**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.

Изображение выглядит как черный, темнота

Автоматически созданное описание

Санкт-Петербург, 2024

Оглавление

[Описание метода 3](#_Toc160999556)

[Блок-схема 4](#_Toc160999557)

[Исходный код метода на языке программирования Python 5](#_Toc160999558)

[Примеры работы программы 6](#_Toc160999559)

[Вывод 9](#_Toc160999560)

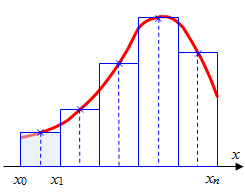
# Описание метода

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

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:

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

На графике это будет выглядеть примерно так:



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

# Блок-схема

Изображение выглядит как диаграмма, зарисовка, текст, рисунок

Автоматически созданное описание

# Исходный код метода на языке программирования Python

|  |
| --- |
| 1. **class** Result: 2. error\_message **=** "Integrated function has discontinuity or does not defined in current interval" 3. has\_discontinuity **=** False 5. **def** first\_function(x: float): 6. **return** 1 **/** x 8. **def** second\_function(x: float): 9. **return** math.sin(x) **/** x 11. **def** third\_function(x: float): 12. **return** x **\*** x **+** 2 14. **def** fourth\_function(x: float): 15. **return** 2 **\*** x **+** 2 17. **def** five\_function(x: float): 18. **return** math.log(x) 20. # How to use this function: 21. # func = Result.get\_function(4) 22. # func(0.01) 23. **def** get\_function(n: int): 24. **if** n **==** 1: 25. **return** Result.first\_function 26. **elif** n **==** 2: 27. **return** Result.second\_function 28. **elif** n **==** 3: 29. **return** Result.third\_function 30. **elif** n **==** 4: 31. **return** Result.fourth\_function 32. **elif** n **==** 5: 33. **return** Result.five\_function 34. **else**: 35. **raise** NotImplementedError(f"Function {n} not defined.") 37. # 38. # Complete the 'calculate\_integral' function below. 39. # 40. # The function is expected to return a DOUBLE. 41. # The function accepts following parameters: 42. #  1. DOUBLE a 43. #  2. DOUBLE b 44. #  3. INTEGER f 45. #  4. DOUBLE epsilon 46. # 48. **def** iterations(a, b, func, epsilon): 49. n **=** 2 50. h **=** abs(b **-** a)**/** n 51. integral\_1 **=** 0 52. integral\_2 **=** epsilon **+** 1 53. **while** abs(integral\_1 **-** integral\_2) >**=** epsilon: 54. integral\_1 **=** integral\_2 55. integral\_2 **=** 0 56. h **=** math.fabs(b **-** a) **/** n 57. **for** i **in** range(n) : 58. **if**  a **+** i **\*** h **==** 0: 59. integral\_2 **+=** 1 **\*** h 60. **else**: 61. integral\_2 **+=** h **\*** func(a **+** i **\*** h) 62. n **\*=** 2 63. **return** integral\_2 64. **def** calculate\_integral(a, b, f, epsilon): 65. func **=** Result.get\_function(f) 66. **if** f **==** 1: 67. **if** a **\*** b < 0: 68. Result.has\_discontinuity **=** True 69. **return** None 70. **if** b < a: 71. **return** (**-**1) **\*** Result.iterations(a, b, func, epsilon) 72. **else**: 73. **return** Result.iterations(a, b, func, epsilon) 74. **elif** f **==** 2: 75. **if** b < a: 76. **return** (**-**1) **\*** Result.iterations(a, b, func, epsilon) 77. **else**: 78. **return** Result.iterations(a, b, func, epsilon) 79. **elif** f **==** 3: 80. **if** b < a: 81. **return** (**-**1) **\*** Result.iterations(a, b, func, epsilon) 82. **else**: 83. **return** Result.iterations(a, b, func, epsilon) 84. **elif** f **==** 4: 85. **if** b < a: 86. **return** (**-**1) **\*** Result.iterations(a, b, func, epsilon) 87. **else**: 88. **return** Result.iterations(a, b, func, epsilon) 89. **elif** f **==** 5: 90. **if** b **\*** a <**=** 0 **or** (b < 0 **and** a < 0): 91. Result.has\_discontinuity **=** True 92. **return** None 93. **if** b < a: 94. **return** (**-**1) **\*** Result.iterations(a, b, func, epsilon) 95. **else**: 96. **return** Result.iterations(a, b, func, epsilon) |

# Примеры работы программы

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

Изображение выглядит как снимок экрана, Шрифт, число

Автоматически созданное описание

1. Третья функция (a < b):

Изображение выглядит как текст, Шрифт, снимок экрана, часы

Автоматически созданное описание

1. Третья функция (a > b):

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описание

1. Разрыв второго рода:

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

1. Пятая функция:

Изображение выглядит как снимок экрана, Шрифт, текст, дизайн

Автоматически созданное описание

# 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(n2).

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.