ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Университет ИТМО

Статистика для анализа данных

Отчёт по лабораторной работе № 1

«Расчет геометрической вероятности»

Выполнили работу:

Семёнов Никита Викторович

Торши Ромдхан

ИСУ: 467414

ИСУ: 467746

Академическая группа:

№ J3113

Санкт-Петербург 2025

#### **Цель работы**

#### Предложить среднеквадратическую аппроксимацию табличной функции многих переменных, проанализировать чувствительность точного решения к ошибкам округления, проверить сходимость расчетных и исходных данных.

**Ход выполнения работы**

 **Выбор метода:**

**Метод Монте-Карло** — это численный метод, основанный на использовании случайных чисел и многократных испытаниях для получения статистических результатов.

**Суть метода Монте-Карло** заключается в приближенном решении вероятностных задач путем многократного случайного моделирования.

 **Выбор параметров:**

* Размер квадрата: a = 1, сторона 2a = 2.
* Радиусы кругов: r ∈ {1.0,0.5,13,0.25,0.2}

 **Расчет истинной вероятности p:**

* Формула: ​.
* Полученные значения:
  + r=1.0:  p≈0.7854
  + r=0.5:  p≈0.1963
  + r≈0.333: p≈0.0873
  + r=0.25:  p≈0.0491
  + r=0.2:  p≈0.0314

 **Генерация случайных точек:**

* Использован метод numpy.random.Generator.uniform.
* Количество точек n варьируется от 1 до 10 000.

 **Построение графиков:**

* График зависимости n).
* График зависимости .

 **Анализ точности:**

* Для каждого r найдено минимальное количество точек N, необходимое для достижения заданной точности .

**Основная часть**

##### **Код программы**

**Main.py**

import numpy as np

import matplotlib.pyplot as plt

a = 1

n\_points = 100000

**# Генерация случайных точек в квадрате [-a, a] x [-a, a]**

rng = np.random.default\_rng()

points = rng.uniform(-a, a, size=(n\_points, 2))

**# Проверка принадлежности кругу**

r\_values = [1.0, 0.5, 1/3, 0.25, 0.2]

for r in r\_values:

in\_circle = np.sum(points[:, 0]\*\*2 + points[:, 1]\*\*2 <= r\*\*2)

p\_hat = in\_circle / n\_points

print(f"r = {r:.3f}, p\_hat = {p\_hat:.5f}, p = {np.pi \* r\*\*2 / 4}")

**lab1.py**

import numpy as np

import matplotlib.pyplot as plt

**# Параметры**

a = 1

r\_values = [1.0, 0.5, 1/3, 0.25, 0.2]

n\_max = 10000

epsilon\_targets = [1e-1, 1e-2, 1e-3, 1e-4]

**# Генерация точек и расчет p\_hat**

rng = np.random.default\_rng()

n\_values = np.arange(1, n\_max + 1, 100)

for r in r\_values:

p\_true = np.pi \* r\*\*2 / 4

p\_hat\_values = []

epsilon\_values = []

for n in n\_values:

points = rng.uniform(-a, a, size=(n, 2))

in\_circle = np.sum(points[:, 0]\*\*2 + points[:, 1]\*\*2 <= r\*\*2)

p\_hat = in\_circle / n

p\_hat\_values.append(p\_hat)

epsilon\_values.append(abs(p\_hat - p\_true))

**# График p\_hat(n)**

plt.figure()

plt.plot(n\_values, p\_hat\_values, label='$\hat{p}(n)$')

plt.axhline(y=p\_true, color='r', linestyle='--', label='$p$')

plt.xlabel('Количество точек, $n$')

plt.ylabel('$\hat{p}$')

plt.legend()

plt.title(f'Оценка вероятности для $r = {r}$')

plt.grid(True)

plt.show()

**# График epsilon(n)**

plt.figure()

plt.plot(n\_values, epsilon\_values, label='$\epsilon(n)$')

plt.xlabel('Количество точек, $n$')

plt.ylabel('$\epsilon$')

plt.legend()

plt.title(f'Ошибка оценки для $r = {r}$')

plt.grid(True)

plt.show()

**# Расчет N(epsilon)**

N\_results = {r: [] for r in r\_values}

for r in r\_values:

p\_true = np.pi \* r\*\*2 / 4

for epsilon in epsilon\_targets:

n = 1

while True:

points = rng.uniform(-a, a, size=(n, 2))

in\_circle = np.sum(points[:, 0]\*\*2 + points[:, 1]\*\*2 <= r\*\*2)

p\_hat = in\_circle / n

if abs(p\_hat - p\_true) <= epsilon:

N\_results[r].append(n)

break

n += 1

**# График N(epsilon)**

plt.figure()

for r in r\_values:

plt.plot(epsilon\_targets, N\_results[r], 'o-', label=f'$r = {r}$')

plt.xscale('log')

plt.yscale('log')

plt.xlabel('Точность, $\epsilon$')

plt.ylabel('Необходимое $N$')

plt.legend()

plt.title('Зависимость $N(\epsilon)$ для разных $r$')

plt.grid(True)

plt.show()

**Графики и Анализ**

r = 1.000, p\_hat = 0.78394, p = 0.7853981633974483

r = 0.500, p\_hat = 0.19492, p = 0.19634954084936207

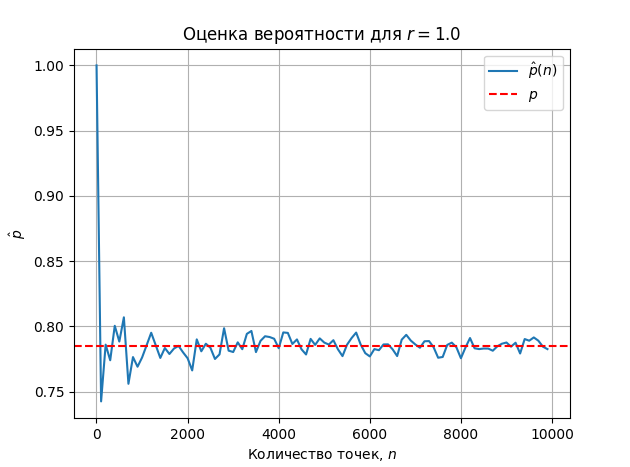
r = 0.333, p\_hat = 0.08604, p = 0.08726646259971647

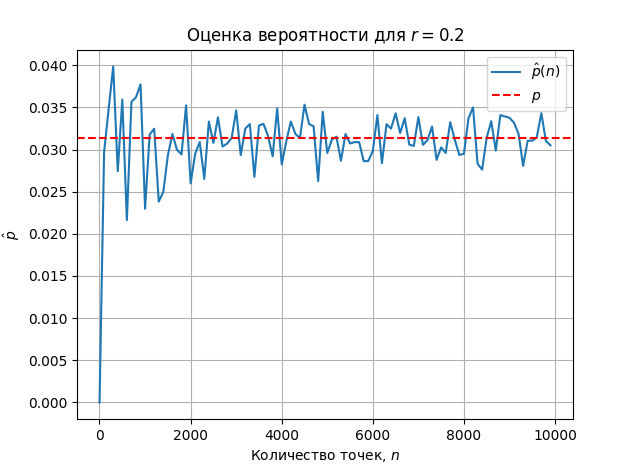
r = 0.250, p\_hat = 0.04861, p = 0.04908738521234052

r = 0.200, p\_hat = 0.03095, p = 0.031415926535897934

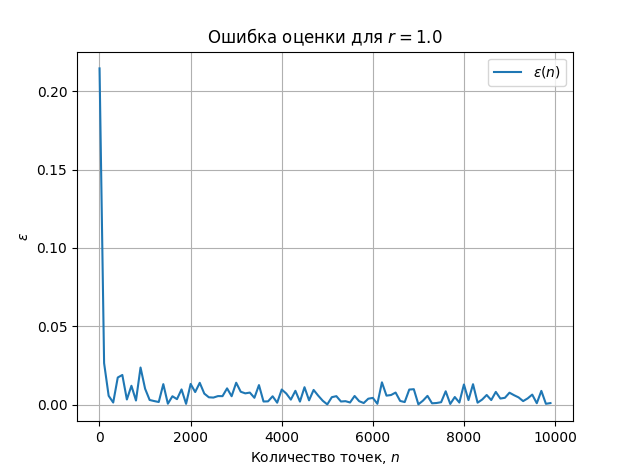
Реальная вероятность попадания точки в круг **p** приблизительно равна вероятности, посчитанной на тестах **p\_hat**.

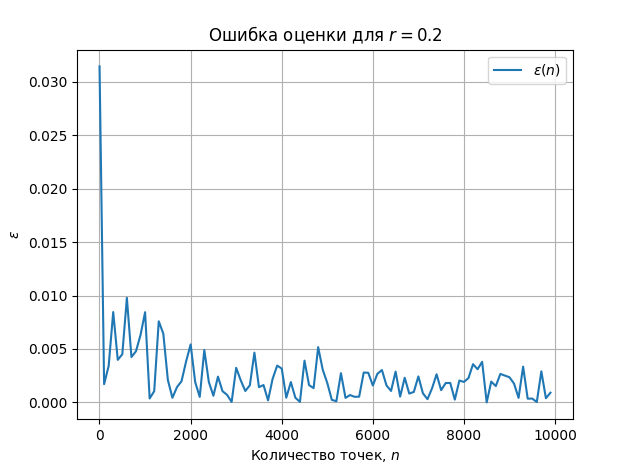
**р** – вычисляется путём деления площади круга на площадь квадрата.

****



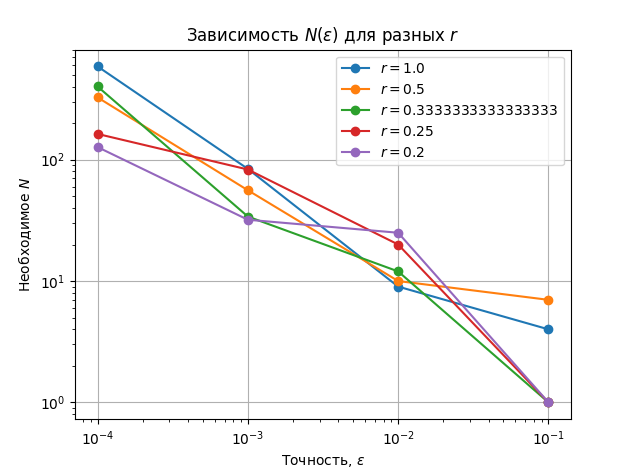
Видно, что оценка **p\*(n)** стремится к истинному значению **p** при увеличении **n**

****

****

Ошибка **(n)** уменьшается с ростом **n**, что подтверждает сходимость оценки.

Так же мы видим, что у меньшего **r** больший разброс. Это возникает из-за того что в маленький круг попадает меньше точек, из-за чего каждая точка оказывает большее влияние на график. А при большом **r** случайность усредняется, так как точек больше и каждая влияет меньше.

****

Чем меньше , тем больше точек **N** требуется для достижения заданной точности.Увеличение  **-** увеличение точности.

**Вывод**

1. Оценка **p\*(n)** сходится к истинной вероятности **p** при увеличении **n**.
2. Ошибка  **(n)** уменьшается с ростом **n**.
3. Для малых **r** требуется больше точек для достижения той же точности, так как вероятность попадания в круг ниже.
4. Зависимость **N()** имеет логарифмический характер: чем выше точность, тем больше испытаний нужно.

В ходе работы мы убедились, что метод Монте-Карло позволяет оценить геометрическую вероятность через отношение площадей, причем точность оценки растет с увеличением количества испытаний.

Метод прост в реализации, но требует значительного числа точек для достижения высокой точности, особенно для малых вероятностей.