Lab Manual: Numerical Analysis



**Lab Manual for Numerical Analysis**

Lab No. 7

INTERPOLATION -

LAGRANGE’S INTERPOLATION

1

Software Engineering Department Bahria University (Karachi Campus)

Lab Manual: Numerical Analysis



**BAHRIA UNIVERSITY KARACHI CAMPUS**

**Department of Software Engineering**

# NUMERICAL ANALYSIS LAB EXPERIMENT # 7

Interpolation - Lagrange’s Interpolation

## OBJECTIVE:

This lab aims to introduce students to **interpolation principles**, with a focus on Lagrange’s interpolation, in order to equip them with practical skills for estimating intermediate values within scientific and engineering contexts.

# Introduction

**Interpolation**, as discussed in previous lab a fundamental concept in mathematics and data analysis, holds a pivotal role in estimating values between established data points. It is essential for making informed decisions and predictions, and we've previously explored its core principles in our academic pursuits.

In our prior chapter, we have extensively delved into two key interpolation formulas that are integral to our understanding: Newton's Backward Formula and Newton's Forward Formula. These formulas prove to be invaluable tools for approximating intermediate values within datasets. Newton's Backward Formula is particularly advantageous when we encounter situations marked by decreasing intervals, while Newton's Forward Formula shines when we work with increasing intervals. Collectively, these formulas provide us with a versatile and robust toolkit for performing interpolation with precision and efficiency. Through these formulas, we gain the ability to make data-driven decisions and predictions, enabling us to navigate the intricate terrain of data analysis and mathematical modeling with confidence and accuracy.

In this lab, we will further enhance our understanding of interpolation by exploring two additional techniques: Newton’s Divided Difference Formula and Lagrange’s Interpolation

# Implementation of Lagrange’s Interpolation

This section will provide a detailed discussion and implementation of **Lagrange Interpolation**, offering a comprehensive understanding of these interpolation techniques and their practical applications.

2

Software Engineering Department Bahria University (Karachi Campus)

Lab Manual: Numerical Analysis



## Lagrange Interpolation

**Lagrange Interpolation** stands as a valuable and versatile technique that plays a pivotal role in mathematical modeling, engineering, and data analysis. Its significance lies in its ability to estimate the value of a function at a specific point, even when the function's form remains unknown. This interpolation method leverages known data points, typically represented as pairs like **(x1, y1)** and **(x2, y2),** as its foundation.

By applying the Lagrange Interpolation Formula, it can effectively calculate the value of y at any arbitrary point, denoted as x = a. The core of this technique lies in the creation of a Lagrange Polynomial, an **nth-degree** expression of the underlying function f(x) where n i.e. degree of polynomial equals the total number of data points. This polynomial's flexibility and precision empower it to bridge the gaps between data points, enabling informed estimations

Lagrange Interpolation proves its worth, particularly in scenarios where the function's explicit form remains elusive, offering a reliable means to navigate the intricate landscape of data analysis and mathematical modeling, making it an indispensable tool in various domains.

**Lagrange First Order Interpolation** Formula (for n = 1) is:

𝒇(𝒙) = (𝒙

(𝒙 − 𝒙𝟏)

𝟎 𝟏

− 𝒙 )

× (𝒚 ) +

(𝒙 − 𝒙𝟎)

𝟎

(𝒙 − 𝒙 )

× 𝒚

𝟏

𝟏 𝟎

and, **Lagrange First Order Interpolation** Formula (for n = 2) is:

𝒇(𝒙) = (𝒙

(𝒙 − 𝒙𝟏)(𝒙 − 𝒙𝟐)

× (𝒚 ) +

(𝒙 − 𝒙𝟎)(𝒙 − 𝒙𝟐)

(𝒙 − 𝒙𝟎)(𝒙 − 𝒙𝟏)

𝟎

× 𝒚 +

× 𝒚

𝟎 𝟏 𝟎 𝟐

− 𝒙 )(𝒙 − 𝒙 )

(𝒙 − 𝒙 )(𝒙 − 𝒙 )

𝟏

𝟐

𝟏 𝟎 𝟏 𝟐

(𝒙 − 𝒙 )(𝒙 − 𝒙 )

𝟐 𝟎 𝟐 𝟏

Just like this, formula for Lagrange Interpolation at **nth order** can be defined as:

𝒇(𝒙) = (𝒙

(𝒙 − 𝒙𝟏)(𝒙 − 𝒙𝟐) … (𝒙 − 𝒙𝒏)

𝟎 𝟏 𝟎 𝟐

− 𝒙 )(𝒙 − 𝒙 ) … (𝒙 − 𝒙 )

× (𝒚 ) +

(𝒙 − 𝒙𝟎)(𝒙 − 𝒙𝟐) … (𝒙 − 𝒙𝒏)

𝟎

× 𝒚 + ⋯

𝟏

𝟎 𝒏

(𝒙 − 𝒙 )(𝒙 − 𝒙 ) … (𝒙 − 𝒙 )

𝟏 𝟎 𝟏 𝟐

𝟏 𝒏

+ (𝒙 − 𝒙 )(𝒙 − 𝒙 ) … (𝒙 − 𝒙 − 𝟏) × 𝒚𝒏

(𝒙 − 𝒙𝟎)(𝒙 − 𝒙𝟏) … (𝒙 − 𝒙𝒏 − 𝟏)

𝒏 𝟎 𝒏 𝟏 𝒏 𝒏

## Implementation in Python

# Define a class to represent data points with x and y values class DataPoint:

def init (self, x, y): self.x = x

self.y = y

# Function for Lagrange Interpolation

def lagrange\_interpolation(data\_points, x\_value): result = 0.0

# Iterate through each data point for i in range(len(data\_points)):

term = data\_points[i].y

# Calculate the Lagrange term for the current data point for j in range(len(data\_points)):

if j != i:

term \*= (x\_value - data\_points[j].x) / (data\_points[i].x - data\_points[j].x)

# Add the term to the result result += term

return result

if name == " main ": data\_points = []

# Input the number of known data points

n = int(input("Enter the number of known data points: "))

# Input x and y values for each data point for i in range(n):

x = float(input(f"Enter x{i + 1}: ")) y = float(input(f"Enter y{i + 1}: ")) data\_points.append(DataPoint(x, y))

# Input the x value for interpolation

x\_value = float(input("Enter the x value for interpolation: "))

# Calculate and print the interpolated value

interpolated\_value = lagrange\_interpolation(data\_points, x\_value) print(f"Interpolated value at {x\_value} is:{interpolated\_value}")

4

Lab Manual: Numerical Analysis



# Lab Tasks

1. Write a python program implementing the Lagrange interpolation formula that considers the following data points, and find the value of y at x = 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 1 | 2 | 3 | 5 |
| y | 10 | 4 | 4 | 7 |

5

Software Engineering Department Bahria University (Karachi Campus)