MA-221(Numerical Analysis)
Course Instructor: Prof. Rajendra K. Ray
TA: Kajal Mittal, Niladri Bose
Lab Assignment-10
Date: 15/04/2025

Instructions

- Solve each problem using Python, C++, and MATLAB.

Question 1: Weather Sensor Calibration

Background

Temperature readings from weather sensors are often taken at fixed time intervals. However, real-time applications may require temperature estimates at times between those intervals. Interpolation particularly Newton Forward and Backward methods for equally spaced data—helps fill in these missing values with reasonable accuracy.

Data Table

Time (hrs)	Temperature (°C)
6	15.0
7	15.9
8	17.1
9	18.2
10	20.5
11	21.8
12	23.0
13	24.1
14	25.2
15	25.9

Tasks

1. Estimate temperature at 8:30 AM using **Newton Forward Interpolation** and **Newton Backward Interpolation**.
2. Estimate temperature at 13:30 PM using **Newton Forward Interpolation** and **Newton Backward Interpolation**.
3. Which gives the best result and why?
4. Calculate absolute error if the actual recorded temperatures were 17.7°C and 24.5°C respectively.
5. Justify why each method was used for its respective point.

Question 2: Fuel Efficiency Curve

Background:

Automotive engineers assess fuel efficiency at specific speeds, but consumer performance often falls between these speeds. Interpolation techniques like Newton's Divided Difference and Lagrange help estimate fuel efficiency for speeds not directly tested, enabling smoother performance predictions and comparison of accuracy under data perturbation.

Data Table

Speed (km/h)	**Efficiency (km/L)**
20	10.5
30	12.5
35	13.7
40	14.8
50	16.0
60	17.2
70	16.8
80	15.5
90	14.2
100	13.0

Tasks

1. Interpolate fuel efficiency at 45 km/h using **Newton Divided Difference** and **Lagrange Interpolation**.
2. Compare interpolated values with actual value 15.4 km/L.
3. Determine which interpolation method is **more accurate** for this dataset.
4. Predict the fuel efficiency at **110 km/h** using both methods. Comment on extrapolation stability and realism of the results.
5. Change the fuel efficiency at **60 km/h from 17.2 to 25.0 km/L** and re-calculate for 45 km/h to see which method is more sensitive to this change and which method is more resilient.

Question 3. Satellite Altitude Estimation

Background:

When satellite telemetry is interrupted or incomplete, estimating the missing altitude data becomes critical for path reconstruction and monitoring. Interpolation provides a way to fill in this gap. Comparing Lagrange and Divided Difference methods sheds light on the stability and reliability of recovery in such sensitive applications.

Data Table

Time (min)	**Altitude (km)**
0	400.0
2	401.3
4	402.7
6	403.6
8	405.0
10	407.8
12	409.0
14	410.2
16	412.0
18	413.5

Tasks

1. Estimate altitude at 9 min using **Lagrange** and **Divided Difference**.
2. Derive and compare theoretical error terms.
3. Remove 10 min data point and re-evaluate sensitivity.
4. Discuss impact of missing data and computational stability.

Question 4. Stock Market Interpolation Challenge

Background:

In financial markets, price feeds can miss ticks due to latency or data loss. Traders often need interpolated values for better trend analysis or algorithmic decisions. This problem explores how different interpolation schemes perform when estimating mid-point prices and how well they adapt to both equal and unequal intervals.

Data Table

Time (hrs)	Price (INR)
9	153.8
10	155.2
11	157.8
12	160.0
13	161.3
14	163.5
15	164.7
16	166.1
17	168.0
18	169.5

Tasks

1. Estimate price at 12:30 PM using **Newton Forward, Backward, Divided Difference, and Lagrange** methods.
2. **Plot** all four interpolation polynomials over the full time range [9 Hours, 18 Hours]
3. Perform a detailed error analysis, showing which method gives the closest value to the actual known price (assuming 160.7 INR) for the respective time.
4. Create a comparison table covering error, time and memory complexity.

Pseudocode for Plotting

Python

```python
import matplotlib.pyplot as plt
import numpy as np

x = np.array([...])  # Replace with actual values
y = np.array([...])
plt.plot(x, y, label='Data')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Data_Visualization")
plt.legend()
plt.show()
```

C++

```cpp
#include <iostream>
#include "matplotlibcpp.h"

namespace plt = matplotlibcpp;

int main() {
    std::vector<double> x = {...};  // Replace with actual values
    std::vector<double> y = {...};
    plt::plot(x, y);
    plt::xlabel("X-axis");
    plt::ylabel("Y-axis");
    plt::title("Data_Visualization");
    plt::show();
    return 0;
}
```

MATLAB

```matlab
x = [...]; % Replace with actual values
y = [...];
plot(x, y);
xlabel('X-axis');
ylabel('Y-axis');
title('Data_Visualization');
grid on;
```

Appendix: Pseudocode and Complexity Analysis

- The loop structure or recursive calls to estimate **Time Complexity** in terms of n (number of data points).
- The size of data structures used (e.g., arrays, tables) to estimate **Memory Complexity**.

A typical pseudocode might look like:

```
For i = 0 to n-1:
    Compute term involving n-i operations
    Accumulate result

=> Time Complexity: O(n^2)
=> Memory Complexity: O(n^2) if 2D table is used
```

Submission

- Submit all source codes with inline comments.
- Include a brief report for each problem (PDF).
- Attach plots for visual comparison.
- Code submission must be modular and readable.