***Г.В. Гулов***

студент кафедры вычислительных систем и программирования

***В.Н. Ассаул***

доцент – научный руководитель

**ПРИМЕНЕНИЕ** **ТЕОРИИ ВЕРОЯТНОСТИ И МАТЕМАТИЧЕСКОЙ СТАТИСТИКИ В АНАЛИТИЧЕСКОЙ ГЕОМЕТРИИ.**

**Ключевые слова:** теория вероятности, аналитическая геометрия, симуляция задач на языках программирования высокого уровня.

**АННОТАЦИЯ**

В этой статье раскрываются основные подходы по решению и симуляции некоторых задач по теории вероятности, для решения которых необходим подход с большим количеством повторений эксперимента, знания аналитической геометрии и языков программирования высокого уровня.

**ВВЕДЕНИЕ**

В современном мире, где идет непрерывное развитие технологий и технических процессов во всех сферах человеческой деятельности, увеличивается востребованность исследований научных тем, находящихся на границах разных разделов наук и затрагивающих не одну, а несколько научных областей.

Развитие таких технологий как машинного обучения и компьютерного зрения сподвигло меня к изучению статистической вероятности попадания точки на различных участках геометрических фигур. В частности, я заинтересовался исследованием задачи: на окружности или эллипсе выбираются три случайные точки, они соединяются и о образуют вписанный треугольник. Найти вероятность того, что получившийся вписанный треугольник пересекает центр геометрической фигуры и определить какова форма треугольника.

Для решения данной задачи необходимо воспользоваться как знаниями теории вероятности, математической статистики, так и аналитической геометрии, линейной алгебры и навыками программирования на языках высокого уровня.

**МАТЕРИАЛЫ И МЕТОДЫ**

1. **Формулировка задачи и её теоретическое решение.**

Размышляя над поставленной проблемой, я понял, что необходимо: во-первых, найти правильное теоретическое решение задачи, во-вторых, рассмотреть несколько методов симуляции с разными подходами для различных геометрических фигур.

## **1.1 Формулировка задачи для окружности.**

На окружности выбираются три случайные точки, они соединяются и о образуют вписанный треугольник. Найти вероятность того, что получившийся вписанный треугольник перекрывает центр окружности.

## **1.2 Теоретическое решение данной задачи.**

Решение данной задачи заключается в том, что где бы ни была выбрана первая точка, диаметр, на котором она лежит, делит круг на два симметричных полукруга, поэтому вторая и третья точки (при условии, что они разные) обязательно должны быть размещены на противоположных половинах круга. Линия, соединяющая вторую и третью точки, тогда должна также лежать выше центра (относительно первой точки - или ниже центра, если первая точка была сверху). Так что если вторая точка находится на расстоянии X от первой точки по периметру круга (в единицах длины периметра), то существует диапазон в пределах в котором можно разместить третью точку. Таким образом, вероятность будет равна:

Мы умножаем интеграл на 2, чтобы отбросить то, что мы можем поменять местами вторую и третью точки. В итоге получаем вероятность равную 25%, что в последствии будет подтверждено симуляциями.

# **2.** **Симуляция для окружности. Метод пересекающихся хорд.**

# **2.1 Симуляция задачи для окружности на языке программирования C++. Метод пересекающихся хорд.**

Метод проверки через пересекающиеся хорды заключается в том, что мы ставим на окружность случайные две точки (они должны быть разными) A и B (Рис. 1). Потом через центр О и эти две точки проводится две хорды (AA1 и BB1). Дуга (противоположенная первым двум точкам) B1A1 заключенная между хордами AA1 и BB1 будет допустимым диапазоном, где может находится третья точка C. Если три случайно сгенерированные точки удовлетворяют данному условию, то треугольник перекрывает центр окружности.

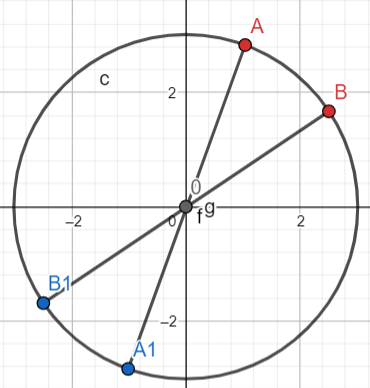


Рис. 1. Метод симуляции через пересекающиеся хорды на окружности.

## **2.2 Алгоритм работы программы.**

Сначала нам необходимо сгенерировать три случайные точки на окружности в диапазоне от 0 до . И для удобства последующих вычислений упорядочить их по возрастанию Рис. 2.

int a = rand() % 360; // создаём случайную точку A

int b = rand() % 360; // создаём случайную точку B

int c = rand() % 360; // создаём случайную точку C

const int n = 3;

int s[n] = {a,b,c};

hoarasort(s, 0, 2); // упорядочиваем их методом быстрой сортировки (Хоара)

a = s[0]; // расставляем упорядоченные точки на окружности

b = s[1];

c = s[2];

Рис. 2 Генерация трёх случайных точек и их упорядочивание.

Для упорядочивания трёх точек по возрастанию используется метод быстрой сортировки (метод Хоара) Рис. 3.

void hoarasort(int\* a, int first, int last)

{

int i = first, j = last;

double tmp, x = a[(first + last) / 2];

do {

while (a[i] < x)

i++;

while (a[j] > x)

j--;

if (i <= j)

{

if (i < j)

{

tmp = a[i];

a[i] = a[j];

a[j] = tmp;

}

i++;

j--;

}

} while (i <= j);

if (i < last)

hoarasort(a, i, last);

if (first < j)

hoarasort(a, first, j);

}

Рис.3. Метод быстрой сортировки Хоара.

Основная проверка вхождения третьей точки в допустимый диапазон происходит Рис. 4.

if(( (a+180)%360 <c && c < (b+180)%360)||((b + 180) % 360 < c && c < (a + 180)% 360))

return true;

else

return false;

Рис. 4. Проверка вхождения третьей точки в допустимый диапазон.

Остаток от деления на берётся для того, чтобы обработать ситуацию, когда из-за больших сгенерированных значений нарушается порядок точек (A, B, C). Все вышеописанные операции выполняются необходимое количество итераций. И в итоге дают результат близкий 25%. Чем больше количество итераций, тем ближе конечный результат к 25%.

## **2.3 Листинг программы.**

Весь листинг программы прикреплён в дополнительных материалах.

## **2.4 Пример результата симуляции в консоли.**

Отработав, программа выдаёт результат своей работы в консоли (Рис. 5).

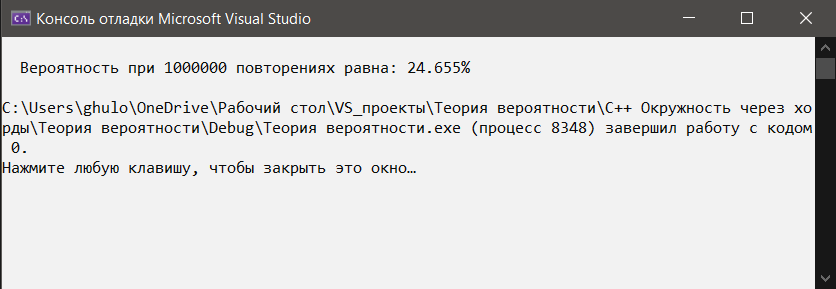


Рис. 5. Пример успешного выполнения работы программы в консоли.

## **2.5 Результаты симуляции для разного количества итераций.**

В зависимости от количества итераций работы программы меняется точность вычислений. В таблице 1 приведены примеры симуляций для разного количества итераций, график зависимости приведён на Рис. 6.

|  |  |
| --- | --- |
| N – количество итераций | P(A)-результат симуляции, % |
| 100 | 27.49 |
| 500 | 25.62 |
| 1000 | 25.32 |
| 2000 | 24.55 |
| 5000 | 24.74 |
| 10000 | 24.79 |
| 50000 | 24.97 |
| 100000 | 25.04 |
| 500000 | 25.01 |

Таблица 1 результатов симуляции при различных количествах итераций.

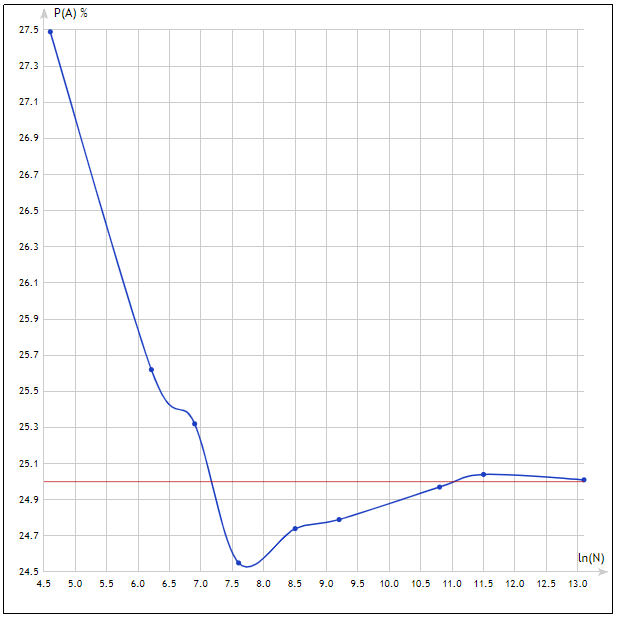


Рис. 6. Синий график - График зависимости вероятности от количества итераций работы программы, Красная линия – вероятность в 25%

## **2.6 Минусы и недостатки программы.**

Данная программа не имеет графического оформления и выводит результат только в консоли. Хорошим решение для данной проблемы будет использование языка программирования JavaScript и оформление визуального интерфейса программы в виде сайта. В программе можно будет вводить количество итераций эксперимента и будет реализовано не консольное отображение полученного результата симуляции.

# **3. Симуляция для эллипса. Метод проверки через векторное умножение.**

## **3.1 Формулировка задачи.**

На эллипсе произвольной большой и малой полуоси выбираются три случайные точки, они соединяются о образуют треугольник. Найти вероятность того, что получившийся вписанный треугольник перекрывает центр эллипса.

## **3.2 Описание метода проверки.**

Данный метод проверки заключается в том, что мы, используя каноническое уравнение эллипса, генерируем три случайные точки на эллипсе. Центром системы координат и одновременно центром эллипса является точка с координатами (0;0). Методом поочередного умножения векторов сторон треугольника на вектор, направленный в центр эллипса (0;0). Мы проверяем условие, что данная точка находится по правую (левую если обходить треугольник против часовой стрелки) сторону от всех сторон треугольника (Рис.7), то есть внутри фигуры, а значит получившийся треугольник из трёх произвольных точек на эллипсе перекрывает цент эллипса.

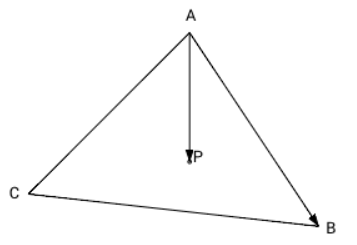
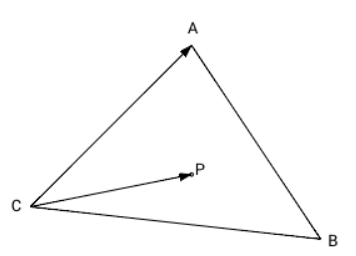
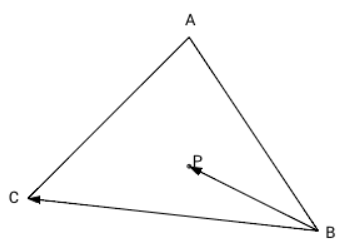


Рис.7. Попарное векторное произведение векторов-сторон треугольника и вектора из вершины в точку (0;0).

**3.3 Алгоритм работы программы.**

Используемый алгоритм проверки основывается на векторном произведении. Векторное произведение векторов a и b, заданного декартовыми координатами в пространстве для 3-х мерного правого ортонормального базиса можно выразить так:

Векторное произведение обладает свойством антикоммутативности:

Это важное свойство, которое будет использоваться для решения задачи.

Для того чтобы определить лежит ли точка P (0;0) — центр эллипса внутри треугольника ABC я вычисляю 3 векторных произведения: ,

Так как наш треугольник и точка в 2-мерном пространстве на плоскости, третья координата Z для трехмерного пространства равна нулю. Согласно формуле:

Можно не вычислять координаты X и Y для векторного произведения, если координата Z векторов-множителей равна нулю - координаты X и Y результата в этом случае всегда равны нулю (результирующий псевдовектор перпендикулярен плоскости треугольника). Знак результата произведения для оставшейся координаты (Z) зависит от относительного положения умножаемых векторов. Если первый вектор (в нашем случае это сторона треугольника) находится правее второго вектора (вектор из вершины в точку P), то координата Z результата будет положительна, если первый вектор будет левее второго - отрицательна, и в противном случае, если оба вектора идут в одном и том же направлении, результат будет равен нулю.

Получив результаты по трем векторным произведениям, остается их проанализировать, чтобы понять лежит ли точка внутри треугольника.

Если получается положительные и отрицательные результаты, точка лежит вне треугольника Рис. 10, если результаты только положительные или только отрицательные, точка - внутри эллипса Рис.8. Если один результат равен нулю, остальные все либо положительны, либо все отрицательны, то точка лежит на стороне треугольника Рис. 9.

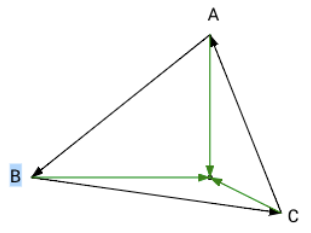
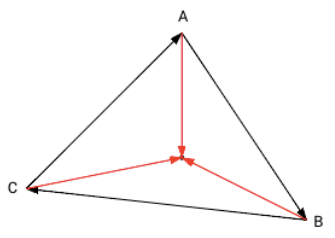


Рис.8. Все результаты положительные или все результаты отрицательные. Точка - внутри треугольника.

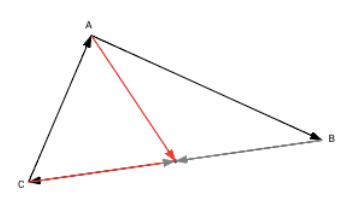


Рис. 9. Один результат равен нулю, остальные все либо положительны, либо все отрицательны. Точка лежит на стороне треугольника. Если два результата равны нулю, точка совпадает с вершиной треугольника.

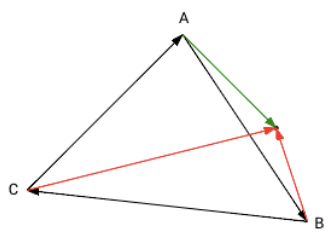


Рис. 10. Если имеются и положительные и отрицательные результаты. Точка лежит вне треугольника.

Проверка точки (0;0) на вхождение в треугольник происходит Рис. 11.

var testInbound = function(a,b,c) {

            var x = (a.x - 0) \* (b.y - a.y) - (b.x - a.x) \* (a.y - 0);

            var y = (b.x - 0) \* (c.y - b.y) - (c.x - b.x) \* (b.y - 0);

            var z = (c.x - 0) \* (a.y - c.y) - (a.x - c.x) \* (c.y - 0);

            if (x > 0 && y > 0 && z > 0 || x < 0 && y < 0 && z < 0) return true;

            return false;

        }

Рис. 11. Проверка вхождения точки (0;0) в треугольник

Что касается остальной части симуляции. Первоначально я, основываясь на введённой пользователем большой полуоси, удвоив её, получаю большую ось — это допустимый диапазон значений точки по оси X (он может быть как положительным, так и отрицательным). Я генерирую случайное число на данном диапазоне. Это происходит Рис. 12.

var x = Math.random()\*(a\_poluos \* 2) - a\_poluos;

Рис. 12. Генерация случайного числа в допустимом диапазоне

Потом основываясь на каноническом уравнении эллипса:

И сгенерированном случайном значении X в диапазоне большой полуоси (малая и большая полуось уже введена пользователем) я вывожу значения Y для имеющегося случайного X:

Также я не забываю, что уравнение для Y из-за корня может принимать как положительные, так и отрицательные значения, поэтому я случайно генерирую либо 0, либо 1, и в зависимости от результата добавляю минус к Y, получая уравнение

Благодаря этому рассматривается как верхняя, так и нижняя часть эллипса. Всё это происходит в части кода на Рис. 13.

  if(Math.round(Math.random())) {

                var y = Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

            } else {

                var y = -Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

            }

Рис. 13. Вычисление координаты по оси OY

Получив три точки с координатами, которые могут находиться на эллипсе с введёнными пользователем малой и большой полуокружностью, происходит проверка попадания центра эллипса в треугольник, описанная в самом начале.

* 1. **Листинг программы.**

Весь листинг программы прикреплён к дополнительным материалам.

**3.5 Примеры работы программы.**

Результат симуляции программы можно видеть в графическом окне браузера Рис. 14.

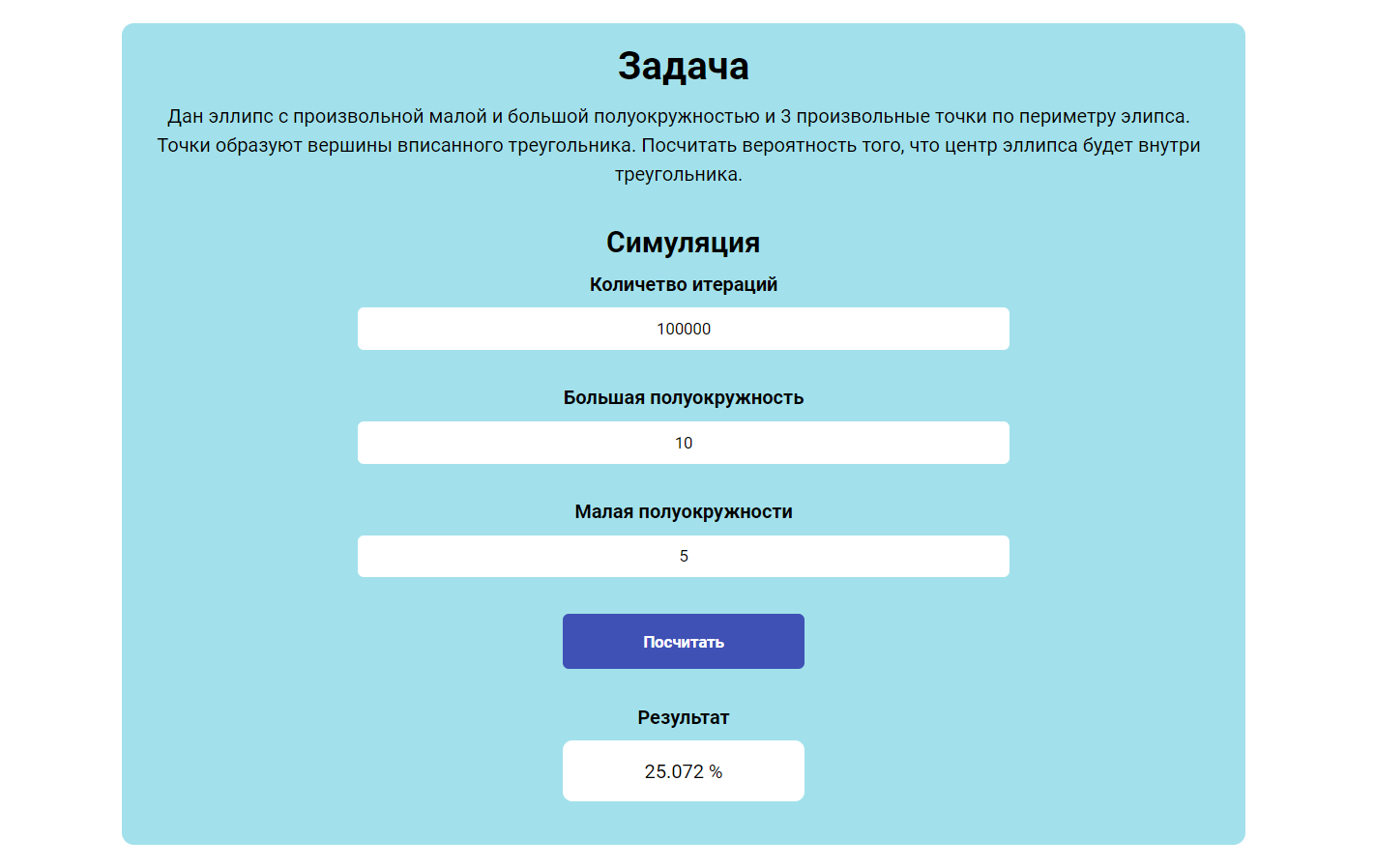
****

Рис. 14. Пример работы программы.

Как видно из симуляции вероятность для эллипса составляет так же, как и для окружности 25% даже на разных отношениях большой малой полуоси.

# **4. Симуляция для окружности и эллипса. Проверка остроугольности вписанного треугольника через условие остроугольности сторон.**

## **4. 1 Формулировка задачи.**

На эллипсе произвольной большой и малой полуоси (если они равны, то рассматривается окружность) выбираются три случайные точки, они соединяются и образуют вписанный треугольник. Найти вероятность того, что получившийся вписанный треугольник является остроугольным.

## **4.2 Описание метода проверки.**

Новая симуляция работает так:

1. Первоначально я, основываясь на введённой пользователем большой полуоси, удвоив её, получаю большую ось — это допустимый диапазон значений точки по оси 0X (он может быть как положительным, так и отрицательным). Я генерирую случайное число на данном диапазоне Рис. 15.

var x = Math.random()\*(a\_poluos \* 2) - a\_poluos;

Рис. 15. Генерация случайного числа в допустимом диапазоне.

Потом, используя каноническое уравнение эллипса , вычисляется значение по оси 0Y, использую выведенное уравнение . Также я не забываю, что уравнение для Y из-за корня может принимать как положительные, так и отрицательные значения, поэтому я случайно генерирую либо 0, либо 1, и в зависимости от результата добавляю минус к Y

. Благодаря этому рассматривается как верхняя, так и нижняя часть эллипса. Всё это происходит Рис. 16:

if(Math.round(Math.random())) {

                var y = Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

            } else {

                var y = -Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

            }

Рис. 16. Вычисление координаты по оси OY

1. Получив три точки с координатами, которые могут находиться на эллипсе с введёнными пользователем малой и большой полуокружностью, происходит проверка остроугольности треугольника. Сначала, зная координаты каждой вершины треугольника, вычисляется длина каждой стороны треугольника по формуле Потом вычисляются их квадраты, а затем проверяется три раза (для каждой стороны) условие

. Если оно выполняется все три раза, то треугольник остроугольный Рис. 17.

function sqr(a) {

            return a \* a;

        }

        function isTriangleObtuse(a, b, c) {

                var a1 = Math.sqrt( sqr(b.x - a.x) + sqr(b.y - a.y) ),

                    b1 = Math.sqrt( sqr(c.x - b.x)  + sqr(c.y - b.y) ),

                    c1 = Math.sqrt( sqr(a.x - c.x)  + sqr(a.y - c.y) );

                var A = sqr(a1), B = sqr(b1), C = sqr(c1);

            return A + B > C && A + C > B && B + C > A;

        }

Рис. 17. Проверка треугольника на остроугольность.

Новая симуляция выдает значение близкое к 25% на одинаковых значениях большой и малой полуоси (когда эллипс становится окружностью). Когда различия между большой и малой полуосью начинают увеличиваться, вероятность остроугольности треугольника начинает постепенно уменьшаться.

## **4.3 Листинг программы.**

Весь листинг программы прикреплён к дополнительным материалам.

## **4.4 Примеры выполнения программы.**

Когда полуоси равны, и эллипс переходи в окружность. Вероятность близится к 25% Рис. 18.

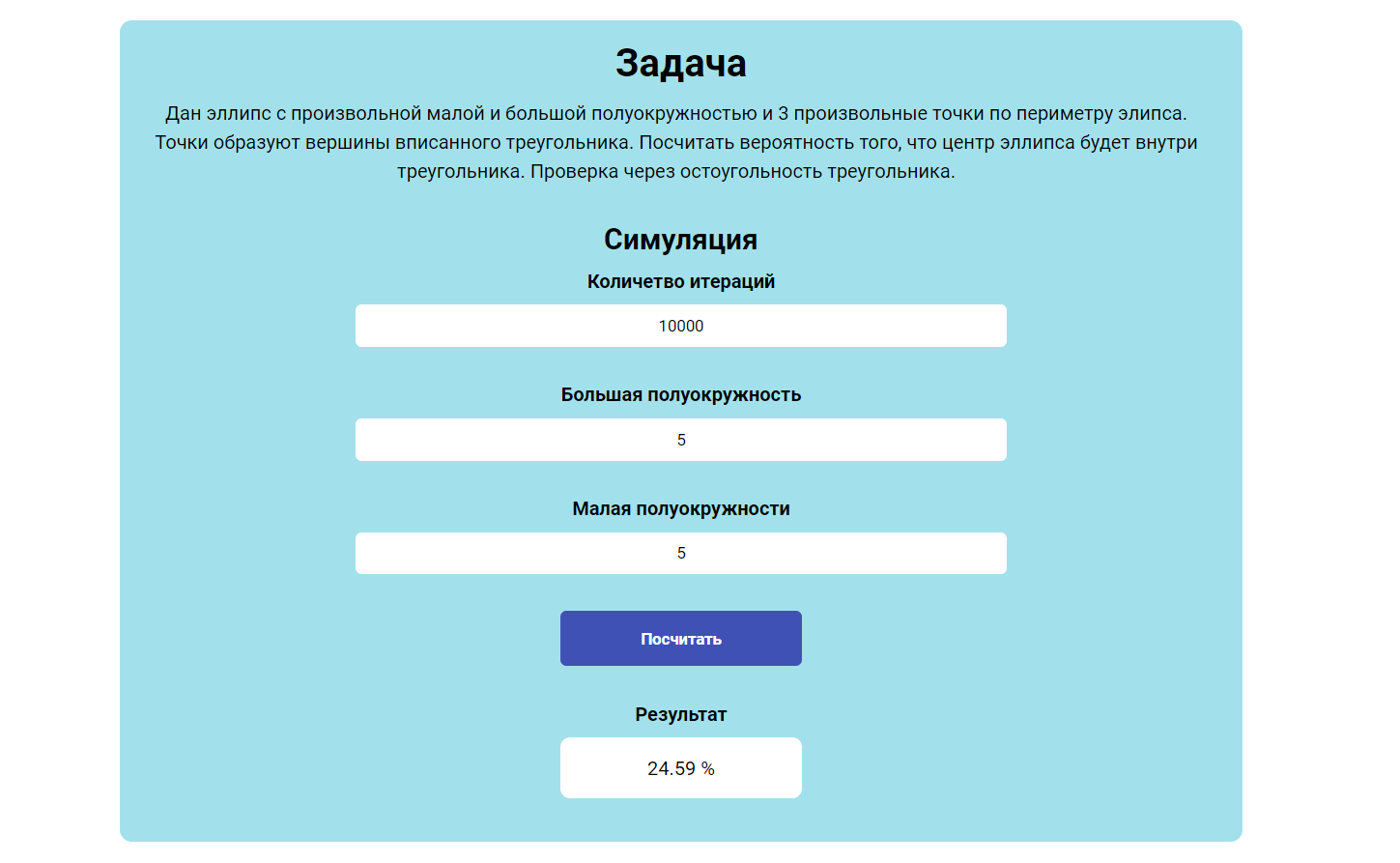
****

Рис. 18. Пример работы программы при равенстве большой и малой полуоси.

Когда значения большой и малой полуоси начинают значительно отличаться, то вероятность остроугольности треугольника начинает постепенно падать Рис. 19, Рис. 20.

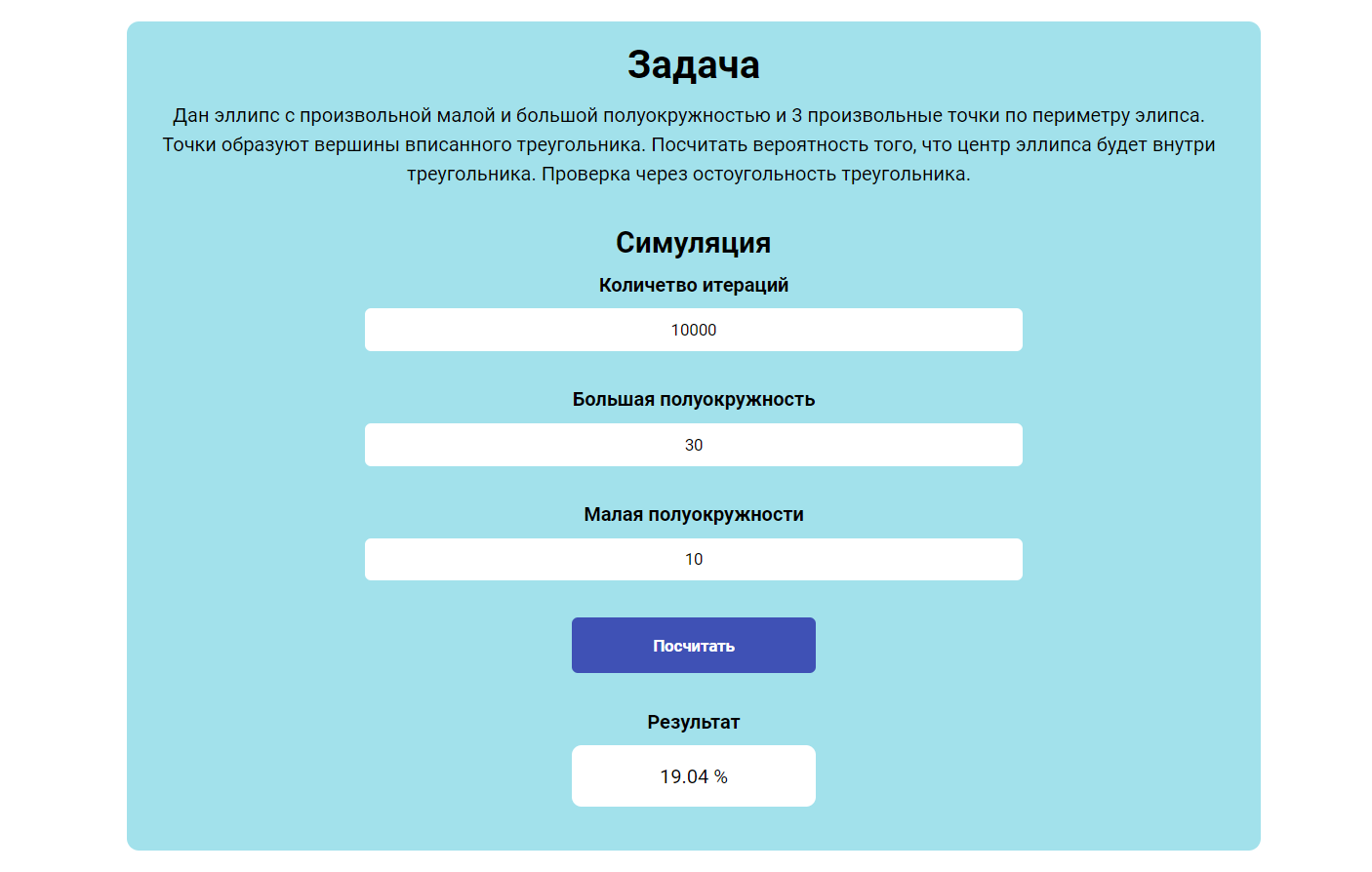


Рис. 19. Результат программы при различии большой и малой полуоси.

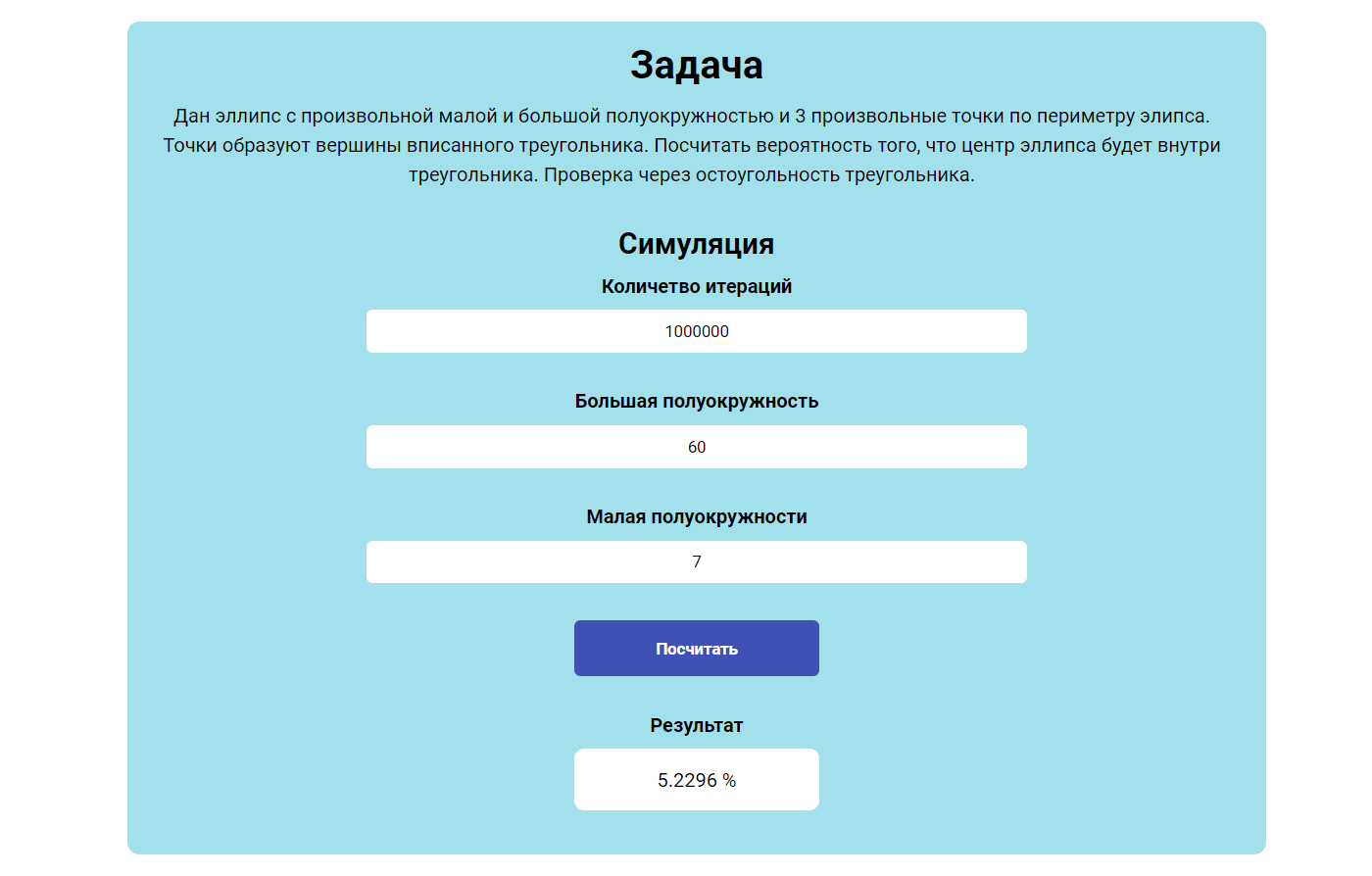


Рис. 20. Результат программы при различии большой и малой полуоси.

**5. Симуляция для эллипса. Сравнение вероятности перекрытия центра эллипса и остроугольности вписанного треугольника (проверка остроугольности через косинусы углов треугольника)**

## **5.1 Описание нововведений.**

Проверка перекрытия центра эллипса вписанным треугольником будет проверяться методом проверки векторного умножения, а проверка остроугольности треугольника будет приходить аналогично предыдущему методу, однако вместо проверки остроугольности треугольника через условие новая симуляция будет проверять остроугольность вписанного треугольника через проверку допустимых значений косинусов углов треугольника. Для остроугольности треугольника необходимо, чтобы все углы треугольника не превосходили , а значит косинусы углов были в пределах от 0 до 0,5.

Для этого нам сначала необходимо вычислить длины сторон треугольника по формуле

Потом используя теорему косинусов находим значение косинуса для рассматриваемого угла . Если все значения косинусов удовлетворяют необходимому нам условию (косинус угла должен быть от 0 до 0,5), то треугольник остроугольный. Всё это будет проверяться в части кода на Рис. 21.

function isTriangleObtuse(a, b, c) {

var a1 = Math.sqrt( sqr(b.x - a.x) + sqr(b.y - a.y) ), // вычисление длины сторон

b1 = Math.sqrt( sqr(c.x - b.x) + sqr(c.y - b.y) ),

c1 = Math.sqrt( sqr(a.x - c.x) + sqr(a.y - c.y) );

var A = sqr(a1), B = sqr(b1), C = sqr(c1) // возведение сторон треугольника в квадрат

// проверка косинусов углов треугольника

return (b1\*b1 + c1\*c1 - a1\*a1)/(2\*b1\*c1) < 1 && (b1\*b1 + c1\*c1 - a1\*a1)/(2\*b1\*c1) > 0 &&

(b1\*b1 + a1\*a1 - c1\*c1)/(2\*b1\*a1) < 1 && (b1\*b1 + a1\*a1 - c1\*c1)/(2\*b1\*a1) > 0 &&

(a1\*a1 + c1\*c1 - b1\*b1)/(2\*a1\*c1) < 1 && (a1\*a1 + c1\*c1 - b1\*b1)/(2\*a1\*c1) > 0

Рис. 21. Проверка косинусов углов треугольников.

## **5.2 Листинг программы.**

Весь листинг программы прикреплён к дополнительным материалам.

## **5.3 Результаты симуляции.**

Как видно из сравнений симуляции вероятность перекрытия центра остаётся на уровне 25%, а вот вероятность остроугольности начинает постепенно падать при увеличении отношений большой и малой полуоси.

## **5.3 Примеры выполнения программы.**

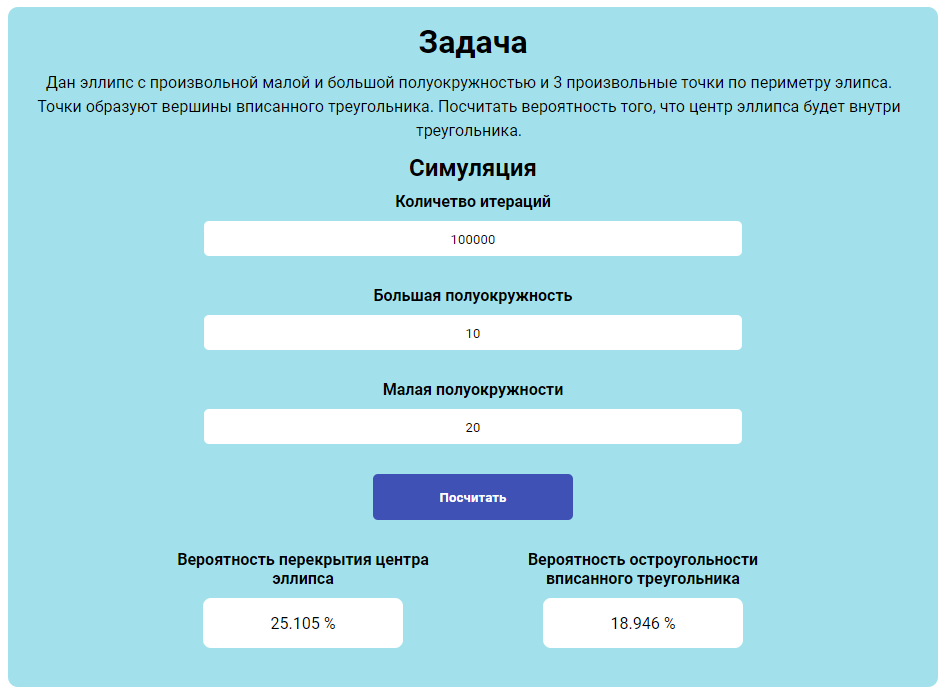


Рис. 23. Результат программы при различии большой и малой полуоси.

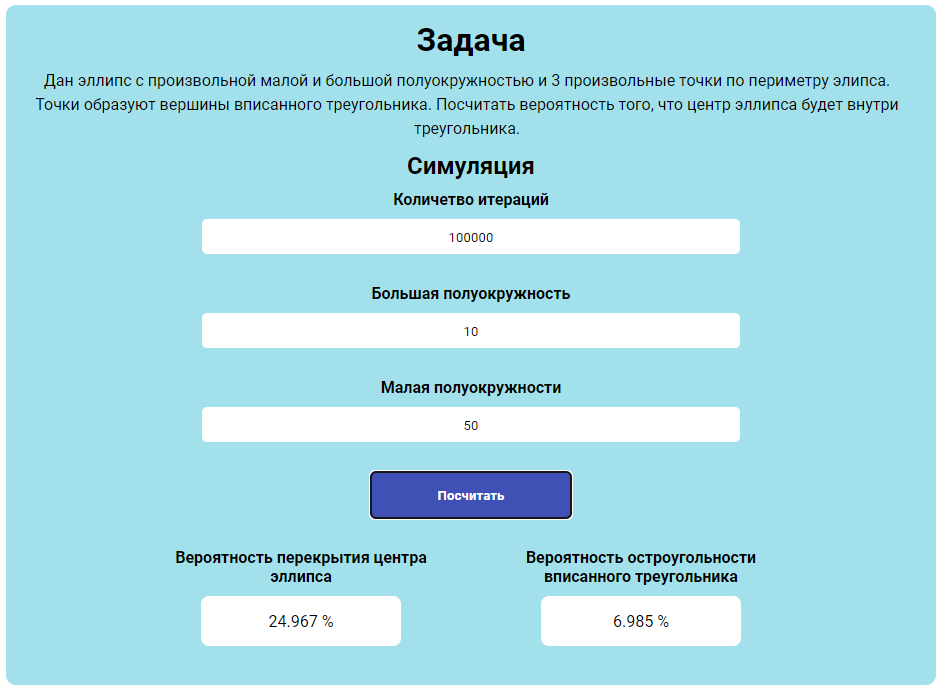


Рис. 24. Результат программы при различии большой и малой полуоси.

**6. Альтернативная генерация координат для точек треугольника на эллипсе за счёт угла между точкой и осью координат. Несовершенства предыдущего метода генерации точек через каноническое уравнение эллипса.**

В своей предыдущей симуляции для генерации координат точек треугольника на эллипсе было использовано каноническое уравнение эллипса . Краткое содержание старого метода генерации координат:

Первоначально я, основываясь на введённой пользователем большой полуоси, удвоив её, получаю большую ось — это допустимый диапазон значений точки по оси 0X (он может быть как положительным, так и отрицательным). Я генерирую случайное число на данном диапазоне Рис. 25.



var x = Math.random()\*(a\_poluos \* 2) - a\_poluos;

Рис. 25. Генерация случайного числа в допустимом диапазоне.

Потом, используя каноническое уравнение эллипса , вычисляется значение по оси 0Y, использую выведенное уравнение . Также я не забываю, что уравнение для Y из-за корня может принимать как положительные, так и отрицательные значения, поэтому я случайно генерирую либо 0, либо 1, и в зависимости от результата добавляю минус к Y

. Благодаря этому рассматривается как верхняя, так и нижняя часть эллипса. Всё это происходит Рис. 26:

if(Math.round(Math.random())) {

                var y = Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

            } else {

                var y = -Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

            }

Рис. 26. Вычисление координаты по оси OY

Однако после проведения рядя испытаний было выяснено, что данная симуляция даёт не корректные значения вероятности для проверки остроугольности треугольника для аналогичных отношений большой и малой полуоси (например, и ), они должны быть приблизительно равны, однако этого не происходит. Результаты симуляции старого метода генерации координат приведены в таблице 2 и на графике на Рис 27.

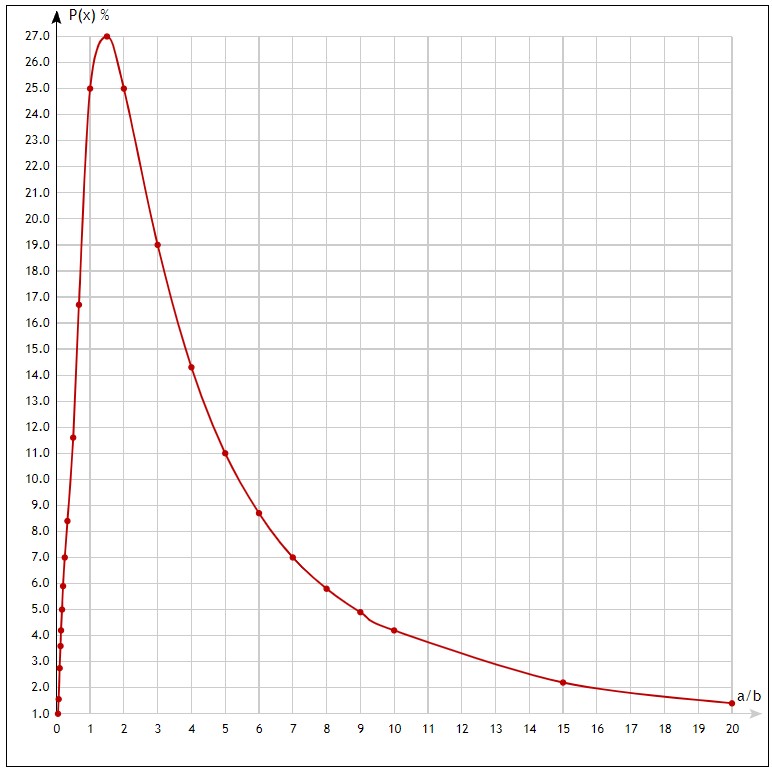


Рис 27. График результатов симуляции.

|  |  |
| --- | --- |
| Отношение большой к малой полуоси | Вероятность остроугольности треугольника  % |
| 0,05 | 1,00 |
| 0,07 | 1,56 |
| 0,10 | 2,75 |
| 0,12 | 3,60 |
| 0,14 | 4,20 |
| 0,17 | 5,00 |
| 0,20 | 5,90 |
| 0,25 | 7,00 |
| 0,33 | 8,40 |
| 0,50 | 11,60 |
| 0,67 | 16,70 |
| 1,00 | 25,00 |
| 1,50 | 27,00 |
| 2,00 | 25,00 |
| 3,00 | 19,00 |
| 4,00 | 14,30 |
| 5,00 | 11,00 |
| 6,00 | 8,70 |
| 7,00 | 7,00 |
| 8,00 | 5,80 |
| 9,00 | 4,90 |
| 10,00 | 4,20 |
| 15,00 | 2,20 |
| 20,00 | 1,40 |

Таблица 2. Результаты симуляции для разного отношения большой и малой полуоси.

Принцип нового метода генерации координат точек треугольника на эллипсе заключается в использовании параметрического уравнения эллипса:

Сначала нам необходимо сгенерировать случайное число t в диапазоне от 0 до . Потом подставить его в параметрическое уравнение эллипса:

Это повторяется три раза и так получаются координаты точек треугольника на эллипсе. Новый метод генерации координат на эллипсе даёт корректный результат для аналогичных отношений большой и малой полуоси (например, и ). Результаты симуляции с новым методом генерации координат приведены в таблице 3 и на графике на Рис 28. Сравнение результатов симуляции при старом и новом методе генерации точек Рис. 29.

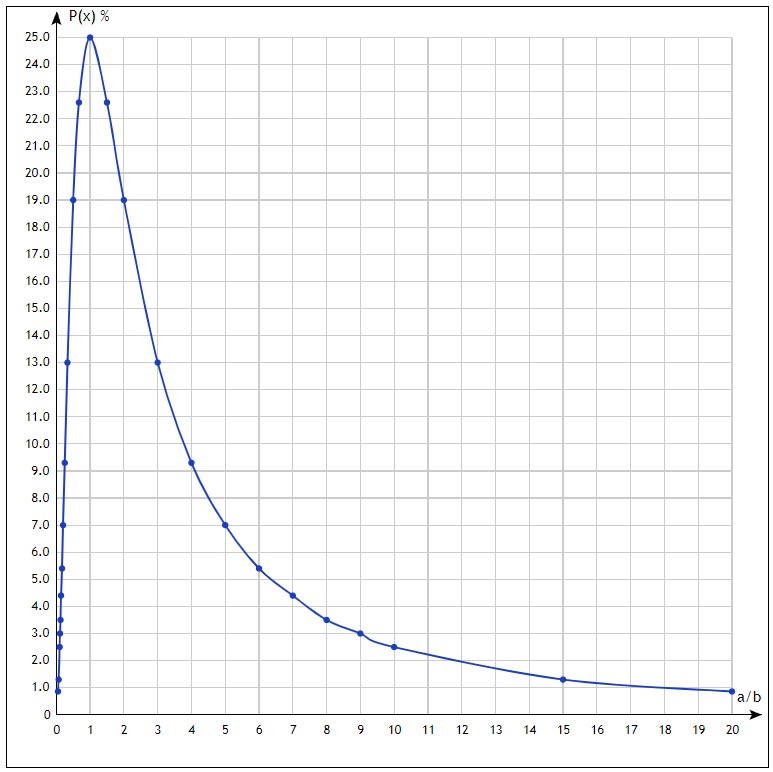


Рис 28. График результатов симуляции.

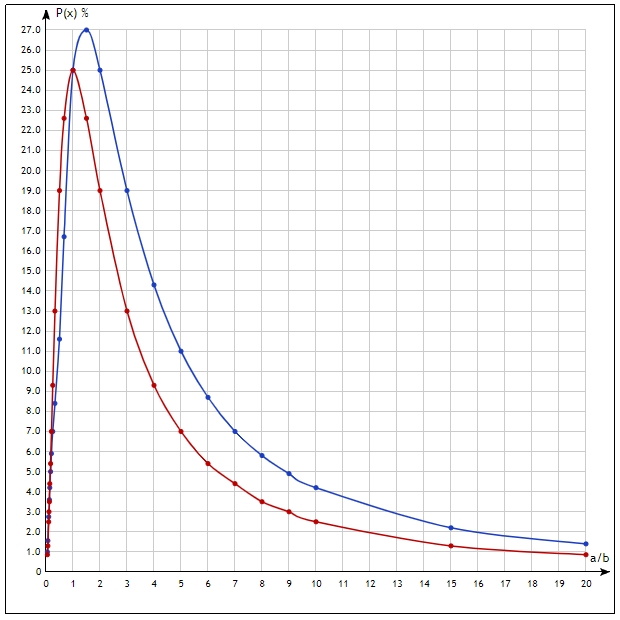


Рис 29. Графики сравнения результатов остроугольности симуляции для эллипса при разном методе генерации точек. Красный – новый метод генерации точек, Синий – старый метод генерации точек.

|  |  |
| --- | --- |
| Отношение большой к малой полуоси | Вероятность перекрытия центра  % |
| 0,05 | 0,86 |
| 0,07 | 1,30 |
| 0,10 | 2,50 |
| 0,12 | 3,50 |
| 0,14 | 4,40 |
| 0,17 | 5,40 |
| 0,20 | 7,00 |
| 0,25 | 9,30 |
| 0,33 | 13,00 |
| 0,50 | 19,00 |
| 0,67 | 22,60 |
| 1,00 | 25,00 |
| 1,50 | 22,60 |
| 2,00 | 19,00 |
| 3,00 | 13,00 |
| 4,00 | 9,30 |
| 5,00 | 7,00 |
| 6,00 | 5,40 |
| 7,00 | 4,40 |
| 8,00 | 3,50 |
| 9,00 | 3,00 |
| 10,00 | 2,50 |
| 15,00 | 1,30 |
| 20,00 | 0,86 |

Таблица 3. Результаты симуляции остроугольности для разного отношения большой и малой полуоси эллипса.

**Результаты и обсуждения.**

Проведя свои исследования, я получил результат, что вероятность перекрытие центра окружности вписанным треугольником образованным тремя случайными точками на окружности равна 25% вне зависимости от метода (симуляции методом пересекающихся хорд, методом векторного умножения и методом проверки условия остроугольности треугольника).

Для эллипса у меня получилась вероятность перекрытия центра треугольником тоже равная 25% (методом векторного умножения векторов сторон треугольника на вектор из вершин треугольника в центр эллипса), однако вероятность остроугольности получившегося треугольника в эллипсе отличается. Если большая и малая полуоси равны, то есть эллипс является окружностью, то вероятность остроугольности получившегося треугольника равна 25%, но если начать изменять соотношение большой и малой полуоси, то вероятность остроугольности треугольника начинает постепенно уменьшаться (это хорошо видно в моей последней симуляции с проверкой остроугольности треугольника в эллипсе, которую можно проверить на моём сайте (дополнительные материалы 1).

Также можно сделать вывод, что для окружности условие покрытия центра равносильно условию остроугольности, но для эллипса оно не работает.

В итоге моих всех работ получается, что для окружности метод пресекающихся хорд, метод проверки через векторное умножение и метод проверки условия остроугольности треугольника дают 25%, для эллипса же получается, что метод проверки через векторное умножение даёт 25%, а метод проверки остроугольности треугольника даёт максимум 25%, если эллипс является окружностью, и постепенно уменьшающуюся вероятность, чем сильнее изменяется соотношение полуосей.

## **Дополнительные материалы**

1. Мои сайты с различными реализациями описываемых симуляций

* для эллипса (генерация точек треугольника через градусы) с проверкой треугольника на остроугольность (через уравнение косинуса)

<https://mrzhizhik.github.io/Checking-an-ellipse-through-vector-multiplication---point-through-degrees/>

* для эллипса (генерация точек треугольника через каноническое уравнение эллипса) с проверкой треугольника на остроугольность (через уравнение косинуса)

<https://mrzhizhik.github.io/Checking-the-ellipse-through-vector-multiplication---point-through-the-canonical-equation/>

* для окружности проверкой через хорды

<https://mrzhizhik.github.io/Checking-a-circle-through-chords/>

* для эллипса с проверкой треугольника на остроугольность (через условие углов)

<https://mrzhizhik.github.io/Checking-an-ellipse-for-acute-angles/>

1. Весь листинг программы методом пересекающихся хорд на языке программирования С++.

#include <iostream>

#include <ctime>

#include <math.h>

#include <stdlib.h>

#include <algorithm>

using namespace std;

void hoarasort(int\* a, int first, int last)

{

int i = first, j = last;

double tmp, x = a[(first + last) / 2];

do {

while (a[i] < x)

i++;

while (a[j] > x)

j--;

if (i <= j)

{

if (i < j)

{

tmp = a[i];

a[i] = a[j];

a[j] = tmp;

}

i++;

j--;

}

} while (i <= j);

if (i < last)

hoarasort(a, i, last);

if (first < j)

hoarasort(a, first, j);

}

bool testInbound()

{

int a = rand() % 360;

int b = rand() % 360;

int c = rand() % 360;

const int n = 3;

int s[n] = {a,b,c};

hoarasort(s, 0, 2);

a = s[0];

b = s[1];

c = s[2];

if (((a+180)%360 <c && c < (b+180)%360)||((b + 180) % 360 < c && c < (a + 180)% 360))

return true;

else

return false;

}

int roundTest(int cnt)

{

int hit = 0;

for (int i = 0; i < cnt; i++) {

hit += testInbound();

}

return hit;

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

int cnt = 1000000;

double total = roundTest(cnt);

cout << endl<<" Вероятность при " << cnt<< " повторениях равна: ";

cout << total \*100/ cnt<<"%"<<endl;

return 0;

}

1. Весь листинг программы на языке программирования JavaScript методом проверки через векторное умножение.

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">

<title>Теория вероятностей</title>

<style>

\*, \*::before, \*::after {

box-sizing: border-box;

font-family: 'Roboto', sans-serif;

}

.container {

width: 930px;

margin: 20px auto;

}

.content {

height: 680px;

border-radius: 10px;

background-color: rgb(162, 225, 235);

padding: 16px;

}

.text-field\_\_input {

display: block ;

margin: 0 auto 30px auto;

text-align:center;

width: 60%;

height: 35px;

border-radius:5px;

outline: none;

border: none;

font-family: 'Roboto', sans-serif;

}

input:hover {

cursor: pointer;

}

button {

display: block;

padding:15px;

border:none;

background-color:#3F51B5;

color:#fff;

font-weight:600;

border-radius:5px;

width: 200px;

margin: 20px auto;

font-family: 'Roboto', sans-serif;

transition: all .5s;

}

button:hover {

cursor: pointer;

background-color:rgb(64, 181, 146);

}

.blok\_\_res{

width: 200px;

height: 50px;

margin: 0 auto;

background-color: #fff;

text-align: center;

color: black;

border-radius: 8px;

display: flex;

align-items: center;

justify-content: center;

}

.title\_\_res {

margin: 30px auto 10px auto;

text-align: center;

font-weight: bold;

}

.title\_\_iterazi {

margin: 0 auto 10px auto;

text-align: center;

font-weight: bold;

}

.content\_\_text {

text-align: center;

line-height: 1.5;

width: 890px;

}

</style>

</head>

<body>

<div class="container">

<div class="content">

<h1 class="title\_\_iterazi">Задача</h1>

<div class="content\_\_text">

Дан эллипс с произвольной малой и большой полуокружностью и 3 произвольные точки по периметру эллипса. Точки образуют вершины вписанного треугольника.

Посчитать вероятность того, что центр эллипса будет внутри треугольника.

</div>

<h2 class="title\_\_res">Симуляция</h2>

<div class="title\_\_iterazi">Количетво итераций</div>

<input class="text-field\_\_input" type="text" placeholder="Количетво итераций" id="iterazi">

<div class="title\_\_res">Большая полуокружность</div>

<input class="text-field\_\_input" type="text" placeholder="Большая полуокружность" id="a\_poluos">

<div class="title\_\_res">Малая полуокружности</div>

<input class="text-field\_\_input" type="text" placeholder="Малая полуокружности" id="b\_poluos">

<button onclick="func()">Посчитать</button>

<div class="title\_\_res">Результат</div>

<div class="blok\_\_res">

<div id="res"></div>

</div>

</div>

</div>

<script>

var rndPoint = function(a\_poluos, b\_poluos) {

var x = Math.random()\*(a\_poluos \* 2) - a\_poluos;

if(Math.round(Math.random())) {

var y = Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

} else {

var y = -Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

}

return {x, y};

}

var testInbound = function(a,b,c) {

var x = (a.x - 0) \* (b.y - a.y) - (b.x - a.x) \* (a.y - 0);

var y = (b.x - 0) \* (c.y - b.y) - (c.x - b.x) \* (b.y - 0);

var z = (c.x - 0) \* (a.y - c.y) - (a.x - c.x) \* (c.y - 0);

if (x > 0 && y > 0 && z > 0 || x < 0 && y < 0 && z < 0) return true;

return false;

}

var roundTest = function(a\_poluos, b\_poluos, cnt) {

var hit = 0;

for (i = cnt; i--;) {

var a = rndPoint(a\_poluos, b\_poluos), b = rndPoint(a\_poluos, b\_poluos), c = rndPoint(a\_poluos, b\_poluos);

hit += testInbound(a, b, c);

}

return hit;

}

function func() {

var cnt = Number(document.getElementById("iterazi").value);

var a\_poluos = Number(document.getElementById("a\_poluos").value);

var b\_poluos = Number(document.getElementById("b\_poluos").value);

var test = roundTest(a\_poluos, b\_poluos, cnt);

let result = 100\*test/cnt;

document.getElementById("res").innerHTML = result+" %";

}

</script>

</body>

</html>

1. Листинг программы на языке программирования JavaScript методом проверки остроугольности вписанного треугольника.

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">

    <title>Теория вероятностей</title>

    <style>

        \*, \*::before, \*::after {

        box-sizing: border-box;

        font-family: 'Roboto', sans-serif;

        }

        .container {

            width: 930px;

            margin: 20px auto;

        }

        .content {

            height: 680px;

            border-radius: 10px;

            background-color: rgb(162, 225, 235);

            padding: 16px;

        }

        .text-field\_\_input {

            display: block ;

            margin: 0 auto 30px auto;

            text-align:center;

            width: 60%;

            height: 35px;

            border-radius:5px;

            outline: none;

            border: none;

            font-family: 'Roboto', sans-serif;

        }

        input:hover {

            cursor: pointer;

        }

        button {

            display: block;

            padding:15px;

            border:none;

            background-color:#3F51B5;

            color:#fff;

            font-weight:600;

            border-radius:5px;

            width: 200px;

            margin: 20px auto;

            font-family: 'Roboto', sans-serif;

            transition: all .5s;

        }

        button:hover {

            cursor: pointer;

            background-color:rgb(64, 181, 146);

        }

        .blok\_\_res{

            width: 200px;

            height: 50px;

            margin: 0 auto;

            background-color: #fff;

            text-align: center;

            color: black;

            border-radius: 8px;

            display: flex;

            align-items: center;

            justify-content: center;

        }

        .title\_\_res {

            margin: 30px auto 10px auto;

            text-align: center;

            font-weight: bold;

        }

        .title\_\_iterazi {

            margin: 0 auto 10px auto;

            text-align: center;

            font-weight: bold;

        }

        .content\_\_text {

            text-align: center;

            line-height: 1.5;

            width: 890px;

        }

       </style>

</head>

<body>

    <div class="container">

        <div class="content">

            <h1 class="title\_\_iterazi">Задача</h1>

            <div class="content\_\_text">

                Дан эллипс с произвольной малой и большой полуокружностью и 3 произвольные точки по периметру элипса. Точки образуют вершины вписанного треугольника.

                Посчитать вероятность того, что центр эллипса будет внутри треугольника. Проверка через остоугольность треугольника.

            </div>

            <h2 class="title\_\_res">Симуляция</h2>

            <div class="title\_\_iterazi">Количетво итераций</div>

            <input class="text-field\_\_input" type="text" placeholder="Количетво итераций" id="iterazi">

            <div class="title\_\_res">Большая полуокружность</div>

            <input class="text-field\_\_input" type="text" placeholder="Большая полуокружность" id="a\_poluos">

            <div class="title\_\_res">Малая полуокружности</div>

            <input class="text-field\_\_input" type="text" placeholder="Малая полуокружности" id="b\_poluos">

            <button onclick="func()">Посчитать</button>

            <div class="title\_\_res">Результат</div>

            <div class="blok\_\_res">

                <div id="res"></div>

            </div>

        </div>

    </div>

    <script>

        var rndPoint = function(a\_poluos, b\_poluos) {

        var x = Math.random()\*(a\_poluos \* 2) - a\_poluos;

            if(Math.round(Math.random())) {

                var y = Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

            } else {

                var y = -Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

            }

        return {x, y};

        }

        function sqr(a) {

            return a \* a;

        }

        function isTriangleObtuse(a, b, c) {

                var a1 = Math.sqrt( sqr(b.x - a.x) + sqr(b.y - a.y) ),

                    b1 = Math.sqrt( sqr(c.x - b.x)  + sqr(c.y - b.y) ),

                    c1 = Math.sqrt( sqr(a.x - c.x)  + sqr(a.y - c.y) );

                var A = sqr(a1), B = sqr(b1), C = sqr(c1);

            return A + B > C && A + C > B && B + C > A;

        }

        var roundTest = function(a\_poluos, b\_poluos, cnt) {

            var hit = 0;

            for (i = cnt; i--;) {

                var a = rndPoint(a\_poluos, b\_poluos), b = rndPoint(a\_poluos, b\_poluos), c = rndPoint(a\_poluos, b\_poluos);

                hit += isTriangleObtuse(a, b, c);

            }

            return hit;

        }

        function func() {

        var cnt = Number(document.getElementById("iterazi").value);

        var a\_poluos = Number(document.getElementById("a\_poluos").value);

        var b\_poluos = Number(document.getElementById("b\_poluos").value);

        var test = roundTest(a\_poluos, b\_poluos, cnt);

        let result = 100\*test/cnt;

        document.getElementById("res").innerHTML = result+" %";

        }

    </script>

</body>

</html>

1. Листинг программы на языке JavaScript cравнения вероятности остроугольности вписанного треугольника и вероятности перекрытия центра эллипса.

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">

<title>Теория вероятностей</title>

<style>

\*, \*::before, \*::after {

box-sizing: border-box;

font-family: 'Roboto', sans-serif;

}

.container {

width: 930px;

margin: 20px auto;

}

.content {

height: 680px;

border-radius: 10px;

background-color: rgb(162, 225, 235);

padding: 16px;

}

.text-field\_\_input {

display: block ;

margin: 0 auto 30px auto;

text-align:center;

width: 60%;

height: 35px;

border-radius:5px;

outline: none;

border: none;

font-family: 'Roboto', sans-serif;

}

input:hover {

cursor: pointer;

}

button {

display: block;

padding:15px;

border:none;

background-color:#3F51B5;

color:#fff;

font-weight:600;

border-radius:5px;

width: 200px;

margin: 20px auto;

font-family: 'Roboto', sans-serif;

transition: all .5s;

}

button:hover {

cursor: pointer;

background-color:rgb(64, 181, 146);

}

.blok\_\_res, .blok\_\_res\_2{

width: 200px;

height: 50px;

margin: 0 70px;

background-color: #fff;

text-align: center;

color: black;

border-radius: 8px;

display: flex;

align-items: center;

justify-content: center;

}

.title\_\_res, .title\_\_res\_2 {

margin: 10px auto 10px auto;

text-align: center;

font-weight: bold;

width: 260px;

}

.title\_\_iterazi {

margin: 0 auto 10px auto;

text-align: center;

font-weight: bold;

}

.content\_\_text {

text-align: center;

line-height: 1.5;

width: 890px;

}

.bloci {

margin: 0 auto;

display: flex;

justify-content: center;

}

</style>

</head>

<body>

<div class="container">

<div class="content">

<h1 class="title\_\_iterazi">Задача</h1>

<div class="content\_\_text">

Дан эллипс с произвольной малой и большой полуокружностью и 3 произвольные точки по периметру элипса. Точки образуют вершины вписанного треугольника.

Посчитать вероятность того, что центр эллипса будет внутри треугольника.

</div>

<h2 class="title\_\_res">Симуляция</h2>

<div class="title\_\_iterazi">Количетво итераций</div>

<input class="text-field\_\_input" type="text" placeholder="Количетво итераций" id="iterazi">

<div class="title\_\_res">Большая полуокружность</div>

<input class="text-field\_\_input" type="text" placeholder="Большая полуокружность" id="a\_poluos">

<div class="title\_\_res">Малая полуокружности</div>

<input class="text-field\_\_input" type="text" placeholder="Малая полуокружности" id="b\_poluos">

<button onclick="func()">Посчитать</button>

<div class="bloci">

<div class="bloc\_1">

<div class="title\_\_res">Вероятность перекрытия центра эллипса</div>

<div class="blok\_\_res">

<div id="res"></div>

</div>

</div>

<div class="bloc\_2">

<div class="title\_\_res\_2">Вероятность остроугольности вписанного треугольника</div>

<div class="blok\_\_res\_2">

<div id="res\_2"></div>

</div>

</div>

</div>

</div>

</div>

<script>

var rndPoint = function(a\_poluos, b\_poluos) {

var x = Math.random()\*(a\_poluos \* 2) - a\_poluos;

if(Math.round(Math.random())) {

var y = Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

} else {

var y = -Math.sqrt((Math.pow(b\_poluos, 2)/Math.pow(a\_poluos, 2))\*(Math.pow(a\_poluos, 2)-Math.pow(x, 2)));

}

return {x, y};

}

var testInbound = function(a,b,c) {

var x = (a.x - 0) \* (b.y - a.y) - (b.x - a.x) \* (a.y - 0);

var y = (b.x - 0) \* (c.y - b.y) - (c.x - b.x) \* (b.y - 0);

var z = (c.x - 0) \* (a.y - c.y) - (a.x - c.x) \* (c.y - 0);

if (x > 0 && y > 0 && z > 0 || x < 0 && y < 0 && z < 0) return true;

return false;

}

function sqr(a) {

return a \* a;

}

function isTriangleObtuse(a, b, c) {

var a1 = Math.sqrt( sqr(b.x - a.x) + sqr(b.y - a.y) ),

b1 = Math.sqrt( sqr(c.x - b.x) + sqr(c.y - b.y) ),

c1 = Math.sqrt( sqr(a.x - c.x) + sqr(a.y - c.y) );

var A = sqr(a1), B = sqr(b1), C = sqr(c1);

return (b1\*b1 + c1\*c1 - a1\*a1)/(2\*b1\*c1) < 1 && (b1\*b1 + c1\*c1 - a1\*a1)/(2\*b1\*c1) > 0 &&

(b1\*b1 + a1\*a1 - c1\*c1)/(2\*b1\*a1) < 1 && (b1\*b1 + a1\*a1 - c1\*c1)/(2\*b1\*a1) > 0 &&

(a1\*a1 + c1\*c1 - b1\*b1)/(2\*a1\*c1) < 1 && (a1\*a1 + c1\*c1 - b1\*b1)/(2\*a1\*c1) > 0

}

var roundTest = function(a\_poluos, b\_poluos, cnt) {

var hit = 0;

for (i = cnt; i--;) {

var a = rndPoint(a\_poluos, b\_poluos), b = rndPoint(a\_poluos, b\_poluos), c = rndPoint(a\_poluos, b\_poluos);

hit += isTriangleObtuse(a, b, c);

}

return hit;

}

var roundTest\_2 = function(a\_poluos, b\_poluos, cnt) {

var hit = 0;

for (i = cnt; i--;) {

var a = rndPoint(a\_poluos, b\_poluos), b = rndPoint(a\_poluos, b\_poluos), c = rndPoint(a\_poluos, b\_poluos);

hit += testInbound(a, b, c);

}

return hit;

}

function func() {

var cnt = Number(document.getElementById("iterazi").value);

var a\_poluos = Number(document.getElementById("a\_poluos").value);

var b\_poluos = Number(document.getElementById("b\_poluos").value);

var test = roundTest(a\_poluos, b\_poluos, cnt);

var test\_2 = roundTest\_2(a\_poluos, b\_poluos, cnt);

let result = 100\*test/cnt;

let result\_2 = 100\*test\_2/cnt;

document.getElementById("res").innerHTML = result+" %";

document.getElementById("res\_2").innerHTML = result\_2+" %";

}

</script>

</body>

</html>