Санкт-Петербургский государственный университет

Прикладная математика и информатика

Отчет по лабораторной работе №4.2-4.3 по вычислительному практикуму

**Приближённое вычисление интеграла по составным квадратурным формулам**

Выполнил:

Чернов Павел Олегович

группа 221

Санкт-Петербург

2022

1. **Введение**

**Приближённое вычисление интеграла по составным квадратурным формулам**

Написать программу для вычисления определенного интеграла при помощи составных квадратурных формул (СКФ).

1. Параметры задачи: пределы интегрирования A, B, весовая функция ρ(x) и функция f(x), m – число промежутков деления [A, B].
2. Для случая ρ(x) ≡ 1 и легко интегрируемой функции f(x) вычислить точно и вывести на печать значение интеграла от ρ(x)∙f(x) по конечному [A, B]. (Обозначим это значение за J).
3. Вычислить приближённо и вывести на печать значение интеграла от ρ(x)∙f(x) по [A, B] при помощи СКФ

* Левых прямоугольников;
* Правых прямоугольников;
* Средних прямоугольников;
* Трапеций;
* Симпсона;

с параметром m. Обозначим эти значения J(h), здесь h= (B-A)/m.

1. Посчитать и вывести на печать │J− J(h)│─ абсолютную фактическую погрешность для каждой составной КФ.
2. Для каждой составной КФ оценить погрешность вычислений для фиксированного набора параметров (теоретически, смотри сводную оценку погрешности для СКФ).
3. Вывести на печать теоретическую оценку.
4. Увеличить m в l раз (здесь l – параметр, натуральное число; запрашивать у пользователя, вводить с клавиатуры).
5. Вычислить приближённо и вывести на печать значения интеграла от ρ(x)∙f(x) по [A, B], посчитанные при помощи составных формул левых, правых, средних прямоугольников, трапеций и Симпсона с новым числом делений [A, B] ‒ m\*l. (Обозначим это новое значение за J(h/l)).
6. Посчитать и вывести на печать абсолютную фактическую погрешность каждой формулы для случая нового числа промежутков разбиения m\*l.
7. Уточнить значения J(h)и J(h/l) по принципу Рунге для каждой СКФ.
8. Посчитать и вывести на печать абсолютные фактические погрешности для уточнённых значений.

Замечание 1:

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

Теоретическая погрешность = Const\*Md+1\*(b – a)\*hd+1, где d – АСТ СКФ, Md+1 = max[a,b]│f(d+1)(x)│, Const = 1/2 для СКФ левых и правых, 1/12 для трапеций, 1/24 для средних и 1/2880 для СКФ Симпсона.

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 СКФ левого прямоугольника, правого прямоугольника, среднего прямоугольника, трапеции, Симпсона.
4. Исходным параметром задачи является функция , а также функции-многочлены . В программе можно задать а) a и b – пределы интегрирования, б) m – количество разбиений заданного промежутка [a;b]. 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 df(double x) { //1-4 производные интегрируемой функции

return pow(x, 2) \* (3 \* log(x) + 1);

}

double d2f(double x) {

return x \* (6 \* log(x) + 5);

}

double d3f(double x) {

return 6 \* log(x) + 11;

}

double d4f(double x) {

return 6 / x;

}

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 df\_0(double x) { //1-4 производные многочлена 0 степени

return 0;

}

double d2f\_0(double x) {

return 0;

}

double d3f\_0(double x) {

return 0;

}

double d4f\_0(double x) {

return 0;

}

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 df\_1(double x) { //1-4 производные многочлена 1 степени

return 4;

}

double d2f\_1(double x) {

return 0;

}

double d3f\_1(double x) {

return 0;

}

double d4f\_1(double x) {

return 0;

}

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 df\_2(double x) { //1-4 производные многочлена 2 степени

return -2 \* x - 8;

}

double d2f\_2(double x) {

return -2;

}

double d3f\_2(double x) {

return 0;

}

double d4f\_2(double x) {

return 0;

}

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 df\_3(double x) { //1-4 производные многочлена 3 степени

return 6 \* pow(x, 2) - 5;

}

double d2f\_3(double x) {

return 12 \* x;

}

double d3f\_3(double x) {

return 12;

}

double d4f\_3(double x) {

return 0;

}

double integr\_3(double x)

{

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

}

#include "header.h"

#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, size\_t m)

{

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

for (size\_t j = 0; j < m; ++j)

{

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

}

sum \*= h;

return(sum);

}

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

{

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

for (size\_t j = 0; j < m; ++j)

{

sum += f(a + (j + 1) \* h);

}

sum \*= h;

return(sum);

}

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

{

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

for (size\_t j = 0; j < m; ++j)

{

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

}

sum \*= h;

return(sum);

}

double trapeze(double (\*f)(double x), double a, double b, size\_t m)

{

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

for (size\_t j = 1; j < m; ++j)

{

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

}

sum \*= 2;

sum += f(a) + f(b);

sum \*= h / 2;

return(sum);

}

double Simpson(double (\*f)(double x), double a, double b, size\_t m) //ACT = 3

{

double sum = 0, sum\_1 = 0, sum\_2 = 0, h = (b - a) / m;

sum += f(a) + f(b);

for (size\_t j = 1; j < m; ++j)

{

sum\_1 += f(a + j \* h);

}

sum\_1 \*= 2;

for (size\_t j = 0; j < m; ++j)

{

sum\_2 += f(a + (j + 0.5) \* h);

}

sum\_2 \*= 4;

sum += sum\_1 + sum\_2;

sum \*= h / 6;

return(sum);

}

double theor\_err\_value(double c, double M, double d, double a, double b, size\_t m)

{

double h = (b - a) / m;

return(c \* M \* (b - a) \* pow(h, d + 1));

}

double max(double x, double y) {

if (x > y) {

return x;

}

else {

return y;

}

}

double approx\_func\_max(double (\*f)(double x), double a, double b, size\_t m) {

double h = (b - a) / m;

double maxi = abs(f(a));

for (size\_t i = 1; i <= m; ++i) {

maxi = max(maxi, abs(f(a + i \* h)));

}

return maxi;

}

void print\_all(int switcher, double a, double b, size\_t m, size\_t l) {

const double c1 = 0.5, c2 = 1.0 / 12.0, c3 = 1.0 / 24.0, c4 = 1.0 / 2880.0;

double left\_rectangl, right\_rectangl, middle\_rectangl, trapez, Simpso;

double left\_ml, right\_ml, middle\_ml, trap\_ml, Simp\_ml;

switch (switcher) {

case 1:

left\_rectangl = left\_rectangle(f, a, b, m), right\_rectangl = right\_rectangle(f, a, b, m), middle\_rectangl = middle\_rectangle(f, a, b, m),

trapez = trapeze(f, a, b, m), Simpso = Simpson(f, a, b, m);

left\_ml = left\_rectangle(f, a, b, m \* l), right\_ml = right\_rectangle(f, a, b, m \* l), middle\_ml = middle\_rectangle(f, a, b, m \* l),

trap\_ml = trapeze(f, a, b, m \* l), Simp\_ml = Simpson(f, a, b, m \* l);

std::cout << "Left rectangle: " << left\_rectangl << "\nAbsolute error value: " << abs(left\_rectangl - Newton\_Leibniz(integr\_of\_f, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c1, approx\_func\_max(df, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(left\_rectangl, left\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_of\_f, a, b)) << "\n\n";

std::cout << "Right rectangle: " << right\_rectangl << "\nAbsolute error value: " << abs(right\_rectangl - Newton\_Leibniz(integr\_of\_f, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c1, approx\_func\_max(df, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(right\_rectangl, right\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_of\_f, a, b)) << "\n\n";

std::cout << "Middle rectangle: " << middle\_rectangl << "\nAbsolute error value: " << abs(middle\_rectangl - Newton\_Leibniz(integr\_of\_f, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c3, approx\_func\_max(df, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(middle\_rectangl, middle\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_of\_f, a, b)) << "\n\n";

std::cout << "Trapeze: " << trapez << "\nAbsolute error value: " << abs(trapez - Newton\_Leibniz(integr\_of\_f, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c2, approx\_func\_max(d2f, a, b, m), 1, a, b, m) <<

"\nRunge correction: " << Runge\_correction(trapez, trap\_ml, 1, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 1, l) - Newton\_Leibniz(integr\_of\_f, a, b)) << "\n\n";

std::cout << "Simpson: " << Simpso << "\nAbsolute error value: " << abs(Simpso - Newton\_Leibniz(integr\_of\_f, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c4, approx\_func\_max(d4f, a, b, m), 3, a, b, m) <<

"\nRunge correction: " << Runge\_correction(Simpso, Simp\_ml, 3, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 3, l) - Newton\_Leibniz(integr\_of\_f, a, b)) << "\n\n";

break;

case 2:

left\_rectangl = left\_rectangle(f\_0, a, b, m), right\_rectangl = right\_rectangle(f\_0, a, b, m), middle\_rectangl = middle\_rectangle(f\_0, a, b, m),

trapez = trapeze(f\_0, a, b, m), Simpso = Simpson(f\_0, a, b, m);

left\_ml = left\_rectangle(f\_0, a, b, m \* l), right\_ml = right\_rectangle(f\_0, a, b, m \* l), middle\_ml = middle\_rectangle(f\_0, a, b, m \* l),

trap\_ml = trapeze(f\_0, a, b, m \* l), Simp\_ml = Simpson(f\_0, a, b, m \* l);

std::cout << "Left rectangle: " << left\_rectangl << "\nAbsolute error value: " << abs(left\_rectangl - Newton\_Leibniz(integr\_0, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c1, approx\_func\_max(df\_0, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(left\_rectangl, left\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_0, a, b)) << "\n\n";

std::cout << "Right rectangle: " << right\_rectangl << "\nAbsolute error value: " << abs(right\_rectangl - Newton\_Leibniz(integr\_0, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c1, approx\_func\_max(df\_0, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(right\_rectangl, right\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_0, a, b)) << "\n\n";

std::cout << "Middle rectangle: " << middle\_rectangl << "\nAbsolute error value: " << abs(middle\_rectangl - Newton\_Leibniz(integr\_0, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c3, approx\_func\_max(df\_0, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(middle\_rectangl, middle\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_0, a, b)) << "\n\n";

std::cout << "Trapeze: " << trapez << "\nAbsolute error value: " << abs(trapez - Newton\_Leibniz(integr\_0, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c2, approx\_func\_max(d2f\_0, a, b, m), 1, a, b, m) <<

"\nRunge correction: " << Runge\_correction(trapez, trap\_ml, 1, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 1, l) - Newton\_Leibniz(integr\_0, a, b)) << "\n\n";

std::cout << "Simpson: " << Simpso << "\nAbsolute error value: " << abs(Simpso - Newton\_Leibniz(integr\_0, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c4, approx\_func\_max(d4f\_0, a, b, m), 3, a, b, m) <<

"\nRunge correction: " << Runge\_correction(Simpso, Simp\_ml, 3, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 3, l) - Newton\_Leibniz(integr\_0, a, b)) << "\n\n";

break;

case 3:

left\_rectangl = left\_rectangle(f\_1, a, b, m), right\_rectangl = right\_rectangle(f\_1, a, b, m), middle\_rectangl = middle\_rectangle(f\_1, a, b, m),

trapez = trapeze(f\_1, a, b, m), Simpso = Simpson(f\_1, a, b, m);

left\_ml = left\_rectangle(f\_1, a, b, m \* l), right\_ml = right\_rectangle(f\_1, a, b, m \* l), middle\_ml = middle\_rectangle(f\_1, a, b, m \* l),

trap\_ml = trapeze(f\_1, a, b, m \* l), Simp\_ml = Simpson(f\_1, a, b, m \* l);

std::cout << "Left rectangle: " << left\_rectangl << "\nAbsolute error value: " << abs(left\_rectangl - Newton\_Leibniz(integr\_1, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c1, approx\_func\_max(df\_1, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(left\_rectangl, left\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_1, a, b)) << "\n\n";

std::cout << "Right rectangle: " << right\_rectangl << "\nAbsolute error value: " << abs(right\_rectangl - Newton\_Leibniz(integr\_1, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c1, approx\_func\_max(df\_1, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(right\_rectangl, right\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_1, a, b)) << "\n\n";

std::cout << "Middle rectangle: " << middle\_rectangl << "\nAbsolute error value: " << abs(middle\_rectangl - Newton\_Leibniz(integr\_1, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c3, approx\_func\_max(df\_1, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(middle\_rectangl, middle\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_1, a, b)) << "\n\n";

std::cout << "Trapeze: " << trapez << "\nAbsolute error value: " << abs(trapez - Newton\_Leibniz(integr\_1, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c2, approx\_func\_max(d2f\_1, a, b, m), 1, a, b, m) <<

"\nRunge correction: " << Runge\_correction(trapez, trap\_ml, 1, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 1, l) - Newton\_Leibniz(integr\_1, a, b)) << "\n\n";

std::cout << "Simpson: " << Simpso << "\nAbsolute error value: " << abs(Simpso - Newton\_Leibniz(integr\_1, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c4, approx\_func\_max(d4f\_1, a, b, m), 3, a, b, m) <<

"\nRunge correction: " << Runge\_correction(Simpso, Simp\_ml, 3, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 3, l) - Newton\_Leibniz(integr\_1, a, b)) << "\n\n";

break;

case 4:

left\_rectangl = left\_rectangle(f\_2, a, b, m), right\_rectangl = right\_rectangle(f\_2, a, b, m), middle\_rectangl = middle\_rectangle(f\_2, a, b, m),

trapez = trapeze(f\_2, a, b, m), Simpso = Simpson(f\_2, a, b, m);

left\_ml = left\_rectangle(f\_2, a, b, m \* l), right\_ml = right\_rectangle(f\_2, a, b, m \* l), middle\_ml = middle\_rectangle(f\_2, a, b, m \* l),

trap\_ml = trapeze(f\_2, a, b, m \* l), Simp\_ml = Simpson(f\_2, a, b, m \* l);

std::cout << "Left rectangle: " << left\_rectangl << "\nAbsolute error value: " << abs(left\_rectangl - Newton\_Leibniz(integr\_2, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c1, approx\_func\_max(df\_2, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(left\_rectangl, left\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_2, a, b)) << "\n\n";

std::cout << "Right rectangle: " << right\_rectangl << "\nAbsolute error value: " << abs(right\_rectangl - Newton\_Leibniz(integr\_2, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c1, approx\_func\_max(df\_2, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(right\_rectangl, right\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_2, a, b)) << "\n\n";

std::cout << "Middle rectangle: " << middle\_rectangl << "\nAbsolute error value: " << abs(middle\_rectangl - Newton\_Leibniz(integr\_2, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c3, approx\_func\_max(df\_2, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(middle\_rectangl, middle\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_2, a, b)) << "\n\n";

std::cout << "Trapeze: " << trapez << "\nAbsolute error value: " << abs(trapez - Newton\_Leibniz(integr\_2, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c2, approx\_func\_max(d2f\_2, a, b, m), 1, a, b, m) <<

"\nRunge correction: " << Runge\_correction(trapez, trap\_ml, 1, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 1, l) - Newton\_Leibniz(integr\_2, a, b)) << "\n\n";

std::cout << "Simpson: " << Simpso << "\nAbsolute error value: " << abs(Simpso - Newton\_Leibniz(integr\_2, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c4, approx\_func\_max(d4f\_2, a, b, m), 3, a, b, m) <<

"\nRunge correction: " << Runge\_correction(Simpso, Simp\_ml, 3, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 3, l) - Newton\_Leibniz(integr\_2, a, b)) << "\n\n";

break;

case 5:

left\_rectangl = left\_rectangle(f\_3, a, b, m), right\_rectangl = right\_rectangle(f\_3, a, b, m), middle\_rectangl = middle\_rectangle(f\_3, a, b, m),

trapez = trapeze(f\_3, a, b, m), Simpso = Simpson(f\_3, a, b, m);

left\_ml = left\_rectangle(f\_3, a, b, m \* l), right\_ml = right\_rectangle(f\_3, a, b, m \* l), middle\_ml = middle\_rectangle(f\_3, a, b, m \* l),

trap\_ml = trapeze(f\_3, a, b, m \* l), Simp\_ml = Simpson(f\_3, a, b, m \* l);

std::cout << "Left rectangle: " << left\_rectangl << "\nAbsolute error value: " << abs(left\_rectangl - Newton\_Leibniz(integr\_3, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c1, approx\_func\_max(df\_3, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(left\_rectangl, left\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_3, a, b)) << "\n\n";

std::cout << "Right rectangle: " << right\_rectangl << "\nAbsolute error value: " << abs(right\_rectangl - Newton\_Leibniz(integr\_3, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c1, approx\_func\_max(df\_3, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(right\_rectangl, right\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_3, a, b)) << "\n\n";

std::cout << "Middle rectangle: " << middle\_rectangl << "\nAbsolute error value: " << abs(middle\_rectangl - Newton\_Leibniz(integr\_3, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c3, approx\_func\_max(df\_3, a, b, m), 0, a, b, m) <<

"\nRunge correction: " << Runge\_correction(middle\_rectangl, middle\_ml, 0, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 0, l) - Newton\_Leibniz(integr\_3, a, b)) << "\n\n";

std::cout << "Trapeze: " << trapez << "\nAbsolute error value: " << abs(trapez - Newton\_Leibniz(integr\_3, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c2, approx\_func\_max(d2f\_3, a, b, m), 1, a, b, m) <<

"\nRunge correction: "<< Runge\_correction(trapez, trap\_ml, 1, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 1, l) - Newton\_Leibniz(integr\_3, a, b)) << "\n\n";

std::cout << "Simpson: " << Simpso << "\nAbsolute error value: " << abs(Simpso - Newton\_Leibniz(integr\_3, a, b)) <<

"\nTheoretical error value: " << theor\_err\_value(c4, approx\_func\_max(d4f\_3, a, b, m), 3, a, b, m) <<

"\nRunge correction: " << Runge\_correction(Simpso, Simp\_ml, 3, l) <<

"\nRunge correction error value: " << abs(Runge\_correction(Simpso, Simp\_ml, 3, l) - Newton\_Leibniz(integr\_3, a, b)) << "\n\n";

break;

}

}

double Runge\_correction(double integr\_step\_m, double integr\_step\_ml, size\_t d, size\_t l)

{

return((pow(l, d + 1) \* integr\_step\_ml - integr\_step\_m) / (pow(l, d + 1) - 1));

}

#include <iostream>

#include <iomanip>

#include "header.h"

#include "functions.h"

int main() {

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

std::cout << "\*\*\*\*\*\*\*\*\*INCLUDED PART OF TASK 4.3\*\*\*\*\*\*\*\*\*\n";

int flag;

do {

std::cout << "Input integration confines a and b: ";

double a, b;

std::cin >> a >> b;

std::cout << "Input m - number of sections of [a;b]: ";

size\_t m;

std::cin >> m;

double h = (b - a) / m;

std::cout << "Length of step = " << h << "\n";

size\_t l;

std::cout << "Input how many times you would like to increase m: ";

std::cin >> l;

int 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";

print\_all(chosen, a, b, m, l);

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 would like to change integration confines and number of sections, press 1. Otherwise press 0\n";

std::cin >> flag;

} while (flag);

return 0;

}

https://github.com/Chernovuk/Chislaki/tree/master/Fourth\_task/4.2\_4.3