

Частное общеобразовательное учреждение
«Школа разговорных языков»

ПРОЕКТ

на тему

**«ВЫЧИСЛЕНИЕ ЗНАЧЕНИЯ π .
ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ.
МЕТОД ПРЯМОУГОЛЬНИКОВ»**

(по Информатике и ИКТ)

ученика 11 класса

Доричева Тимофея Константиновича

Санкт-Петербург

2024

Содержание

1 Введение	2
2 Краткая историческая справка	2
3 Теоретическое обоснование используемого метода	3
4 Суть метода прямоугольников	3
5 Расчёты	4
6 Анализ полученных результатов	5
7 Заключение	5
8 Приложение. Код построения графика	6
Список иллюстраций	7
Список таблиц	7
Список литературы	7

1 Введение

Цель данного проекта — наглядно пояснить один из численных методов вычисления значения π — метод прямоугольников.

Работа выполнена с использованием языка программирования [Python](#), библиотек [NumPy](#), [Matplotlib](#), [Seaborn](#).

Работа оформлена в издательской системе [L^AT_EX](#).

2 Краткая историческая справка

Число π (произносится «пи») — математическая постоянная, равная отношению длины окружности к её диаметру.

Число π иррационально, то есть его значение не может быть точно выражено в виде дроби $\frac{m}{n}$, где m — целое число, а n — натуральное. Следовательно, его десятичное представление никогда не заканчивается и не является периодическим.

Иррациональность числа π была впервые доказана Иоганном Ламбертом в 1761 году путём разложения тангенса в непрерывную дробь.

Первые 50 знаков мантиссы числа π [1, Википедия]:

$$\pi \approx 3.14159265358979323846264338327950288419716939937510 \dots \quad (1)$$

3 Теоретическое обоснование используемого метода

Общеизвестно, что площадь круга S вычисляется по формуле:

$$S = \pi \times R^2 \quad (2)$$

где R — радиус круга

Из 2 имеем:

$$\pi = \frac{S}{R^2} \quad (3)$$

А для окружности единичного радиуса ($R = 1$),

$$\pi = S \quad (4)$$

Также общеизвестно, что уравнение окружности в декартовой системе координат имеет вид:

$$x^2 + y^2 = R^2 \quad (5)$$

А для единичной окружности в первой четверти декартовой системы координат:

$$y = \sqrt{1 - x^2}, x \in [0, 1] \quad (6)$$

Таким образом, площадь *четверти* круга будет численно равна площади фигуры, ограниченной осями координат и графиком функции 6.

Данную площадь можно вычислить с помощью определённого интеграла:

$$S/4 = \int_0^1 \sqrt{1 - x^2} dx \quad (7)$$

Тогда, из 4 и 7 имеем окончательную рабочую расчётную формулу:

$$\pi = 4 \times \int_0^1 \sqrt{1 - x^2} dx \quad (8)$$

4 Суть метода прямоугольников

Для нахождения значения интеграла 8 воспользуемся методом прямоугольников. Суть метода состоит в разбиении интервала интегрирования на равные промежутки и аппроксимации приращения функции на интервале константой. Далее находятся и суммируются площади полученных прямоугольников, и при количестве интервалов разбиения $n \rightarrow \infty$, полученное значение стремится к значению интеграла.

В зависимости от того, какое значение функции принимают за основу, различают методы левых, правых и серединных прямоугольников.

Так, для любого элементарного интервала $[a, b]$ и метода *левых* прямоугольников имеем:

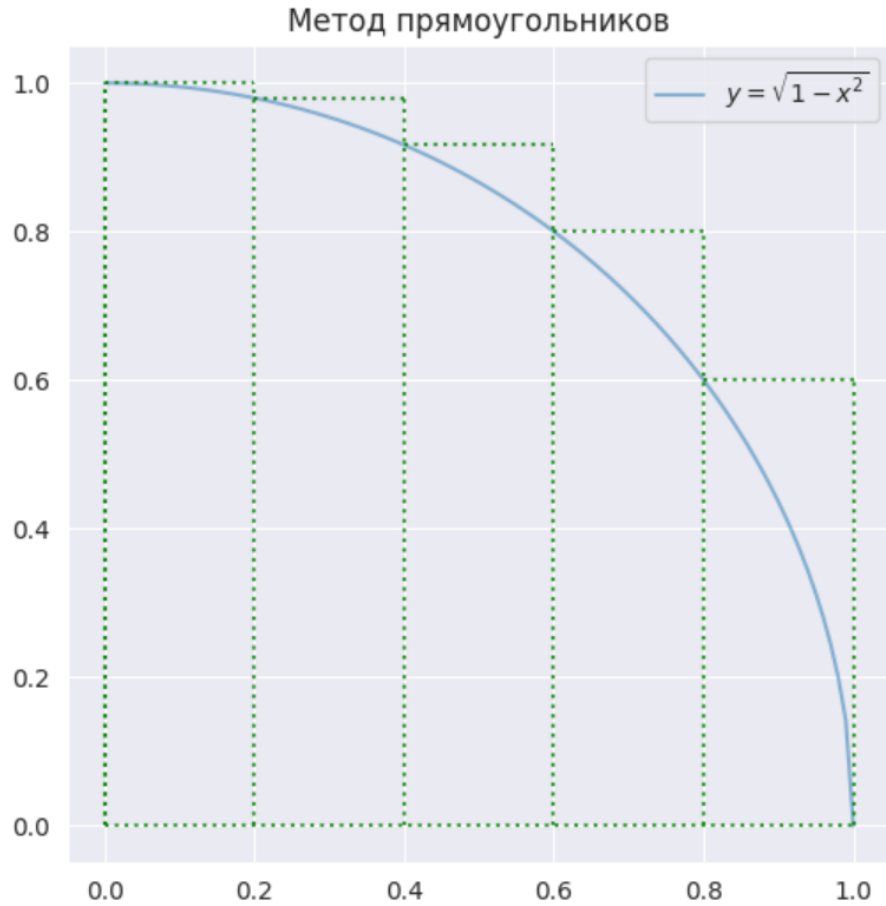


Рис. 1: Метод левых прямоугольников

$$\int_a^b f(x) dx \approx f(a) \times (b - a) \quad (9)$$

Тогда из 8, заменив интеграл суммой, имеем:

$$\pi = 4 \times \int_0^1 \sqrt{1 - x^2} dx \approx 4 \times \sum_{n=0}^{n-1} f(x_i) \times (x_{i+1} - x_i) \quad (10)$$

Подробнее метод описан в [2].

5 Расчёты

Для вычисления (10) напомним на языке программирования Python функцию, которая будет принимать на вход аргумент `n` — количество интервалов интегрирования (`num_intervals`) и возвращать значение π и разницу с известным, базовым значением — ошибку. За базовое возьмём значение, используемое в библиотеке NumPy:

$$\pi \approx 3.141592653589793$$

Точность — 15 знаков мантиссы, сравни с (1).

```

1 import numpy as np
2
3 def pi_rectangles(num_intervals: int) -> tuple[np.float64, np.float64]:
4
5     if num_intervals == 0:
6         return np.NaN, np.NaN
7
8     dx = np.double(1.0) / num_intervals
9     x = np.linspace(0, 1, num_intervals, endpoint=False)
10    y = np.sqrt(1 - x**2)
11    ds = y * dx
12    pi = np.sum(ds) * 4
13    return pi, pi - np.pi

```

Листинг 1: Функция вычисления π и ошибки

Необходимо отметить, что библиотека NumPy поддерживает т. н. векторизацию — один из способов реализации параллельных вычислений. Поэтому позволяет писать эффективный и компактный код и в некоторых случаях, как выше, избегать циклов, работая с векторами.

Так, к примеру, строка 7 листинга 1 — получение вектора значений аргумента x , а строки 8, 9 — вычисление всех значений функции y во всех точках x , и всех значений площадей прямоугольников ds .

6 Анализ полученных результатов

В таблице 1 приведены вычисленные значения константы π в зависимости от количества интервалов интегрирования n — количества прямоугольников, а также значения ошибки. Жирным шрифтом выделены верные цифры.

Таблица 1: Результаты вычислений

n	Значение π	Ошибка
1	4.0	0.858407346410207
10	3.304518326248318	0.162925672658520
100	3.160417031779045	0.018824378189252
1000	3.143555466911028	0.001962813321235
10000	3.141791477611323	0.000198824021529
100000	3.141612616401987	0.000019962812194
1000000	3.141594652413811	0.000001998824017

На рисунке 2 ось абсцисс для наглядности выбрана в логарифмическом масштабе.

7 Заключение

В данной работе продемонстрирован один из численных методов вычисления значения константы π — метод прямоугольников.

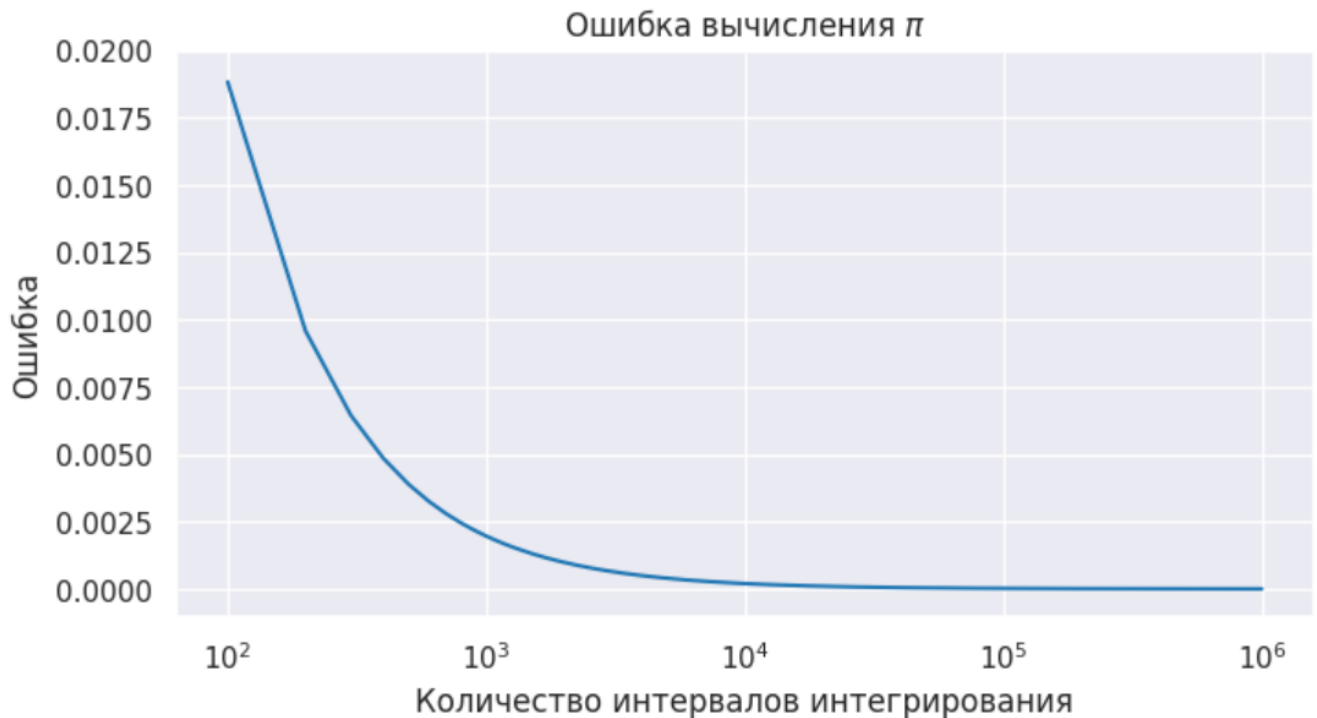


Рис. 2: Ошибка вычислений как функция количества интервалов интегрирования

8 Приложение. Код построения графика

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  import seaborn as sns
4
5  # Значения функции
6  x = np.linspace(0, 1, 100, endpoint=True)
7  y = np.sqrt(1 - x**2)
8
9  # Значения функции для прямоугольников
10 xr = np.linspace(0, 1, 5, endpoint=False)
11 yr = np.sqrt(1 - xr**2)
12
13 # Построение графика
14 fig, ax = plt.subplots(figsize=(6, 6))
15 sb_ax = sns.lineplot(y=y, x=x, alpha=0.5, label="$y = \sqrt{1 - x^2}$")
16 sb_ax.set(title = 'Метод прямоугольников')
17 ax.set_xlim(-0.05, 1.05)
18 ax.set_ylim(-0.05, 1.05)
19
20 # Построение прямоугольников
21 sb_ax.hlines(
22     y=yr,
23     xmin=xr[:],
24     xmax=[*xr[1:], 1.0],
25     colors='green',
26     linestyles='dotted',
27     alpha=0.8,
28 )
29 sb_ax.vlines(

```

```

30     x=[*xr, 1.0],
31     ymin=0,
32     ymax=[1.0, *yr[::]],
33     colors='green',
34     linestyle='dotted',
35     alpha=0.8,
36 )
37 ax.vlines(
38     x=0.0,
39     ymin=0,
40     ymax=1.0,
41     colors='green',
42     linestyle='dotted',
43     alpha=0.8,
44 )
45 ax.hlines(
46     y=0.0,
47     xmin=0,
48     xmax=1.0,
49     colors='green',
50     linestyle='dotted',
51     alpha=0.8,
52 )

```

Листинг 2: Код построения рис.1

Список иллюстраций

1	Метод левых прямоугольников	4
2	Ошибка вычислений как функция количества интервалов интегрирования . .	6

Список таблиц

1	Результаты вычислений	5
---	---------------------------------	---

Список литературы

- [1] [Википедия. Число Пи](#)
- [2] [Википедия. Метод прямоугольников](#)
- [3] Библиотека Numerical Python (NumPy). numpy.org
- [4] Графическая библиотека Seaborn. seaborn.pydata.org