Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и информатики

Лабораторная работа №10 по дисциплине

Численные методы

«Решение интегралов с помощью квадратурных формул Ньютона-Котеса»

Выполнил

студент гр.5030102/20001 Дрекалов Н.С.

Преподаватель Козлов К.Н.

Санкт-Петербург

2023

Оглавление

[**Формулировка задачи** 3](#_Toc163335585)

[**Формализация** 3](#_Toc163335586)

[**Вычисление интеграла формулой 3/8** 3](#_Toc163335587)

[**Предварительный анализ задачи** 4](#_Toc163335588)

[**Тестовый пример к методам** 5](#_Toc163335589)

[**Контрольные тесты** 6](#_Toc163335590)

[**Модульная структура программы** 7](#_Toc163335591)

[C++ 7](#_Toc163335592)

[**Численный анализ методов** 9](#_Toc163335593)

[**Вывод** 12](#_Toc163335594)

## **Формулировка задачи**

Необходимо вычислить интеграл от заданной функции на заданном отрезке с нужной точность

## **Формализация**

* Задана подынтегральная функция промежуток и точность
* Необходимо вычислить определённый интеграл с помощью квадратурной формулы Ньютона­Котеса– формулы Симпсона 3/8 таким образом, чтобы точность достигала .

## **Вычисление интеграла формулой 3/8**

*Условия применимости:*

1. Функция должна в худшем случае иметь лишь конечное число разрывов.

*Алгоритм метода:*

Необходимо вычислить такое, что (

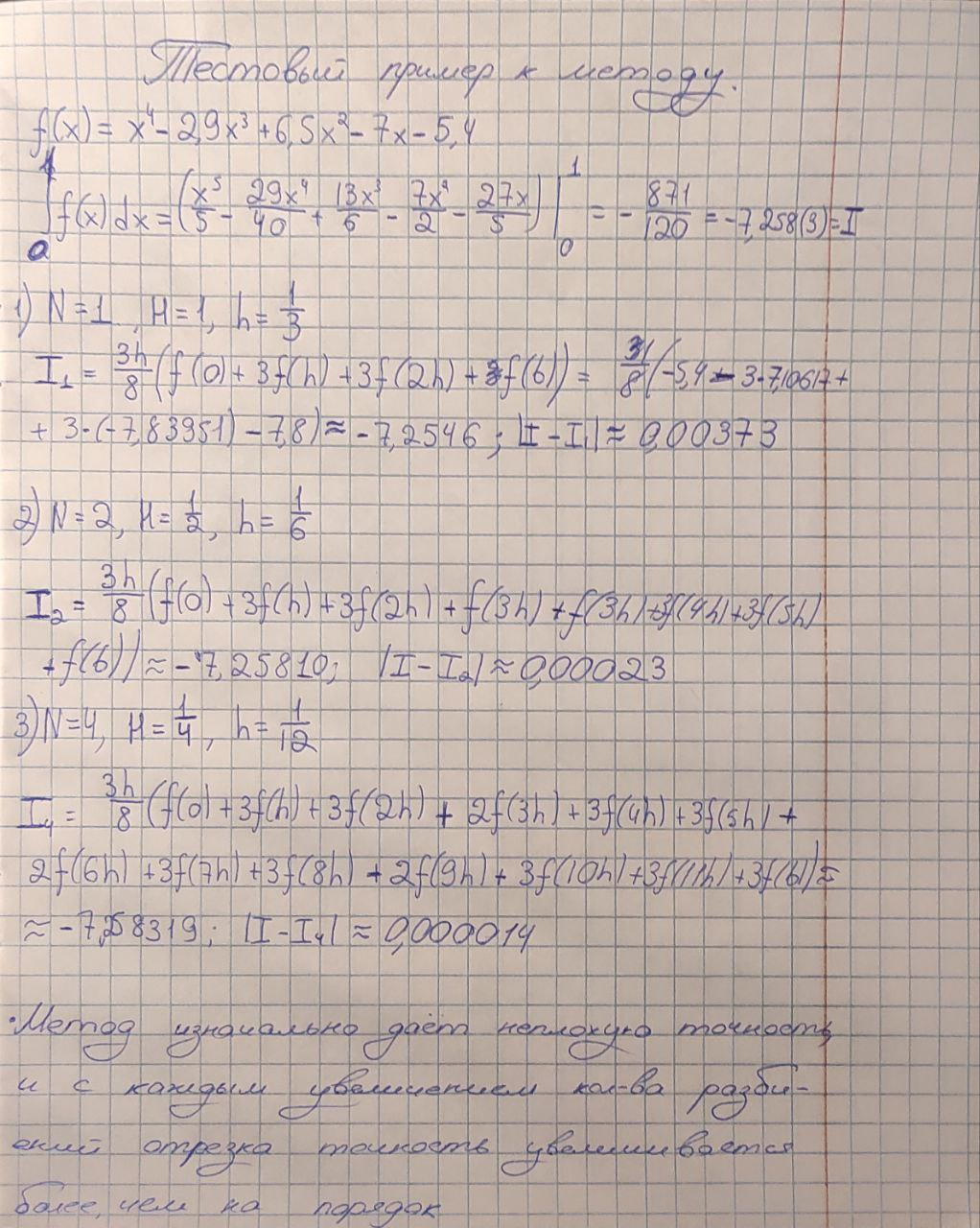
1. Необходимо выбрать число отрезков разбиений. Начнём с N = 1 и посчитаем для него , где
2. Приравниваем
3. Увеличиваем число отрезков разбиение в 2 раза. Тогда .
4. Для каждого из отрезков разбиений вычисляем интеграл по формуле
5. Приравниваем
6. Проверяем выполнение правила Рунге: . Если выполняется, то - значение интеграла с точностью , иначе повторяем действия, начиная с п.2

## **Предварительный анализ задачи**

*Анализ функций на непрерывность:*

* непрерывна на всем множестве вещественных чисел она интегрируема.

## **Тестовый пример к методам**



## **Контрольные тесты**

*Построим графики зависимостей*

* Фактической ошибки от заданной точности с биссектрисой.
* Числа итераций от заданной точности.
* Фактической ошибки от длины отрезка разбиения.

## **Модульная структура программы**

Выполнение нужных действий достигается с помощью 2-х программ: на языках python и C++. При этом C++ программа вычисляет значение интеграла на заданном промежутке с заданной точностью, в то время как python программа занимается анализом результата (расчёт различных ошибок, построение графиков и т. п.). Далее будут перечислены объявления функций.

### C++

integral.hpp

using Function = double (&)(double);  
  
/// \class Интеграл от заданной функции  
class IntegralCalculator {  
 // Ссылка на функцию  
 Function f;  
 // Порядок метода (=4)  
 static int const methodOrder;  
 // Знаменатель в правиле Рунге (=15)  
 static double const rungeRuleDenominator;  
   
 /// \brief Функция проверяет правило Рунге для двух значений интеграла и нужной точности  
 /// \param I\_2n значение интеграла при 2N разбиений  
 /// \param I\_n значение интеграла при N разбиений  
 /// \param epsilon нужная точность  
 /// \return Выполняется ли правило Рунге (true/false)  
 static bool checkRungeRule(double I\_2n, double I\_n, double epsilon);  
public:  
 /// \class Результат вычислений интеграла  
 class Result {  
 public:  
 // Значение интеграла   
 double value;  
 // Число итераций, которое понадобилось для вычисления значения  
 long iterCnt;  
 // Длина отрезка разбиения  
 double h;  
 };  
   
 /// \brief Конструктор класса   
 /// \param f подинтегральная функция  
 explicit IntegralCalculator(Function f);  
   
 /// \brief функция считает значение интеграла на заданном отрезке с заданной точностью  
 /// \param a левая граница отрезка  
 /// \param b правая граница отрезка  
 /// \param epsilon требуемая точность  
 /// \return Значение интеграла  
 Result calculate(double a, double b, double epsilon);  
};

**main.cpp**

/// \brief Подынтегральная функция  
double f(double x) {  
 return x\*x\*x\*x - 2.9\*x\*x\*x + 6.5\*x\*x - 7\*x - 5.4;  
}  
  
  
/// \class gRPC server класс для обмена данными  
class EvaluateIntegralService : public nm::EvaluateIntegral::Service {  
 /// \brief функция принимает поток данных (a, b, epsilon) и возвращает поток результатов (value, iter\_cnt, h)  
 /// \param context контекст обмена  
 /// \param stream[in][out] переменная для обмена данными  
 /// \return статус обработки (OK)  
 grpc::Status Evaluate(  
 grpc::ServerContext \*context,  
 grpc::ServerReaderWriter<nm::Integral, nm::Requirements> \*stream) override

## **Численный анализ методов**

Интеграл вычислялся на отрезке для диапазона точностей

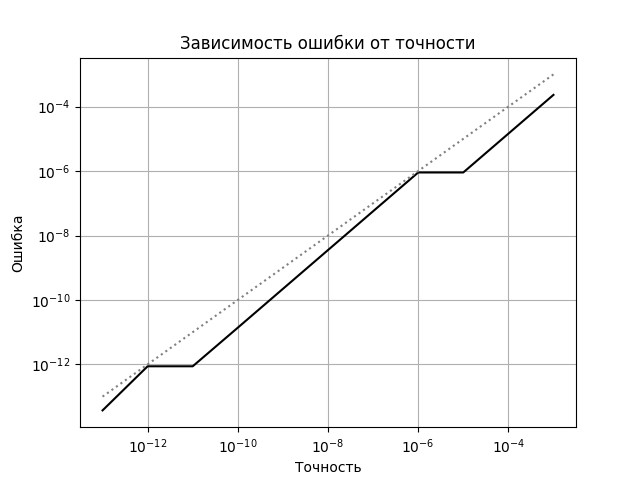


Рисунок 1. График зависимости фактической ошибки от заданной точности.

Из графика видно, что каждая заданная точность достигалась (т. к. весь график лежит ниже биссектрисы. При этом на графике нет ситуаций, когда происходило бы избыточное уточнение.

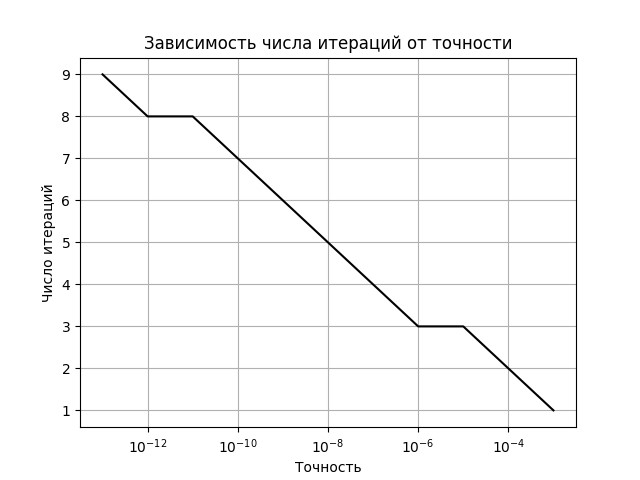


Рисунок 2. График зависимости числа итераций от заданной точности

Из графика видно, что число итераций (почти) линейно зависит от показателя степени точности. Причём нелинейность возникает лишь в тех ситуациях, когда метод за одну итерацию получил уменьшение точности на 2 порядка.

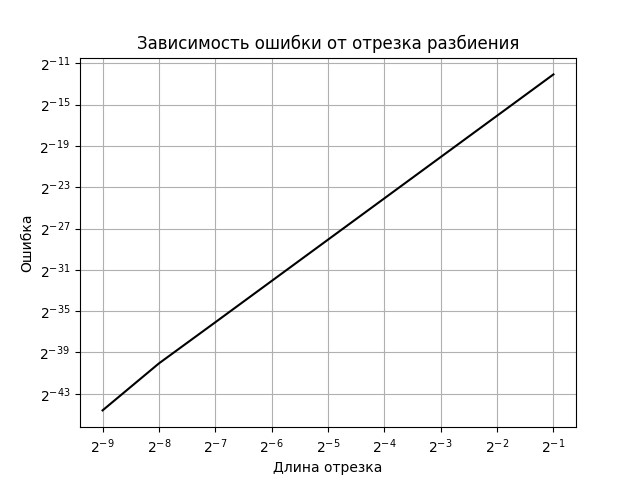


Рисунок 3. График зависимости фактической ошибки от длины отрезка разбиения

Из графика видно, что при уменьшении длины отрезка в раза ошибка уменьшается примерно в раза. То есть порядок точности равен .

Коэффициент, полученный с помощью вычисления методом наименьших квадратов для степеней двоек, равен

## **Вывод**

В лабораторной работе мне удалось вычислить интеграл с помощью формулы 3/8 с требуемой точности.

В ходе исследования была проанализирована зависимости ошибки и числа итераций от заданной точности, которые подтвердили правильность выполнения вычислений.

Также по Рис. 3 и с помощью МНК было установлено, что метод Симпсона 3/8 имеет порядок 4. Точно найденная константа имеет значение