

Ссылки на ресурсы

[Репозиторий A1](#)

Посылка: 349327518

Монте-Карло

```
1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4 #include <random>
5 #include <iomanip>
6 #include <algorithm>
7
8 struct Circle {
9     double x, y, r;
10 };
11 bool is_inside_intersection(double px, double py, const std::vector<Circle>& circles) {
12     for (const auto& c : circles) {
13         if (std::pow(px - c.x, 2) + std::pow(py - c.y, 2) > std::pow(c.r, 2)) {
14             return false;
15         }
16     }
17     return true;
18 }
19
20 int main() {
21     std::ios_base::sync_with_stdio(false);
22     std::cin.tie(NULL);
23     std::vector<Circle> circles(3);
24     for (int i = 0; i < 3; ++i) {
25         std::cin >> circles[i].x >> circles[i].y >> circles[i].r;
26     }
27     double min_x = circles[0].x - circles[0].r;
28     double max_x = circles[0].x + circles[0].r;
29     double min_y = circles[0].y - circles[0].r;
30     double max_y = circles[0].y + circles[0].r;
31
32     for (int i = 1; i < 3; ++i) {
```

```

33     min_x = std::min(min_x, circles[i].x - circles[i].r);
34     max_x = std::max(max_x, circles[i].x + circles[i].r);
35     min_y = std::min(min_y, circles[i].y - circles[i].r);
36     max_y = std::max(max_y, circles[i].y + circles[i].r);
37 }
38 const long long n_points = 2000000;
39 double box_area = (max_x - min_x) * (max_y - min_y);
40 long long points_inside = 0;
41
42 std::random_device rd;
43 std::mt19937 gen(rd());
44 std::uniform_real_distribution<> dis_x(min_x, max_x);
45 std::uniform_real_distribution<> dis_y(min_y, max_y);
46
47 for (long long i = 0; i < n_points; ++i) {
48     double rand_x = dis_x(gen);
49     double rand_y = dis_y(gen);
50     if (is_inside_intersection(rand_x, rand_y, circles)) {
51         points_inside++;
52     }
53 }
54
55 double estimated_area = (static_cast<double>(points_inside) / n_points) * box_area;
56 std::cout << std::fixed << std::setprecision(20) << estimated_area << std::endl;
57 return 0;
58 }
59

```

График 1: Зависимость приближенной площади от числа точек (N)

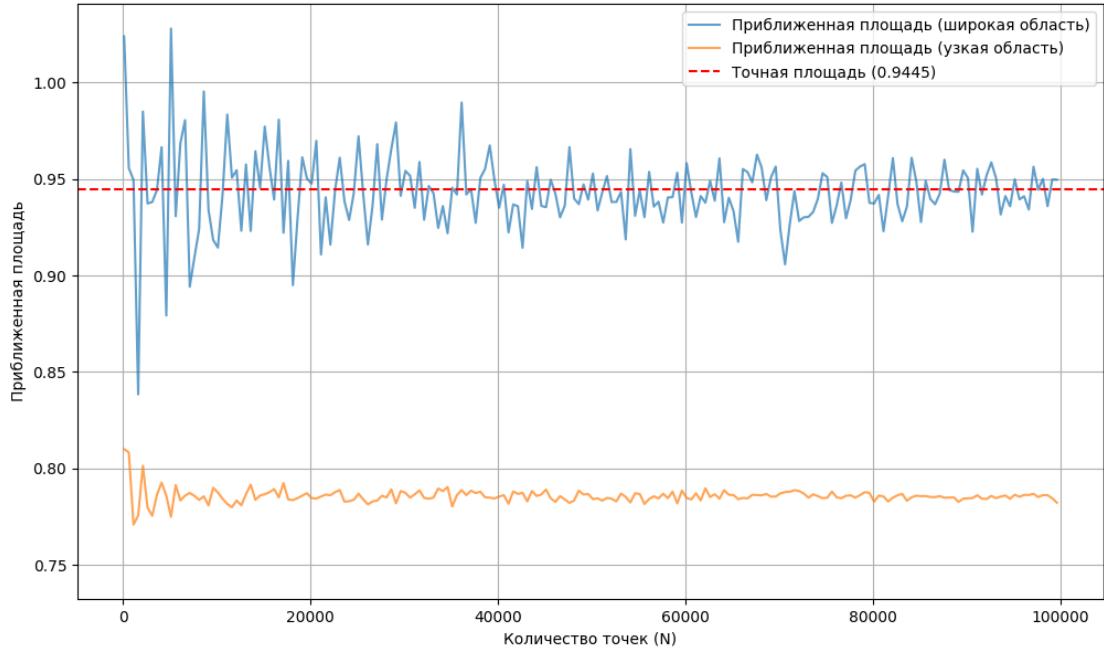


Рис. 1: График 1: Зависимость приближенного значения площади от числа точек N.

График 2: Зависимость относительной ошибки от числа точек (N)

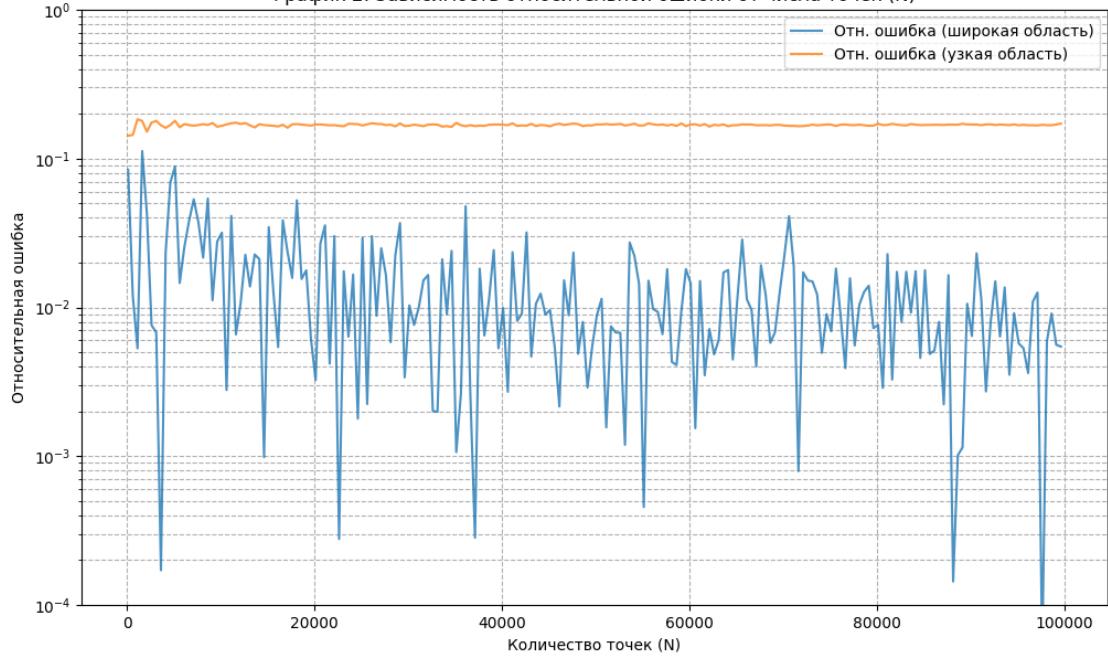


Рис. 2: График 2: Зависимость относительной ошибки от числа точек N .

Выводы

Для широкой области, полностью покрывающей фигуру, приближенное значение площади сходится к точному значению с ростом числа точек (N). Соответственно, относительная ошибка уменьшается, что подтверждает работоспособность метода.

Для узкой области, которая не покрывает всю фигуру, результат оказывается систематически неверным. Приближенная площадь стабильно занижена, а относительная ошибка остается высокой независимо от увеличения N .

Эксперимент доказывает, что ключевым условием точности метода Монте-Карло является выбор ограничивающей области, которая полностью содержит измеряемый объект. Увеличение числа итераций (N) позволяет снизить случайную погрешность, но бессильно против систематической ошибки, вызванной неправильно заданными границами.