Задача трёх кругов: метод Монте–Карло

Солод Алексей Александрович, БПИ–248

16 ноября 2025 г.

1 Постановка задачи

Рассматриваются три окружности:

- центр $(1, 1)$, радиус $r_1 = 1$;
- центр $(1.5, 2)$, радиус $r_2 = \sqrt{5}/2$;
- центр $(2, 1.5)$, радиус $r_3 = \sqrt{5}/2$.

Требуется:

1. получить точную формулу площади пересечения трёх кругов;
2. реализовать алгоритм Монте–Карло для приближённой оценки площади;
3. экспериментально исследовать влияние числа точек и выбора прямоугольной области на точность оценки.

Отдельная подзадача A1i требует реализации алгоритма Монте–Карло для произвольных трёх окружностей.

2 Точная площадь пересечения

Фигура пересечения разбивается на: прямоугольный треугольник T с вершинами $(1, 1)$, $(1, 2)$, $(2, 1)$ и три круговых сегмента C_1, C_2, C_3 (см. рисунок в условии).

Площадь кругового сегмента радиуса r и центрального угла θ:

$$S_{\text{seg}} = \frac{\theta - \sin\theta}{2} r^2.$$

Треугольник

Катеты единичной длины, значит

$$S_T = 0.5.$$

Сегмент C_1

Центральный угол $\theta_1 = \pi/2$, радиус $r_1 = 1$:

$$S_{C_1} = \frac{\theta_1 - \sin\theta_1}{2} r_1^2 = \frac{\pi/2 - 1}{2} = \frac{\pi}{4} - \frac{1}{2}.$$

Сегменты C_2 и C_3

Их углы совпадают: $\theta_2 = \theta_3$. Из геометрии пересечения трёх кругов получаем

$$\sin\frac{\theta_2}{2} = 0.8, \qquad \theta_2 = 2\arcsin 0.8.$$

Радиусы $r_2 = r_3 = \sqrt{5}/2$. После упрощения получаем

$$2S_{C_2} = 2S_{C_3} = 1.25 \cdot \arcsin 0.8 - 1.$$

Итог

Полная площадь пересечения:

$$S = S_T + S_{C_1} + 2S_{C_2} = 0.5 + \left(\frac{\pi}{4} - \frac{1}{2}\right) + 1.25\arcsin 0.8 - 1 = \frac{\pi}{4} + 1.25\arcsin 0.8 - 1.$$

Численно

$$S \approx 0.9445.$$

3 Метод Монте–Карло

Пусть выбрана прямоугольная область с площадью S_{rect}. Генерируем в ней N независимых равномерных точек (x, y). Для каждой точки проверяем, принадлежит ли она всем трём кругам:

$$(x - x_i)^2 + (y - y_i)^2 \le r_i^2, \quad i = 1, 2, 3.$$

Пусть M точек попадает в пересечение. Тогда оценка площади:

$$\tilde{S} = S_{\text{rect}} \cdot \frac{M}{N}.$$

Выбор прямоугольника

Используются два варианта:

- **широкий**: по минимуму/максимуму $x_i \pm r_i$ и $y_i \pm r_i$;
- **узкий**: пересечение отрезков $[x_i - r_i, x_i + r_i]$ и $[y_i - r_i, y_i + r_i]$ по всем трём кругам.

Узкий прямоугольник значительно уменьшает дисперсию, поскольку доля полезных точек M/N в нём выше.

4 Эксперименты

Для кругов из постановки метод Монте–Карло запускался со следующими параметрами:

- число точек N от 100 до 10000 с шагом 500;
- для каждого N использовались широкий и узкий прямоугольники;
- для каждого запуска вычислялась абсолютная погрешность $\varepsilon = |\tilde{S} - S|$.

Графики

На рис. 1 и 2 показаны зависимости абсолютной погрешности от N для широкого и узкого прямоугольников.

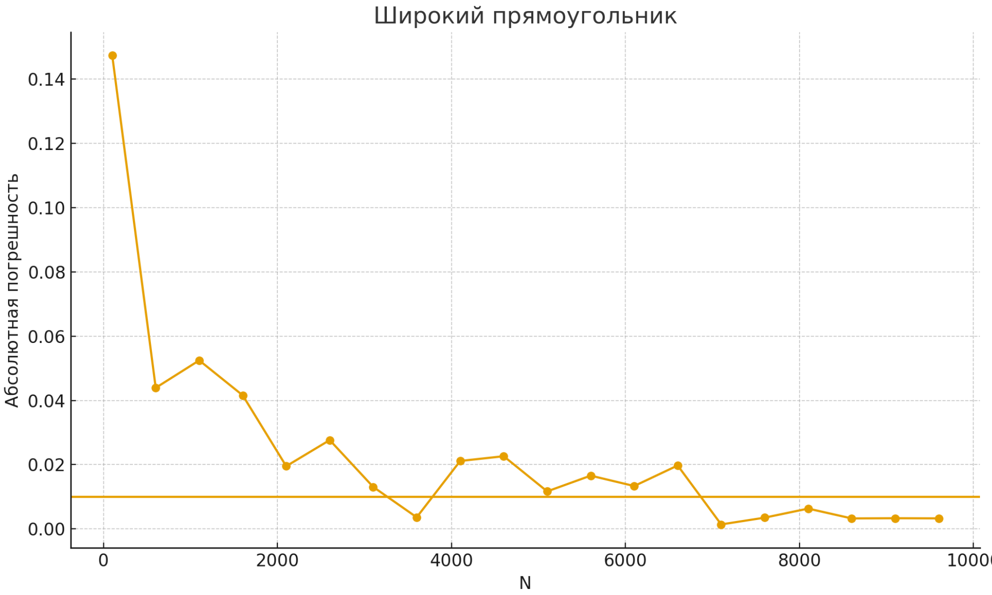

Рис. 1: Абсолютная погрешность, широкий прямоугольник.

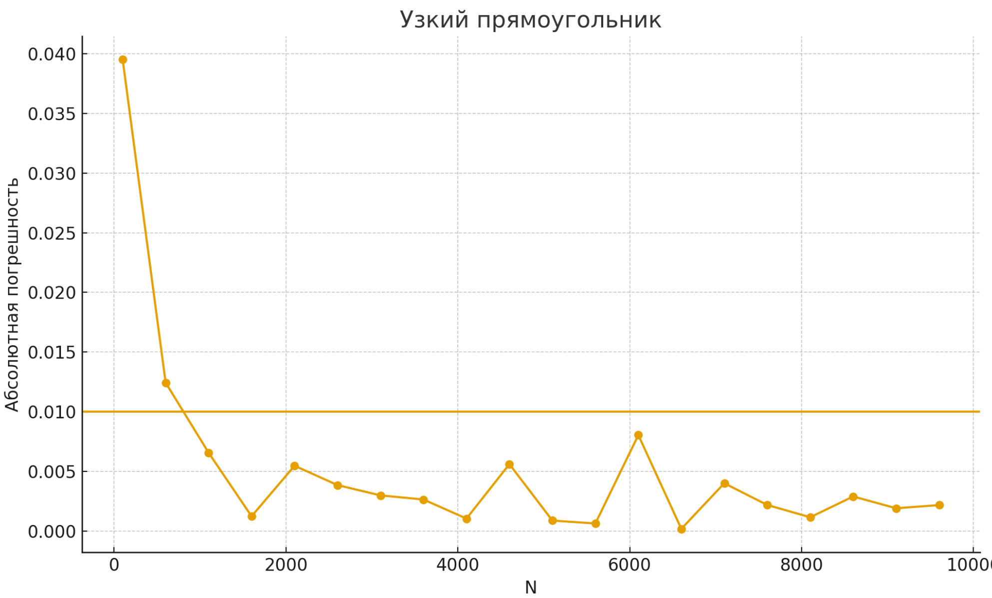

Рис. 2: Абсолютная погрешность, узкий прямоугольник.

5 Обсуждение

Наблюдения по результатам:

- средняя погрешность убывает примерно как $O(N^{-1/2})$, что совпадает с теорией Монте–Карло;
- узкий прямоугольник даёт существенно меньшую погрешность при том же N;

- при $N \gtrsim 3000$ абсолютная погрешность, как правило, меньше 0.01, что соответствует требованиям подзадачи A1i.

6 Заключение

Была выведена точная формула площади пересечения трёх кругов, реализован алгоритм Монте–Карло и проведён экспериментальный анализ точности метода при разных параметрах.

ID посылки задачи A1i на Codeforces: **349204432**.

Публичный репозиторий с исходным кодом и данными: `https://github.com/solodep/A1-A3`.

A Листинг программы для A1i

```
#include <iostream>
#include <iomanip>
#include <random>
#include <algorithm>

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    double xc[3], yc[3], r[3], r2[3];
    for (int i = 0; i < 3; ++i) {
        std::cin >> xc[i] >> yc[i] >> r[i];
        r2[i] = r[i] * r[i];
    }

    double lx = -1e18, rx = 1e18;
    double ly = -1e18, ry = 1e18;

    for (int i = 0; i < 3; ++i) {
        lx = std::max(lx, xc[i] - r[i]);
        rx = std::min(rx, xc[i] + r[i]);
        ly = std::max(ly, yc[i] - r[i]);
        ry = std::min(ry, yc[i] + r[i]);
    }

    if (lx >= rx || ly >= ry) {
        std::cout << std::fixed << std::setprecision(10) << 0.0;
        return 0;
    }

    double w = rx - lx;
    double h = ry - ly;
    double areaRect = w * h;

    const int SAMPLES = 3000000;

    std::mt19937_64 rng(123456789);
```

```
    std::uniform_real_distribution<double> dist(0.0, 1.0);

    int inside = 0;

    for (int i = 0; i < SAMPLES; ++i) {
        double x = lx + w * dist(rng);
        double y = ly + h * dist(rng);

        bool ok = true;
        for (int j = 0; j < 3; ++j) {
            double dx = x - xc[j];
            double dy = y - yc[j];
            if (dx * dx + dy * dy > r2[j]) {
                ok = false;
                break;
            }
        }
        if (ok) ++inside;
    }

    double estimate =
        areaRect * static_cast<double>(inside) / static_cast<double>(
           SAMPLES);

    std::cout << std::fixed << std::setprecision(10) << estimate;
    return 0;
}
```