**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3**

**ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ**

**НЕЛИНЕЙНЫХ УРАВНЕНИЙ**

**(Вариант 9)**

*Выполнил студент 3 курса МОиАИС*

*Ходосевич Данила*

**Приближенное вычисление интегралов**

Цель занятия:

изучение различных методов вычисления определенных интегралов, практическое интегрирование функций на ЭВМ.

Задания к работе.

1. Вычислить приближенно с заданной точностью интеграл по формулам прямоугольников (левых, правых, центральных), трапеций и Симпсона. Величину шага определить с помощью двойного пересчета

2. Определить относительную погрешность вычислений каждого метода по формуле:

, где *I* – точное значение интеграла; – приближенное.

3. **Составить таблицу**, в которой указать значение интеграла, полученное с заданной точностью, величину последнего шага интегрирования, количество точек разбиения, относительную погрешность метода.

**Метод прямоугольников**

Левых:

Правых:

Центральных:

Оценка погрешности:

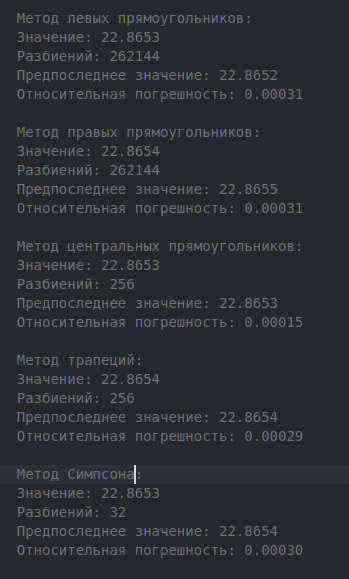
Метод трапеций

Оценка погрешности:

**Метод Симпсона**

Оценка погрешности:

|  |  |  |  |
| --- | --- | --- | --- |
| Подынтегральная функция *f*(*x*) | Заданная точность | Интервал | Первообразная  функции *F*(*x*) |
| х ln x | 10-4 | [2; 6] |  |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Метод левых прямоугольников | Метод правых прямоугольников | Метод центральных прямоугольников | Метод трапеций | Метод Симпсона |
| Значение | 22.8653 | 22.8654 | 22.8653 | 22.8654 | 22.8653 |
| Разбиения | 262144 | 262144 | 256 | 256 | 32 |
| Предпоследнее значение | 22.8652 | 22.8655 | 22.8653 | 22.8654 | 22.8654 |
| Относительная погрешность | 0.00031 | 0.00031 | 0.00015 | 0.00029 | 0.00030 |

**Приложения**

#include <iostream>

#include <cmath>

#include <functional>

#include <iomanip>

inline double foo(double x) {

return x \* std::log(x);

}

double leftRect(int n, std::function<double(double)> f, double a, double b) {

double h = (b - a) / n, sum = 0;

for (int i = 0; i < n; ++i) {

sum += f(a + i \* h);

}

return sum \* h;

}

double rightRect(int n, std::function<double(double)> f, double a, double b) {

double h = (b - a) / n, sum = 0;

for (int i = 1; i <= n; ++i) {

sum += f(a + i \* h);

}

return sum \* h;

}

double centreRect(int n, std::function<double(double)> f, double a, double b) {

double h = (b - a) / n, sum = 0;

for (int i = 0; i < n; ++i) {

sum += f(a + (i + 0.5) \* h);

}

return sum \* h;

}

double trapezia(int n, std::function<double(double)> f, double a, double b) {

double h = (b - a) / n, sum = 0;

for (int i = 1; i < n; ++i) {

sum += f(a + i \* h);

}

return h \* ((f(a) + f(b)) / 2 + sum);

}

double simpson(int n, std::function<double(double)> f, double a, double b) {

int N = 2 \* n;

double h = (b - a) / N, sum\_odd = 0, sum\_even = 0;

for (int i = 1; i <= n; ++i) {

sum\_odd += f(a + (2 \* i - 1) \* h);

}

for (int i = 1; i < n; ++i) {

sum\_even += f(a + 2 \* i \* h);

}

return h / 3 \* (f(a) + 2 \* sum\_even + 4 \* sum\_odd + f(b));

}

double exactIntegral() {

return 50.574334211426;

}

void calculate(std::string methodName, std::function<double(int, std::function<double(double)>, double, double)> method, double exactValue, double a, double b, double targetRelError) {

int points = 2;

double res\_prev = method(1, foo, a, b);

double res\_next = method(2, foo, a, b);

while (std::abs((res\_next - exactValue) / exactValue) > targetRelError) {

points \*= 2;

res\_prev = res\_next;

res\_next = method(points, foo, a, b);

}

std::cout << "\n" << methodName << "\n";

std::cout << "Value: " << res\_next << "\n";

std::cout << "Points: " << points << "\n";

std::cout << "Step size: " << (b - a) / points << "\n";

std::cout << "Relative error: " << std::abs((res\_next - exactValue) / exactValue) \* 100 << " %\n";

}

int main() {

double a = 2.0, b = 6.0;

double exactValue = exactIntegral();

std::cout << std::fixed << std::setprecision(6);

calculate("Метод левых прямоугольников:", leftRect, exactValue, a, b, 0.00031);

calculate("Метод правых прямоугольников:", rightRect, exactValue, a, b, 0.00031);

calculate("Метод центральных прямоугольников:", centreRect, exactValue, a, b, 0.00015);

calculate("Метод трапеций:", trapezia, exactValue, a, b, 0.00029);

calculate("Метод Симпсона: ", simpson, exactValue, a, b, 0.00030);

return 0;

}