A1. Задача трех кругов. Отчет

Головина Арина БПИ244
ID посылки: [349237694]
Ссылка на репозиторий: https://github.com/AMGolovina/Algorithm/tree/main/SET3/A1

Код задачи A1i (широкая прямоугольная область):

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

double monteCarlo(const std::vector<double>& A,
                  const std::vector<double>& B,
                  const std::vector<double>& C) {
   double x1 = A[0], y1 = A[1], r1 = A[2];
   double x2 = B[0], y2 = B[1], r2 = B[2];
   double x3 = C[0], y3 = C[1], r3 = C[2];

   double xmin = std::min(x1 - r1, x2 - r2);
   xmin = std::min(xmin, x3 - r3);

   double xmax = std::max(x1 + r1, x2 + r2);
   xmax = std::max(xmax, x3 + r3);

   double ymin = std::min(y1 - r1, y2 - r2);
   ymin = std::min(ymin, y3 - r3);

   double ymax = std::max(y1 + r1, y2 + r2);
   ymax = std::max(ymax, y3 + r3);

   double width  = xmax - xmin;
   double height = ymax - ymin;
   double rectArea = width * height;

   const int N = 1000000;

   std::random_device rd;
   std::mt19937_64 gen(rd());
   std::uniform_real_distribution<double> distX(xmin, xmax);
   std::uniform_real_distribution<double> distY(ymin, ymax);

   int cnt = 0;

   for (int i = 0; i < N; ++i) {
       double x = distX(gen);
       double y = distY(gen);
```

```
        double dx1 = x - x1, dy1 = y - y1;
        double dx2 = x - x2, dy2 = y - y2;
        double dx3 = x - x3, dy3 = y - y3;

        bool insideAll =
            (dx1 * dx1 + dy1 * dy1 <= r1 * r1) &&
            (dx2 * dx2 + dy2 * dy2 <= r2 * r2) &&
            (dx3 * dx3 + dy3 * dy3 <= r3 * r3);

        if (insideAll)
            ++cnt;
    }

    return rectArea * static_cast<double>(cnt) /
static_cast<double>(N);
}

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

    std::vector<double> A;
    for (int i = 0; i < 3; i++) {
        double x;
        std::cin >> x;
        A.push_back(x);
    }

    std::vector<double> B;
    for (int i = 0; i < 3; i++) {
        double x;
        std::cin >> x;
        B.push_back(x);
    }

    std::vector<double> C;
    for (int i = 0; i < 3; i++) {
        double x;
        std::cin >> x;
        C.push_back(x);
    }

    std::cout << monteCarlo(A, B, C);
}
```

Код задачи A1i с подставленными значения(узкая прямоугольная область):

```
#include <iostream>
```

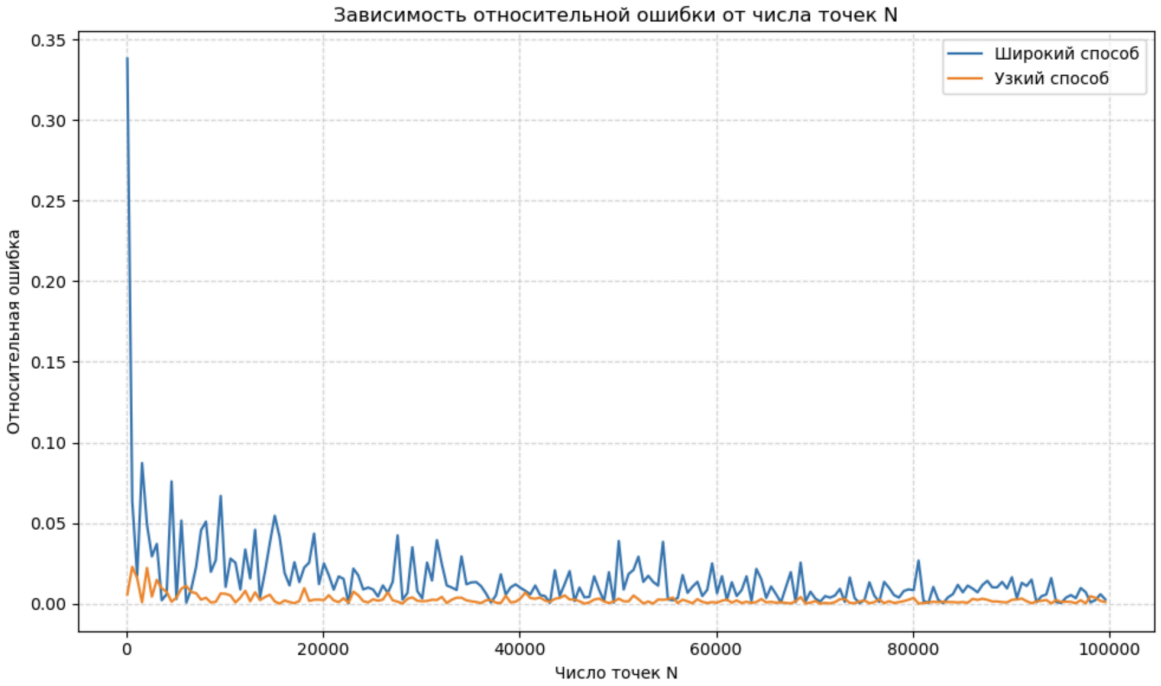
```cpp
#include <vector>
#include <random>
#include <algorithm>

double monteCarlo(const std::vector<double>& A,
                  const std::vector<double>& B,
                  const std::vector<double>& C) {
    double x1 = A[0], y1 = A[1], r1 = A[2];
    double x2 = B[0], y2 = B[1], r2 = B[2];
    double x3 = C[0], y3 = C[1], r3 = C[2];

    double xmin = std::max(x1 - r1, x2 - r2);
    xmin = std::max(xmin, x3 - r3);

    double xmax = std::min(x1 + r1, x2 + r2);
    xmax = std::min(xmax, x3 + r3);

    double ymin = std::max(y1 - r1, y2 - r2);
    ymin = std::max(ymin, y3 - r3);

    double ymax = std::min(y1 + r1, y2 + r2);
    ymax = std::min(ymax, y3 + r3);

    if (xmin >= xmax || ymin >= ymax)
        return 0.0;

    double width  = xmax - xmin;
    double height = ymax - ymin;
    double rectArea = width * height;

    const int N = 1000000;

    std::random_device rd;
    std::mt19937_64 gen(rd());
    std::uniform_real_distribution<double> distX(xmin, xmax);
    std::uniform_real_distribution<double> distY(ymin, ymax);

    int cnt = 0;

    for (int i = 0; i < N; ++i) {
        double x = distX(gen);
        double y = distY(gen);

        double dx1 = x - x1, dy1 = y - y1;
        double dx2 = x - x2, dy2 = y - y2;
        double dx3 = x - x3, dy3 = y - y3;

        bool insideAll =
```

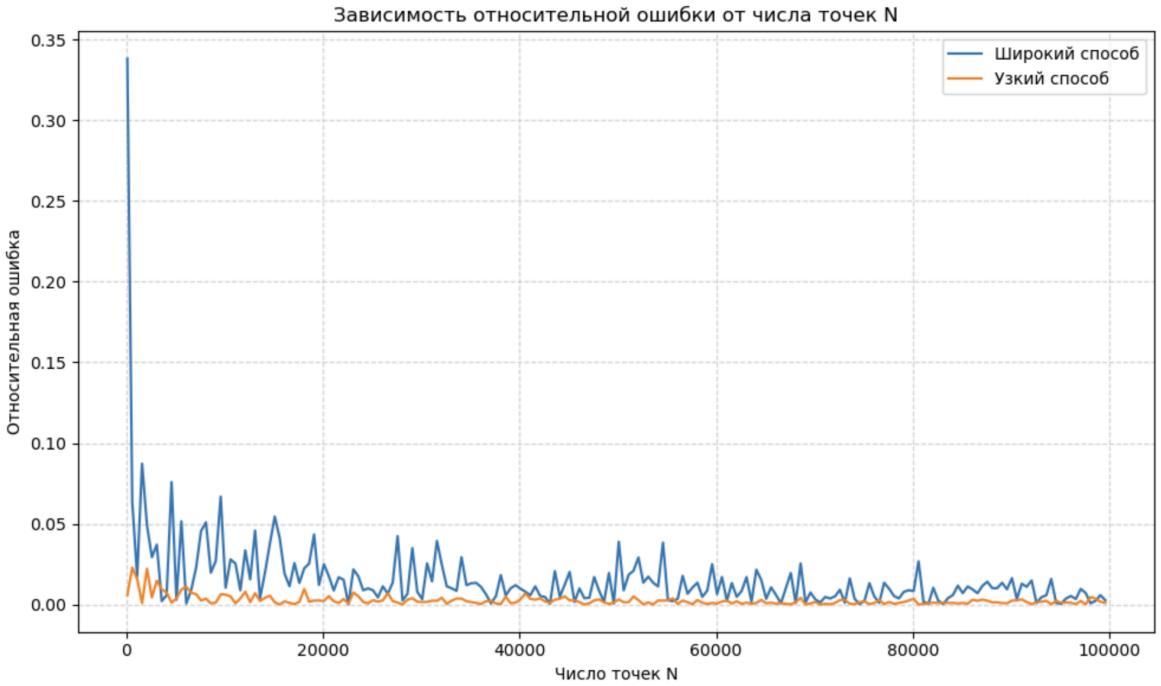
```
            (dx1 * dx1 + dy1 * dy1 <= r1 * r1) &&
            (dx2 * dx2 + dy2 * dy2 <= r2 * r2) &&
            (dx3 * dx3 + dy3 * dy3 <= r3 * r3);

        if (insideAll)
            ++cnt;
    }

    return rectArea * static_cast<double>(cnt) /
static_cast<double>(N);
}

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

    std::vector<double> A = {1.0, 1.0, 1.0};
    std::vector<double> B = {1.5, 2.0, std::sqrt(5.0) / 2.0};
    std::vector<double> C = {2.0, 1.5, std::sqrt(5.0) / 2.0};

    std::cout << monteCarlo(A, B, C);
}
```

Вывод для задачи из A1: 0.944903

Сравнительный анализ и построение графиков:

Data.WideWay - данные, полученные широким способом.
Data.NarrowWay - данные, полученные узким способом.





Выводы:

При увеличении числа точек N, приближенные значения стремятся к точному.
При меньшем числе точек, узкий метод дает более устойчивые значения.