DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

Faculties of Computer Science



Bachelors of Science in Computer Science and Information

Technology

(BSc. CSIT)

Course: Numerical Method (CSC-207)

Class of 2027/Semester: III

A Lab Report On:

**Interpolation and Regression**

**Submitted to:**

Ganesh Thapa

Faculty, Department of Computer Science

**Submitted by:**

Name:

Roll No:

# LAB 2

**TITLE: Interpolation and Regression**

**OBJECTIVES:**

To study different interpolation methods used to find the value for independent variable using provided reference data.

**THEORY:**

*Define Interpolation and Regression*

*Explain Newton's Forward Interpolation, Lagrange's Interpolation and Newton's Divided Interpolation methods briefly (Explain the concept, formulas and also include difference tables whenever required)*

**PRACTICAL IMPLEMENTATION**

**TASK 1: NEWTON'S FORWARD INTERPOLATION METHOD**

**ALGORITHM:**

Input: n, x[20], y[20], xp

Output: yp

1. Declare variables n, i, j as integer.
2. Declare variables x[20] and y[20] to store the x and y values.
3. Declare a 2D array diff[20][20] to store the forward differences.
4. Declare variables xp, p, h, nr, dr, and yp as floating point numbers for calculations.
5. Read n, x, y
6. Read xp
7. Calculate Differences:
   1. Calculate h = x[1] – x[0] as the difference between the second and first x values.
   2. For i from 0 to n-1:
      1. Calculate diff[i][1] as y[i+1] - y[i].
   3. For j from 2 to n:

For i from 0 to n-j:

1. Calculate diff[i][j] as diff[i+1][j-1] - diff[i][j-1].
2. Calculate Interpolation:
3. Calculate p as (xp - x[0]) / h.
4. Initialize yp with y[0].
5. For k from 1 to n-1:
6. Calculate nr \*= (p - k + 1).
   1. Calculate dr \*= k.
   2. Update yp += (nr / dr) \* diff[i][k].
7. Output Result:

Print the interpolated value yp for the given xp..

**SOURCE CODE:**

*Source Code must be handwritten*

**OUTPUT:**

*Output must be screenshot of your output and must display your name as well.*

**TASK 2: LAGRANGE'S INTERPOLATION METHOD**

**ALGORITHM:**

Input: n, x[100], y[100], xp

Output: yp

1. Declare arrays x[100] and y[100] to store the x and y values.
2. Declare variables xp and yp for calculations.
3. Declare variables n, i, j, and p.
4. Read n, x, y
5. Read xp

6. Interpolate using Lagrange's method:

a. For i from 1 to n:

i. Initialize p to 1.

ii. For j from 1 to n:

- If i is not equal to j, calculate p \*= (xp - x[j]) / (x[i] - x[j]).

iii. Update yp += p \* y[i].

7. Output Result:

- Print the interpolated value yp for the given xp.

**SOURCE CODE:**

**OUTPUT:**

**TASK 3: NEWTON'S DIVIDED DIFFERENCE INTERPOLATION**

**ALGORITHM:**

Algorithm Newton Interpolation:

Input: n, x[10], y[10], k

Output: f

1. Initialize variables:

- Declare arrays x[10], y[10], and p[10] to store x values, y values, and divided differences.

- Declare variables k, f, n, f1, f2, i, j = 1;

2. Read n, x, y

3. Read k

4. Calculate Divided Differences:

a. Use a do-while loop until n becomes 1:

i. For i from 1 to n-1:

- Calculate p[i] = (y[i+1] - y[i]) / (x[i+j] - x[i]).

- Update y[i] = p[i].

ii. Initialize f1 = 1.

iii. For i from 1 to j:

- Calculate f1 \*= (k - x[i]).

iv. Update f2 += y[1] \* f1.

v. Decrement n and increment j.

5. Calculate Interpolation:

- Update f += f2.

6. Output Result:

- Print the interpolated value f for the given k.

**SOURCE CODE:**

**OUTPUT:**

**TASK 4: LINEAR REGRESSION**

Solving the non- linear equation x3 - 2x – 5 correct upto 3 decimal places using secant method.

**ALGORITHM:**

1. Start
2. Initialize variables: n,i as integer
3. Initialize variable x[20],y[20], sumX = 0, sumX2 = 0, sumY = 0, sumXY = 0, a, b as floating point number.

4. Read Number of Data (n)

5. For i=1 to n:

Read Xi and Yi

Next i

6. Calculate Required Sum

For i=1 to n:

sumX = sumX + Xi

sumX2 = sumX2 + Xi \* Xi

sumY = sumY + Yi

sumXY = sumXY + Xi \* Yi

Next i

7. Calculate Required Constant a and b of y = a + bx:

b = (n \* sumXY - sumX \* sumY)/(n\*sumX2 - sumX \* sumX)

a = (sumY - b\*sumX)/n

8. Display value of a and b

9. Print the equation of the best fit line: y = a + bx.

**SOURCE CODE:**

**OUTPUT:**

**CONCLUSION:** *Discuss about the lesson learnt after completion of this lab.*