Лабораторная работа №4 «Численное интегрирование»

Николаева Ксения, 9 группа

Содержание

1 Постановка задачи и используемые ресурсы

Задание 1. Вычислить интеграл

$$\int_2^9 \sqrt{\frac{9-2x}{2x-21}}dx \tag{1}$$

с точностью $\varepsilon = 10^{-7}$, используя составные квадратурные формулы трапеций и Симпсона, и правило Рунге оценки погрешности.

Задание 2. Вычислить приближенное значение интеграла из задания 1, используя квадратурную формулу наивысшей алгебраической степени точности (НАСТ) с 5 узлами.

Для реализации вычислений использовался язык программирования C++.

2 Теория и расчетные формулы

2.1 Составная квадратурная формула трапеций

Пусть отрезок $[a, b]$, на котором вычисляется интеграл, разбит на N равных отрезков узлами $a = x_0, x_1, ...x_N = b$, расстояние между двумя соседними узлами равно $h = (b - a)/N$. Тогда приближенное значение интеграла можно вычислить по формуле трапеций:

$$\int_a^b f(x)dx \approx Q_h = \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{N-1} f(x_i) + f(x_N)\right] \tag{2}$$

Порядок точности формулы $m = 2$.

2.2 Составная квадратурная формула Симпсона

Пусть отрезок $[a, b]$, на котором вычисляется интеграл, разбит на N равных отрезков узлами $a = x_0, x_1, ...x_N = b$, расстояние между двумя соседними узлами равно $h = (b - a)/N$. Пусть также возможно вычислить значение подынтегральной функции в точках $x_{i+1/2} = x_i + \frac{h}{2}, i = \overline{0, N-1}$. Тогда приближенное значение интеграла можно вычислить по формуле Симпсона:

$$\int_a^b f(x)dx \approx Q_h = \frac{h}{6}\sum_{i=0}^{N-1}\left[f(x_i) + 4f(x_{i+1/2}) + f(x_{i+1})\right] \tag{3}$$

Порядок точности формулы $m = 4$.

2.3 Правило Рунге

Обозначим приближенное значение определенного интеграла, вычисленное с помощью составных квадратурных формул как Q_h, где $h = (b - a)/N$ – расстояние между двумя соседними узлами.

При уменьшении расстояния h вдвое, получим $Q_{h/2}$. Апостериорная оценка погрешности вычислений по правилу Рунге вычисляется следующим образом:

$$R_{h/2} = \frac{Q_{h/2} - Q_h}{2^m - 1} \tag{4}$$

Здесь m – порядок точности соответствующей квадратурной формулы.

2.4 Квадратурная формула наивысшей алгебраической степени точности

Формула НАСТ для вычисления приближенного значения определенного интеграла с весовой функцией $\rho(x)$ для количества узлов k в общем виде:

$$\int_a^b \rho(x)f(x)dx \approx \sum_{k=0}^{n} A_k f(x_k) \tag{5}$$

где $n = k - 1$ – количество подотрезков $[a, b]$.

Для вычисления приближенного значения необходимо знать A_k и x_k. Чтобы получившаяся формула имела наивысшую алгебраическую степень точности $(2n+1)$, для нахождения КФ используется следующий алгоритм (для пределов интегрирования $[-1, 1]$):

1. Вводится многочлен вида:

$$\omega_{n+1}(x) = x^{n+1} + a_0 x^n + ... + a_n \tag{6}$$

2. Решается СЛАУ:

$$\int_{-1}^{1} \rho(x)\omega_{n+1}(x)x^i dx = 0, \quad i = 0, ..., n \tag{7}$$

   Отсюда находятся коэффициенты многочлена из п.1

3. Многочлен из п.1 приравнивается к 0 и решается уравнение, откуда находятся x_k.

4. Коэффициенты A_k находятся по формуле:

$$A_k = \int_{-1}^{1} \rho(x) \prod_{j=0, j\neq k}^{n} \frac{x - x_j}{x_k - x_j} dx, \quad k = 0, ..., n \tag{8}$$

Для перехода от стандартного отрезка $[-1, 1]$ к произвольному отрезку $[a, b]$ используется подстановка:

$$x = \frac{b-a}{2}t + \frac{b+a}{2}, \quad dx = \frac{b-a}{2}dt \tag{9}$$

Тогда интеграл принимает вид:

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{b+a}{2}\right) dt \tag{10}$$

3 Результаты

3.1 Результаты вычислений по составным квадратурным формулам

Таблица 1: Результаты вычислений по составным квадратурным формулам

КФ	Число разбиений	Шаг	Приближенное значение	Оценка погрешност
Трапеций	$N = 2$	$h = 1.5$	$Q_h = 3.2320508$	-
	$4N = 8$	$h/2 = 0.75$	$Q_{h/2} = 3.1652187$	$R_{h/2} = -0.0222774$
	$8N = 16$	$h/4 = 0.375$	$Q_{h/4} = 3.1475782$	$R_{h/4} = -0.0058802$
	$16N = 32$	$h/8 = 0.1875$	$Q_{h/8} = 3.1430944$	$R_{h/8} = -0.0014946$
	$32N = 64$	$h/16 = 0.09375$	$Q_{h/16} = 3.1419684$	$R_{h/16} = -0.0003753$
	$64N = 128$	$h/32 = 0.046875$	$Q_{h/32} = 3.1416866$	$R_{h/32} = -0.0000939$
	$128N = 256$	$h/64 = 0.0234375$	$Q_{h/64} = 3.1416161$	$R_{h/64} = -0.0000235$
	$256N = 512$	$h/128 = 0.0117188$	$Q_{h/128} = 3.1415985$	$R_{h/128} = -0.0000059$
	$512N = 1024$	$h/256 = 0.0058594$	$Q_{h/256} = 3.1415941$	$R_{h/256} = -0.0000015$
	$1024N = 2048$	$h/512 = 0.0029297$	$Q_{h/1024} = 3.1415930$	$R_{h/1024} = -0.000000$
	$2048N = 4096$	$h/1024 = 0.0014648$	$Q_{h/1024} = 3.1415927$	$R_{h/1024} = -0.000000$
Симпсона	$N = 2$	$h = 1.5$	$Q_h = 3.1429414$	-
	$4N = 8$	$h/2 = 0.75$	$Q_{h/2} = 3.1416980$	$R_{h/2} = -0.0000829$
	$8N = 16$	$h/4 = 0.375$	$Q_{h/4} = 3.1415998$	$R_{h/4} = -0.0000065$
	$16N = 32$	$h/8 = 0.1875$	$Q_{h/8} = 3.1415931$	$R_{h/8} = -0.0000004$
	$32N = 64$	$h/16 = 0.09375$	$Q_{h/16} = 3.1415927$	$R_{h/16} = -0.0000000$

Точное значение интеграла, полученное аналитически: $I = \pi = 3.1415926$.

3.2 Результаты вычислений по квадратурной формуле НАСТ

Для квадратурной формулы НАСТ с 5 узлами были использованы следующие веса и узлы:

k	x_k	A_k
0	-0.906180	0.236927
1	-0.538469	0.478629
2	0.000000	0.568889
3	0.538469	0.478629
4	0.906180	0.236927

Приближенное значение интеграла, вычисленное с помощью КФ НАСТ с 5 узлами: $Q = 3.1415905$.

Абсолютная погрешность: $|I - Q| = 0.0000021$.

4 Выводы

1. Использование составных квадратурных формул трапеций и Симпсона в сочетании с правилом Рунге для достижения заданной точности является эффективным способом вычисления приближенного значения определенного интеграла.

2. Для достижения заданной точности $\varepsilon = 10^{-7}$ при использовании формулы трапеций потребовалось 16384 разбиений, в то время как при использовании формулы

Симпсона было достаточно 512 разбиений. Это демонстрирует превосходство формулы Симпсона в эффективности, что объясняется более высоким порядком точности ($m = 4$ для формулы Симпсона против $m = 2$ для формулы трапеций).

3. Квадратурная формула наивысшей алгебраической степени точности (НАСТ) с 5 узлами демонстрирует высокую точность, обеспечивая погрешность порядка 10^{-7} без необходимости разбиения отрезка интегрирования на мелкие части. Это связано с тем, что формула НАСТ оптимально выбирает как узлы, так и веса интегрирования.

4. При решении практических задач численного интегрирования рекомендуется:

   - Для интегралов с гладкими подынтегральными функциями на небольших отрезках использовать КФ НАСТ
   - При необходимости интегрирования на большом отрезке или для функций с особенностями использовать составные квадратурные формулы, предпочтительно формулу Симпсона как более эффективную

Таким образом, при выборе метода численного интегрирования следует учитывать требуемую точность, характер подынтегральной функции и вычислительные ресурсы.

5 Листинг программы с комментариями

```
   #include <iostream>
#include <cmath>
#include <iomanip>
#include <limits>
#include <algorithm>

using namespace std;

#define M_PI 3.14159265358979323846

// f(x) = sqrt((9-2x)/(2x-21))
double f(double x) {
    return sqrt((9 - 2 * x) / (2 * x - 21));
}

// КФ трапеций
double trapezoidalRule(double a, double b, int n) {
    double h = (b - a) / n;
    double sum = 0.5 * (f(a) + f(b));

    for (int i = 1; i < n; i++) {
        double x = a + i * h;
        sum += f(x);
    }

    return h * sum;
}

// КФ Симпсона
double simpsonRule(double a, double b, int n) {
    double h = (b - a) / n;
    double sum = 0.0;

    for (int i = 0; i < n; i++) {
        double x0 = a + i * h;
        double x1 = a + (i + 1) * h;
        double xm = (x0 + x1) / 2.0;

        sum += f(x0) + 4 * f(xm) + f(x1);
    }

    return (h / 6.0) * sum;
}

// Метод численного интегрирования с использованием правила Рунге
double rungeIntegration(double a, double b, double eps, int m, bool useTrapezoidal) {
    int n = 2; // Начальное количество разбиений
    int k = 0; // Счетчик итераций
```

```
    double h = (b - a) / n; // Начальный шаг

    // Вычисляем первое приближение интеграла
    double Q_h = useTrapezoidal ? trapezoidalRule(a, b, n) : simpsonRule(a, b, n);
    cout << "N=" << n << ", h=" << h << ", Q_h=" << setprecision(7) << fixed << Q_h << e

    double Q_h_div_2, R_h_div_2;

    do {
        n *= 2;
        k++;
        h /= 2;

        // Вычисляем следующее приближение интеграла с шагом h/2
        Q_h_div_2 = useTrapezoidal ? trapezoidalRule(a, b, n) : simpsonRule(a, b, n);

        // Оценка погрешности по правилу Рунге
        R_h_div_2 = (Q_h_div_2 - Q_h) / (pow(2, m) - 1);

        cout << n << "N=" << 2 * n << ", h/" << pow(2, k) << "=" << h << ", Q_h/2^" << k
            << ", R_h/2^" << k << "=" << R_h_div_2 << endl;

        Q_h = Q_h_div_2;

    } while (fabs(R_h_div_2) > eps);

    return Q_h_div_2;
}

// Вычисление интеграла с помощью КФ НАСТ с 5 узлами
double gaussLegendreIntegration(double a, double b) {
    // Узлы КФ НАСТ (корни многочлена Лежандра) на отрезке [-1, 1]
    const double x_k[5] = { -0.9061798459, -0.5384693101, 0.0, 0.5384693101, 0.906179845

    // Веса КФ НАСТ
    const double A_k[5] = { 0.2369268851, 0.4786286705, 0.5688888889, 0.4786286705, 0.23

    double sum = 0.0;

    // Вычисление интеграла
    double mid = (b + a) / 2.0;
    double half_length = (b - a) / 2.0;

    for (int i = 0; i < 5; i++) {
        // Преобразование координаты от [-1, 1] к [a, b]
        double x = mid + half_length * x_k[i];
        sum += A_k[i] * f(x);
    }
    return half_length * sum;
}
```

```
int main() {
    setlocale(LC_ALL, "ru");

    double a = 6.0;
    double b = 9.0;
    double eps = 1e-7;
    double exact_value = M_PI;

    cout << "Численное интегрирование интеграла sqrt((9-2x)/(2x-21)) от " << a << " до "
    cout << "Требуемая точность: " << eps << endl;
    cout << "Точное значение интеграла: " << setprecision(7) << fixed << exact_value <<

    cout << "Квадратурная формула трапеций:" << endl;
    double trap_result = rungeIntegration(a, b, eps, 2, true);
    cout << "Итоговое значение (КФ трапеций): " << trap_result << endl;
    cout << "Абсолютная погрешность: " << fabs(trap_result - exact_value) << endl << end

    cout << "Квадратурная формула Симпсона:" << endl;
    double simpson_result = rungeIntegration(a, b, eps, 4, false);
    cout << "Итоговое значение (КФ Симпсона): " << simpson_result << endl;
    cout << "Абсолютная погрешность: " << fabs(simpson_result - exact_value) << endl <<

    cout << "Квадратурная формула НАСТ с 5 узлами:" << endl;
    double gauss_result = gaussLegendreIntegration(a, b);
    cout << "Значение интеграла (КФ НАСТ): " << setprecision(7) << fixed << gauss_result
    cout << "Абсолютная погрешность: " << fabs(gauss_result - exact_value) << endl;

    return 0;
}
```