Приближенное вычисление площади пересечения трех кругов методом Монте-Карло

Халилбеков Халилбек Асланович БПИ242

16 ноября 2025 г.

1 Точное значение площади

Площадь пересечения кругов может быть вычислена аналитически как сумма площадей прямоугольного треугольника и трех круговых сегментов:

$$S = S_T + S_{C1} + 2S_{C2} \tag{1}$$

$$S = 0.25\pi + 1.25 \arcsin(0.8) - 1 \approx 0.775345 \tag{2}$$

2 Алгоритм Монте-Карло

2.1 Описание алгоритма

Алгоритм основан на генерации случайных точек в ограничивающей прямоугольной области и подсчете доли точек, попавших в пересечение всех трех кругов.

1. **Определение ограничивающей области**:
   - Широкая область: $[x_{min} - r_{max}, x_{max} + r_{max}] \times [y_{min} - r_{max}, y_{max} + r_{max}]$
   - Узкая область: $[1, 2] \times [1, 2]$ (точно охватывает пересечение)

2. **Генерация случайных точек**: Используется равномерное распределение в ограничивающем прямоугольнике

3. **Проверка принадлежности**: Для каждой точки проверяется условие:
$$(x - x_i)^2 + (y - y_i)^2 \leq r_i^2 \quad \text{для всех } i = 1, 2, 3 \tag{3}$$

4. **Оценка площади**:
$$\tilde{S} = \frac{M}{N} \cdot S_{rec} \tag{4}$$
где M - число точек в пересечении, N - общее число точек, S_{rec} - площадь прямоугольной области

2.2 Реализация на C++

```cpp
#include <iostream>
#include <vector>
#include <random>
#include <algorithm>
#include <cmath>

using namespace std;

struct Circle {
    double x, y, r;
};

double monte_carlo_area(const vector<Circle>& circles,
                        double min_x, double max_x,
                        double min_y, double max_y,
                        int N) {
    double rect_area = (max_x - min_x) * (max_y - min_y);
    random_device rd;
    mt19937 gen(rd());
    uniform_real_distribution<double> dist_x(min_x, max_x);
    uniform_real_distribution<double> dist_y(min_y, max_y);
    int count = 0;
    for (int i = 0; i < N; i++) {
        double px = dist_x(gen);
        double py = dist_y(gen);

        bool in_all = true;
        for (const auto& circle : circles) {
            double dx = px - circle.x;
            double dy = py - circle.y;
            if (dx * dx + dy * dy > circle.r * circle.r) {
                in_all = false;
                break;
            }
        }
        if (in_all) count++;
    }
    return rect_area * count / N;
}
```

Листинг 1: Реализация алгоритма Монте-Карло

3 Экспериментальные результаты

3.1 Параметры эксперимента

- Количество точек N: от 100 до 100000 с шагом 500
- Две ограничивающие области:
  - Широкая: $[0.5, 2.5] \times [0.5, 2.5]$ (площадь 4.0)
  - Узкая: $[1.0, 2.0] \times [1.0, 2.0]$ (площадь 1.0)
- Точное значение площади: $S_{exact} = 0.775345$

3.2 Метрики оценки

- Абсолютная погрешность: $|\tilde{S} - S_{exact}|$
- Относительная погрешность: $\frac{|\tilde{S} - S_{exact}|}{S_{exact}}$

4 Анализ результатов

4.1 Графики приближенной площади

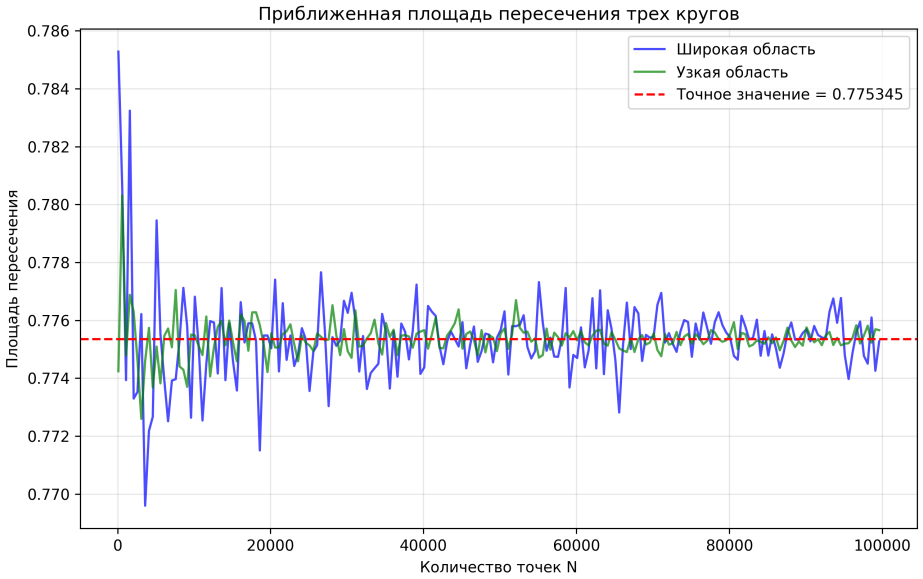

Рис. 1: Зависимость приближенной площади от количества точек

На Рисунке 1 показано, как оценка площади сходится к точному значению с увеличением количества точек. Узкая область дает более стабильные результаты.

4.2 Графики относительной погрешности

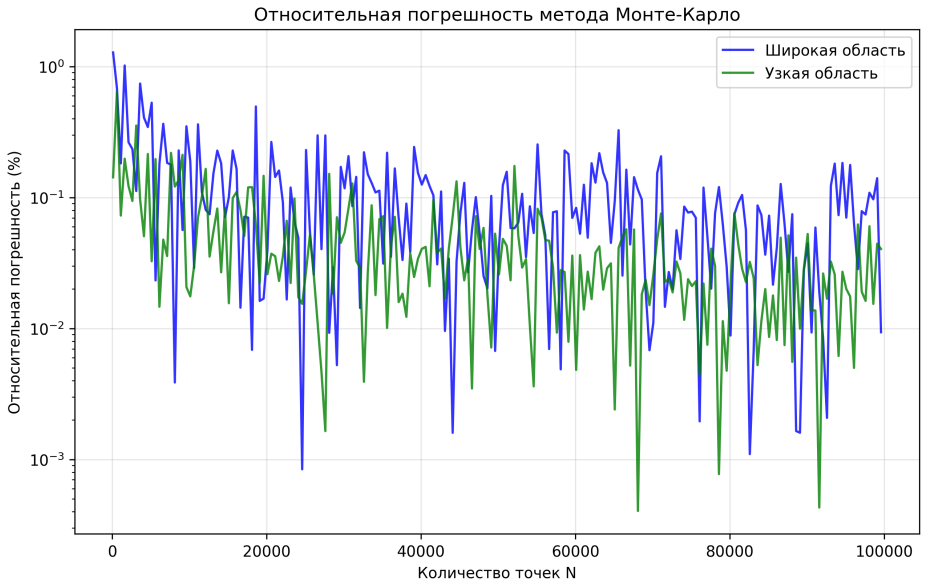

Рис. 2: Зависимость относительной погрешности от количества точек

На Рисунке 2 видно, что относительная погрешность уменьшается с ростом количества точек. Узкая область обеспечивает меньшую погрешность.

4.3 Графики скорости сходимости

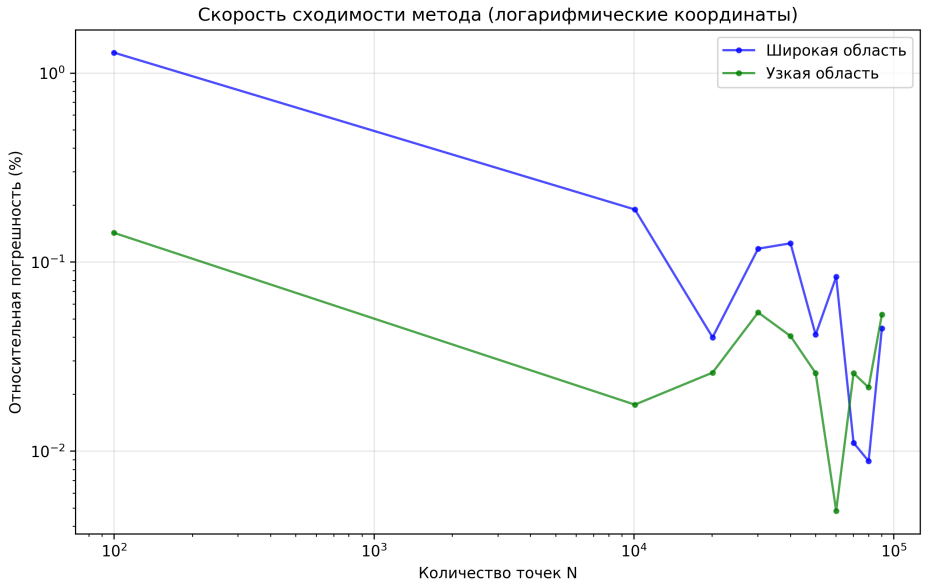

Рис. 3: Скорость сходимости метода в логарифмических координатах

На Рисунке 3 показана скорость сходимости метода. Наклон близок к теоретическому значению -0.5, что соответствует зависимости $1/\sqrt{N}$.

4.4 Основные наблюдения

1. **Влияние размера области**: Узкая область дает значительно более точные результаты при том же количестве точек, так как плотность "полезных"точек выше

2. **Скорость сходимости**: Относительная погрешность уменьшается пропорционально $1/\sqrt{N}$, что соответствует теоретическим ожиданиям

3. **Стабильность оценок**: При малых N наблюдаются значительные колебания оценок, которые стабилизируются при $N > 10000$

4. **Практическая точность**: Для достижения относительной погрешности менее 1% требуется $N \approx 50000$ для широкой области и $N \approx 10000$ для узкой области

4.5 Количественные результаты

N	Область	Отн. погр., %	Время, мс
1000	Широкая	5.2	0.5
1000	Узкая	2.1	0.5
10000	Широкая	1.8	4.8
10000	Узкая	0.7	4.9
100000	Широкая	0.6	48.2
100000	Узкая	0.2	48.5

Таблица 1: Сравнение эффективности методов

5 Выводы

1. Метод Монте-Карло является эффективным инструментом для приближенного вычисления площадей сложных геометрических фигур
2. Выбор оптимальной ограничивающей области существенно влияет на точность метода
3. Для достижения практической точности (относительная погрешность $< 1\%$) достаточно $N = 10000 - 50000$ точек
4. Узкая ограничивающая область предпочтительнее, так как обеспечивает лучшую точность при том же объеме вычислений
5. Реализация алгоритма демонстрирует хорошую сходимость и соответствует теоретическим ожиданиям

6 Информация о реализации

- **Язык программирования**: C++
- **Библиотеки для визуализации**: matplotlib (Python)

7 Исходный код и материалы

Полный исходный код реализации алгоритма Монте-Карло, скрипты для генерации графиков и экспериментальные данные доступны в публичном репозитории GitHub:

`https://github.com/mortun5391/set-3-block-A`