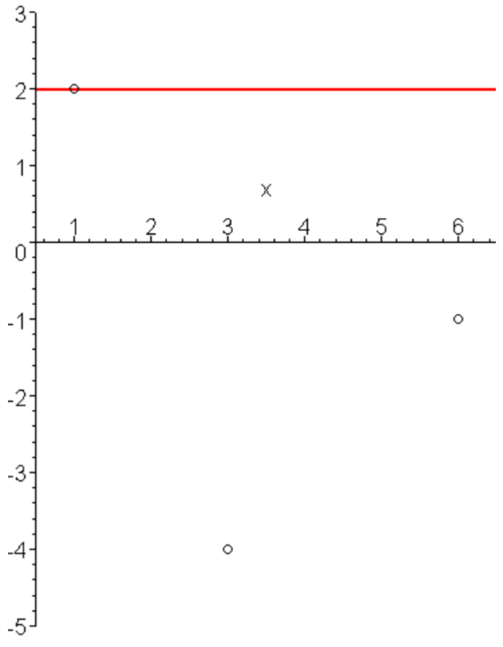

Figure 2. Constant interpolant through the first point.

We will do this by finding a sequence of polynomials $\boldsymbol{p_0(x)}$, $\boldsymbol{p_1(x)}$, and $\boldsymbol{p_2(x)}$ of degrees 0, 1, and 2 where the last will interpolate all three points.

Constant (n = 1):

To do this, let us start from the beginning with a single point $(\boldsymbol{x_1}, \boldsymbol{y_1})$. The polynomial passing through this point is $\boldsymbol{p_0(x) = y_1}$. For sample points, $\boldsymbol{p_0(x) = 2}$ is shown above in Figure 2.

Linear (n = 2):

Having found $\boldsymbol{p_0(x)}$, we will find $\boldsymbol{p_1(x)}$ by defining a polynomial $\Delta\boldsymbol{p_1(x)}$ such that:

- $\boldsymbol{\Delta p_1(x_1) = 0}$
- $\boldsymbol{\Delta p_1(x_2) = y_2 - p_0(x_2)}$

The first requirement makes it quite clear that the delta-polynomial must be of the form $\boldsymbol{\Delta p_1(x) = c_2(x - x_1)}$. We will use the second condition to determine $\boldsymbol{c_2}$:

$$\boldsymbol{c_2(x_2 - x_1) = y_2 - y_1}$$

Dividing through by $\boldsymbol{x_2 - x_1}$, we get:

$$\boldsymbol{c_2 = \frac{(y_2 - y_1)}{(x_2 - x_1)}}$$

Given the example points, $\boldsymbol{c_2 = (-4 - 2)/(3 - 1) = -3}$, and Figure 3 shows $\boldsymbol{p_0(x) = 2}$ in blue, $\boldsymbol{\Delta p_1(x) = 2 - 3(x - 1)}$ in magenta, and summing these together, we get that $\boldsymbol{p_1(x)}$ passes through both the first and second points.

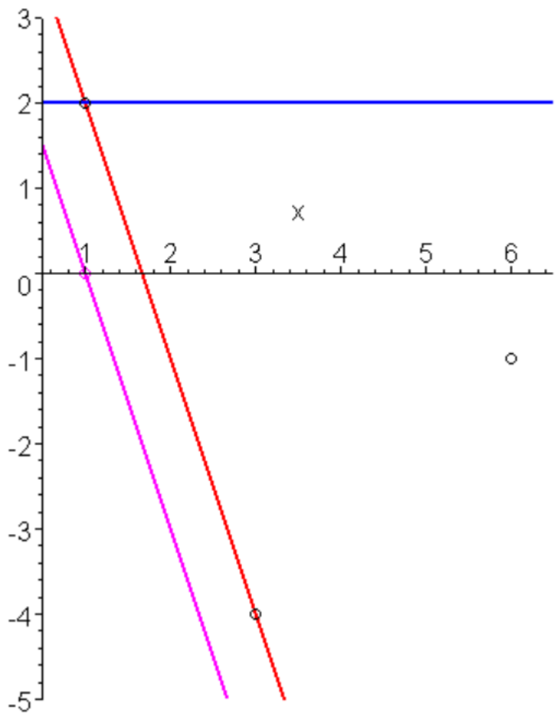

Figure 3. Linear interpolant through the first and second points.

Quadratic:

Given that the function $\boldsymbol{p_2(x)}$ passes through two points, we want to find a function $\boldsymbol{\Delta p_2(x)}$ such that $\boldsymbol{p_2(x) = p_1(x) + \Delta p_2(x)}$ passes through all three points.

By solving, we will get an equation like this:

$$\boldsymbol{p_2(x) = y_1 + y_{21}(x - x_1) + y_{321}(x - x_1)(x - x_2)}$$

where, $\boldsymbol{y_{21} = \frac{(y_2 - y_1)}{(x_2 - x_1)}}$; $\boldsymbol{y_{32} = \frac{(y_3 - y_2)}{(x_3 - x_2)}}$ and $\boldsymbol{y_{321} = \frac{(y_{32} - y_{21})}{(x_3 - x_1)}}$.

In general, if we define the interpolating Newton polynomial of degree $\boldsymbol{n - 1}$ as:

$$\boldsymbol{p_{n-1}(x) = y_1 + y_{21}(x - x_1) + y_{321}(x - x_1)(x - x_2) + \ldots\ldots + y_{n\ldots321}(x - x_1)(x - x_2)\ldots.(x - x_{n-1})}$$

---

5. CONTRIBUTION BY TEAM MEMBERS

I.	**Krishnapriya:**	Introduction to Polynomial Interpolation
II.	**Abhiram:**	Computation method- Vandermonde Matrix.
III.	**Amanpreet:**	Application- Karatsuba Algorithm.
IV.	**Ayushi:**	Application- Tom-Cook Multiplication.
V.	**Navya:**	Application- Shamir's Secret Sharing Scheme.
VI.	**Pradeep:**	Application- Secure multi-party computation (SMPC).

---

6. REFERENCES:

- https://en.wikipedia.org/wiki/Lagrange_polynomial
- https://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/05Interpolation/
- https://www.youtube.com/watch?v=B2wqZKZv3O0
- https://www.youtube.com/watch?v=XK4G5Ndy-m8
- https://www2.math.uconn.edu/~leykekhman/courses/MATH3795/Lectures/Lecture_14_poly_interp.pdf