Санкт-Петербургский политехнический университет
Высшая школа прикладной математики
и вычислительной физики, ФизМех

Направление подготовки
«Прикладная математика и информатика»

Отчет по лабораторной работе №3
«Численное интегрирование с помощью формул Ньютона-Котеса»

Дисциплина: «Численные методы»

Выполнил студент гр. 5030102/00003 Анищенко М.Д.

Преподаватель: Курц В.В.

Санкт-Петербург 2022

Оглавление

Глава 1

Формулировка задачи и её формализация

1.1 Формулировка задачи

Представим определённый интеграл на промежутке $[a, b]$ функции $F(x)$ в виде

$$\int_a^b F(x)dx = \int_a^b p(x)f(x)dx$$

где $p(x)$ — весовая функция. Необходимо вычислить приближенное значение интеграла, используя квадратурные формулы:

$$\int_a^b p(x)f(x)dx \approx \sum_{k=1}^{n} A_k f(x_k)$$

где A_k — коэффициенты, а x_k — узлы квадратурной функции.

1.2 Формализация

В данном варианте необходимо решить задачу, используя методы левых и правых прямоугольников, для следующей функции на отрезке $[0, 3]$:

$$F(x) = x^2 \cos(2x) + 1$$

Формула прямоугольников — это одна из формул Ньютона-Котеса, поэтому

$$p(x) = 1, h = \frac{b-a}{n}, x_i = a + ih$$

Глава 2

Алгоритм метода и условия его применимости

2.1 Алгоритм метода

Приближенное значение интеграла функции по формуле прямоугольников находится, как сумма значений интегралов на каждом отрезке разбиения по формулам:
Для левых прямоугольников:

$$\int\limits_a^b F(x)dx = \int\limits_a^b f(x)dx \approx h\sum_{i=0}^{n-1} f(x_i)$$

Для правых прямоугольников:

$$\int\limits_a^b F(x)dx = \int\limits_a^b f(x)dx \approx h\sum_{i=1}^{n} f(x_i)$$

Для достижения заданной точности будем использовать правило Рунге:

$$\Delta_{2n} \approx 3|I_{2n} - I_n|$$

2.2 Условия применимости

Функция $F(x)$ должна быть непрерывна на отрезке $[a, b]$.

Глава 3

Анализ задачи

Для корректной работы метода необходимо задать узлы, согласно формуле $x_i = a + ih$. Также выбранная функция Должна удовлетворять условиям применимости. Произведение степенной и тригонометрической функций непрерывно на отрезке $[a, b]$.

Глава 4

Тестовый пример

Тестовый пример выполним для нашей функции $F(x) = x^2 \cos(2x) + 1$ на отрезке $[0, 3]$. Точное значение интеграла найдем классическим способом. Опуская подробные шаги, получаем

$$\int\limits_0^3 x^2 \cos(2x) + 1 dx = \frac{17\sin(6) + 6\cos(6) + 12}{4} \approx 3.252739$$

Теперь воспользуемся двумя вариациями метода

4.1 Левые прямоугольники

1. $$\int\limits_0^3 x^2 \cos(2x) + 1 dx = h \cdot f(x_0) = (b - a) \cdot f(a) = 3 \cdot 1 = 3$$

2. $$\int\limits_0^3 x^2 \cos(2x) + 1 dx = h \cdot (f(x_0) + f(x_1)) = \frac{b - a}{n} \cdot (f(a) + f(a + h)) =$$

$$= \frac{3-0}{2} \cdot (f(0) + f(1.5)) = 1.5 \cdot (1 + (-1.227)) = -0.3405$$

3.
$$\int_0^3 x^2 \cos(2x) + 1dx = h \cdot (f(x_0) + f(x_1) + f(x_2) + f(x_3)) =$$
$$= \frac{b-a}{n} \cdot (f(a) + f(a+h) + f(a+2h) + f(a+3h)) =$$
$$= \frac{3}{4}(f(0) + f(0.75) + f(1.5) + f(2.25)) =$$
$$= 0.75 \cdot (1 + 1.04 - 1.227 - 0.0672) = 0.55935$$

4.2 Правые прямоугольники

1.
$$\int_0^3 x^2 \cos(2x) + 1dx = h \cdot f(x_1) = (b-a) \cdot f(b) = 3 \cdot 9.642 = 28.926$$

2.
$$\int_0^3 x^2 \cos(2x) + 1dx = h \cdot (f(x_1) + f(x_2)) = \frac{b-a}{n} \cdot (f(a+h) + f(b)) =$$
$$= \frac{3-0}{2} \cdot (f(1.5) + f(3)) = 1.5 \cdot (-1.227 + 9.642) = 12.6225$$

3.
$$\int_0^3 x^2 \cos(2x) + 1dx = h \cdot (f(x_0) + f(x_1) + f(x_2) + f(x_3)) =$$
$$= \frac{b-a}{n} \cdot (f(a+h) + f(a+2h) + f(a+3h) + f(b)) =$$
$$= \frac{3}{4}(f(0.75) + f(1.5) + f(2.25) + f(3)) =$$
$$= 0.75 \cdot (1.04 - 1.227 - 0.0672 + 9.642) = 7.041$$

4.3 Вывод и сравнение

Как мы видим, погрешность интегрирования очень большая, при этом значения интегралов, найденные с помощью левых и правых прямоугольников, сильно разнятся друг с другом. Отметим, что при нахождении интеграла с помощью программы точность 0.1 была достигнута при $n = 64$.

Глава 5

Контрольные тесты

В ходе данной работы проведем следующие исследования:

1. Изучим зависимость погрешности интегрирования от заданной точности. Будем менять точность от 10^{-7} до 10^{-6}.
2. Изучим зависимость количества разбиений (подотрезков) от заданной точности.
3. Оптимизируем алгоритм так, чтобы было меньше вычислений функции $f(x)$, т.е. значения в повторяющихся точках хранились в памяти, а не вычислялись заново. Затем изучим зависимость количества вызовов функции $f(x)$ от заданной точности.

Глава 6

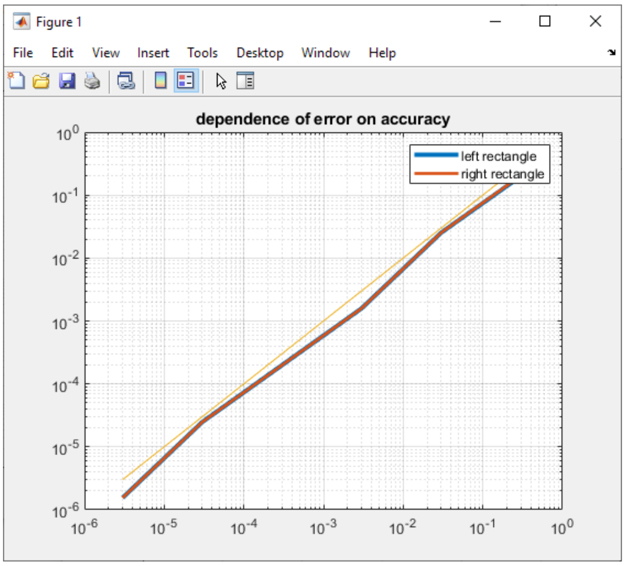
Модульная структура программы

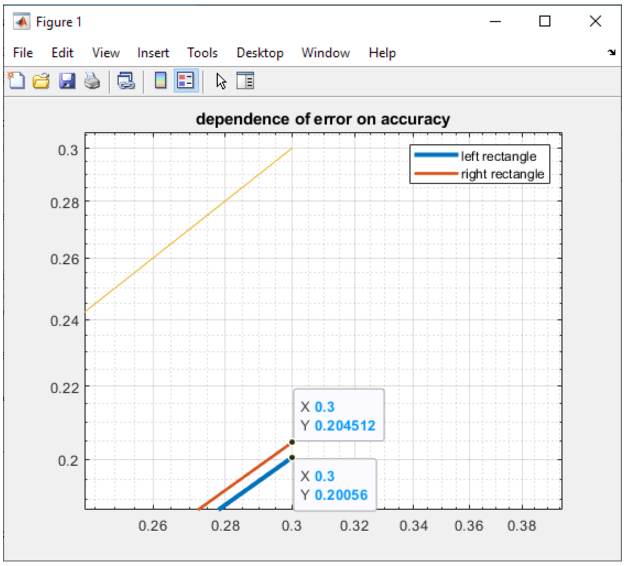
- double f(double x) — вычисление значения функции в точке x.
- double Formula(double a, double b, unsigned long long int n, int side) — функция, выполняющая вычисления на текущем приближении.
- double FormulaOpt(double a, double b, unsigned long long int n, int side, double I) — аналогияная функция для оптимизированного варианта метода.
- void Method(double a, double b, int side, FILE* file) — функция, проверяющая условия окончания расчетов с помощью правила Рунге, а также записывающая результаты в файл.
- void MethodOpt(double a, double b, int side, FILE* file) — аналогияная функция для оптимизированного варианта метода.

Глава 7

Численный анализ

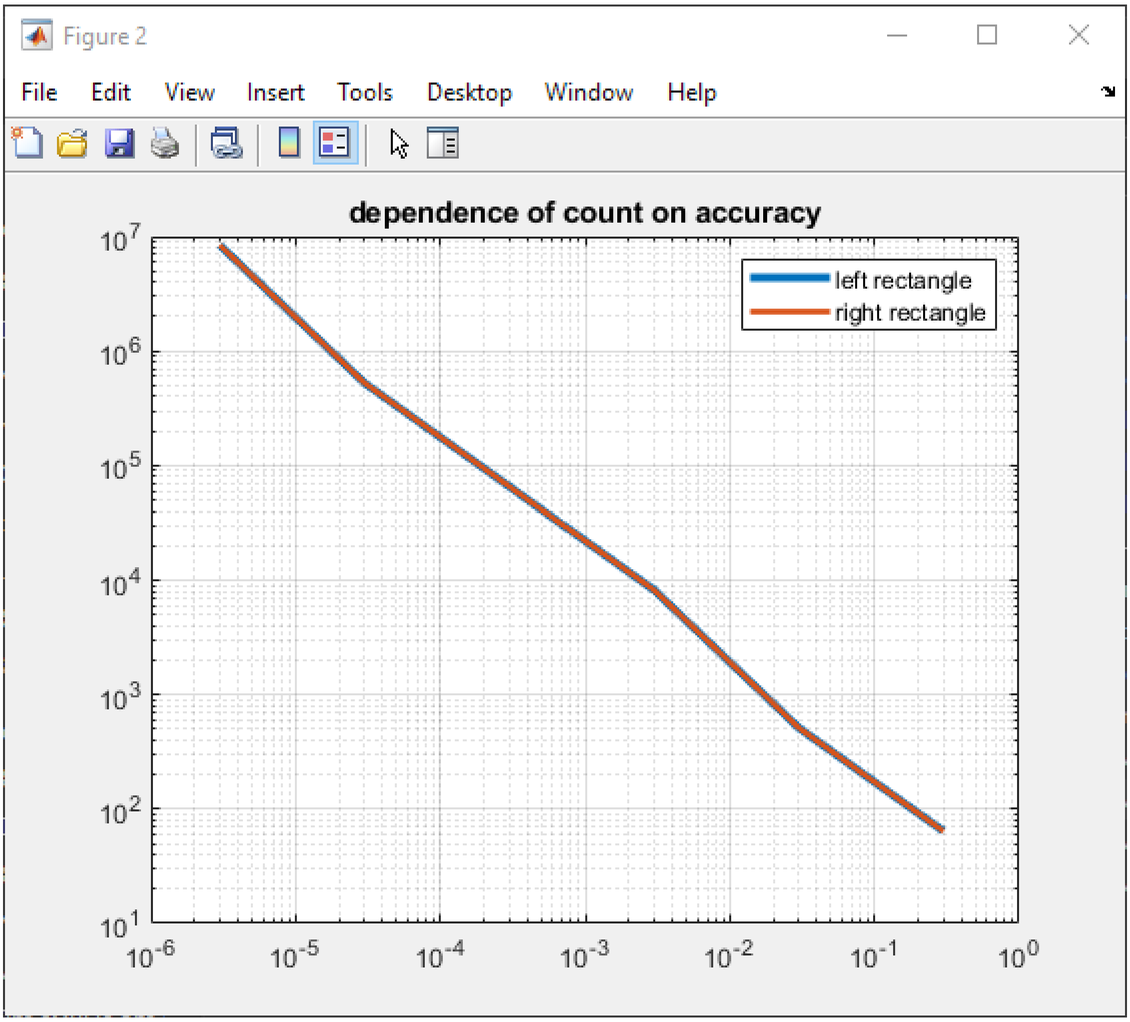
7.1 Зависимость погрешности от точности





Мы можем видеть, что для левых и правых прямоугольников для данной функции на выбранном отрезке значения инттегралов довольно близки. Так получается из-за того, что даже для точности 0.1 требуется 64 разбиения, что на таком коротком отрезке влечет за собой малую разницу между двумя вариациями данного метода.

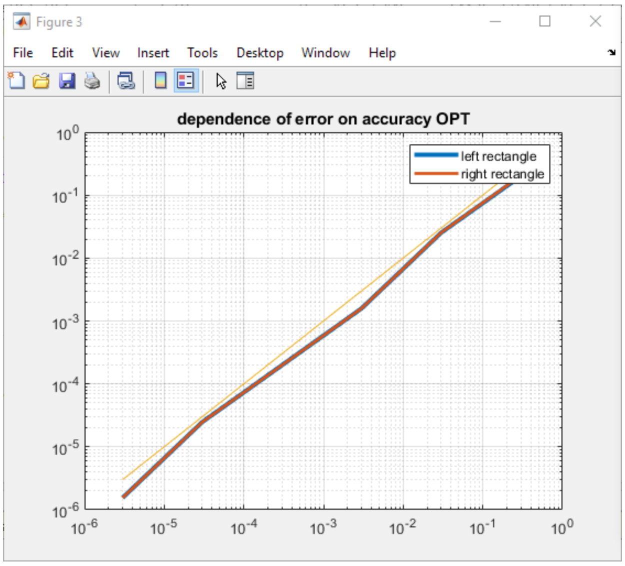
7.2 Зависимость количества отрезков от точности

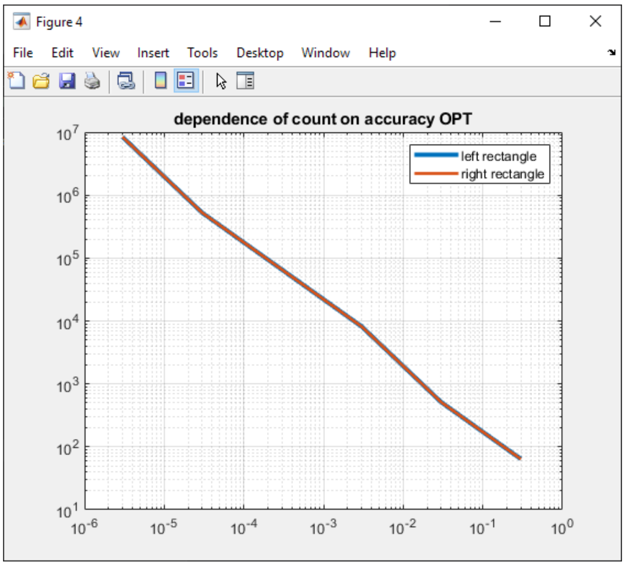


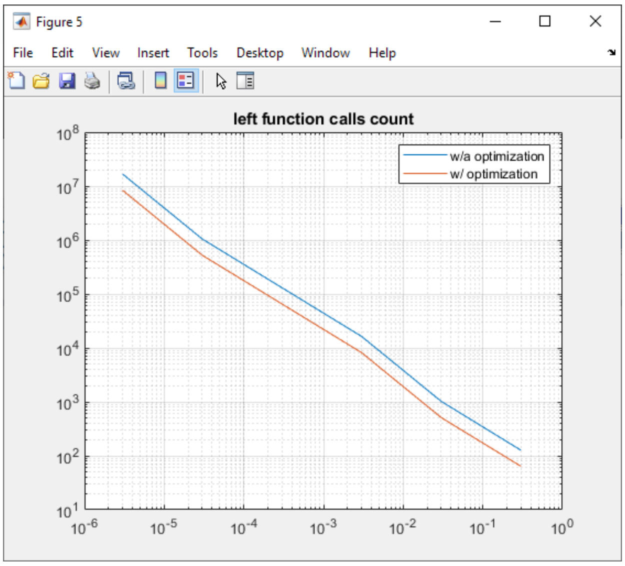
Заметим, что количество отрезков разбиения совпадает для методов левых и правых прямоугольников.

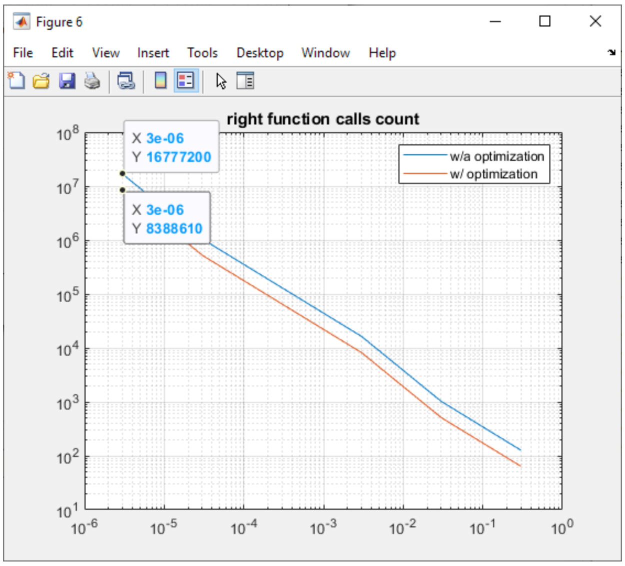
7.3 Оптимизация метода

Построим графики зависимости погрешности интегрирования от точности, количества отрезков разбиения и количества вызовов функции $f(x)$ от точности для оптимизированного метода.









Мы видим, что погрешность и количество отрезков остались теми же, как и предполагалось (поскольку мы оптимизируем только объем вычичслений, а не их качество). При этом количество вызовов функций уменьшилось вдвое. Так как по правилу Рунге мы увеличичваем количество отрезков вдвое на каждом следующем приближении, то их количество равно 2^k, где k — номер шага. Количество вызовов функций прямо пропорционально количеству разбиений. Поэтому при оптимизации мы запоминаем результат прошлого приближения, а вычисляем значения функции только в новых точках, которые лежат между уже имеющимися. Их ровно столько же, поэтому вместо $2n$ вычислений на каждом следующем приближении мы выполняем лишь n. Отсюда уменьшение количества вызовов функции $f(x)$ в два раза.

Глава 8

Общие выводы

По результатам проведенных исследований, можем отметить, что метод прямоугольников (как правых, так и левых) сходится к значению интеграла с требуемой точностью, причем линейно. Оптимизация алгоритма помогает сократить вдвое объем вычислений. При этом количество отрезков разбиения все равно растет достаточно быстро, из-за чего данный метод становится менее предпочтителен по сравнению с другими (трапеций или 3/8), хоть и является самым простым. На практике же он применяется реже всего, так как даже для низких точностей требует большого количества вычислений.