МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт вычислительной математики и информационных технологий

Кафедра прикладной математики

ОТЧЕТ

ПО СЕМЕСТРОВОЙ РАБОТЕ

по дисциплине «Теория вероятностей»

Вариант № 14

Выполнил: студент гр. 09-022

Тепляков Н.А.

Проверил: доц. С.В. Симушкин

Казань 2021

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ:

Изучить метод Монте-Карло и с его помощью вычислить интеграл.

2 ИСХОДНЫЕ ДАННЫЕ И ЗАДАНИЕ:

С помощью метода случайного моделирования вычислить интеграл

$$\iint_{\Omega} h(x,y)dxdy,$$

где область $\Omega = \{(x,y): a \le x \le b, 0 \le y \le g(x)\}$ с некоторой функцией g.

$a = 2; b = 4$; $g_{(x)} = \sqrt{x-1}$; $h(x,y) = \frac{\sin(x+y)+\cos(x-y)}{(x+y)}$.

3 ХОД РАБОТЫ

3.1 Описание метода Монте-Карло

$$\iint_{\Omega} h(x,y)dxdy = S_{\Omega} \iint_{\Omega} h(x,y)\frac{1}{S_{\Omega}} I_{\Omega}(x,y)dxdy$$

где S_{Ω} – площадь Ω, и функция $f(x,y) = \frac{1}{S_{\Omega}} I_{\Omega}(x,y), (x,y) \in \Omega$, есть функция плотности равномерного на Ω распределения. Другими словами, интеграл совпадает (с точностью до множителя S_{Ω}) с математическим ожиданием относительно случайного вектора (ξ,η) с этим равномерным распределением:

$$\iint_{\Omega} h(x,y)dxdy = S_{\Omega}\, \mathbf{E}[h(\xi,\eta)].$$

Для приближённого вычисления математического ожидания применим закон больших чисел.

Теорема. Пусть $(\xi_1,\eta_1), (\xi_2,\eta_2), \dots, (\xi_n,\eta_n)$ – посл-ть независимых случайных векторов. Если $J = \iint_{\Omega} h(x,y)dxdy < \infty$, то при $n \to \infty$ среднее арифметическое

$$\frac{1}{n}\sum_{i=1}^{n} h(\xi_i,\eta_i) \xrightarrow{\mathbf{P}} J.$$

Таким образом, для вычисления интеграла достаточно сгенерировать большое число равномерных случайных векторов и найти среднее арифметическое всех значений функции h в этих точках.

Генерировать случайные точки в произвольной области сложно. Можно поступить следующим образом:

1. Охватить область Ω минимально узкой простой областью Ω' – прямоугольником.

2. Как известно, равномерное распределение в прямоугольнике (со сторонами, параллельными осям), можно задать с помощью равномерного распределения каждой компоненты, т.е. если $(x_1, \dots, x_n), (y_1, \dots, y_n)$ – реализации случайных величин с соответствующими равномерными распределениями (на сторонах прямоугольника), то пары $(x_1, y_1), \dots, (x_n, y_n)$ – суть реализации случайных величин с равномерным распределением внутри прямоугольника Ω'.

3. Оставим в наборе чисел $(x_1, y_1), \dots, (x_n, y_n)$ только те, которые попадают в Ω, т.е. $0 \le y_i \le g(x_i)$. Пусть их будет k штук: $(x_1{}', y_1{}'), \dots, (x_k{}', y_k{}')$

4. Среднее арифметическое

$$\frac{1}{k}\sum_{i=1}^{k} h(x_i{}', y_i{}')$$

по закону больших чисел будет оценкой для $\mathbf{E}[h(\xi, \eta)]$.

5. Отношение $\frac{k}{n}$ также по закону больших чисел есть оценка для вероятности попадания в область Ω. По определению равномерного распределения в Ω' вероятность попадания в любую область равна отношению площади области на площадь всего прямоугольника, т.е. $\frac{k}{n} \cdot S_{\Omega'}$ - оценка площади S_Ω.

6. Оценка для искомого интеграла J равна

$$\frac{1}{k}\sum_{i=1}^{k} h(x_i{}', y_i{}') \ \ \frac{k}{n} \cdot S_{\Omega'} = S_{\Omega'} \cdot \frac{1}{n}\sum_{i=1}^{k} h(x_i{}', y_i{}')$$

$$\xi \sim U(0,1) => \ \ (b-a) * \xi + a$$

3.2 Описание алгоритма

1. Выбрать минимальный прямоугольник, который охватит данную область;
2. Сгенерировать в этом прямоугольнике $(x_1, \dots, x_n), \dots, (y_1, \dots, y_n)$ случайные величины;
3. Отберём только те $(x_1, y_1), \dots, (x_n, y_n)$, которые попадают в Ω, т.е. $0 \le y_i \le g(x_i)$.
4. Найдём сумму $h(x_i, y_i)$ в отобранных точках (обозначим: $h(x_i{}', y_i{}')$).
5. Нахождение интеграла по формуле: $I = S_{\Omega'} \cdot \frac{1}{n}\sum_{i=1}^{k} h(x_i{}', y_i{}')$.

3.3 Пошаговая реализация алгоритма

Для решения задачи реализуем программу на C++, с подключением библиотеки GLFW спецификации OpenGL (для вывода результатов на экран).

1. Выбрать минимальный прямоугольник, который охватит данную область. Для этого найдём максимум функции $g(x)$. $max\ g(x) = \sqrt{3}$; где $x \in [2\,;4]$. Данный максимум будет высотой прямоугольника.

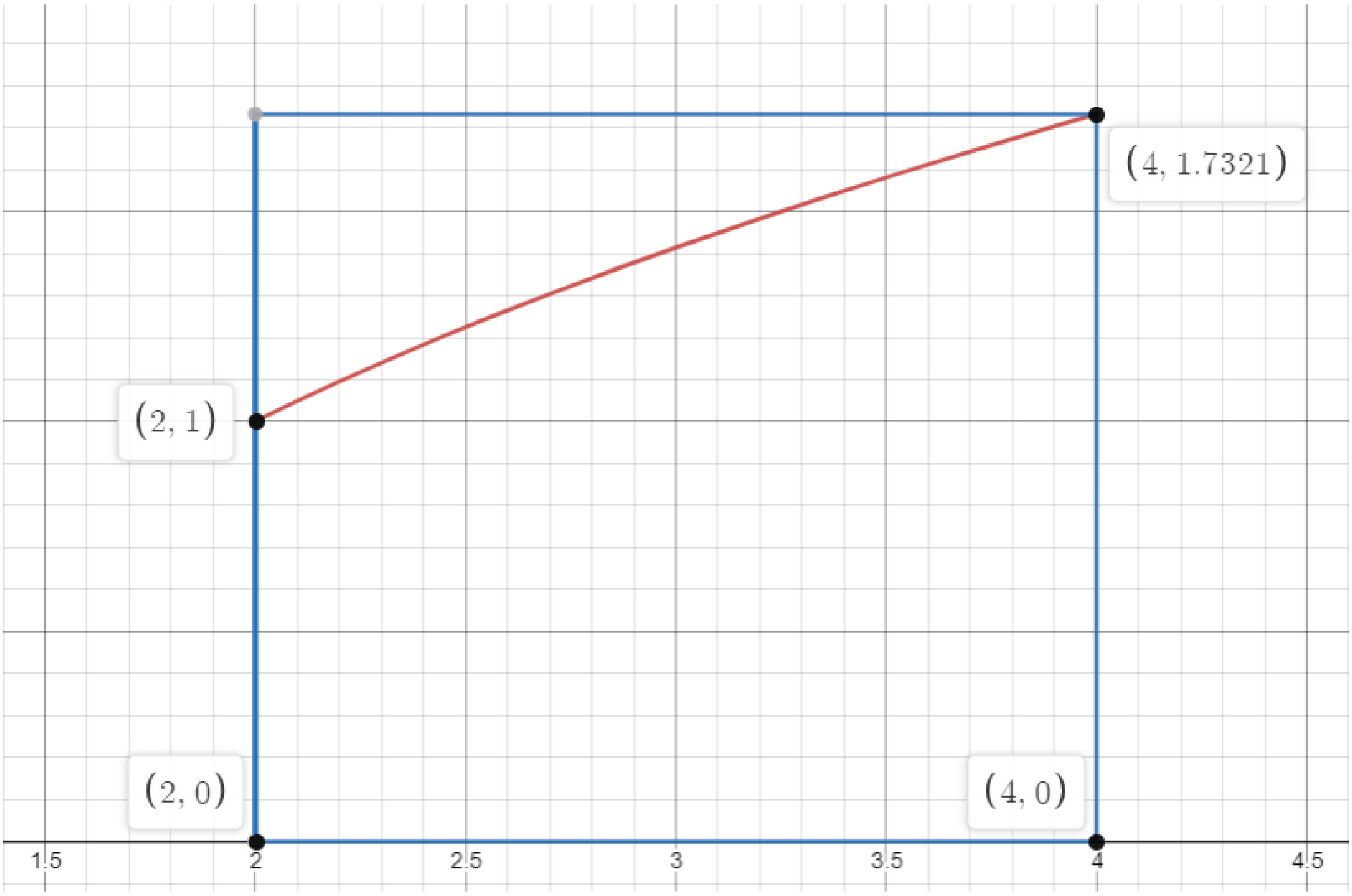

Рисунок 1 g(x) на [2;4] в среде desmos, ограниченная наименьшим прямоугольником

2.Сгенерируем случайные точки в границах прямоугольника, которые мы определили на первом шаге.

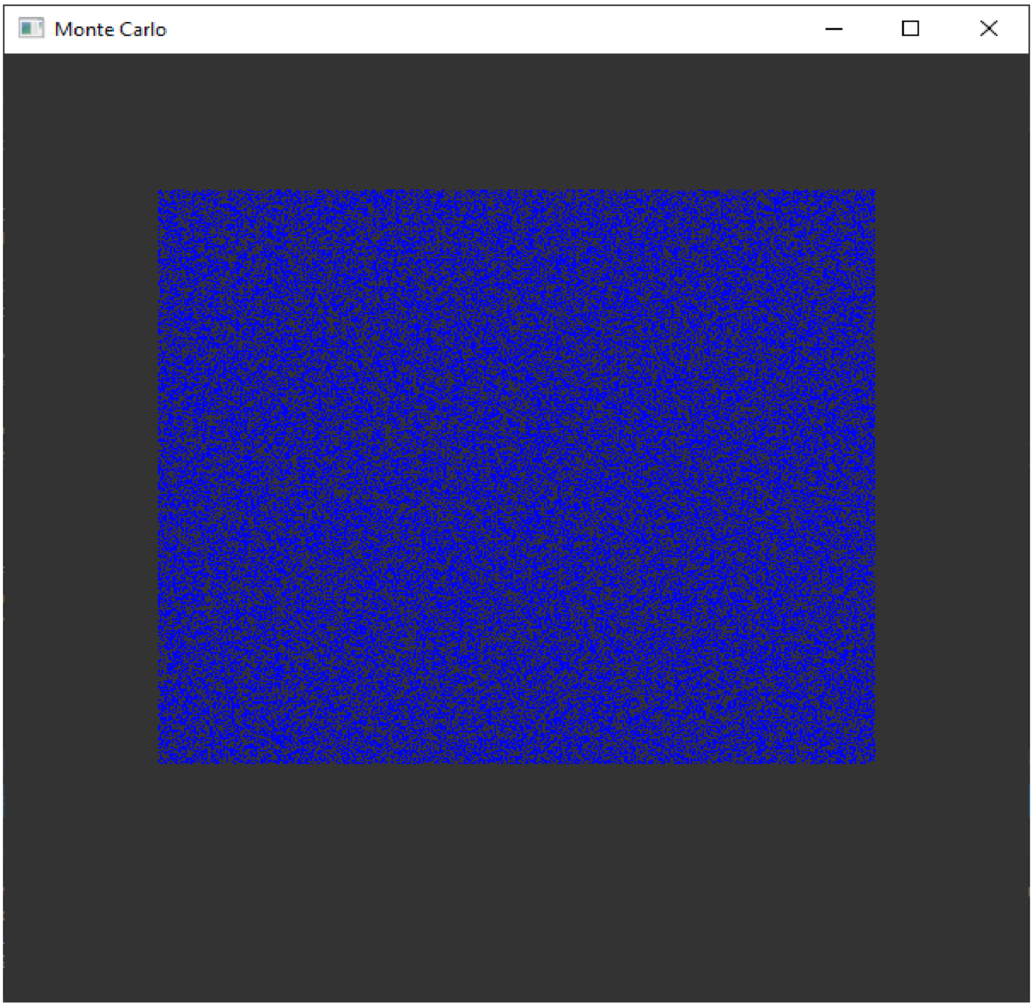

Рисунок 2 Сгенерировано 100000 точек

3. Сделаем отбор точек. Выберем только те, которые ниже нашей функции $g(x)$.

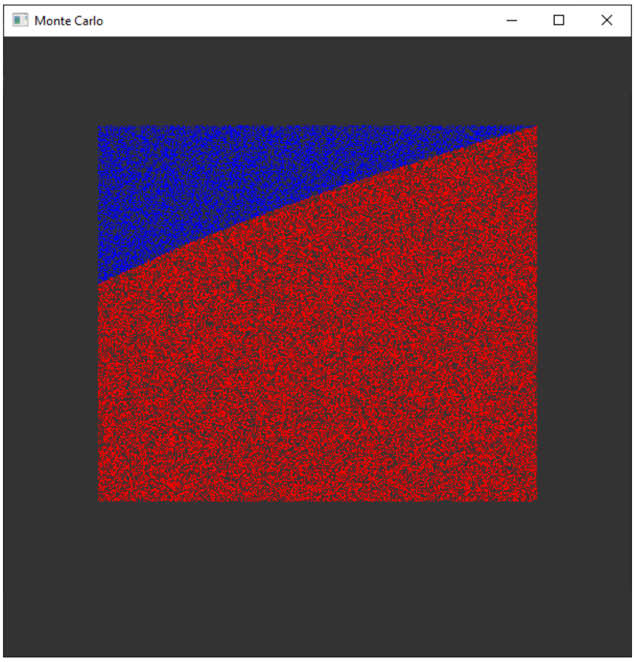

Рисунок 4 Произвели отбор (Красные точки - отобранные)

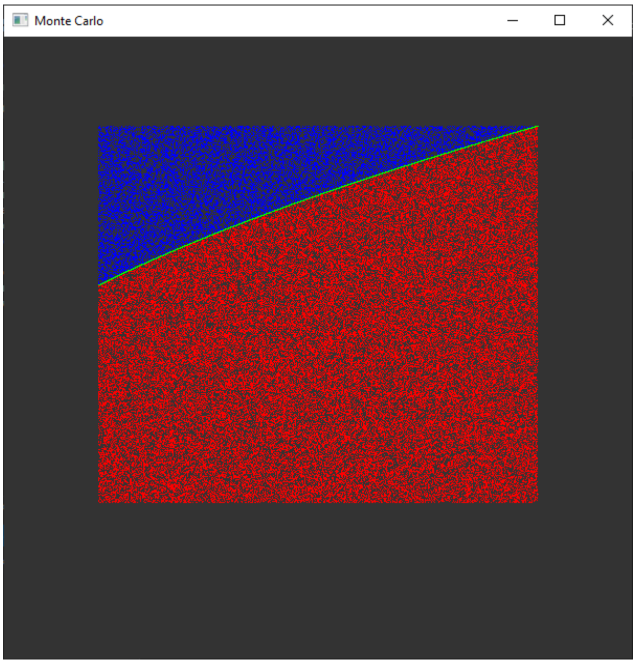

Рисунок 4 Для наглядности на рисунке отмечена функция g(x) зелёным цветом

4. Вычислим $h(x_i, y_i)$ в каждой из отобранных точек и найдём их сумму.

Должен сделать замечание, что если вычислить функцию $h(x, y)$, то мы получим график поверхности. В моём случае $h(x, y) = \frac{\sin(x+y)+\cos(x-y)}{(x+y)}$.

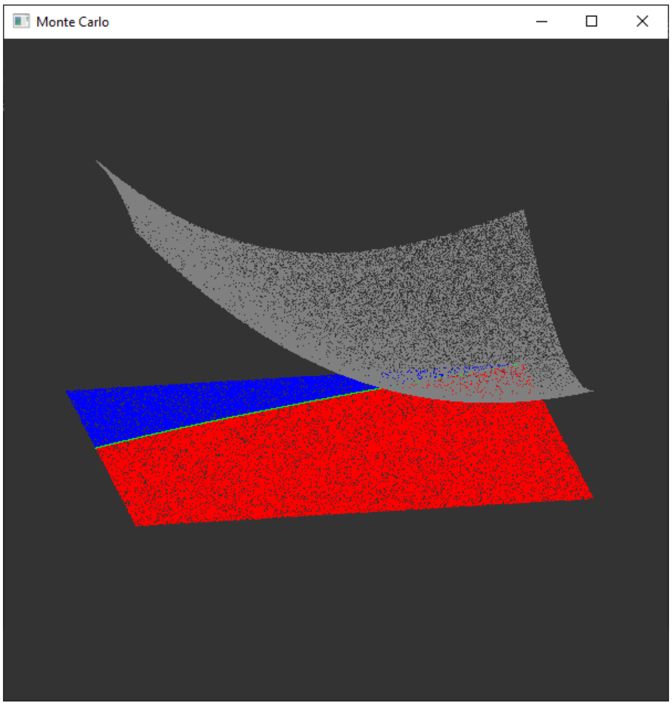

Рисунок 6 Поверхность h(x,y) ракурс №1

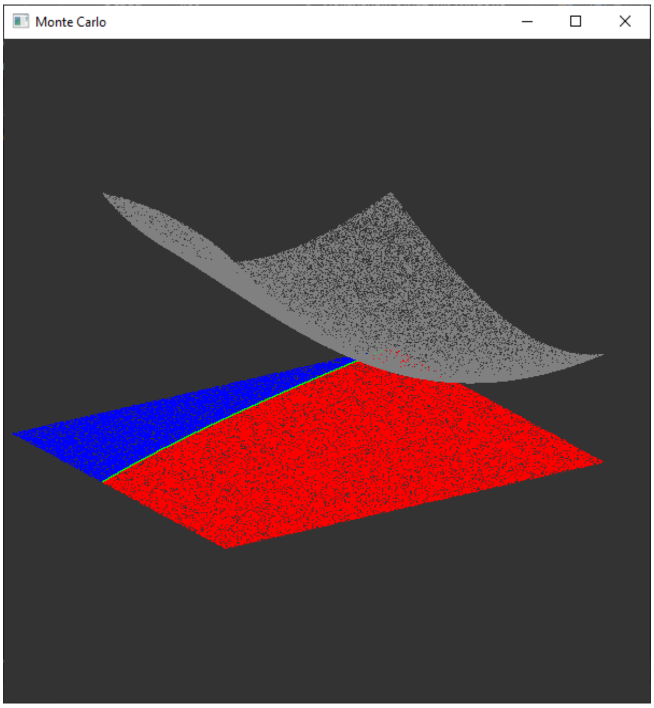

Рисунок 6 Поверхность h(x,y) ракурс №2

5. Вычислим интеграл по формуле: $I = S_{\Omega'} \cdot \frac{1}{n} \sum_{i=1}^{k} h(x_i', y_i')$.

Мой результат -0.680754863698567 при 100000 точках.

3.4 Анализ полученного результата

При 100тыс.точек I = -0.680754863698567. Увеличим количество сгенерированных точек, чтобы повысить точность. Программа готова, поэтому проделывать всё с самого начала не нужно, стоит лишь поменять числовое значение в коде.

При 10млн. I = -0.678984922795595. Точность увеличилась, и она является вполне достаточной.

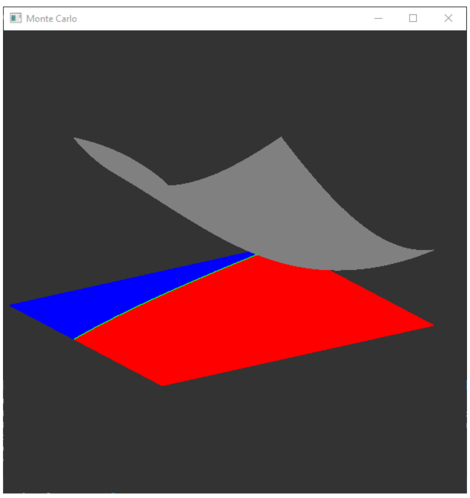

Рисунок 7 Поверхность h(x,y) при 10млн. точек

4 ЗАКЛЮЧЕНИЕ

Был изучен и применён на практике численный метод Монте-Карло. В ходе работы с помощью этого метода был вычислен заданный интеграл на отрезке. Были получены новые знания и навыки. Работа проделана без особых трудностей и сдана в срок.

5 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ Р 7.0.100-2018 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления».
2. ГОСТ 2.105-95 «ЕСКД. Общие требования к текстовым документам».
3. Соболь Илья Меерович. «Метод Монте-Карло» Год выпуска 1968
4. Интернет ресурс №1

6 ЛИСТИНГ

```
#include <vector>
#include "GLFW/glfw3.h"
#include <iostream>
#include "Functions.h"

int main()
{
    int n = 1e7;
    double S = 2.0 * sqrt(3.0);
    std::vector<double> index(n,0);
    std::vector<double> x;
    std::vector<double> y;

    InitPoints(x, y, n);
    SelectPoints(x, y, index, n);
    double sum_h = SumH(x, y, index, n);
    double I = S/n*sum_h;
    printf("%.15f\n", I);
#pragma region OpenGLDraws
    GLFWwindow* window;
    if (!glfwInit())
        return -1;
    window = glfwCreateWindow(600, 600, "Monte Carlo", NULL, NULL);
    if (!window)
    {
        glfwTerminate();
        return -1;
    }
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-0.1,0.1 ,-0.1,0.1,0.2,1000);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glfwMakeContextCurrent(window);
    while (!glfwWindowShouldClose(window))
    {
        glClear(GL_COLOR_BUFFER_BIT);
        glPushMatrix();
        glRotatef(-70, 1, 0, 0);
        glRotatef(20, 0, 0, 1);
        glTranslatef(0,0,-0.28);
        glPointSize(1.0);
```

```
        glBegin(GL_POINTS);
        for (int i = 0; i < n; i++){
            if (index[i] == 1) {
                glColor3f(1, 0, 0);
                glVertex2d(x[i] * 0.7 - 2.1, y[i] * 0.7 - 0.5);
            }
            else {
                glColor3f(0, 0, 1);
                glVertex2d(x[i] * 0.7 - 2.1, y[i] * 0.7 - 0.5);
            }

        }
        for (int i = 0; i < n; i++)
        {
            glColor3f(0, 1, 0);
            glVertex2d(x[i] * 0.7 - 2.1, sqrt(x[i] - 1) * 0.7 - 0.5);
        }
        glEnd();
        glTranslatef(0, 0, 0.7);
        glBegin(GL_POINTS);
        for (int i = 0; i < n; i++)
        {
            if (index[i] == 1) {
                glColor3f(0.5, 0.5, 0.5);
                glVertex3d(x[i] * 0.7 - 2.1, y[i] * 0.7 - 0.5, (sin(x[i] + y[i]) + cos(x[i] -
y[i])) / (x[i] + y[i]));
            }
        }
        glEnd();
        glPopMatrix();
        glfwSwapBuffers(window);
        glfwPollEvents();
        glClearColor(0.2, 0.2, 0.2, 1.0);
    }
    glfwTerminate();
#pragma endregion
}
```

Файл "Functions.h":

```
#pragma once
double SumH(std::vector<double>& x, std::vector<double>& y, std::vector<double>& index, int n);
void SelectPoints(std::vector<double>& x, std::vector<double>& y, std::vector<double>& index, int
n);
void InitPoints(std::vector<double>& x, std::vector<double>& y, int n);
```

Файл "Functions.cpp":

```
#include <vector>
#include <iostream>
#include "Functions.h"

double SumH(std::vector<double>& x, std::vector<double>& y, std::vector<double>& index, int n) {
    double expected_val = 0;
    for (int i = 0; i < n; i++) {
        if (index[i] == 1) {
            expected_val += (sin(x[i] + y[i]) + cos(x[i] - y[i])) / (x[i] + y[i]);
        }
    }
    return expected_val;
}
void SelectPoints(std::vector<double>& x, std::vector<double>& y, std::vector<double>& index, int
n) {
    for (int i = 0; i < n; i++) {
        double g_x = sqrt(x[i] - 1);
        if (y[i] <= g_x) {
            index[i] = 1;
        }
```

```
    }
}
void InitPoints(std::vector<double>& x, std::vector<double>& y, int n)
{
    double sq = sqrt(3.0);
    for (int i = 0; i < n; i++)
    {
        x.emplace_back(((double)rand() / (double)RAND_MAX) * 2.0 + 2.0);
        y.emplace_back(((double)rand() / (double)RAND_MAX) * sq);
    }
}
```