Белорусский Государственный Университет

Методы Численного Анализа

Лабораторная работа 8

Численное интегрирование

Рак Алексей

1 июня 2017 г.

Постановка Задачи

$$f(x) = \frac{2^x}{1-4^x} \qquad a = -2 \qquad b = -1 \qquad \epsilon = 10^{-5}$$

Дан интеграл $\int_a^b$. Требуется:

1) Используя квадратурное правило с погрешностью $O(h)$ (правило левых прямоугольников) вычислить по правилу Рунге интеграл с точностью ϵ.

2) Определить шаг h в квадратурных формулах трапеции и Симпсона, достаточный для получения результата с точностью ϵ.

3) Найти количество n узлов квадратурной формулы НАСТ, достаточное для достижения точности ϵ и рассчитать при этом приближенное значение интеграла.

Алгоритм решения и формулы

1) Для первой итерации возьмём $n = 1,\ x_0 = a,\ h = b - a$. Интеграл будем приближённо высчитывать по формуле левых прямоугольников:

$$I_h \approx h \sum_{i=0}^{n-1} f(a + ih)$$

Поскольку формула левых прямоугольников имеет порядок точности $p = 1$, то будем проводить итерации пока не будет выполнено:

$$|I_{\frac{h}{2}} - I_h| < \epsilon$$

После чего находим интеграл по формуле:

$$I \approx I_{\frac{h}{2}} + I_{\frac{h}{2}} - I_h$$

Дополнительно оценим погрешность формулы левых прямоугольников:

$$|R_{lr} \leq \frac{M(b-a)^2}{2n}$$

где $M = \sup_{x\in[-2,-1]} |f'(x)|$ вычислим аналитически $M = \frac{10 \ln 2}{9}$

2) Будем находить количество промежутков n, используемых в квадратурной формуле трапеции, применяя погрешности:

$$|R_{tr}| \leq \frac{Mh^2(b-a)}{12}$$

где $M = \sup_{x\in[-2,-1]} |f''(x)| = \frac{(2+6*8+32)\ln^2 2}{3^3}$ откуда вычисляем максимальный возможный для данного ϵ шаг:

$$h \leq \sqrt{|\frac{12\epsilon}{(b-a)M}|}$$

Аналогично для формулы Симпсона получаем:

$$h \leq \sqrt[4]{\frac{180\epsilon}{(b-a)M}}$$

3) Для нахождения количества узлов n квадратурной формулы НАСТ, достаточное для достижения точности ϵ, воспользуемся оценкой погрешности:

$$|R_{Gauss} \leq \frac{(b-a)2^{2n}(n!)^4 M}{(2n+1)((2n)!)^3}$$

где $M = \sup_{x\in[-2,-1]} |f^{2n}(x)|$. Будем последовательно увеличивать n до тех пор, пока не станет выполено:

$$\frac{(b-a)2^{2n}(n!)^4 M}{(2n+1)((2n)!)^3} < \epsilon$$

Листинг

```
#include <cmath>
#include <iostream>

double function(double x) {
    return pow(2.0, x) / (1 - pow(4.0, x));
}

double left_rectangle(double a, double b, double eps) {
    double old_int = function(a) * (b - a);
    int n = 2;
    while (true) {
        double new_int = 0;
        double h = (b - a) / n;
        for (int i = 0; i < n; ++i) {
            new_int += h * function(a + i * h);
        }
        if (fabs(new_int - old_int) < eps) {
            old_int = new_int + new_int - old_int;
            break;
        }
        old_int = new_int;
        n *= 2;
    }
    return old_int;
}

double get_trapeze_h(double a, double b, double df2max, double eps) {
    return sqrt(12 * eps / (b - a) / df2max);
}

double get_Simpson_h(double a, double b, double df4max, double eps) {
    return pow(eps * 180 / (b - a) / df4max, 0.25);
}

double factorial(int n) {
    double fact = 1;
    int i = 1;
    while (++i <= n) {
        fact *= i;
    }
    return fact;
}

int get_gauss_n(double a, double b, double eps) {
    int n = 1;
    double r = 1;
    double dfmax[16] = {0.8, 1.6, 5, 20, 100, 350, 2500, 2e4, 1.4e5, 1.2e6, 1.2e7, 1.2e8, 1.2e9,
                        1.4e10, 2e11, 4e12};
    while (n < 9 && r > eps) {
        r = dfmax[2 * n - 1] * (b - a) * pow(2.0, 2.0 * n) * pow(factorial(1 * n), 4.0) /
            (2.0 * n + 1.0) / pow(factorial(2 * n), 3.0);
        ++n;
    }
    return n - 1;
}

int main() {
    double df1max = log(2.0) * (pow(2.0, -1.0) + pow(8.0, -1.0)) / pow(1.0 - pow(4.0, -1.0), 2.0);
    double df2max = pow(log(2.0), 2.0) * (pow(2.0, -1.0) + 6.0 * pow(8.0, -1.0) + pow(32.0, -1.0)) /
                    pow(1.0 - pow(4.0, -1.0), 3.0);
    double df4max = pow(log(2.0), 4.0) *
```

```
                    (pow(2.0, -1.0) + 76.0 * pow(8.0, -1.0) + 230 * pow(32.0, -1.0) +
                     76.0 * pow(128.0, -1.0) + pow(512.0, -1.0)) / pow(1.0 - pow(4.0, -1.0), 5.0);
    double a = -2;
    double b = -1;
    double eps = 1e-5;
    std::cout « "Function = " « log(1.8) / log(4.0) « std::endl;
    std::cout « "Left Rectangle integral = " « left_rectangle(a, b, eps) « std::endl;
    std::cout « "Eps = " « fabs(left_rectangle(a, b, eps) - log(1.8) / log(4.0)) « std::endl;
    std::cout « "Trapeze h = " « get_trapeze_h(a, b, df2max, eps) « std::endl;
    std::cout « "Simpson h = " « get_Simpson_h(a, b, df4max, eps) « std::endl;
    std::cout « "Gauss n = " « get_gauss_n(a, b, eps) « std::endl;
    return 0;
}
```

Результаты

Точное значение интеграла: 0.423998
Значение интеграла вычисленное с помощью правила Рунге: 0.423998
Разница между точным и приближенным значением: $8.70284e^{-11}$
Шаг для метода трапеций: 0.0090686
Шаг для метода Симпсона: 0.100999
Число точек для метода Гаусса: 8

Вывод

Отсюда видно, что методу Гаусса понадобилось наименьшее число точек, также неплохо себя проявил метод Симпсона, число точек для которого чуть меньше чем для метода Гаусса. Метод трапеций работает на порядок хуже метода Симпсона и метода Гаусса.