Numerical Analysis

Assignment #3

Submitted to: Mam Sadaf Khalid

Date: May 31, 2021

Submitted by
Bazila Afridi (008)
Saad Iqbal (030)
Sarmad Ashfaq Chaudhary (033)
Muhammad Hayyan Khan (042)
Manal Talat (046)

Section: BSE-6A

DEPARTMENT OF SOFTWARE ENGINEERING

BAHRIA UNIVERSITY ISLAMABAD

Implement the concept of Numerical Differentiation using programming techniques of C++. Your programming solution should cater the following interpolation formulae:

1. Stirling's formula.
2. Bessel's formula.
3. Everett's formula.
4. Gauss Forward difference formula.
5. Gauss Backward difference formula.

Formula (assigned): Bessel's Interpolation

Formula Derivation:

day / date:

Bessel's Formula Derivations

⇒ Guass's forward interpolation formula is:

$$y_P = y_0 + p\Delta y_0 + \frac{P(P-1)}{2!}\Delta^2 y_{-1} + \frac{(P+1)P(P-1)}{3!}\Delta^3 y_{-1} + \frac{(P+1)P(P-1)(P+1)}{4!}\Delta^4 y_{-2} + \cdots$$

⇒ Now we will write odd terms starting from third term as a sum of its halves.

$$y_P = y_0 + p\Delta y_0 + \frac{1}{2}\frac{P(P-1)}{2!}\Delta^2 y_{-1} + \frac{1}{2}\frac{P(P-1)}{2!}\Delta^2 y_{-1} + \frac{(P+1)P(P-1)}{3!}\Delta^3 y_{-1} +$$
$$\frac{1}{2}\frac{(P+1)P(P-1)}{4!}\Delta^4 y_{-2} + \frac{1}{2}\frac{(P+1)P(P-1)}{4!}\Delta^4 y_{-2} + \cdots$$

⇒ Now from forward interpolation difference table we have:

$$\Delta^2 y_{-1} = \Delta^2 y_0 - \Delta^3 y_{-1}. \quad \text{—— (1)}$$
$$\text{and } \Delta^4 y_{-2} = \Delta^4 y_{-1} - \Delta^5 y_{-2} \quad \text{etc. —— (2)}$$

⇒ Now substituting the equations (1) and (2) in the second half terms of the above equation we have:

$$y_P = y_0 + p\Delta y_0 + \frac{1}{2}\frac{P(P-1)}{2!}\Delta^2 y_{-1} + \frac{1}{2}\frac{P(P-1)}{2!}(\Delta^2 y_0 - \Delta^3 y_{-1}) + \frac{(P+1)P(P-1)}{3!}\Delta^3 y_{-1} +$$
$$\frac{1}{2}\frac{(P+1)P(P-1)}{4!}\Delta^4 y_{-2} + \frac{1}{2}\frac{(P+1)P(P-1)}{4!}(\Delta^4 y_{-1} - \Delta^5 y_{-2}) + \cdots$$

$$\Rightarrow y_P = y_0 + P\Delta y_0 + \frac{1}{2}\frac{P(P-1)}{2!}\Delta^2 y_{-1} + \frac{P(P+1)}{2!}\left(\frac{\Delta^2 y_0}{2} - \frac{\Delta^3 y_{-1}}{2}\right) + \frac{(P+1)P(P-1)}{3!}\Delta^3 y_{-1}$$
$$+ \frac{1}{2}\frac{(P+1)P(P-1)(P-2)}{4!}\Delta^4 y_{-2} + \frac{(P+1)P(P-1)(P-2)}{4!}\left(\frac{\Delta^4 y_{-1}}{2} - \frac{\Delta^5 y_{-2}}{2}\right) + \cdots$$

$$\Rightarrow y_P = y_0 + P\Delta y_0 + \frac{P(P-1)}{2!}\cdot\frac{\Delta^2 y_{-1}}{2} + \frac{P(P-1)}{2!}\cdot\frac{\Delta^2 y_0}{2} - \frac{P(P-1)}{2!}\cdot\frac{\Delta^3 y_{-1}}{2} + \frac{(P+1)P(P-1)}{3\times 2!}\Delta^3 y_{-1}$$
$$+ \frac{(P+1)P(P-1)(P-2)}{4!}\cdot\frac{\Delta^4 y_{-2}}{2} + \frac{(P+1)P(P-1)(P-2)}{4!}\cdot\frac{\Delta^4 y_{-1}}{2} - \cdots$$

day / date:

$$\Rightarrow y_P = y_0 + P\Delta y_0 + \frac{P(P-1)}{2!}\left(\frac{\Delta^2 y_{-1} + \Delta^2 y_0}{2}\right) + \frac{P(P-1)}{2!}\left(-\frac{\Delta^3 y_{-1}}{2} + \frac{(P+1)}{3}\Delta^3 y_{-1}\right) + \frac{(P+1)P(P-1)(P-2)}{4!}\left(\frac{\Delta^4 y_{-2} + \Delta^4 y_{-1}}{2}\right) + \dots$$

$$\Rightarrow y_P = y_0 + P\Delta y_0 + \frac{P(P-1)}{2!}\left(\frac{\Delta^2 y_{-1} + \Delta^2 y_0}{2}\right) + \frac{(P-\frac{1}{2})P(P-1)}{3!}\Delta^3 y_{-1} + \frac{(P+1)P(P-1)(P-2)}{4!}\left(\frac{\Delta^4 y_{-2} + \Delta^4 y_{-1}}{2}\right) + \dots$$

Above equation is known as Bessel's formula

$\Rightarrow$ In the central difference notation, above equation becomes:

$$\boxed{y_P = y_0 + P\delta y_{1/2} + \frac{P(P-1)}{2!}\mu\delta^2 y_{1/2} + \frac{(P-\frac{1}{2})P(P-1)}{3!}\delta^3 y_{1/2} + \frac{(P+1)P(P-1)(P-2)}{4!}\mu\delta^4 y_{1/2} + \dots}$$

Bessel's Interpolation Formula:

Bessel's formula follows the path through the difference table:

$$\begin{tikzcd}
x_0 & f_0 \arrow[dr] & & \delta^2 f_0 \arrow[dr] & & \delta^4 f_0 \\
& & \delta f_{1/2} \arrow[ur] \arrow[dr] & & \delta^3 f_{1/2} \arrow[ur] \arrow[dr] & \\
x_1 & f_1 & & \delta^2 f_1 \arrow[ur] & & \delta^4 f_1
\end{tikzcd}$$

Bessel's formula is expressed as follows:

$$f_p = f_0 + p\delta f_{1/2} + \frac{p(p-1)}{2 \cdot 2!}(\delta^2 f_0 + \delta^2 f_1) + \frac{p(p-1)(p-1/2)}{3!}\delta^3 f_{1/2} + \frac{(p+1)p(p-1)(p-2)}{2 \cdot 4!}(\delta^4 f_0 + \delta^4 f_1) + \dots$$

Code: [(View on Github)]

bessel.cpp

```
#include <iostream>
#include <time.h>
using namespace std;

class BesselsInterpolation
{
private:
    int n;
    double xp, xo, p;
    int origin_index;

    // dynamic arrays for difference table
    double *x, *fx;
    double *delta, *delta_sq, *delta_cube, *delta_quad;

public:
    // generate table itself for 5 terms
    BesselsInterpolation(double xp)
    {
        this->xp = xp;
        this->n = 5;
        x = new double[n];
        fx = new double[n];

        generateValues();
    }

    // initialize class with xp and no. of terms
    BesselsInterpolation(double xp, int num)
    {
        this->xp = xp;
        this->n = num;
        x = new double[n];
        fx = new double[n];

        getInputValues();
```

```
}

void generateValues()
{
    srand(1);
    for (int i = 0; i < n; i++)
    {
        x[i] = i + 1;
        fx[i] = rand() % 10;
    }
}

// fill difference table from user input
void getInputValues()
{
    cout << "\n";
    for (int i = 0; i < n; i++)
    {
        cout << "Enter value for x[" << i << "]: ";
        cin >> x[i];

        cout << "Enter value for fx[" << i << "]: ";
        cin >> fx[i];
    }
}

// construct difference table
void buildDifferenceTable()
{
    delta = new double[n];
    delta_sq = new double[n];
    delta_cube = new double[n];
    delta_quad = new double[n];

    // calculate values of delta
    for (int i = n - 1; i >= 0; i--)
    {
        if (i - 1 >= 0)
            delta[i] = fx[i] - fx[i - 1];
    }
```

```
    // calculate values of delta square
    for (int j = n - 1; j >= 0; j--)
    {
        if (j - 1 >= 0)
            delta_sq[j] = delta[j] - delta[j - 1];
    }

    // calculate values of delta cube
    for (int k = n - 1; k >= 0; k--)
    {
        if (k - 1 >= 0)
            delta_cube[k] = delta_sq[k] - delta_sq[k - 1];
    }

    // calculate values of delta quad
    for (int l = n - 1; l >= 0; l--)
    {
        if (l - 1 >= 0)
            delta_quad[l] = delta_cube[l] - delta_cube[l - 1];
    }

    // print difference table
    cout << "\nx: ";
    for (int i = 0; i < n; i++)
        cout << "\t\t" << x[i];

    cout << "\nfx: ";
    for (int i = 0; i < n; i++)
        cout << "\t\t" << fx[i];

    cout << "\ndelta: ";
    for (int i = 1; i < n; i++)
        cout << "\t\t" << delta[i];

    cout << "\ndelta^2: ";
    for (int i = 2; i < n; i++)
        cout << "\t\t" << delta_sq[i];

    cout << "\ndelta^3: ";
```

```cpp
    for (int i = 3; i < n; i++)
        cout << "\t\t" << delta_cube[i];

    cout << "\ndelta^4: ";
    for (int i = 4; i < n; i++)
        cout << "\t\t" << delta_quad[i];
}

// find step size
double findInterval()
{
    double h = x[1] - x[0];
    return h;
}

// calculate p for a specified origin
double calculateP(int xo)
{
    return ((xp - xo) / findInterval());
}

// select origin for interpolation
void findOrigin()
{
    for (int i = 0; i < n; i++)
    {
        p = calculateP(x[i]);
        xo = x[i];
        origin_index = i;

        if (p >= 0 && p <= 1)
            break;
        else
            continue;
    }

    cout << "\n\nOrigin (selected): " << xo;
    cout << "\nOrigin Index: " << origin_index;
    cout << "\nValue of P at Origin: " << p;
}
```

```
// calculate factorial for a number
int factorial(int n)
{
    if (n > 1)
        return n * factorial(n - 1);
    else
        return 1;
}

// calculate bessel interpolation's first term
double calcFirstTerm()
{
    return fx[origin_index]; // value of f at origin
}

// calculate bessel interpolation's second term
double calcSecondTerm()
{
    double sigma1by2 = delta[origin_index];
    return (p * sigma1by2);
}

// calculate bessel interpolation's third term
double calcThirdTerm()
{
    double sigmaSq0 = delta_sq[origin_index];
    double sigmaSq1 = delta_sq[origin_index + 1];

    return ((p * (p - 1) * (sigmaSq0 + sigmaSq1)) / (2 * factorial(2)));
}

// calculate bessel interpolation's fourth term
double calcFourthTerm()
{
    double sigmaCube1by2 = delta_cube[origin_index];

    return ((p * (p - 1) * (p - 1 / 2) * sigmaCube1by2) / (factorial(3)));
}
```

```
    // calculate bessel interpolation's fifth term
    double calcFifthTerm()
    {
        double sigmaQuad0 = delta_quad[origin_index];
        double sigmaQuad1 = delta_quad[origin_index + 1];

        return (((p + 1) * p * (p - 1) * (p - 2) * (sigmaQuad0 + sigmaQuad1)) / (2
* factorial(4)));
    }

    void findBesselResult()
    {
        double result = calcFirstTerm() + calcSecondTerm() + calcThirdTerm() +
calcFourthTerm() + calcFifthTerm();
        cout << "\n\n-> Bessel's Interpolation Result: " << result << endl
             << endl;
    }
};
```

main.cpp

```
#include <iostream>
#include "bessel.cpp"
using namespace std;

int main()
{
    cout << "\n\n --- Bessel's Interpolation ---";

    double xp;
    cout << "\nInterpolate for: ";
    cin >> xp;

    int choice;
    cout << "\nSelect an option: " << endl
         << "1. Auto-generate values" << endl
         << "2. Specify values" << endl;
    cin >> choice;

    if (choice == 1)
```

```
    {
        BesselsInterpolation BI(xp);
        BI.buildDifferenceTable();
        BI.findOrigin();
        BI.findBesselResult();
    }
    else
    {
        int num;
        cout << "\nNo. of terms in table: ";
        cin >> num;

        BesselsInterpolation BI(xp, num);
        BI.buildDifferenceTable();
        BI.findOrigin();
        BI.findBesselResult();
    }

    return 0;
}
```

Output:

(User specified values)

```
 --- Bessel's Interpolation ---
Interpolate for: 27.4

Select an option:
1. Auto-generate values
2. Specify values
2

No. of terms in table: 6

Enter value for x[0]: 25
Enter value for fx[0]: 4.000
Enter value for x[1]: 26
Enter value for fx[1]: 3.846
Enter value for x[2]: 27
Enter value for fx[2]: 3.704
Enter value for x[3]: 28
Enter value for fx[3]: 3.571
Enter value for x[4]: 29
Enter value for fx[4]: 3.448
Enter value for x[5]: 30
Enter value for fx[5]: 3.333

x:              25              26              27              28              29              30
fx:             4               3.846           3.704           3.571           3.448           3.333
delta:          -0.154          -0.142          -0.133          -0.123          -0.115
delta^2:                0.012           0.009           0.01            0.008
delta^3:                -0.003          0.001           -0.002
delta^4:                0.004           -0.003

Origin (selected): 27
Origin Index: 2
Value of P at Origin: 0.4

-> Bessel's Interpolation Result: 3.64498

(base) sarmad@Sarmads-Macbook NA Asgn 3 % 
```

(Auto-generated values)

```
 --- Bessel's Interpolation ---
Interpolate for: 27.4

Select an option:
1. Auto-generate values
2. Specify values
1

x:              1               2               3               4               5
fx:             7               9               3               8               0
delta:          2               -6              5               -8
delta^2:                -8              11              -13
delta^3:                19              -24
delta^4:                -43

Origin (selected): 5
Origin Index: 4
Value of P at Origin: 22.4

-> Bessel's Interpolation Result: -249679
```

REFERENCES

(Book) Chapter 3.4.2 - Bessel's Interpolation, Numerical Analysis with C++