Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа по вычислительной математике №3**

Численное интегрирование

Преподаватель: Малышева Татьяна Алексеевна

Выполнила: Голованова Дарья Владимировна

Группа: Р3222

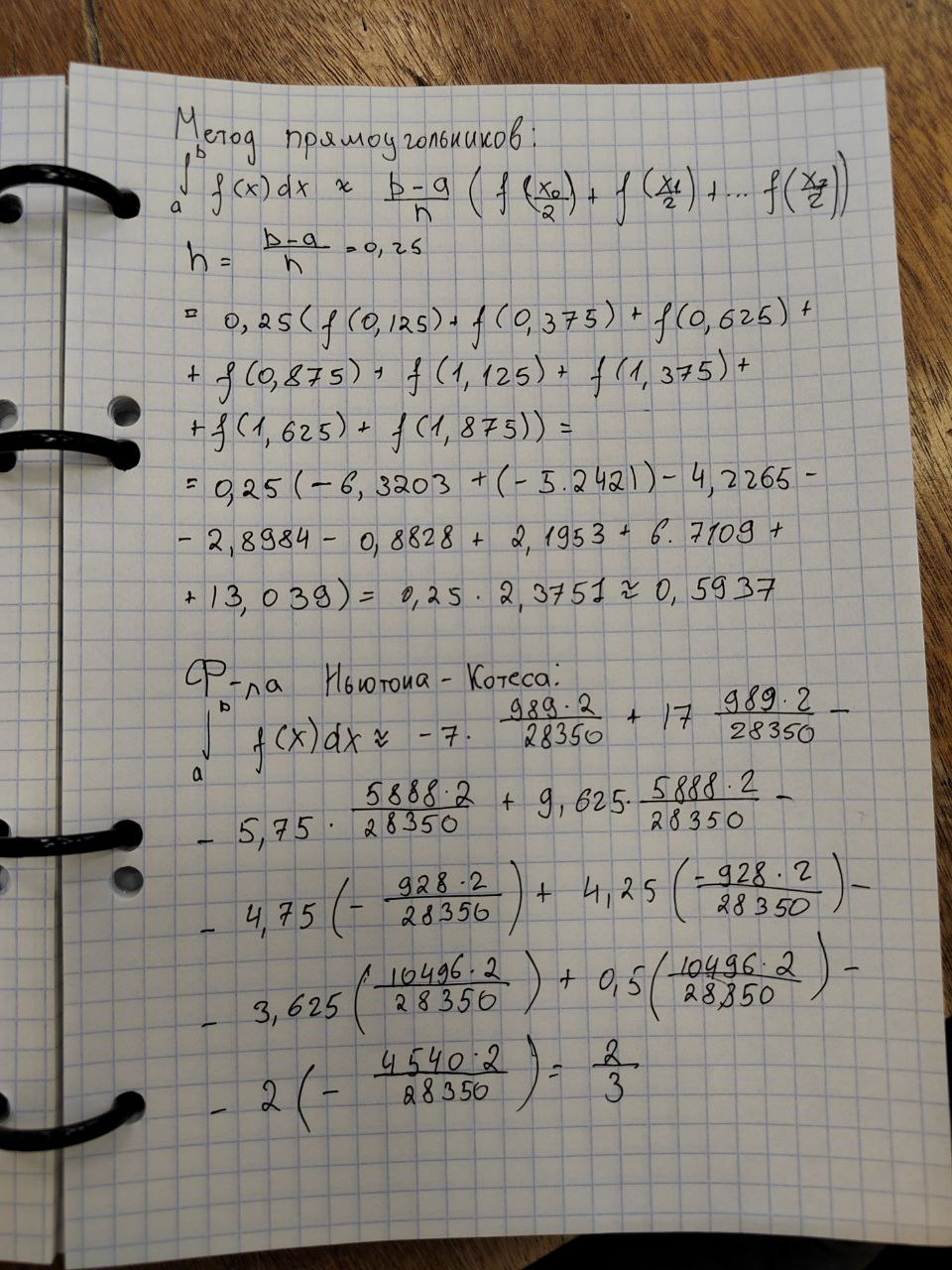
Санкт-Петербург,

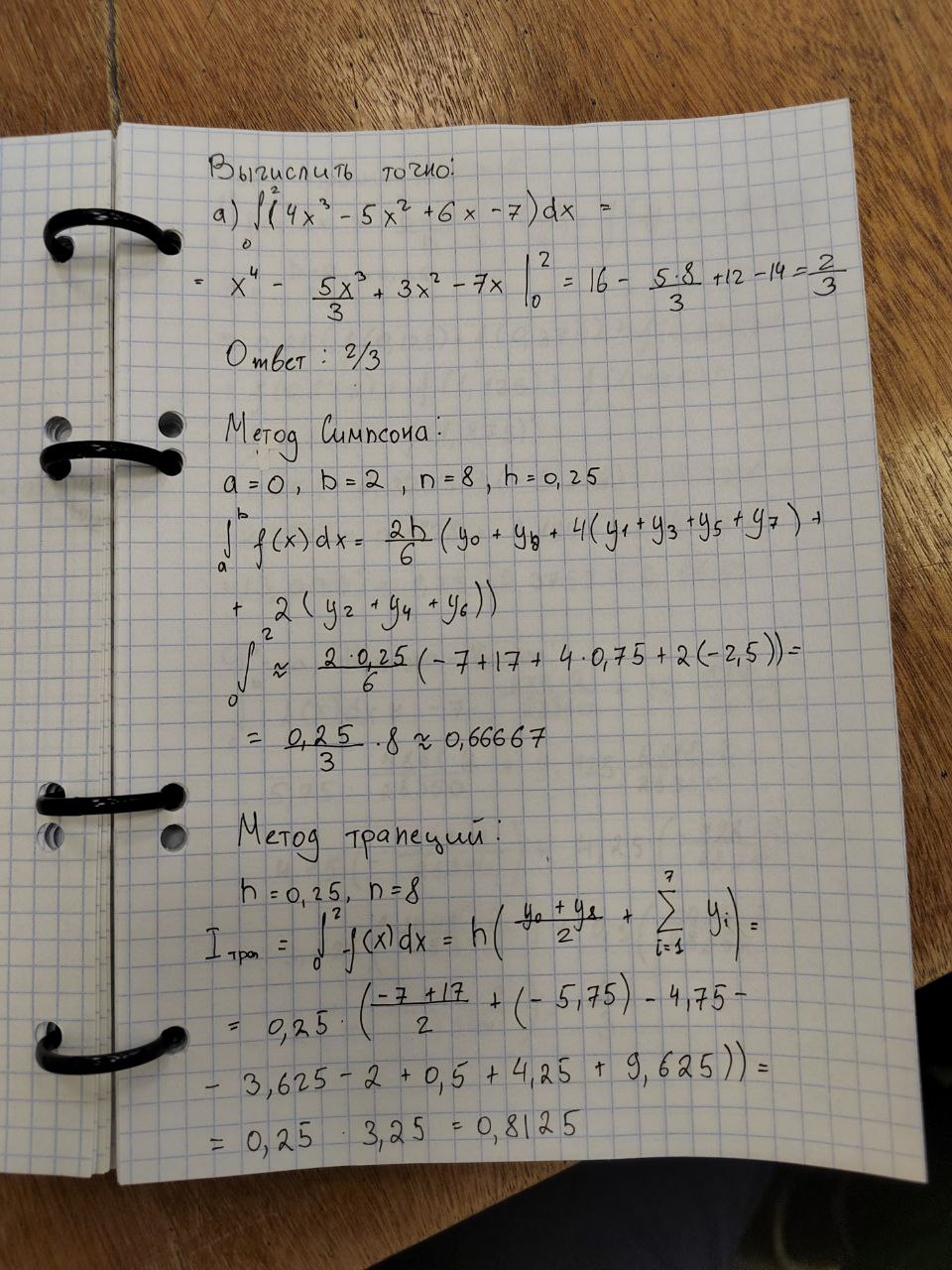
2022г

# Цель работы

Найти приближенное значение определенного интеграла с требуемой точностью различными численными методами.

# Вычислительная реализация задачи





# Программная реализация задачи

# Описание использованного метода

## Метод Симпсона:

Суть метода заключается в том, что на определенном промежутке дуга некоторой

параболы в общем случае теснее прилегает кривой y = f(x), чем хорда, соединяющая

концы дуги этой кривой (как в методе трапеций).

В связи с этим значения площадей соответствующих элементарных трапеций,

ограниченных сверху дугами парабол, являются более близкими к значениям площадей

соответствующих частичных криволинейных трапеций, ограниченных сверху дугой

кривой y = f(x), чем значения площадей соответствующих прямолинейных трапеций.

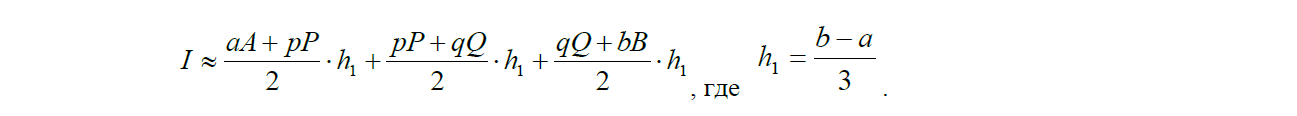
## 

Рассмотрим функцию *y* = *f(x)* такую, что на отрезке [a; b] она положительна и непрерывна. Найдем площадь криволинейной трапеции aABb (Рис1.).

Разобьём отрезок [*a, b*] точкой *c = (a+b)/2* пополам и в точке *C*(*c*, *f(c)*) и проведем касательную к графику *y* = *f(x).*

После этого разделим [*a, b*] точками *p* и *q* на три равные части и проведем через них прямые *x* = *p* и *x* = *q*.

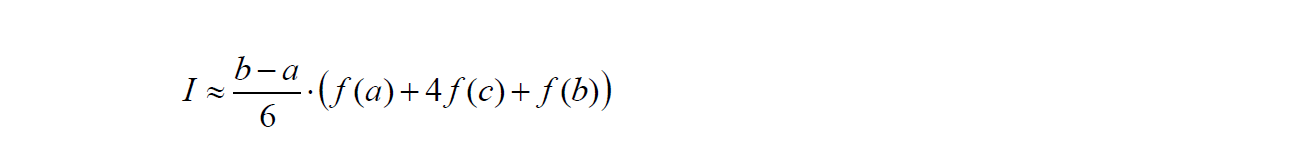
Пусть *P* и *Q* – точки пересечения этих прямых с касательной. Соединив *A* с *P* и *B* с *Q*, получим три трапеции *aAPp, pPQq, qQBb*. Тогда площадь трапеции *aABb* можно приближенно посчитать по следующей формуле



Откуда получаем

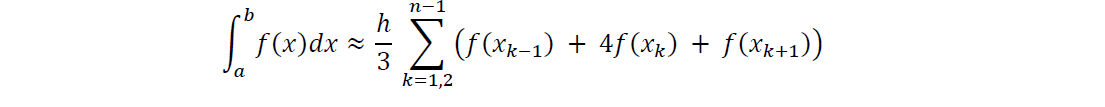


Заметим, что *aA* = *f(a), bB* = *f(b), а pP* + *qQ* = 2*f(c)* (как средняя линия трапеции), в итоге получаем формулу Симпсона:



Общий вид формулы Симпсона:

Если кратко, то суть метода заключается в том, что мы делим подынтегральную функцию на n(четное) равных отрезков, на каждом из которых аппроксимируем значение этой функции параболой, затем можем вычислить искомый интеграл по формуле:



При этом мы вычисляем значение интеграла по формуле для разбиения на n и 2n отрезков, после чего вычисляем погрешность по формуле Рунге.



где для метода Симпсона 𝜃 = 1/15

При этом если найденная погрешность оказывается больше заданной точности, то увеличиваем *n* в два раза и снова вычисляем интегралы. Затем повторяем увеличение *n* до тех пор погрешность не окажется меньше нужной точности.

## Метод прямоугольников:

Пусть есть функция , непрерывная на отрезке , тогда можем вычислить значение интеграла . Воспользуемся заменой определенного интеграла интегральной суммой. Разобьем отрезок на n частей где i и выбираем точку со значением Существует определенный тип интегральных сумм при бесконечном уменьшении длины такой части. Это выражается формулой λ =, тогда получаем, что любая из таких интегральных сумм – приближенное значение интеграла

Суть метода прямоугольниковзаключается в том, что приближенное значение считается интегральной суммой.

В качестве точек могут выбираться левые, правые или средние точки отрезков, то получаем формулы левых, правых и средних прямоугольников.

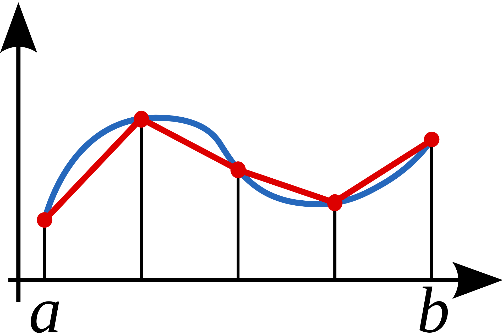
Обозначаем

Метод левых прямоугольников:

Метод правых прямоугольников:

Метод средних прямоугольников:

## Метод трапеций:

Метод трапеций – метод численного интегрирования функции от одной переменной. Суть данного метода заключается в замене на каждом элементарном отрезке подынтегральной функции на линейную функцию. Площадь под графиком функции аппроксимируется прямоугольными трапециями.

Если отрезок является элементарным и не подвергается дальнейшему разбиению, значение интеграла можно найти по формуле:

Это простое применение формулы для площади трапеции – произведение полусуммы оснований, которыми в данном случае являются значения функции в крайних точках отрезка, на высоту (длину отрезка интегрирования). Погрешность аппроксимации можно оценить через максимум второй производной

Тогда формула для интегрирования на всё отрезке :

При этом мы вычисляем значение интеграла по формуле для разбиения на и отрезков, после чего вычисляем погрешность по формуле Рунге.

При этом если найденная погрешность оказывается больше заданной точности, то увеличиваем n в два раза и снова вычисляем интегралы. Затем повторяем увеличение n до тех пор погрешность не окажется меньше нужной точности.

# Выводы

Выполнив данную лабораторную работу, я пришла к выводу, что все приведенные в вариантах методы интегрирования схожи между собой в том, что всегда производится разбиение функции на *n* отрезков, а затем интерполирование на каждом из них.

Рассмотрю подробнее все три метода, чтобы это доказать.

1. Метод прямоугольников:

1.1 Метод левых прямоугольников

1.2 Метод средних прямоугольников

1.3 Метод правых прямоугольников

Метод заключается в том, что мы разбиваем фигуру под графиком на прямоугольники и считаем интеграл как сумму площадей этих прямоугольников с учетом значения функции в левой точке (метод левых прямоугольников) в средней точке (метод средних прямоугольников) и в правой точке (метод правых прямоугольников). В силу того, что в методе средних прямоугольников симметрия не нарушается, то погрешность у данного метода будет меньше, чем у левых и правых прямоугольников.

2. Метод трапеций:

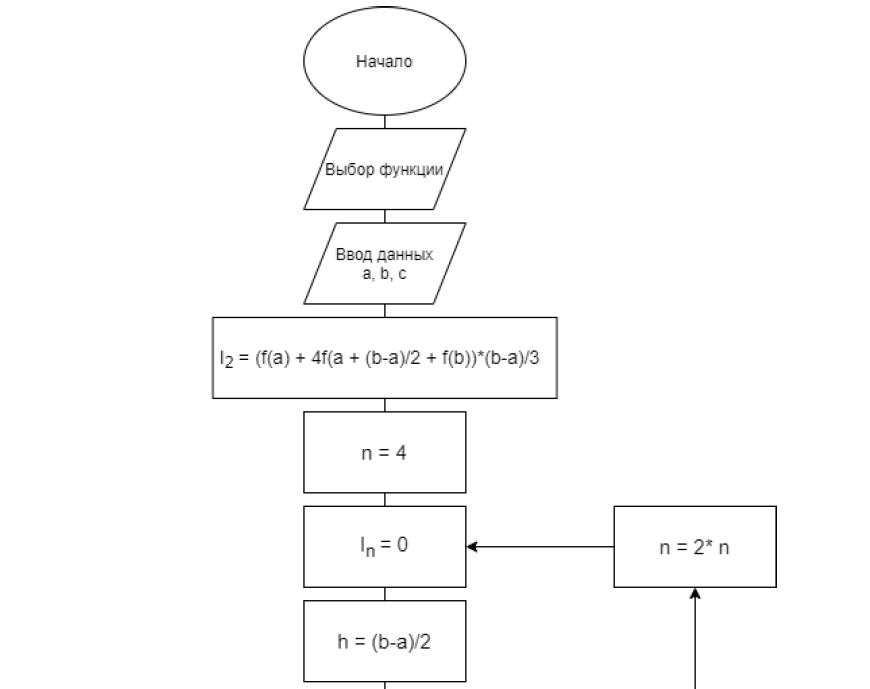
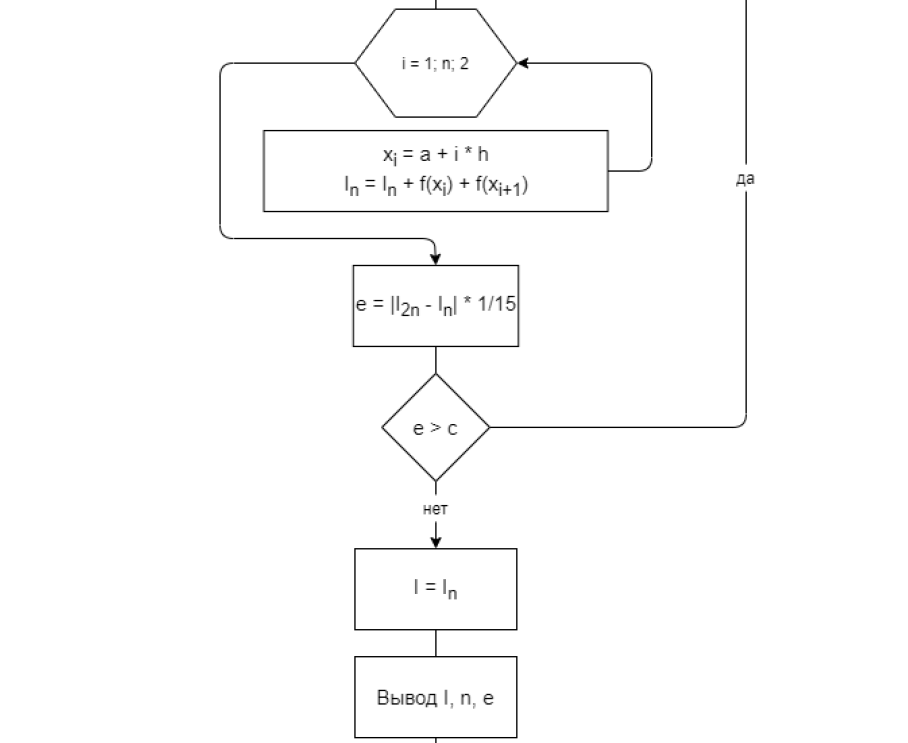
В данном методе мы аппроксимируем функцию к прямой и считаем интеграл как площадь многоугольника, образованного получившимися трапециями. Данный метод менее точный чем средние прямоугольники так как значение в средней точке точнее, чем полусумма значений на концах.

3. Метод Симпсона:

Метод Симпсона является самым точным из всех представленных методов, так как мы аппроксимируем функцию параболой (которая зачастую находится гораздо ближе к графику (в сравнении с прямой)).

# Блок-схемы

Метод Симпсона:



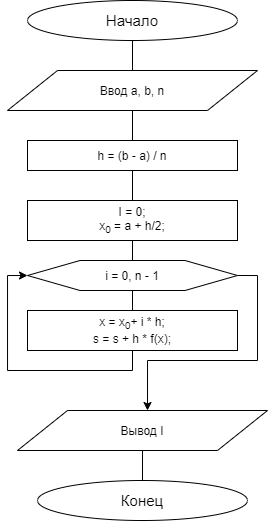
# 

Метод левых прямоугольников:

Изображение выглядит как текст

Автоматически созданное описание

Метод средних прямоугольников:



Метод левых прямоугольников:

Изображение выглядит как текст

Автоматически созданное описание

Метод правых прямоугольников:

Изображение выглядит как текст

Автоматически созданное описание

Метод трапеций:

**Изображение выглядит как часы

Автоматически созданное описание**

# Листинг численных методов:

## Проверка точности методом Руне

```

def count\_abstract\_integral\_rune\_check(result\_formula, a, b, e):

num\_of\_intervals = 4

h = interval\_width(a, b, num\_of\_intervals)

result = result\_formula(h)

prev\_result = float('inf')

while(abs(result - prev\_result) > e):

num\_of\_intervals \*= 2

h = interval\_width(a, b, num\_of\_intervals)

prev\_result = result

result = result\_formula(h)

return result, num\_of\_intervals

def interval\_width(a, b, num\_of\_intervals):

return abs(b - a) / num\_of\_intervals

```

## Метод Прямоугольников

```

def count\_integral\_start(f, a, b, e):

result\_start = lambda h : sum([f(i) for i in np.arange(a, b, h)]) \* h

return count\_abstract\_integral\_rune\_check(result\_start, a, b, e \* 3)

def count\_integral\_middle(f, a, b, e):

result\_middle = lambda h : sum([f(i) for i in np.arange(a + h/2, b, h)]) \* h

return count\_abstract\_integral\_rune\_check(result\_middle, a, b, e \* 3)

def count\_integral\_stop(f, a, b, e):

result\_stop = lambda h : sum([f(i) for i in np.arange(a + h, b, h)]) \* h

return count\_abstract\_integral\_rune\_check(result\_stop, a, b, e \* 3)

```

## Метод Трапеции

```

result\_formula = lambda h : (sum([f(i) for i in np.arange(a + h, b, h)]) + (f(a) + f(b)) / 2)\* h

integral, num\_of\_intervals = count\_abstract\_integral\_rune\_check(result\_formula, a, b, e \* 3)

```

## Метод Симпсона

```

def calculate\_integral\_simpson\_method(f, a, b, e):

result\_formula = lambda h: simpson\_formula(h, a, b, f)

result, num\_of\_intervals = count\_abstract\_integral\_rune\_check(result\_formula, a, b, e \* 15)

print(f"simpson integral={result}, while the number of intervals was={num\_of\_intervals}")

def simpson\_formula(h, a, b, f):

num\_of\_intervals = abs(b - a) / h

odd\_sum = count\_odd\_sum(f, a, b, num\_of\_intervals)

even\_sum = count\_even\_sum(f, a, b, num\_of\_intervals)

return h/3 \* (f(a) + 4 \* odd\_sum + 2 \* even\_sum + f(b))

def count\_even\_sum(f, a, b, num\_of\_intervals):

h = count\_interval\_width(a, b, num\_of\_intervals)

return sum([f(i) for i in np.arange(a + h, b, 2\*h)])

def count\_odd\_sum(f, a, b, num\_of\_intervals):

h = count\_interval\_width(a, b, num\_of\_intervals)

return sum([f(i) for i in np.arange(a + 2\*h, b, 2\*h)])

def count\_interval\_width(a, b, num\_of\_intervals):

return abs(b-a) / num\_of\_intervals

```

# Примеры и результаты работы программы

## Пример 1

