Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №4

по дисциплине «**Вычислительная математика**»

Автор: Баянов Равиль Динарович

Факультет: ПИиКТ

Группа: P3234

Преподаватель: Перл О. В.

Санкт-Петербург, 2024

Оглавление

Описание метода

Мы имеем функцию y = f(x), которая непрерывна и что самое главное дифференцируема на отрезке [a, b]. Нужно вычислить значение интеграла

$$\int_a^b f(x)dx$$

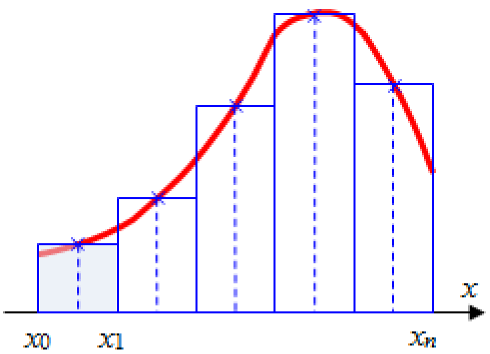
Будем пользоваться понятием неопределённого интеграла. Интеграл – это сумма бесконечного числа умножений. Следовательно, на отрезке [a, b] разобьём нашу функцию на какое-то количество n отрезков. Получим прямоугольники, расположенные по всей нашей функции. И дальше будем на вершине прямоугольника брать какую-то случайную точку, для того чтобы затем получить сумму изменяющейся величины для нахождения интеграла. То, какую точку на отрезке мы выберем, зависит от метода, который мы используем. Есть три разновидности метода прямоугольников для вычисления определённого интеграла: левый, средний, правый. Для точности, искомого значения будем пользоваться методом средних прямоугольников. А это значит, что формула для вычисления определённого интеграла будем выглядеть так:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(c_i)(x_i - x_{i-1})$$

Где, c_i – это середина каждого отрезка нашего разбиения. Формулу можно также немного видоизменить с использованием переменной h, которая будет являться шагом для наших прямоугольников.

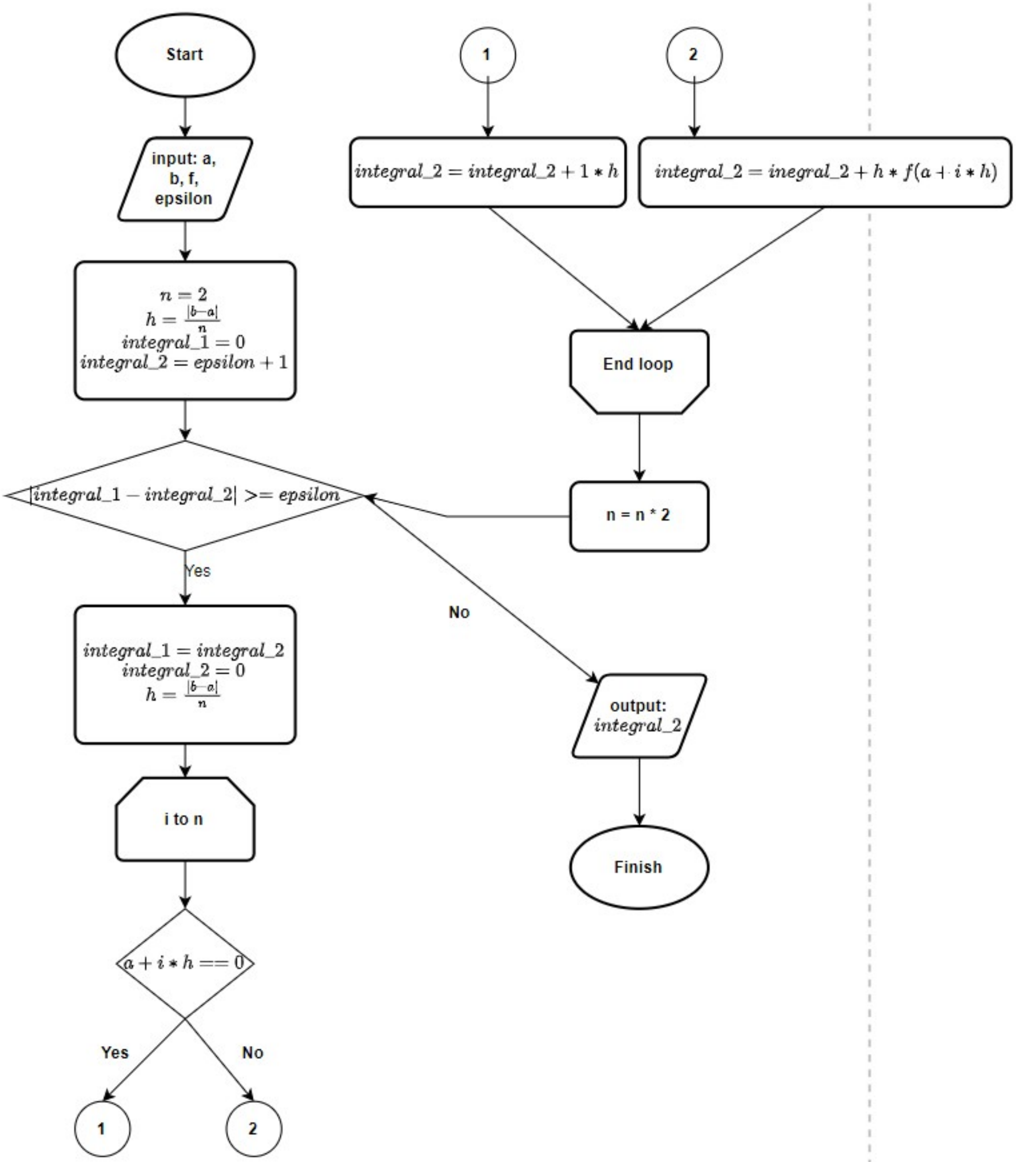
$$\int_a^b f(x)dx \approx h\sum_{i=1}^{n} f(c_i)\left(x_{i-1} + \frac{h}{2}\right)$$

На графике это будет выглядеть примерно так:



Собственно, теперь реализуем этот алгоритм в коде, но перед этим создадим блок-схему алгоритма.

Блок-схема



Start
input: a, b, f, epsilon
$n = 2$
$h = \frac{|b-a|}{n}$
$integral\_1 = 0$
$integral\_2 = epsilon + 1$
$|integral\_1 - integral\_2| >= epsilon$
Yes
$integral\_1 = integral\_2$
$integral\_2 = 0$
$h = \frac{|b-a|}{n}$
i to n
$a + i * h == 0$
Yes
No
1
2
1
2
$integral\_2 = integral\_2 + 1 * h$
$integral\_2 = inegral\_2 + h * f(a + i * h)$
End loop
n = n * 2
No
output: $integral\_2$
Finish

Исходный код метода на языке программирования Python

```
class Result:
  error_message = "Integrated function has discontinuity or does not defined in current interval"
  has_discontinuity = False

  def first_function(x: float):
    return 1 / x

  def second_function(x: float):
    return math.sin(x) / x

  def third_function(x: float):
    return x * x + 2

  def fourth_function(x: float):
    return 2 * x + 2

  def five_function(x: float):
    return math.log(x)

  # How to use this function:
  # func = Result.get_function(4)
  # func(0.01)
  def get_function(n: int):
    if n == 1:
      return Result.first_function
    elif n == 2:
      return Result.second_function
    elif n == 3:
      return Result.third_function
    elif n == 4:
      return Result.fourth_function
    elif n == 5:
      return Result.five_function
    else:
      raise NotImplementedError(f"Function {n} not defined.")

  #
  # Complete the 'calculate_integral' function below.
  #
```

```
# The function is expected to return a DOUBLE.
# The function accepts following parameters:
#  1. DOUBLE a
#  2. DOUBLE b
#  3. INTEGER f
#  4. DOUBLE epsilon
#

def iterations(a, b, func, epsilon):
  n = 2
  h = abs(b - a)/ n
  integral_1 = 0
  integral_2 = epsilon + 1
  while abs(integral_1 - integral_2) >= epsilon:
    integral_1 = integral_2
    integral_2 = 0
    h = math.fabs(b - a) / n
    for i in range(n) :
      if a + i * h == 0:
        integral_2 += 1 * h
      else:
        integral_2 += h * func(a + i * h)
    n *= 2
  return integral_2
def calculate_integral(a, b, f, epsilon):
  func = Result.get_function(f)
  if f == 1:
    if a * b < 0:
      Result.has_discontinuity = True
      return None
    if b < a:
      return (-1) * Result.iterations(a, b, func, epsilon)
    else:
      return Result.iterations(a, b, func, epsilon)
  elif f == 2:
    if b < a:
      return (-1) * Result.iterations(a, b, func, epsilon)
    else:
      return Result.iterations(a, b, func, epsilon)
  elif f == 3:
    if b < a:
      return (-1) * Result.iterations(a, b, func, epsilon)
    else:
      return Result.iterations(a, b, func, epsilon)
```

```
    elif f == 4:
      if b < a:
        return (-1) * Result.iterations(a, b, func, epsilon)
      else:
        return Result.iterations(a, b, func, epsilon)
    elif f == 5:
      if b * a <= 0 or (b < 0 and a < 0):
        Result.has_discontinuity = True
        return None
      if b < a:
        return (-1) * Result.iterations(a, b, func, epsilon)
      else:
        return Result.iterations(a, b, func, epsilon)
```

Примеры работы программы

1. Случай, когда длина отрезка равна нулю (a = b):

```
1
1
3
0.1
0.0
```

2. Третья функция (a < b):

```
1
4
3
0.001
26.99931335868314
```

3. Третья функция (a > b):

```
4
1
3
0.001
-26.99931335868314
```

4. Разрыв второго рода:

```
-1
1
1
0.001
Integrated function has discontinuity or does not defined in current interval
```

5. Пятая функция:

```
1
2
5
0.001
0.3856173006306909
```

Вывод

Метод средних квадратов для вычисления определённых интегралов является одним из многих известных методов, используемых для решения этой задачи.

Сравним этот метод с другими численными методами.

- Метод средних прямоугольников чуть более точный, чем левый и правый метод прямоугольников, так как он использует больше точек для приближения площади под кривой интегрирования.
- Метод трапеций чуть менее точный, чем метод средних прямоугольников, но требует меньше вычислений.
- Метод Симпсона требует меньше вычислений, чем метод средних квадратов, но чуть менее точный.

Таким образом, метод средних прямоугольников является одним из самых точных методов для вычисления определённого интеграла. Однако, он требует больше вычислений, чем другие методы.

Главное условие для применимости метода средних прямоугольников – это непрерывность функции на заданном интервале. Разрывы функции нужно либо устранять, либо сообщать об ошибке, если функция имеет разрывы второго рода. Также не стоит забывать про выбор точек разбиения, например, если у нас есть область, на которой функция изменяется очень быстро, то необходимо выбирать более плотное разбиение в этой области.

Алгоритмическая сложность интегрирования методом средних прямоугольников примерно равна $O(n)$, но если мы ещё будем учитывать точность, к которой мы должны стремиться, то алгоритмическая сложность времени нашего метода может возрасти до $O(n^2)$.

Нетрудно, также заметить, что погрешность данного метода интегрирования уменьшается при уменьшении шага интегрирования h. Также погрешность зависит от гладкости функции и стоит избегать использования нашего метода для вычисления интегралов от функции с большими значениями второй производной.

В заключении, можно сделать вывод, что метод средних прямоугольников крайне простой и точный, но не всегда самый быстрый и применимый метод для вычисления определённого интеграла.