Lab 03: Lagrange Interpolation for Curve Fitting

Objective

To understand and implement the **Lagrang Interpolation** for curve fitting on data provided by the user.

Theory

Lagrange interpolation is a polynomial interpolation method used to estimate values between known data points. Given a set of $n+1$ data points $(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$, the Lagrange interpolating polynomial $P(x)$ is constructed as:

$$P(x) = \sum_{i=0}^{n} y_i \cdot L_i(x)$$

where $L_i(x)$ are the Lagrange basis polynomials defined by:

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Each $L_i(x)$ is equal to 1 at $x = x_i$ and 0 at all other x_j ($j \neq i$), ensuring that $P(x_i) = y_i$. This method provides an exact fit for the given points and is widely used in numerical analysis for curve fitting and interpolation.

Algorithm

1. Input the number of data points n.
2. Input the data points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$.
3. Input the value x_p at which interpolation is required.
4. For each i from 0 to n:
   - Compute the Lagrange basis polynomial $L_i(x_p) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x_p - x_j}{x_i - x_j}$.
5. Compute the interpolated value $P(x_p) = \sum_{i=0}^{n} y_i \cdot L_i(x_p)$.
6. Output the interpolated value $P(x_p)$.

Implementation

```c
# include <stdio.h>

# define N 10

// Function to perform Lagrange interpolation
float lagrange_interpolation(float x[], float y[], int n, float xp) {
    float L[N];
    float yp = 0.0f;
    for (int i = 0; i < n; i++) {
        L[i] = 1.0f;
        for (int j = 0; j < n; j++) {
            if (j != i) {
                L[i] *= (xp - x[j]) / (x[i] - x[j]);
            }
        }
        yp += y[i] * L[i];
    }

    // Print the Lagrange basis polynomials
    printf("Lagrange basis polynomials:\n");
    for (int i = 0; i < n; i++)
    {
        printf("L[%d] = %.2f\n", i, L[i]);
    }

    return yp;
}

// Main function
int main(int argc, char const *argv[])
{
    float x[N], y[N], xp, yp;
    int i, j, n;

    printf("Enter the number of data points: ");
    scanf("%d", &n);

    printf("Enter the data points (x y):\n");
    for (i = 0; i < n; i++)
    {
        scanf("%f %f", &x[i], &y[i]);
    }

    printf("Enter the value of x for interpolation: ");
    scanf("%f", &xp);

    yp = lagrange_interpolation(x, y, n, xp);
    printf("\nThe interpolated value at x = %.2f is y = %.2f\n", xp, yp);

    return 0;
}
```

Output:

```
Enter the number of data points: 4
Enter the data points (x y):
1 3
2 5
3 7
4 9
Enter the value of x for interpolation: 2.5
Lagrange basis polynomials:
L[0] = -0.06
L[1] = 0.56
L[2] = 0.56
L[3] = -0.06

The interpolated value at x = 2.50 is y = 6.00
```

Discussion

The Lagrange interpolation method provides a systematic approach to estimate unknown values within the range of a discrete set of known data points. In this lab, implementing the algorithm helped reinforce the concepts of polynomial interpolation and the importance of basis polynomials. While the method is straightforward for a small number of points, it can become computationally intensive and less stable as the dataset grows. Careful handling of input data and numerical calculations is essential to ensure accurate results.

Conclusion

Lagrange interpolation is a powerful and straightforward technique for constructing a polynomial that passes exactly through a given set of data points. It is particularly effective for small datasets, providing accurate interpolation between known values. However, its computational complexity increases with more data points, and it may suffer from numerical instability for large datasets. Overall, Lagrange interpolation is well-suited for problems where the number of points is limited and an exact fit is required.

Name: Kushal Prasad Joshi
Roll No: 230345
Faculty: Science and Technology
Program: BE Computer
Group: B