МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Вычислительная Математика»

Тема: Составные формулы прямоугольников, трапеций, Симпсона

Студент гр. 8381 _____ СергеевА.Д.

Преподаватель _____ Щеголева Н.Л.

Санкт-Петербург

2019

Цель работы.

Изучение и сравнение различных методов численного интегрирования на примере составных формул прямоугольников, трапеций, Симпсона.

Краткое изложение основных теоретических понятий.

Пусть требуется найти определенный интеграл $I = \int_a^b f(x)dx,$ где функция $f(x)$ непрерывна на отрезке $[a,b]$.

Для приближенного вычисления интегралов чаще всего подынтегральную функцию заменяют «близкой» ей вспомогательной функцией, интеграл от которой вычисляется аналитически. За приближенное значение интеграла принимают значение интеграла от вспомогательной функции.

Заменим функцию на отрезке $[a,b]$ её значением в середине отрезка. Искомый интеграл, равный площади криволинейной фигуры, заменяется на площадь прямоугольника. Из геометрических соображений нетрудно записать *формулу прямоугольников*:

$$\int_a^b f(x)dx \approx f\left(\frac{a+b}{2}\right)(b-a).$$

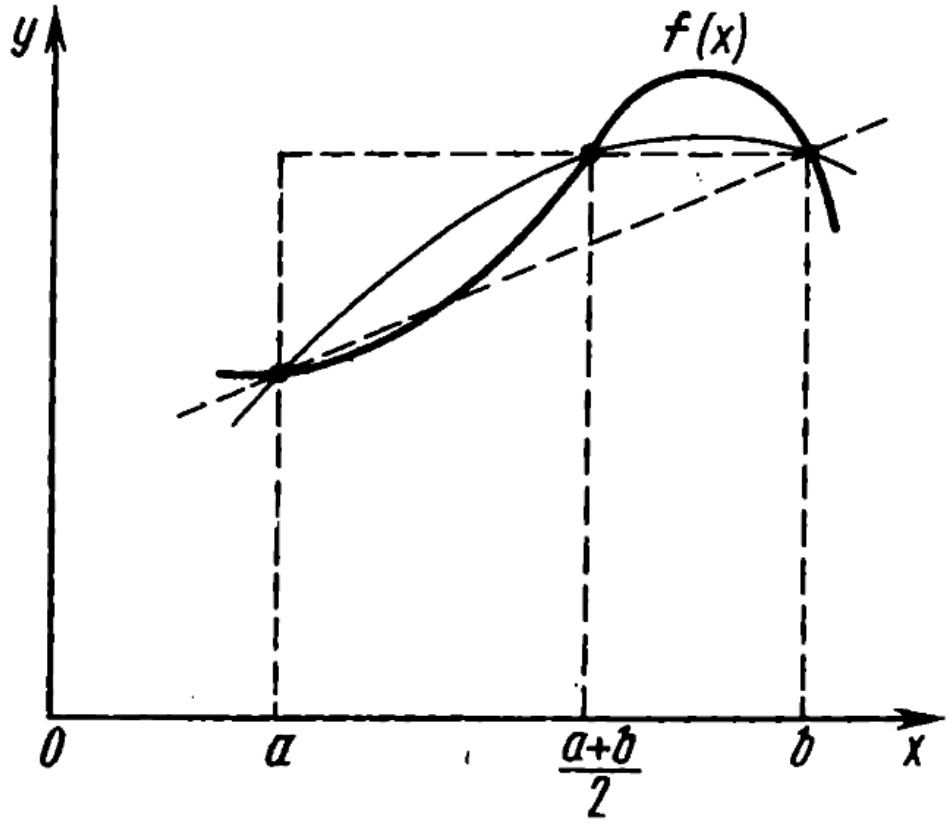
Приблизив $f(x)$ линейной функцией и вычислив площадь соответствующей трапеции, получим *формулу трапеций*:

$$\int_a^b f(x)dx \approx \frac{1}{2}(f(a)+f(b))(b-a).$$

Если же приблизить подынтегральную функцию параболой, проходящей через точки $(a,f(a)),\left(\frac{a+b}{2},f\left(\frac{a+b}{2}\right)\right),(b,f(b)),$ то получим *формулу Симпсона (парабол)*:

$$\int_a^b f(x)dx \approx \frac{1}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)(b-a).$$

Все три формулы хорошо иллюстрируются геометрически и представлены на рисунке.



Для повышения точности интегрирования применяют составные формулы. Для этого разбивают отрезок $[a,b]$ на четное $n=2m$ число отрезков длины $h=\frac{b-a}{n}$ и на каждом из отрезков длины $2h$ применяют соответствующую формулу. Таким образом получают *составные формулы прямоугольников, трапеций и Симпсона*.

На сетке $x_i=a+ih, y_i=f(x_i), i=\overline{0,2m}$ составные формулы имеют следующий вид:

• формула прямоугольников:

$$\int_a^b f(x)dx = h\sum_{i=0}^{n-1} f\left(x_i+\frac{h}{2}\right)+R_1;$$

$$R_1 \approx \frac{h^2}{24}\int_a^b f''(x) = o(h^2);$$

• формула трапеций:

$$\int_a^b f(x)dx = \frac{h}{2}\sum_{i=0}^{n-1}\left(f(x_i)+f(x_{i+1})\right)+R_2;$$

$$R_2 \approx -\frac{h^2}{12}\int_a^b f''(x) = o\left(h^2\right);$$

• формула Симпсона:

$$\int_a^b f(x)dx = h\sum_{i=0}^{n-1}\left(f(x_{2i})+4f(x_{2i+1})+f(x_{2i+2})\right)+R_1;$$

$$R_3 \approx -\frac{h^4}{180}\int_a^b f^{IV}(x)d = o\left(h^4\right);$$

где $R_1, R_2, R_3 -$ остаточные члены. Оценки остаточных членов получены в предположении, что соответствующие производные $f(x)$ непрерывны на $[a,b]$. При $n \to \infty$ приближенные значения интегралов для всех трех формул (в предположении отсутствия погрешностей округления) стремятся к точному значению интеграла.

Любая попытка сравнить достоинства приведенных формул связана с вопросом типа «что больше $h^2 f''(x)$ или $h^4 f''''(x)$?» Ответ зависит от свойств интегрируемой функции. Можно лишь утверждать, что остаточный член формулы прямоугольников примерно вдвое меньше, чем формулы трапеций, и оба они имеют порядок h^2. А остаточный член формулы Симпсона убывает быстрее – со скоростью h^4.

Для практической оценки погрешности квадратурной можно использовать правило Рунге. Для этого проводят вычисления на сетках с шагом h и $\frac{h}{2}$, получают приближенные значения интеграла I_h $I_{\frac{h}{2}}$ и за окончательные значения интеграла принимают величины:

• $I_{h/2} + \frac{I_{h/2} - I_h}{3}$ – для формулы прямоугольников;

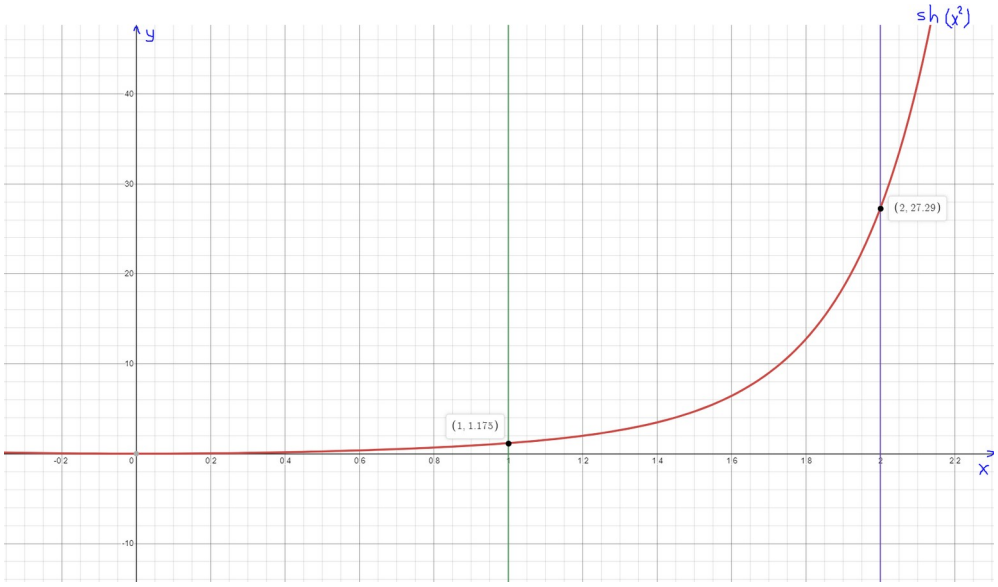
- $I_{h/2} - \frac{I_{h/2} - I_h}{3}$ – для формулы трапеций;
- $I_{h/2} - \frac{I_{h/2} - I_h}{15}$ – для формулы Симпсона.

При этом за погрешность приближённого значения интеграла принимается величина $\frac{|I_{h/2} - I_h|}{2^k - 1}$, причём для формул прямоугольников и трапеций $k = 2$, для формулы Симпсона $k = 4$.

Такую оценку погрешностей применяют обычно для построения адаптивных алгоритмов, т.е. таких алгоритмов, которые автоматически так определяют величину шага h, что результат удовлетворяет требуемой точности.

Постановка задачи с кратким описанием порядка выполнения работы.

При помощи онлайн-калькулятора *desmos* был постороен график функции $f(x)$ на заданном отрезке $[a,b]$.



Используя квадратурные формулы прямоугольников, трапеций и Симпсона, необходимо вычислить значения заданного интеграла и, применив правило Рунге, найти наименьшее значение n (наибольшее значение шага h), при котором каждая из указанных формул дает приближенное значение интеграла с погрешностью ε, не превышающей заданную.

Для выполнения работы было принято решение использовать язык программирования *kotlin*. Для разработки использовалась *IDE IntelliJ IDEA* и операционная система *Windows 10*. Были написаны функции *f*, *f2* и *f4*, вычисляющие значение функции, первой и четвёртой её производной в заданной точке соответственно. Также были написаны методы *rect*, *trap* и *simp*, вычисляющие значения интеграла функции $f(x)$ на отрезке $[a,b]$ при помощи составных формул прямоугольников, трапеций и Симпсона. Для каждого из этих методов была написана ещё одна функция, вычисляющая окончательное приближённое значение интеграла и его погрешность по правилу Рунге, *rHead*, *tHead* и *sHead* соответственно. Была реализована головная функция *main*, производящая расчёты по заданным данным и выводящая результаты в консоль. Исходный код программы содержится в Приложении А.

Необходимые графики и таблицы с краткими выводами.

По результатам работы программы была построена таблица. В ней представлены результаты вычисления значения интеграла тремя названными методами, а также зависимость погрешности вычисления от заданной точности ε и количества делений сетки на заданном отрезке.

Метод	ε	n, кол-во делений	Значение интеграла	Погрешность вычислений
Прямоугольников	1	2	7.391	0.19182
Прямоугольников	0.1	4	7.4247	0.05638

Прямоугольников	0.01	16	7.4273	0.00372
Прямоугольников	0.001	32	7.4274	0.00093
Прямоугольников	0.0001	128	7.4274	0.00006
Трапеций	1	2	2.0527	0.87804
Трапеций	0.1	64	7.1501	0.06761
Трапеций	0.01	512	7.392	0.00883
Трапеций	0.001	8192	7.4251	0.00056
Трапеций	0.0001	65536	7.4270	0.00007
Симпсона	1	2	1.1658	0.08327
Симпсона	0.1	2	1.1658	0.08327
Симпсона	0.01	256	7.3146	0.00694
Симпсона	0.001	2048	7.4132	0.00089
Симпсона	0.0001	32768	7.4265	0.00006

Сравнительная оценка применяемых для вычисления формул.

Из полученных данных видно, что с уменьшением коэффициента ε, количество делений сетки, необходимое для обеспечения заданной точности растёт.

Практические исследования показали, что для данной функции метод прямоугольников достигает высокой точности быстрее, чем метод трапеций и метод Симпсона, из которых, в свою очередь, немного быстрее точность растёт у метода Симпсона.

Такая зависимость обусловлена свойствами интегрируемой фкнкции.

Общий вывод по проделанной работе.

В результате проделанной работы, был сделан вывод о том, что точность вычисления значения интеграла по составным формулам зависит от количества отрезков, на которые разбит заданный отрезок $[a,b]$. Какой из трёх описанных методов эффективнее — зависит от свойств подынтегральной функции.

Приложение А: исходный код программы.

Файл Main.kt:

```
package wd

import kotlin.math.*

const val max = 0.00001

/*
* Function int sh(x^2)dx in 1, 2; var 24.
* */
fun f(x: Double): Double = sinh(x.pow(2))

fun f2(x: Double): Double = 2*x*cosh(x.pow(2))

fun f4(x: Double): Double = 48*x.pow(2)*cosh(x.pow(2)) + 4*(3 + 4*x.pow(4))*sinh(x.pow(2))


fun rect(a: Double, b: Double, h: Double): Double {
    val n = ((b - a) / h).toInt()
    var result = 0.0
    for (i in 0 until n) result += f(a + i*h + h/2)
    val r1 = (h.pow(2) / 24) * (f2((a + b) / 2) * (b - a))
    return result * h + r1
}

fun trap(a: Double, b: Double, h: Double): Double {
    val n = ((b - a) / h).toInt()
    var result = 0.0
    for (i in 0 until n-1) result += f(a + i*h) + f(a + (i+1)*h)
    val r2 = -(h.pow(2) / 12) * (0.5 * (f2(a) + f2(b)) * (b - a))
    return result * h/2 + r2
}

fun simp(a: Double, b: Double, h: Double): Double {
    val n = ((b - a) / h).toInt()
    var result = 0.0
    for (i in 0 until n/2-1) result += f(a + 2*i*h) + 4*f(a + (2*i+1)*h) + f(a + (2*i+2)*h)
    val r3 = -(h.pow(4) / 180) * ((1/6) * (f4(a) + 4 * f4((a + b) / 2) + f4(b)) * (a - b))
    return result * h/3 + r3
}



fun rHead(a: Double, b: Double, h: Double): Pair<Double, Double> {
    val st = rect(a, b, h/2) + (rect(a, b, h/2) - rect(a, b, h)) / 3.0
    val app = (rect(a, b, h/2) - rect(a, b, h)).absoluteValue / 3.0 // 3 = (2^2 - 1)
    return Pair(st, app)
}
```

```
fun tHead(a: Double, b: Double, h: Double): Pair<Double, Double> {
    val st = trap(a, b, h/2) - (trap(a, b, h/2) - trap(a, b, h)) / 3.0
    val app = (trap(a, b, h/2) - trap(a, b, h)).absoluteValue / 3.0 // 3 = (2^2 - 1)
    return Pair(st, app)
}

fun sHead(a: Double, b: Double, h: Double): Pair<Double, Double> {
    val st = simp(a, b, h/2) - (simp(a, b, h/2) - simp(a, b, h)) / 15.0
    val app = (simp(a, b, h/2) - simp(a, b, h)).absoluteValue / 15.0 // 15 = (2^4 - 1)
    return Pair(st, app)
}



fun main() {
    val a = 1.0
    val b = 2.0
    var epsilon = 1.0
    var n = 2
    var res: Pair<Double, Double>

    println("METHOD OF RECTANGLES:")
    while (epsilon > max) {
        res = rHead(a, b, (b-a)/n)

        while (res.second > epsilon) {
            n *= 2
            res = rHead(a, b, (b-a)/n)
        }
         println("\tFor (epsilon = %4f), (n = ${n}) expression value is %4f, approximation is
%4f".format(epsilon, res.first, res.second))

        epsilon /= 10
    }

    epsilon = 1.0
    n = 2
    println("\nMETHOD OF TRAPEZOIDS:")
    while (epsilon > max) {
        res = tHead(a, b, (b-a)/n)

        while (res.second > epsilon) {
            n *= 2
            res = tHead(a, b, (b-a)/n)
        }
         println("\tFor (epsilon = %4f), (n = ${n}) expression value is %4f, approximation is
%4f".format(epsilon, res.first, res.second))

        epsilon /= 10
    }
```

```
            epsilon = 1.0
            n = 2
            println("\nMETHOD OF SIMPSON:")
            while (epsilon > max) {
               res = sHead(a, b, (b-a)/n)

               while (res.second > epsilon) {
                  n *= 2
                  res = sHead(a, b, (b-a)/n)
               }
                println("\tFor (epsilon = %4f), (n = ${n}) expression value is %4f, approximation is
%4f".format(epsilon, res.first, res.second))

               epsilon /= 10
            }
         }
```