Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Лабораторная работа №1 по дисциплине “Статистика и анализ данных ”

Семестр II

Выполнили:
студенты
Андреева Наталия Леонидовна
гр. J3110
ИСУ 465004
Ширкунова Мария Михайловна
гр. J3111
ИСУ 468094

Отчет сдан:
23.03.2025

Санкт-Петербург
2025

Содержание

1 Ход выполнения работы

В ходе лабораторной работы были выполнены следующие шаги:

1. Выбор значений радиусов кругов.
2. Расчёт истинной геометрической вероятности.
3. Генерация случайных точек и оценка вероятности.
4. Построение графиков $\hat{p}(n)$ и $\epsilon(n)$.
5. Расчёт необходимого количества точек N для заданной точности ϵ.
6. Построение графиков $N(\epsilon)$.

Подробное описание выполнения этих шагов представлено в следующем пункте.

2 Основная часть

2.1 Выбор значений радиусов кругов

Было выбрано 5 значений радиусов кругов r из интервала $(0, 2a]$, где $a = 2$. Значения радиусов были получены по формуле $r_k = \frac{a}{k+1}$, где $k = 1, 2, 3, 4, 5$.

2.2 Расчет истинной геометрической вероятности

Для каждого значения радиуса r была рассчитана истинная геометрическая вероятность p по формуле:

$$p = \frac{\text{Площадь круга}}{\text{Площадь квадрата}} = \frac{\pi r^2}{(2a)^2}.$$

2.3 Генерация случайных точек и оценка вероятности

Для каждого значения радиуса r:

1. С помощью генератора случайных чисел `numpy.random.default_rng()` были сгенерированы координаты (x, y) для $n = 1000$ точек в квадрате Ω со стороной $2a$.
2. Для каждой точки проверялась принадлежность кругу $A(r)$ по условию:

$$x^2 + y^2 \leq r^2.$$

3. Была рассчитана доля точек $\hat{p}(n)$, попавших в круг, как отношение числа точек внутри круга к общему количеству точек n.

2.4 Построение графиков $\hat{p}(n)$ и $\epsilon(n)$

Для каждого значения радиуса r были построены графики:

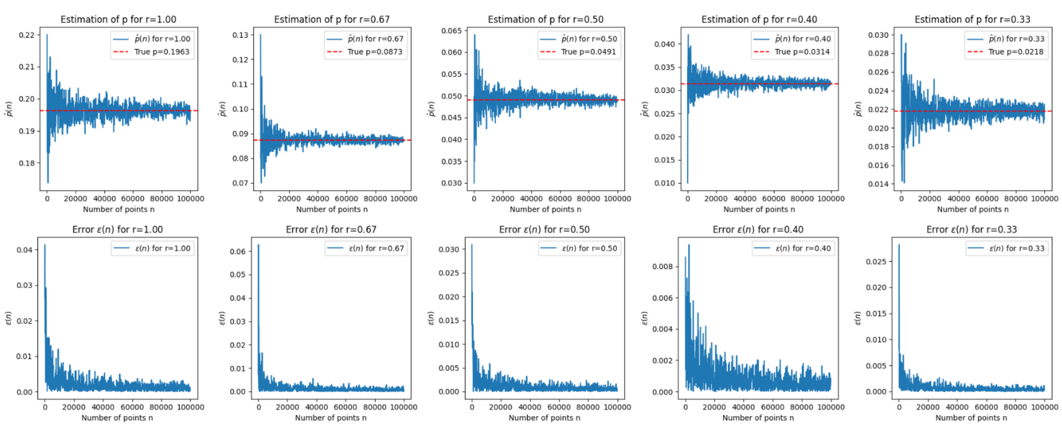

Рис. 1: Графики $\hat{p}(n), \epsilon(n)$ для различных радиусов

2.5 Расчет необходимого количества точек N для заданной точности ϵ_i

Для каждого значения радиуса r и заданной точности ϵ_i (где $\epsilon_i \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$):

1. Было выполнено моделирование с увеличением количества точек n до тех пор, пока ошибка $\epsilon(n) = |\hat{p}(n) - p|$ не станет меньше ϵ_i.
2. Зафиксировано количество точек N, необходимое для достижения заданной точности.

2.6 Построение графиков $N(\epsilon)$

Для каждого значения радиуса r был построен график зависимости количества точек N от точности ϵ:

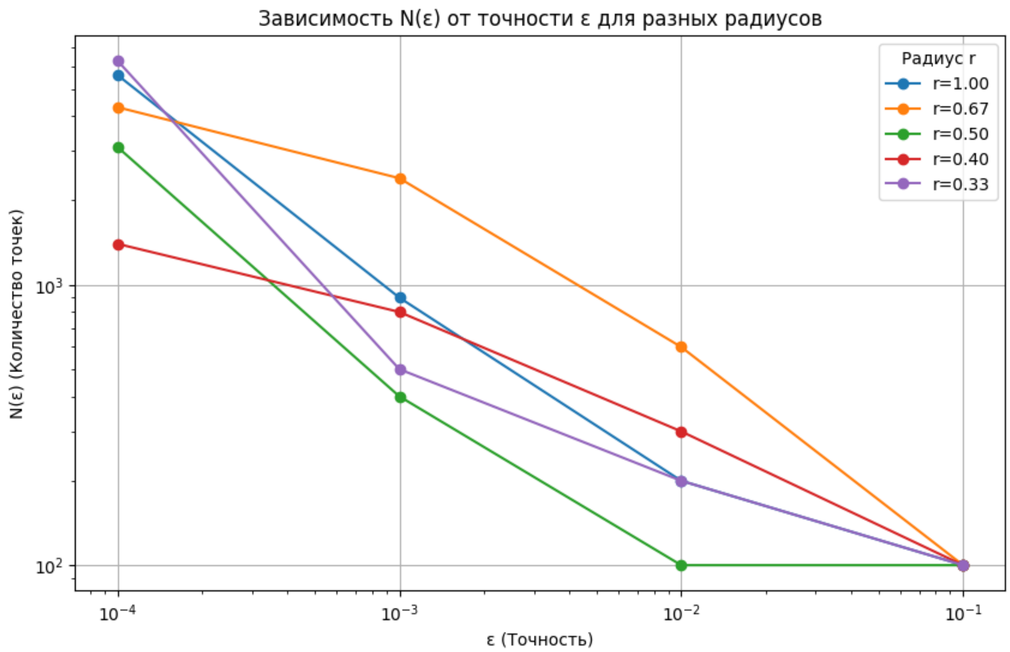

Рис. 2: График $N(\epsilon)$ для различных радиусов

3 Заключение

1. **Эффективность метода Монте-Карло**: Результаты работы подтверждают, что метод Монте-Карло является эффективным инструментом для оценки геометрической вероятности. Оценка $\hat{p}(n)$ сходится к истинной вероятности p с увеличением количества точек n.

2. **Зависимость ошибки от количества точек**: Ошибка $\epsilon(n)$ уменьшается с увеличением количества точек n. Это подтверждает теоретические ожидания и демонстрирует устойчивость метода.

3. **Зависимость N от точности ϵ**: Для достижения высокой точности ϵ требуется значительно больше точек N. Это особенно важно при работе с малыми значениями ϵ.

4 Листинг кода

```
import numpy as np
import matplotlib.pyplot as plt

# Параметры
a = 2
n_values = np.arange(100, 100000, 100)
epsilon_values = [0.1, 0.01, 0.001, 0.0001]
radii = [a/(k+1) for k in range(1, 6)]

# Функция для расчета истинной вероятности
def true_probability(r, a):
    return (np.pi * r**2) / (4 * a**2)

# Функция для генерации точек и расчета доли попавших в круг
def estimate_probability(r, a, n):
    x = np.random.uniform(-a, a, n)
    y = np.random.uniform(-a, a, n)
    inside = (x**2 + y**2) <= r**2
    return np.mean(inside)

# Создаем фигуру и оси для 5 графиков в строке
fig, axes = plt.subplots(1, 5, figsize=(20, 4))  # 1 строка, 5 столбцов

# Построение графиков для каждого значения радиуса
for i, r in enumerate(radii):
    p = true_probability(r, a)
    p_hat = [estimate_probability(r, a, n) for n in n_values]
    epsilon = [abs(ph - p) for ph in p_hat]

    # График p(n)
    axes[i].plot(n_values, p_hat, label=f'$\hat{{p}}(n)$ for r={r:.2f}')
    axes[i].axhline(y=p, color='r', linestyle='--', label=f'True p={p:.4f}')
    axes[i].set_xlabel('Number of points n')
    axes[i].set_ylabel('$\hat{p}(n)$')
    axes[i].legend()
    axes[i].set_title(f'Estimation of p for r={r:.2f}')

plt.tight_layout()  # Автоматическая настройка расстояний между графиками
plt.show()

# Аналогично для графиков ошибок (n)
fig, axes = plt.subplots(1, 5, figsize=(20, 4))

for i, r in enumerate(radii):
    p = true_probability(r, a)
    p_hat = [estimate_probability(r, a, n) for n in n_values]
    epsilon = [abs(ph - p) for ph in p_hat]

    # График (n)
    axes[i].plot(n_values, epsilon, label=f'$\epsilon(n)$ for r={r:.2f}')
    axes[i].set_xlabel('Number of points n')
    axes[i].set_ylabel('$\epsilon(n)$')
    axes[i].legend()
    axes[i].set_title(f'Error $\epsilon(n)$ for r={r:.2f}')
```

```
plt.tight_layout()
plt.show()

# Создаем фигуру для объединенного графика
plt.figure(figsize=(10, 6))

# Для каждого значения радиуса вычисляем N() и строим график
for r in radii:
    p = true_probability(r, a)
    N_epsilon = []
    for epsilon in epsilon_values:
        n = 100
        while True:
            p_hat = estimate_probability(r, a, n)
            if abs(p_hat - p) <= epsilon:
                N_epsilon.append(n)
                break
            n += 100

    # Рисуем линию для текущего радиуса
    plt.plot(epsilon_values, N_epsilon, marker='o', label=f'r={r:.2f}')

# Настройки графика
plt.xlabel(' (Точность)')
plt.ylabel('N() (Количество точек)')
plt.title('Зависимость N() от точности  для разных радиусов')
plt.legend(title='Радиус r')
plt.grid(True)
plt.xscale('log')  # Логарифмическая шкала для оси X
plt.yscale('log')  # Логарифмическая шкала для оси Y
plt.show()
```