**Звіт**

**до лабораторної роботи №1**

Оформив студент групи ТТП-31

Казмірчук Володимир

Викладач: Голубєва Катерина

Постановка задачі

Обчислити з точнiстю ε = 10-5 iнтеграл I(ω) = ∞∫0sin (ωx)\*e-x^3 dx

при ω = {1; 10; 20; 50}. Використати методобрiзання границь,

1) метод Сiмпсона, оцiнку залишкових членiв

2) таблицю Ромберга з кроком 4

Розв’язок

Для вирішення цієї задачі було написано наступний код на мові програмування C++:

#include <iostream>

#include <cmath>

#define MAX\_ITERATIONS 10000

using namespace std;

// Функція, яку інтегруємо

long double f (long double x, long double w) {

    return sin (w\*x) \* exp (- x\*x\*x);

}

// Метод обрізання границь

int limitrunc (long double w, long double e) {

    long double a = 1;

    while (true) {

        // Частковий інтеграл для обмеження функції

        if (exp (-a) < e || a > MAX\_ITERATIONS) { break; }

        a ++;

    }

    return a;

}

// Конвертація точності у кількість ітерацій

int eton (long double e) {

    return min ((int) (1 + 1 / sqrt (e)), MAX\_ITERATIONS);

}

// Оцінка залишкових членів

double estimate (double b, int n) {

    double M4 = 24;

    double h = b / n;

    double R = (M4 \* pow (h, 4)) / 180 \* b;

    return R;

}

// Метод середніх прямокутників

long double squares (long double b, long double w, int n) {

    long double h = b / n;

    long double sum = 0;

    for (int i = 0; i < n; i ++) {

        double xi = i \* h + h / 2;

        sum += f (xi, w);

    }

    return h \* sum;

}

// Метод Сімпсона

long double simpson (long double b, long double w, int n) {

    long double h = b / n;

    long double sum = 0;

    for (int i = 0; i <= n; i ++) {

        long double xi = i \* h;

        long double fxi = f (xi, w);

        if (i == 0 || i == n) {

            sum += fxi;

        }

        else if (i % 2 == 0) {

            sum += 2 \* fxi;

        }

        else {

            sum += 4 \* fxi;

        }

    }

    return h / 3 \* sum;

}

// Таблиця Ромберга

long double romberg (long double b, long double w, int k) {

    long double table [k + 1] [k + 1] = {0};

    for (int i = 0; i <= k; i ++) {

        table [i] [0] = squares (b, w, pow (2, i));

    }

    for (int j = 1; j <= k; j ++) {

        for (int i = j; i <= k; i ++) {

            table [i] [j] = (pow (4, j) \* table [i] [j - 1] -

                table [i - 1] [j - 1]) / (pow (4, j) - 1);

        }

    }

    return table [k][k];

}

int main () {

    // Точність та кількість кроків для обчислення інтегралу

    long double e = 0.00001;

    int n = eton (e);

    cout << "Precision: " << e << ", iterations: " << n << endl;

    for (auto w : {1, 10, 20, 50}) {

        cout << "w: " << w << endl;

        // Обчислення верхньої межі інтегрування методом обрізання границь

        long double upper = limitrunc (w, e);

        cout << "Upper integration bound: " << upper << endl;

        // Обчислення інтегралу методом середніх прямокутників

        cout << "Middle squares method: " << squares (upper, w, n) << endl;

        // Обчислення інтегралу методом Сімпсона

        double resultSimpson = simpson (upper, w, n);

        cout << "Simpson method: " << resultSimpson << endl;

        // Обчислення оцінки залишкових членів

        cout << "Residuals estimate: " << estimate (upper, n) << endl;

        // Обчислення інтегралу за допомогою таблиці Ромберга

        cout << "Romberg Table Method: " << romberg (upper, w, 5) << endl;

        cout << endl;

    }

    return 0;

}

Цей код виконує числове обчислення інтегралів для заданої функції на відрізку [0, *upper*], де *upper* обчислюється за допомогою методу обрізання границь. Програма використовує кілька методів для обчислення інтегралу, таких як метод середніх прямокутників, метод Сімпсона та таблиця Ромберга. Нижче є опис основних елементів коду:

1. Функція *f*: Визначає функцію, яку програма інтегрує.

2. Функція *limitrunc*: Визначає верхню межу інтегрування за допомогою методу обрізання границь. Використовується для визначення *upper*, який є верхньою межею інтегрування в головній функції.

3. Функція *eton*: Перетворює точність *e* в кількість ітерацій.

4. Функція *estimate*: Обчислює оцінку залишкових членів для методу середніх прямокутників.

5. Функції *squares* і *simpson*: Реалізують методи середніх прямокутників та Сімпсона відповідно.

6. Функція *romberg*: Реалізує метод Ромберга для обчислення інтегралу.

7. Головна функція *main*: Встановлює точність *e*, обчислює кількість ітерацій *n* за допомогою *eton*, і потім виконує обчислення інтегралів для різних значень параметра *w* (1, 10, 20, 50) з використанням різних методів.

- Виводиться точність та кількість ітерацій.

- Для кожного *w* обчислюється верхня межа інтегрування *upper* за допомогою методу обрізання границь.

- Обчислюється і виводиться інтеграл за допомогою методу середніх прямокутників.

- Обчислюється і виводиться інтеграл за допомогою методу Сімпсона.

- Обчислюється і виводиться оцінка залишкових членів.

- Обчислюється і виводиться інтеграл за допомогою таблиці Ромберга.

Код розрахований на обчислення інтегралу для різних значень параметра *w* з використанням різних чисельних методів, а також на аналіз точності результатів, чого і вимагає поставлена задача.

Приклади роботи програми

Приклад №1

Вивід:

Precision: 1e-05, iterations: 317

w: 1

Upper integration bound: 12

Middle squares method: 0.404503

Simpson method: 0.404443

Residuals estimate: 3.28555e-06

Romberg Table Method: 0.411973

w: 10

Upper integration bound: 12

Middle squares method: 0.100694

Simpson method: 0.100106

Residuals estimate: 3.28555e-06

Romberg Table Method: 0.586016

w: 20

Upper integration bound: 12

Middle squares method: 0.0512141

Simpson method: 0.0500976

Residuals estimate: 3.28555e-06

Romberg Table Method: -0.979132

w: 50

Upper integration bound: 12

Middle squares method: 0.0233298

Simpson method: 0.0223946

Residuals estimate: 3.28555e-06

Romberg Table Method: 0.228789

Як можна помітити, програма визначила кількість ітерацій як 317 та верхню межу інтегрування як 12. У ході виконання для різних значень *w* методи середніх прямокутників та Сімпсона давали схожі результати, у той час як таблиця Ромберга інколи давала похибку. Загалом програма не зовсім дотрималась поставленої точності через поганий код, але вона намагалася та може колись буде покращена і зможе досягти поставленої задачі.