Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Казанский (Приволжский) федеральный университет»
Институт вычислительной математики и информационных технологий
Кафедра прикладной математики и искусственного интеллекта

ОТЧЕТ

по дисциплине «Численные методы»

Тема: «Вычисление интеграла квадратурными формулами»

Вариант 8

Выполнил: студент гр. 09-321,
Лещенко Е.Ф.
Проверил: старший преподаватель
Глазырина О.В.

Казань, 2025

СОДЕРЖАНИЕ

ПОСТАНОВКА ЗАДАЧИ 3
ХОД РАБОТЫ 4
2.1 Метод левых прямоугольников 4
2.2 Метод центральных прямоугольников 6
2.3 Метод трапеций 7
2.4 Метод Симпсона 9
2.5 Метод Гаусса 11
ВЫВОД 13
ЛИСТИНГ ПРОГРАММЫ 14

ПОСТАНОВКА ЗАДАЧИ

В математической физике нередко применяется функция ошибки erf:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \tag{1}$$

В виде цели работы поставлена реализация различных квадратурных форм и сравнение их эффективность.

Сначала функция (1) раскладывается в ряд Тейлора в окрестности $x = 0$. Это позволяет приближённо вычислять значения на отрезке $[a, b]$ с точностью ε, которые станут эталонными для оценки других методов. Далее интеграл вычисляется численно с помощью пяти квадратур:

1. Метод левых прямоугольников 2.1;
2. Метод центральных прямоугольников 2.2;
3. Метод трапеций 2.3;
4. Метод Симпсона 2.4;
5. Метод Гаусса 2.5.

Для каждого метода анализируется, как точность зависит от числа разбиений N. Сравнение проводится по разности между приближённым значением I_N и I_0 из ряда Тейлора — будем считать его эталонным. Результаты сводятся в таблицу, где для каждого $x_i \in [a, b]$ указаны: I_0, I_N, абсолютная погрешность $|I_0 - I_N|$ и минимальное N для заданной точности.

В завершение методы сравниваются по точности для выявления наиболее эффективного метода.

ХОД РАБОТЫ

Разложим функцию (1) в ряд Тейлора в окрестности нуля. Разложение экспоненциальной функции имеет вид:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

Пусть $x = e^{-t^2}$. Тогда:

$$e^{-t^2} = \sum_{n=0}^{\infty} \frac{(-t^2)^n}{n!} = \sum_{n=0}^{\infty} (-1)^n \frac{t^{2n}}{n!}.$$

Проинтегрируем полученный ряд на отрезке от 0 до x:

$$\int_0^x \sum_{n=0}^{\infty} (-1)^n \frac{t^{2n}}{n!} dt = \sum_{n=0}^{\infty} (-1)^n \frac{1}{n!} \int_0^x t^{2n} dt = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{n!(2n+1)}.$$

Таким образом, разложение функции (1) в ряд Тейлора в окрестности нуля имеет вид:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{n!(2n+1)}. \tag{2}$$

Для перехода к численному интегрированию для начала вычислим функцию (1) в одиннадцати узлах на отрезке $[0, 2]$ с точностью $\varepsilon = 10^{-6}$. Будем использовать равноудаленные узлы, поэтому $h = 0.2$.

2.1 Метод левых прямоугольников

Используем составную квадратурную формулу левых прямоугольников:

$$I_N = h \sum_{i=1}^{N-1} f(x_i). \tag{3}$$

Вычислим значения функции (1) с помощью разложения в ряд Тейлора (2) и сопоставим их с полученными результатами. Результаты занесены в таблицу 1. Возьмем в (3) количество разбиений $N = 6$. График изображен на рис. ??.

x	$J_0(x)$	$J_n(x)$	$J_0(x) - J_n(x)$	n
0.00	0.000000	0.000000	0	1
0.20	0.222703	0.222703	5.42728×10^{-7}	8192
0.40	0.428392	0.428393	5.0994×10^{-7}	65536
0.60	0.603856	0.603857	7.74047×10^{-7}	131072
0.80	0.742101	0.742102	8.48733×10^{-7}	262144
1.00	0.842701	0.842701	6.93895×10^{-7}	524288
1.20	0.910314	0.910315	9.12102×10^{-7}	524288
1.40	0.952285	0.952286	5.94847×10^{-7}	1048576
1.60	0.976348	0.976349	7.42921×10^{-7}	1048576
1.80	0.989091	0.989091	8.64023×10^{-7}	1048576
2.00	0.995322	0.995323	4.18471×10^{-7}	2097152

Таблица 1. Результаты вычислений составной квадратурной формулой левых прямоугольников.

Как видно по таблице, в некоторых точках нам понадобилось производить целых 2 097 152 разбиений.

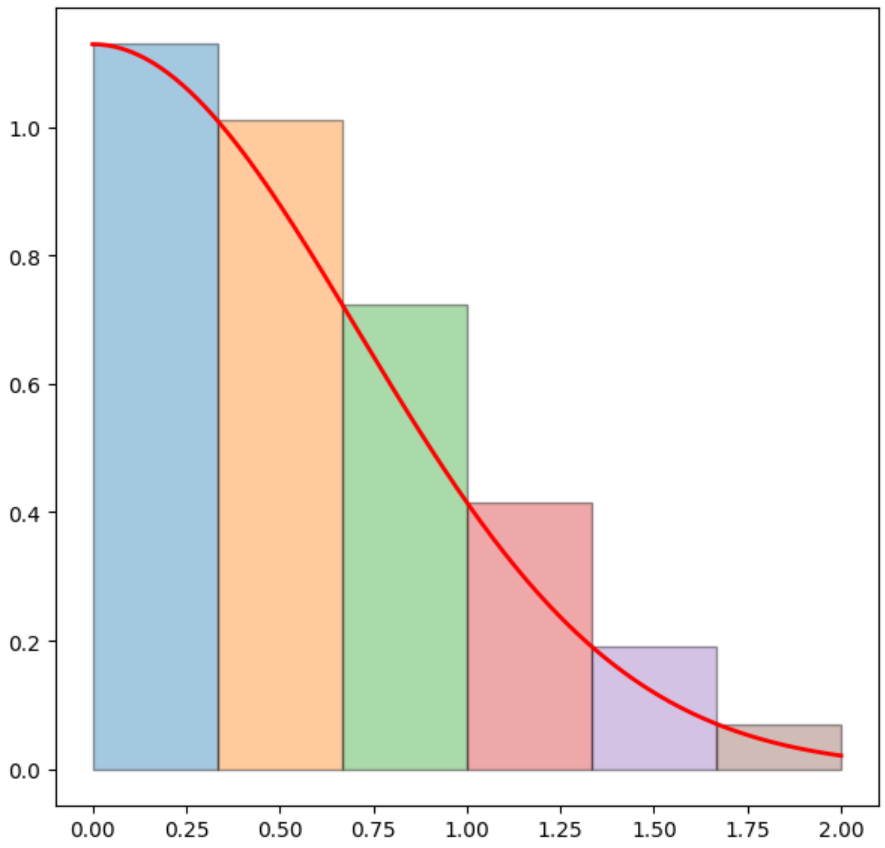

Рис. 1. Составная квадратурная формула левых прямоугольников с 6 разбиениями.

Обратим внимание на рисунок: проблема метода сразу становится очевидной — в связи с монотонностью функции значительная часть прямоугольников выходит за подынтегральную функцию. Это и вызывает потребность в настолько больших количествах разбиений.

2.2 Метод центральных прямоугольников

Определим значения функции (1), применив составную квадратурную формулу метода центральных прямоугольников:

$$I_N = h \sum_{i=1}^{N} f\left(\frac{x_i + x_{i-1}}{2}\right). \tag{4}$$

Сравним полученные данные с результатами, рассчитанными с помощью ряда Тейлора. Итоги представлены в таблице 2. Возьмем в (4) количество разбиений $N = 6$. График этой формулы изображен на рис. 2.

x	$J_0(x)$	$J_n(x)$	$J_0(x) - J_n(x)$	n
0.00	0.000000	0.000000	0	1
0.20	0.222703	0.222703	1.79112×10^{-7}	64
0.40	0.428392	0.428393	3.13795×10^{-7}	128
0.60	0.603856	0.603856	2.09474×10^{-7}	256
0.80	0.742101	0.742101	1.31681×10^{-7}	512
1.00	0.842701	0.842701	1.45625×10^{-7}	512
1.20	0.910314	0.910314	7.35829×10^{-8}	512
1.40	0.952285	0.952285	8.63214×10^{-8}	512
1.60	0.976348	0.976348	6.21656×10^{-8}	512
1.80	0.989091	0.989091	2.6117×10^{-7}	256
2.00	0.995322	0.995322	1.00505×10^{-7}	256

Таблица 2. Результаты вычислений составной квадратурной формулой центральных прямоугольников.

Этот метод обладает той же простотой, что и метод левых прямоугольников 2.1, однако на порядок более высокой точностью. Максимальное количество разбиений в этот раз составляет только 512.

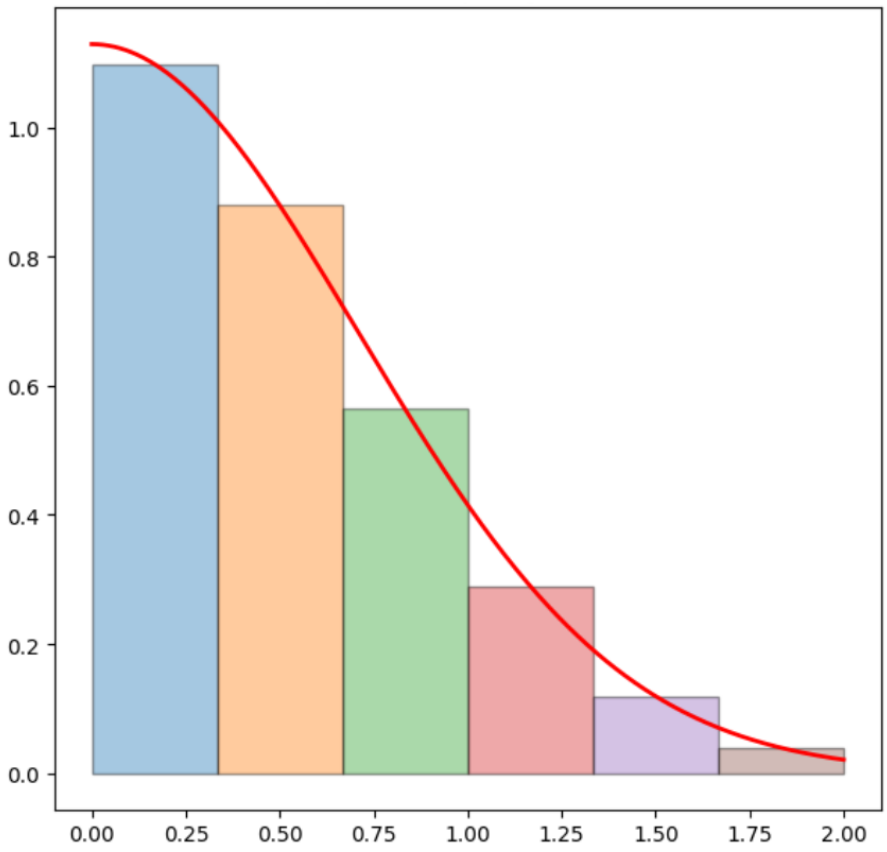

Рис. 2. Составная квадратурная формула центральных прямоугольников с 6 разбиениями.

Анализируя рисунок 2 было сделано предположение, что преимущество над методом 2.1 заключается в первом и втором разбиении. Однако проверка показала: если для точек $x = 1,\ x = n - 1$ использовать метод центральных прямоугольников, то эффективность сильно страдает: приходится совершать до 8 388 608 разбиений.

2.3 Метод трапеций

Рассчитаем (1), применив составную квадратурную формулу метода трапеций:

$$I_N = h \sum_{i=1}^{N-1} \frac{f(x_i) + f(x_{i+1})}{2}. \tag{5}$$

Сопоставим полученные данные с результатами, найденными с использованием ряда Тейлора. Все итоги записаны в таблицу 3. Возьмем в (5) количество разбиений $N = 6$. График этой формулы изображен на рис. 3.

x	$J_0(x)$	$J_n(x)$	$J_0(x) - J_n(x)$	n
0.00	0.000000	0.000000	0	1
0.20	0.222703	0.222703	8.55699×10^{-8}	128
0.40	0.428392	0.428392	1.55708×10^{-7}	256
0.60	0.603856	0.603856	1.1486×10^{-7}	512
0.80	0.742101	0.742101	1.5884×10^{-7}	512
1.00	0.842701	0.842701	2.50252×10^{-7}	512
1.20	0.910314	0.910314	3.66987×10^{-7}	512
1.40	0.952285	0.952285	3.2961×10^{-7}	512
1.60	0.976348	0.976348	2.78573×10^{-7}	512
1.80	0.989091	0.989090	2.304×10^{-7}	512
2.00	0.995322	0.995322	2.14844×10^{-7}	512

Таблица 3. Результаты вычислений составной квадратурной формулой трапеций.

В этот раз максимальное количество необходимых разбиений не изменилось в сравнении с 2.2 и составило 512. Однако пик приходится на край отрезка разбиения $[0, 2]$, в то время как в 2.2 он находился по центру.

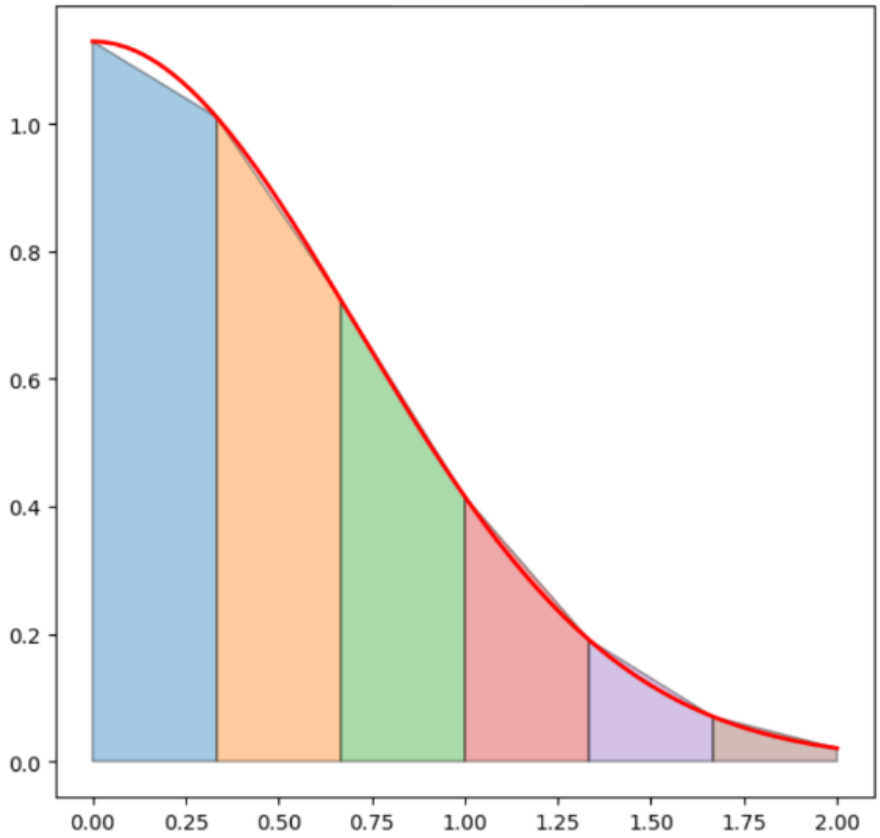

Рис. 3. Составная квадратурная формула трапеций с 6 разбиениями.

2.4 Метод Симпсона

Построим квадратурную формулу Симпсона с тремя узлами

$$x_1 = a, \qquad x_2 = \frac{a+b}{2}, \qquad x_3 = b,$$

для вычисления интеграла

$$I = \int_a^b f(x)dx.$$

Параболу можно представить с помощью интерполяционного многочлена Лагранжа второй степени $L_2(x)$ для функции $f(x)$, который будет проходить через точки $(x_1, f(x_1))$, $(x_2, f(x_2))$, $(x_3, f(x_3))$. Тогда интеграл приближенно равен:

$$I \approx \int_a^b L_2(x)dx.$$

Найденный интеграл можно представить с помощью весовых функций ω_i.

$$\int_a^b L_2(x)dx = \omega_1 f(a) + \omega_2 f\left(\frac{a+b}{2}\right) + \omega_3 f(b).$$

Благодаря симметрии узлов относительно центра отрезка, веса подчиняются равенству $\omega_1 = \omega_3$, и при этом должно соблюдаться следующее:

$$\omega_1 + \omega_2 + \omega_3 = b - a.$$

Вычислим веса ω_i через интегралы от базисных функций:

$$\omega_2 = (b-a)\int_0^1 \hat{l}_1(t)dt = (b-a)\int_0^1 4t(1-t)dt = \frac{2}{3}(b-a).$$

Тогда получаем, что

$$2\omega_1 + \frac{2}{3}(b-a) = b - a \quad \Rightarrow \qquad \omega_1 = \omega_3 = \frac{b-a}{6}.$$

Подставляя найденные веса, получаем формулу Симпсона:

$$I \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right). \tag{6}$$

Вычислим, используя составную квадратурную формулу Симпсона:

$$I_N = \frac{h}{6}\sum_{i=1}^{N-1} f(x_i) + 4f\left(\frac{x_{i+1}+x_i}{2}\right) + f(x_{i+1}).$$

Полученные данные сравним со значениями, вычисленными рядом Тейлора. Результаты занесены в таблицу 4. График формулы (6) изображен на рис. 4.

x	$J_0(x)$	$J_n(x)$	$J_0(x) - J_n(x)$	n
0.00	0.000000	0.000000	0	1
0.20	0.222703	0.222703	8.15721×10^{-9}	4
0.40	0.428392	0.428392	9.74687×10^{-9}	8
0.60	0.603856	0.603856	4.06359×10^{-8}	8
0.80	0.742101	0.742101	4.19496×10^{-8}	16
1.00	0.842701	0.842701	2.24613×10^{-8}	16
1.20	0.910314	0.910314	4.78853×10^{-8}	8
1.40	0.952285	0.952285	6.90624×10^{-8}	16
1.60	0.976348	0.976348	9.26237×10^{-8}	16
1.80	0.989091	0.989090	1.28231×10^{-7}	16
2.00	0.995322	0.995322	1.14107×10^{-7}	32

Таблица 4. Результаты вычислений составной квадратурной формулой Симпсона.

Полученный результат можно назвать серьезным — потребовалось сделать не более 32 разбиений для удовлетворения необходимой точности. Логика алгоритма стала сложнее, что компенсируется на порядок более высокой скоростью.

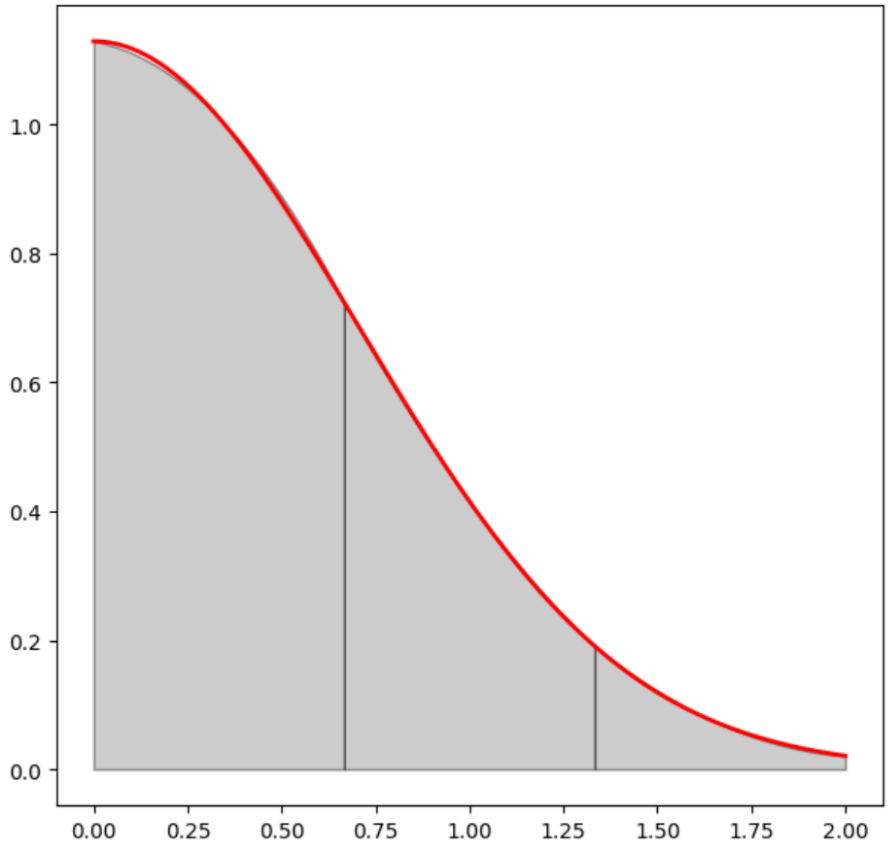

Рис. 4. Квадратурная формула Симпсона с 3 разбиениями.

На рисунке 4 видно, что сплайн из 3 парабол уже практически полностью покрывает площадь под графиком.

2.5 Метод Гаусса

Вычислим (1), используя составную квадратурную формулу Гаусса:

$$I_N = \frac{h}{2} \sum_{i=1}^{N-1} f\left(x_i + \frac{h}{2}\left(1 - \frac{1}{\sqrt{3}}\right)\right) + f\left(x_i + \frac{h}{2}\left(1 + \frac{1}{\sqrt{3}}\right)\right). \qquad (7)$$

Сравним полученные результаты с данными, рассчитанными с использованием ряда Тейлора. Все итоги отражены в таблице 5.

x	$J_0(x)$	$J_n(x)$	$J_0(x) - J_n(x)$	n
0.00	0.000000	0.000000	0	1
0.20	0.222703	0.222703	5.61481×10^{-8}	2
0.40	0.428392	0.428392	5.17605×10^{-9}	8
0.60	0.603856	0.603856	3.83412×10^{-8}	8
0.80	0.742101	0.742101	3.01009×10^{-8}	16
1.00	0.842701	0.842701	7.80269×10^{-9}	16
1.20	0.910314	0.910314	9.01245×10^{-8}	8
1.40	0.952285	0.952285	4.11593×10^{-8}	16
1.60	0.976348	0.976348	2.39354×10^{-8}	16
1.80	0.989091	0.989091	2.54125×10^{-8}	16
2.00	0.995322	0.995322	6.29995×10^{-8}	16

Таблица 5. Результаты вычислений составной квадратурной формулой Гаусса.

Самый эффективный метод. При фактически эквивалентной сложности алгоритма методу 2.4 необходимое количество разбиений уменьшилось вдвое и не превысило 16. Однако концепция метода намного сложнее чем рассмотренные выше — это не попытка замощения площади под графиком другими функциями, поэтому попытка изобразить процесс графически лишена своего смысла.

ВЫВОД

В процессе исследования были реализованы и протестированы 5 квадратурных методов. Был проведен анализ и избран самый эффективный из них.

Метод левых прямоугольников оказался самым бесполезным: с аналогичной сложностью лучше избирать метод центральных прямоугольников, который лучше во всем. Тем не менее он все так же мало эффективен. Метод трапеций не показал лучших результатов, чем метод центральных прямоугольников.

Пускай число разбиений $N = 512$ и является предпочтительнее $2\,097\,152$, они все еще проигрывают двум самым эффективным методам — Гаусса и Симпсона. Первый оказался в 2 раза эффективней второго. Ошибка метода Симсона оказалась несколько меньше. Однако это преимущество сомнительно, поскольку метод Гаусса является более быстрым, из-за чего при желании можно добиться даже большой точности, если увеличить число разбиений.

```
#pragma once
#include <complex>
#include <functional>

namespace integral_methods {
constexpr static double GAUSSIAN_FACTOR_L = 1 - 1/std::sqrt(3);
constexpr static double GAUSSIAN_FACTOR_G = 1 + 1/std::sqrt(3);

inline double calculate_left_integral(
    std::vector<double> dots, const std::function<double(double)> &f)
{
    double result = 0;
    for (int i = 0; i < dots.size() - 1; i++) {
        result += (dots[i + 1] - dots[i]) * f(dots[i]);
    }

    return result;
}

inline double calculate_right_integral(
    std::vector<double> dots, const std::function<double(double)> &f)
{
    double result = 0;
    for (int i = 1; i < dots.size(); i++) {
        result += (dots[i + 1] - dots[i]) * f(dots[i + 1]);
    }

    return result;
}

inline double calculate_central_integral(
    std::vector<double> dots, const std::function<double(double)> &f)
{
    double result = 0;
    for (int i = 0; i < dots.size() - 1; i++) {
        auto left = dots[i];
        auto right = dots[i + 1];
        auto central = (left + right) / 2;

        result += (right - left) * f(central);
```

```
    }

    return result;
}

inline double calculate_trapezoid_integral(
    std::vector<double> dots, const std::function<double(double)> &f)
{
    double result = 0;
    for (int i = 0; i < dots.size() - 1; i++) {
        auto left = dots[i];
        auto right = dots[i + 1];

        result += (right - left) * (f(left) + f(right)) / 2;
    }

    return result;
}

inline double calculate_simpson_integral(
    std::vector<double> dots, const std::function<double(double)> &f)
{
    double result = 0;
    for (int i = 0; i < dots.size() - 1; i++) {
        auto left = dots[i];
        auto right = dots[i + 1];
        auto f_factor = f(left) + 4 * f((left + right) / 2) + f(right);

        result += (right - left) / 6 * f_factor;
    }

    return result;
}

inline double calculate_gaussian_integral(
    std::vector<double> dots, const std::function<double(double)> &f)
{
    double result = 0;

    for (int i = 0; i < dots.size() - 1; i++) {
        auto left = dots[i];
```

```
        auto right = dots[i + 1];
        auto h_half = (right - left) / 2;
        auto f_factor = f(left + h_half * GAUSSIAN_FACTOR_L)
                      + f(left + h_half * GAUSSIAN_FACTOR_G);

        result += h_half * f_factor;
    }

    return result;
}
} // namespace integral_methods
```