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

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 LaTeX.

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]:

$$\pi \approx 3.14159265358979323846264338327950288419716939937510\ldots \tag{1}$$

3 Теоретическое обоснование используемого метода

Общеизвестно, что площадь круга S вычисляется по формуле:

$$S = \pi \times R^2 \tag{2}$$

где R — радиус круга
Из 2 имеем:

$$\pi = \frac{S}{R^2} \tag{3}$$

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

$$\pi = S \tag{4}$$

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

$$x^2 + y^2 = R^2 \tag{5}$$

А для единичной окружности в первой четверти декартовой системы координат:

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

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:

$$S/4 = \int_0^1 \sqrt{1 - x^2}\, dx \tag{7}$$

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

$$\pi = 4 \times \int_0^1 \sqrt{1 - x^2}\, dx \tag{8}$$

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 $n \to \infty$, 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:

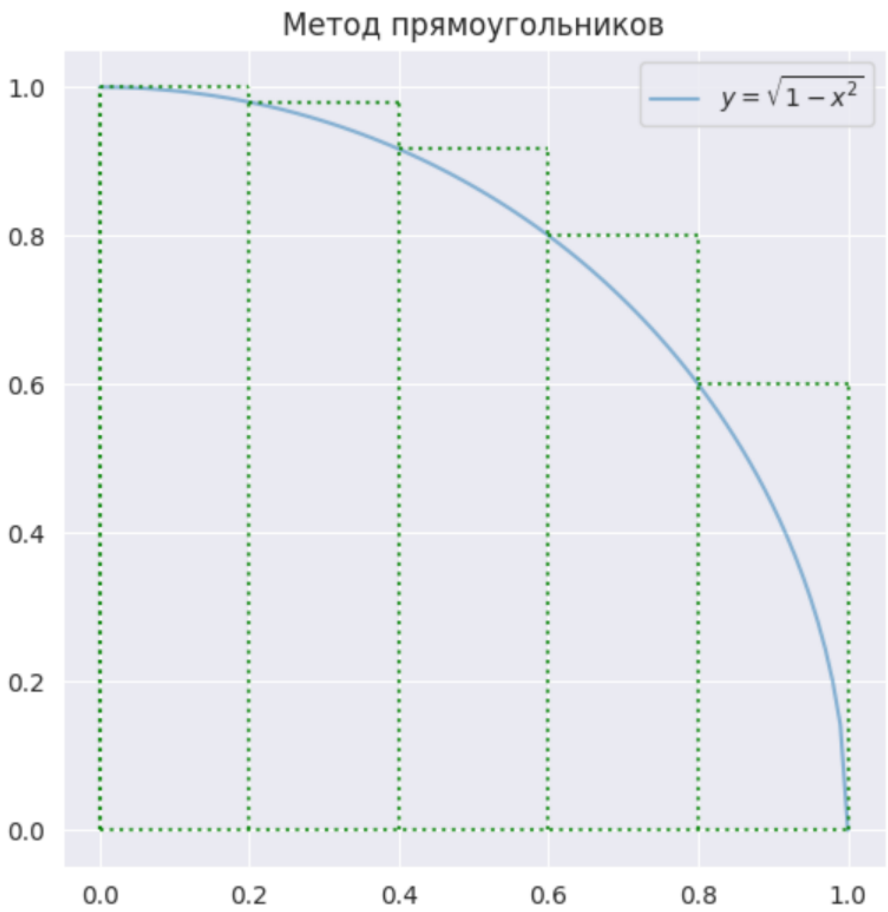

Рис. 1: Метод левых прямоугольников

$$\int_a^b f(x)\,dx \approx f(a) \times (b - a) \tag{9}$$

Тогда из 8, заменив интеграл суммой, имеем:

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

Подробнее метод описан в [2].

5 Расчёты

Для вычисления (10) напишем 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).

```python
1 def pi_rectangles(num_intervals: int) -> tuple[np.float64, np.float64]:
2 
3     if num_intervals == 0:
4         return np.NaN, np.NaN
5 
6     dx = np.double(1.0) / num_intervals
7     x = np.linspace(0, 1, num_intervals, endpoint=False)
8     y = np.sqrt(1 - x**2)
9     ds = y * dx
10    pi = np.sum(ds) * 4
11    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	**3.1**60417031779045	0.018824378189252
1000	**3.14**3555466911028	0.001962813321235
10000	**3.141**791477611323	0.000198824021529
100000	**3.141**612616401987	0.000019962812194
1000000	**3.14159**4652413811	0.000001998824017

На рисунке 2 ось ординат для наглядности выбрана в логарифмическом масштабе. Из графика в частности видно, что для вычисления значения π выбранным методом с точностью до пятого знака (10^{-5}) необходимо выбрать $n \geq 2 \times 10^5$.

7 Заключение

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

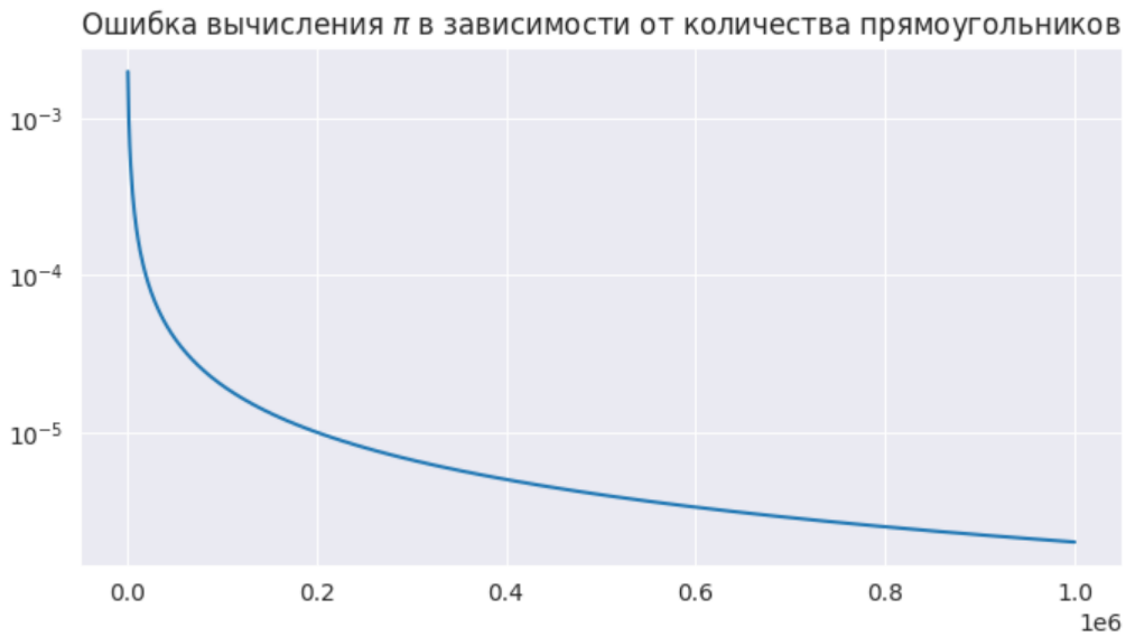

Рис. 2: Ошибка вычислений как функция количества интервалов интегрирования

8 Приложение. Код построения графика

```
import matplotlib.pyplot as plt
import seaborn as sns

# Значения функции
x = np.linspace(0, 1, 100, endpoint=True)
y = np.sqrt(1 - x**2)

# Значения функции для прямоугольников
xr = np.linspace(0, 1, 5, endpoint=False)
yr = np.sqrt(1 - xr**2)

# Построение графика
fig, ax = plt.subplots(figsize=(6, 6))
sb_ax = sns.lineplot(y=y, x=x,alpha=0.5, label="$y = \sqrt{1 - x^2}$")
sb_ax.set(title = 'Метод прямоугольников')
ax.set_xlim(-0.05,1.05)
ax.set_ylim(-0.05,1.05)

# Построение прямоугольников
sb_ax.hlines(
    y=yr,
    xmin=xr[:],
    xmax=[*xr[1:], 1.0],
    colors='green',
    linestyles='dotted',
    alpha=0.8,
)
sb_ax.vlines(
    x=[*xr, 1.0],
    ymin=0,
    ymax=[1.0, *yr[:]],
```

```
            colors='green',
            linestyles='dotted',
            alpha=0.8,
        )
        ax.vlines(
            x=0.0,
            ymin=0,
            ymax=1.0,
            colors='green',
            linestyles='dotted',
            alpha=0.8,
        )
        ax.hlines(
            y=0.0,
            xmin=0,
            xmax=1.0,
            colors='green',
            linestyles='dotted',
            alpha=0.8,
        )
```

Листинг 2: Код построения рис.1

Список иллюстраций

Список таблиц

Список литературы

[1] Википедия. Число Пи

[2] Википедия. Метод прямоугольников

[3] Библиотека Numerical Python (NumPy). numpy.org

[4] Графическая библиотека Seaborn. seaborn.pydata.org