Санкт-Петербургский государственный университет

Прикладная математика и информатика

Отчет по лабораторной работе №4.1 по вычислительному практикуму

**Приближённое вычисление интеграла по квадратурным формулам**

Выполнил:

Чернов Павел Олегович

группа 221

Санкт-Петербург

2022

1. **Введение**

**Приближённое вычисление интеграла по квадратурным формулам**

Написать программу для вычисления определенного интеграла при помощи предложенных квадратурных формул (КФ).

1. Параметры задачи: пределы интегрирования a, b (запрашивать у пользователя, вводить с клавиатуры).
2. Для случая легко интегрируемой функции f(x) (выбрать на свое усмотрение) вычислить точно и вывести на печать значение интеграла от f(x) по конечному [a, b] (считаем вес ρ(x) ≡ 1). (Обозначим это значение за J).
3. Вычислить приближённо и вывести на печать значение интеграла от f(x) по [a, b] при помощи
   * КФ левого прямоугольника;
   * КФ правого прямоугольника;
   * КФ среднего прямоугольника;
   * КФ трапеции;
   * КФ Симпсона (или парабол);
   * КФ3/8.
4. Посчитать и вывести на печать абсолютную фактическую погрешность для каждой КФ.

Замечание 1:

Обязательно описать в программе кроме произвольной функции также функции-многочлены от нулевой до третьей степени включительно.

1. **Ход работы**
2. Представленная ниже программа позволяет аппроксимировано вычислить значение собственного интеграла с заданными пределами интегрирования и весовой функцией, константно равной единице.
3. Для приближенного вычисления решений используются КФ левого прямоугольника, правого прямоугольника, среднего прямоугольника, трапеции, Симпсона, трех восьмых.
4. Исходным параметром задачи является функция , а также функции-многочлены . В программе можно задать а) a и b – пределы интегрирования. Ввод осуществляется при помощи клавиатуры.
5. Программа написана на языке С++, в расчетах используются переменные типа double.
6. **Приложения**

А) Программа

#include "functions.h"

double f(double x) { //интегрируемая функция

return log(x) \* pow(x, 3);

}

double integr\_of\_f(double x) { //Ее интеграл

return pow(x, 4) \* (4 \* log(x) - 1) / 16;

}

double f\_0(double x) //интегрируемая функция, многочлен 0 степени

{

return(7);

}

double integr\_0(double x) //интеграл многочлена 0 степени

{

return(7 \* x);

}

double f\_1(double x) //интегрируемая функция, многочлен 1 степени

{

return(4 \* x - 3);

}

double integr\_1(double x) //интеграл многочлена 1 степени

{

return 2 \* pow(x, 2) - 3 \* x;

}

double f\_2(double x) //интегрируемая функция, многочлен 2 степени

{

return(-pow(x, 2) - 8 \* x + 1);

}

double integr\_2(double x) //интеграл многочлена 2 степени

{

return -pow(x, 3) / 3 - 4 \* pow(x, 2) + x;

}

double f\_3(double x) //интегрируемая функция, многочлен 3 степени

{

return 2 \* pow(x, 3) - 5 \* x - 2;

}

double integr\_3(double x)

{

return pow(x, 4) / 2 - 2.5 \* pow(x, 2) - 2 \* x; //интеграл многочлена 3 степени

}

#include <iomanip>

#include <iostream>

#include "functions.h"

double Newton\_Leibniz(double (\*integr)(double x), double a, double b)

{

return(integr(b) - integr(a));

}

double left\_rectangle(double (\*f)(double x), double a, double b)

{

return((b - a) \* f(a));

}

double right\_rectangle(double (\*f)(double x), double a, double b)

{

return((b - a) \* f(b));

}

double middle\_rectangle(double (\*f)(double x), double a, double b)

{

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

}

double trapeze(double (\*f)(double x), double a, double b)

{

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

}

double Simpson(double (\*f)(double x), double a, double b)

{

return((b - a) / 6 \* (f(a) + 4 \* f((a + b) / 2) + f(b)));

}

double three\_eighths(double (\*f)(double x), double a, double b)

{

double h = (b - a) / 3;

return((b - a) \* (0.125 \* f(a) + 0.375 \* f(a + h) + 0.375 \* f(a + 2 \* h) + 0.125 \* f(b)));

}

void print\_all(double (\*f)(double x), double (\*integr)(double x), double a, double b) {

std::cout << "Left rectangle: " << left\_rectangle(f, a, b) << "\nAbsolute error value: " << abs(left\_rectangle(f, a, b) - Newton\_Leibniz(integr, a, b)) << "\n\n";

std::cout << "Right rectangle: " << right\_rectangle(f, a, b) << "\nAbsolute error value: " << abs(right\_rectangle(f, a, b) - Newton\_Leibniz(integr, a, b)) << "\n\n";

std::cout << "Middle rectangle: " << middle\_rectangle(f, a, b) << "\nAbsolute error value: " << abs(middle\_rectangle(f, a, b) - Newton\_Leibniz(integr, a, b)) << "\n\n";

std::cout << "Trapeze: " << trapeze(f, a, b) << "\nAbsolute error value: " << abs(trapeze(f, a, b) - Newton\_Leibniz(integr, a, b)) << "\n\n";

std::cout << "Simpson: " << Simpson(f, a, b) << "\nAbsolute error value: " << abs(Simpson(f, a, b) - Newton\_Leibniz(integr, a, b)) << "\n\n";

std::cout << "Three\_eighths: " << three\_eighths(f, a, b) << "\nAbsolute error value: " << abs(three\_eighths(f, a, b) - Newton\_Leibniz(integr, a, b)) << "\n\n";

}

int main() {

std::cout << std::fixed << std::setprecision(16);

bool flag;

do {

std::cout << "Input integration confines: ";

double a, b; //Пределы интегрирования

std::cin >> a >> b;

bool switching\_fs;

do {

std::cout << "Choose function which you want to integrate:\n" <<

"1. f = ln(x)\*x^3\n" << "2. f = 7\n" << "3. f = 4\*x - 3\n" << "4. f = -x^2 - 8\*x + 1\n" << "5. f = 2\*x^3 - 5\*x - 2\n";

int chosen;

std::cin >> chosen;

std::cout << "Result by all methods = \n";

switch (chosen) {

case 1:

print\_all(f, integr\_of\_f, a, b);

break;

case 2:

print\_all(f\_0, integr\_0, a, b);

break;

case 3:

print\_all(f\_1, integr\_1, a, b);

break;

case 4:

print\_all(f\_2, integr\_2, a, b);

break;

case 5:

print\_all(f\_3, integr\_3, a, b);

break;

}

std::cout << "If you would like to choose another function, press 1, otherwise 0\n";

std::cin >> switching\_fs;

} while (switching\_fs);

std::cout << "If you want change integration confines, press 1, otherwise 0\n";

std::cin >> flag;

} while (flag);

return 0;

}

https://github.com/Chernovuk/Chislaki/tree/master/Fourth\_task/4.1