Interpolation
Lab Manual 02

Prepared By:
Md. Azmain Yakin Srizon
Assistant Professor
Department of Computer Sciecne & Engineering
Rajshahi University of Engineering & Technology

Reference Book:
Introductory Methods of Numerical Analysis by S. S. Sastry (5th Edition)
Last Updated: October 6, 2024

1 Newton's Forward Interpolation

1.1 Pseudo Code

```
1. Initialize arrays x[] = {1, 3, 5, 7} and y[] = {24, 120, 336, 720}
2. Define a 2D array diffTable[n][n] where n is the number of data points
3. Fill the first column of diffTable with y[] values
4. for j = 1 to n-1
       for i = 0 to n-j-1
           diffTable[i][j] = diffTable[i+1][j-1] - diffTable[i][j-1]
5. Set h = x[1] - x[0]
6. Set p = (desired_x - x[0]) / h
7. Set result = y[0]
8. for i = 1 to n-1
       product = p
       for k = 1 to i-1
           product = product * (p - k)
       result = result + (product * diffTable[0][i]) / factorial(i)
9. Print the forward difference table
10. Print the result (interpolated value of y(desired_x))
```

1.2 Sample Input

- Given data points:

x	$y(x)$
1	24
3	120
5	336
7	720

- Desired value of x: 8

1.3 Sample Output

Forward Difference Table

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
1	24	96	120	48
3	120	216	168	
5	336	384		
7	720			

The value of $y(8)$ is: 990

2 Newton's Backward Interpolation

2.1 Pseudo Code

```
1. Initialize arrays x[] = {1, 3, 5, 7} and y[] = {24, 120, 336, 720}
2. Define a 2D array diffTable[n][n] where n is the number of data points
3. Fill the first column of diffTable with y[] values
4. for j = 1 to n-1
       for i = n-1 down to j
           diffTable[i][j] = diffTable[i][j-1] - diffTable[i-1][j-1]
5. Set h = x[1] - x[0]
6. Set p = (desired_x - x[n-1]) / h
7. Set result = y[n-1]
8. for i = 1 to n-1
       product = p
       for k = 1 to i-1
           product = product * (p + k)
       result = result + (product * diffTable[n-1][i]) / factorial(i)
9. Print the backward difference table
10. Print the result (interpolated value of y(desired_x))
```

2.2 Sample Input

- Given data points:

x	$y(x)$
1	24
3	120
5	336
7	720

- Desired value of x: 8

2.3 Sample Output

Backward Difference Table

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
1	24			
3	120	96		
5	336	216	120	
7	720	384	168	48

The value of $y(8)$ is: 990

3 Gauss' Forward Central Interpolation

3.1 Pseudo Code

```
1. Initialize arrays x[] = {1, 3, 5, 7} and y[] = {24, 120, 336, 720}
2. Define a 2D array diffTable[n][n] where n is the number of data points
3. Fill the first column of diffTable with y[] values
4. for j = 1 to n-1
       for i = 0 to n-j-1
           diffTable[i][j] = diffTable[i+1][j-1] - diffTable[i][j-1]
5. Set mid = n / 2 (choose the central point)
6. Set h = x[1] - x[0]
7. Set p = (desired_x - x[mid]) / h
8. Set result = y[mid]
9. for i = 1 to n-1
       product = p
       for k = 1 to i-1
           if k is even, product = product * (p + k/2)
           if k is odd, product = product * (p - (k+1)/2)
       result = result + (product * diffTable[mid - i/2][i]) / factorial(i)
10. Print the forward difference table
11. Print the result (interpolated value of y(desired_x))
```

3.2 Sample Input

- Given data points:

x	$y(x)$
1	24
3	120
5	336
7	720

- Desired value of x: 8

3.3 Sample Output

Forward Difference Table

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
1	24	96	120	48
3	120	216	168	
5	336	384		
7	720			

The value of $y(8)$ is: 975

4 Gauss' Backward Central Interpolation

4.1 Pseudo Code

```
1. Initialize arrays x[] = {1, 3, 5, 7} and y[] = {24, 120, 336, 720}
2. Define a 2D array diffTable[n][n] where n is the number of data points
3. Fill the first column of diffTable with y[] values
4. for j = 1 to n-1
       for i = n-1 down to j
           diffTable[i][j] = diffTable[i][j-1] - diffTable[i-1][j-1]
5. Set mid = n / 2 (choose the central point)
6. Set h = x[1] - x[0]
7. Set p = (desired_x - x[mid]) / h
8. Set result = y[mid]
9. for i = 1 to n-1
       product = p
       for k = 1 to i-1
           if k is odd, product = product * (p + (k+1)/2)
           if k is even, product = product * (p - k/2)
       result = result + (product * diffTable[mid + (i-1)/2][i]) / factorial(i)
10. Print the backward difference table
11. Print the result (interpolated value of y(desired_x))
```

4.2 Sample Input

- Given data points:

x	$y(x)$
1	24
3	120
5	336
7	720

- Desired value of x: 8

4.3 Sample Output

Backward Difference Table

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
1	24			
3	120	96		
5	336	216	120	
7	720	384	168	48

The value of $y(8)$ is: 900

5 Lagrange Interpolation

5.1 Pseudo Code

```
1. Initialize arrays x[] = {1, 3, 5, 7} and y[] = {24, 120, 336, 720}
2. Set the number of data points n = size of x[]
3. Set value = 8 (the x-value to interpolate)
4. Initialize result = 0.0
5. for i = 0 to n-1:
       Initialize term = y[i]
       for j = 0 to n-1:
           if j != i:
               term = term * (value - x[j]) / (x[i] - x[j])
       Add term to result
6. Print the result (interpolated value of y(value))
```

5.2 Sample Input

- Given data points:

x	$y(x)$
1	24
3	120
5	336
7	720

- Desired value of x: 8

5.3 Sample Output

The value of $y(8)$ using Lagrange Interpolation is: 990.00

6 Comparison of Results Using Different Interpolation Methods

6.1 Question

The following results were obtained using different interpolation methods to find the value of $y(8)$ based on the data points:

x	$y(x)$
1	24
3	120
5	336
7	720

The interpolation methods and their results were:

- Newton's Forward Interpolation: 990
- Newton's Backward Interpolation: 990
- Gauss' Forward Central Interpolation: 975
- Gauss' Backward Central Interpolation: 900
- Lagrange Interpolation: 990

Why are the results from Gauss' forward central interpolation and backward central interpolation different from those of Newton's and Lagrange's methods?

6.2 Explanation

The differences in the results arise due to the fundamental way each interpolation method computes the interpolated value and the specific focus of each method. Let's examine the reasons:

6.2.1 Newton's Forward and Backward Interpolation (990)

Both Newton's forward and backward interpolation produced the same result of 990 because the interpolation point $x = 8$ lies outside the given data points (the data ends at $x = 7$).

- **Newton's forward interpolation** works best for points near the beginning of the dataset, while **Newton's backward interpolation** is best for points near the end. Since $x = 8$ is near the end of the dataset, both methods provide very similar results.
- The fact that both methods give 990 suggests that the underlying polynomial fits the data well near $x = 8$, and thus the result is accurate.

6.2.2 Gauss' Forward Central Interpolation (975)

Gauss' forward central interpolation focuses on points around the middle of the dataset and uses forward differences starting from the central point.

- Since Gauss' method is centered, and $x = 8$ lies outside the dataset, it struggles to capture the upward trend in y. The interpolation point is distant from the central points $x = 3$ and $x = 5$, and thus the method **underestimates** the value, giving 975.
- This happens because Gauss' forward central interpolation does not fully capture the higher-order differences beyond the range of the dataset, leading to an underestimation.

6.2.3 Gauss' Backward Central Interpolation (900)

Gauss' backward central interpolation also focuses on points near the middle of the dataset but uses backward differences.

- Since $x = 8$ is beyond the dataset, the backward central interpolation again underestimates the value, providing 900, which is even lower than the forward central result.
- This method uses backward differences, which diminish faster when extrapolating beyond the dataset, thus leading to a significant underestimation.

6.2.4 Lagrange Interpolation (990)

Lagrange interpolation uses all data points equally, without focusing on the position of the interpolated value relative to the dataset.

- Lagrange interpolation constructs a polynomial that passes through all the data points, making it accurate for both interpolation and extrapolation.
- Since Lagrange interpolation does not depend on where the interpolation point is within the dataset, it provides a consistent result of 990, the same as Newton's methods.

6.2.5 Why the Differences?

- **Extrapolation vs. Interpolation**: Since $x = 8$ is outside the given data points, we are extrapolating, which can introduce differences in accuracy. Newton's forward/backward and Lagrange handle extrapolation better, whereas Gauss' central interpolation is more suited to interpolation within the dataset.
- **Focus of Each Method**: Gauss' central methods focus on the middle of the dataset, so they are less accurate for points beyond the dataset's range, while Newton's methods work best at the dataset's boundaries.
- **Handling of Higher-Order Differences**: Newton's and Lagrange's methods accurately capture higher-order differences, while Gauss' central interpolation does not fully account for these when extrapolating beyond the dataset, leading to underestimation.

6.3 Conclusion

Newton's forward, backward, and Lagrange interpolation methods provided the same result of 990 because they handle extrapolation near the edge of the dataset well. On the other hand, Gauss' forward and backward central interpolation methods, which focus on points near the center of the dataset, struggled with extrapolation and produced underestimated results of 975 and 900, respectively.

7 Exercises

Example 3.6, 3.8, 3.9, 3.13, 3.15 from the book.

THE END