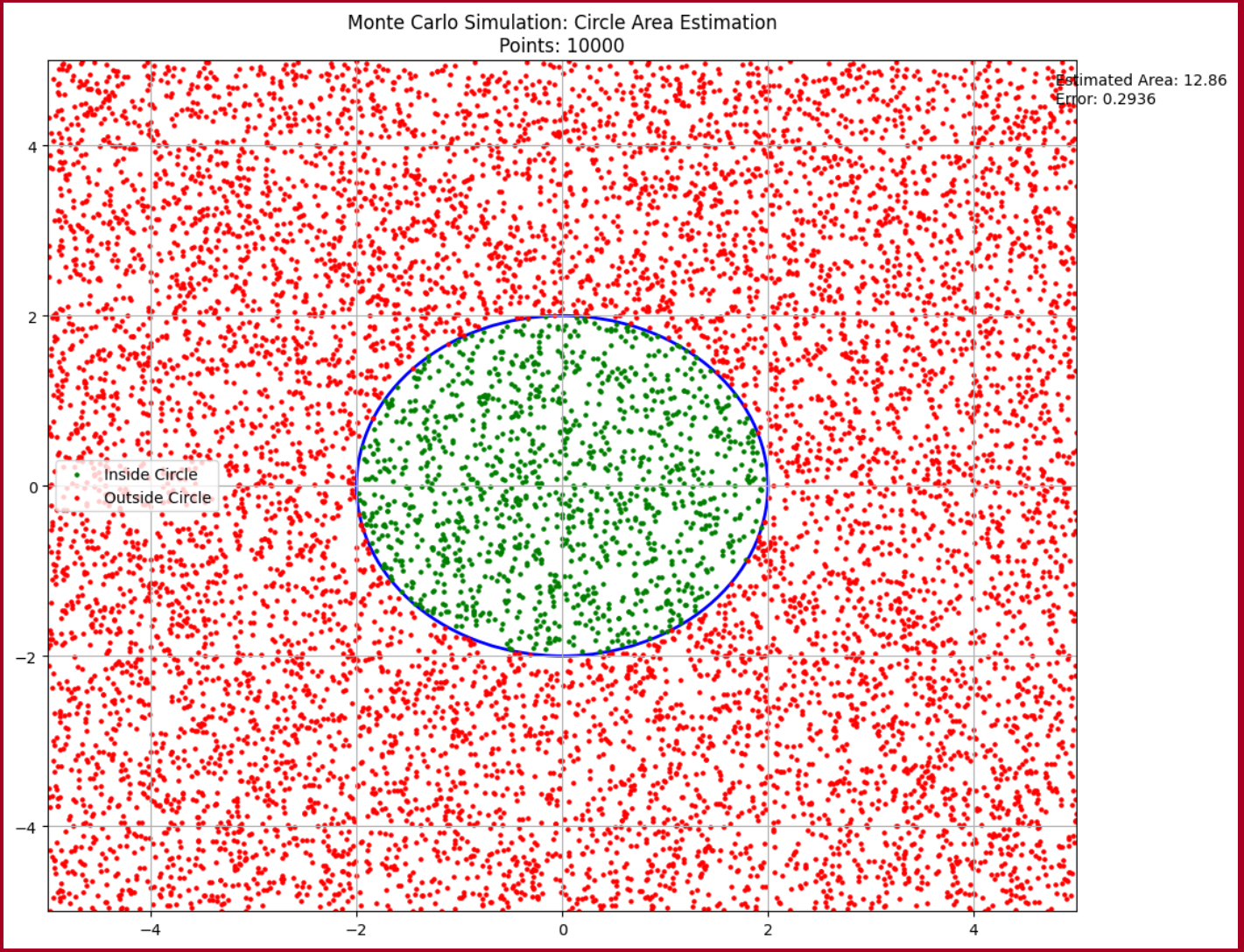
Метод Монте-Карло

Журавский Игорь

Что это за метод?

Метод Монте-Карло - это численный метод решения математических задач при помощи моделирования случайных величин

Идея метода



- не попали

- попали

Преимущества

1. Простота понимания и реализации
2. Универсальность применения
3. Точность

Недостатки

1. Зависимость от количества точек
2. Случайная природа
3. Неэффективность в высокоразмерных пространствах

Область применения

1. моделирование облучения твёрдых тел ионами в физике;
2. моделирование поведения разреженных газов
3. исследования поведения разных тел при столкновении
4. алгоритмы оптимизации и нахождения кратчайшего пути решения
5. решение сложных интегралов (или когда их очень много)
6. предсказание астрономических наблюдений
7. поиск в дереве в различных алгоритмах

Задача

Даны фигуры на координатной плоскости: треугольник, круг и квадрат. Нам известны их площади. Нужно рассчитать площади фигур с помощью метода Монте-Карло и сравнить вычисленное значение с реальным.

Круг

```
buljad *
class Circle:
    buljad
    def __init__(self, cx, cy, radius):
        self.cx = cx
        self.cy = cy
        self.radius = radius

    1 usage (1 dynamic)  buljad
    def is_inside(self, x, y):
        return (x - self.cx) ** 2 + (y - self.cy) ** 2 <= self.radius ** 2

    new *
    def area(self):
        return np.pi * self.radius ** 2
```

Квадрат

```
buljad *
class Square:
    buljad
    def __init__(self, sx_min, sx_max, sy_min, sy_max):
        self.sx_min = sx_min
        self.sx_max = sx_max
        self.sy_min = sy_min
        self.sy_max = sy_max

    1 usage (1 dynamic)   buljad
    def is_inside(self, x, y):
        return self.sx_min <= x <= self.sx_max and self.sy_min <= y <= self.sy_max

    new *
    def area(self):
        return (self.sx_max - self.sx_min) * (self.sy_max - self.sy_min)
```

Треугольник

```
class Triangle:
    buljad
    def __init__(self, vertices):
        self.vertices = vertices

    1 usage (1 dynamic)  buljad
    def is_inside(self, x, y):
        x1, y1 = self.vertices[0]
        x2, y2 = self.vertices[1]
        x3, y3 = self.vertices[2]

        A = 0.5 * np.abs(x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2))
        A1 = 0.5 * np.abs(x * (y2 - y3) + x2 * (y3 - y) + x3 * (y - y2))
        A2 = 0.5 * np.abs(x1 * (y - y3) + x * (y3 - y1) + x3 * (y1 - y))
        A3 = 0.5 * np.abs(x1 * (y2 - y) + x2 * (y - y1) + x * (y1 - y2))

        return np.isclose(A, A1 + A2 + A3)

    new *
    def area(self):
        x1, y1 = self.vertices[0]
        x2, y2 = self.vertices[1]
        x3, y3 = self.vertices[2]
        return 0.5 * np.abs(x1 * (y2 - y3) + x2 * (y3 - y1) + x3 * (y1 - y2))
```

Функция расчета площади

Estimated Area =
(Total Number of Points) /
(Number of Points Inside Shape)
× Area of Domain

Функция расчета площади

```python
def calculate_area_monte_carlo(shape, area_domain, num_points):
    x_min, x_max, y_min, y_max = area_domain
    inside_shape_count = 0

    for _ in range(num_points):
        x = random.uniform(x_min, x_max)
        y = random.uniform(y_min, y_max)

        if shape.is_inside(x, y):
            inside_shape_count += 1

    area_of_domain = (x_max - x_min) * (y_max - y_min)
    estimated_area = (inside_shape_count / num_points) * area_of_domain

    return estimated_area
```

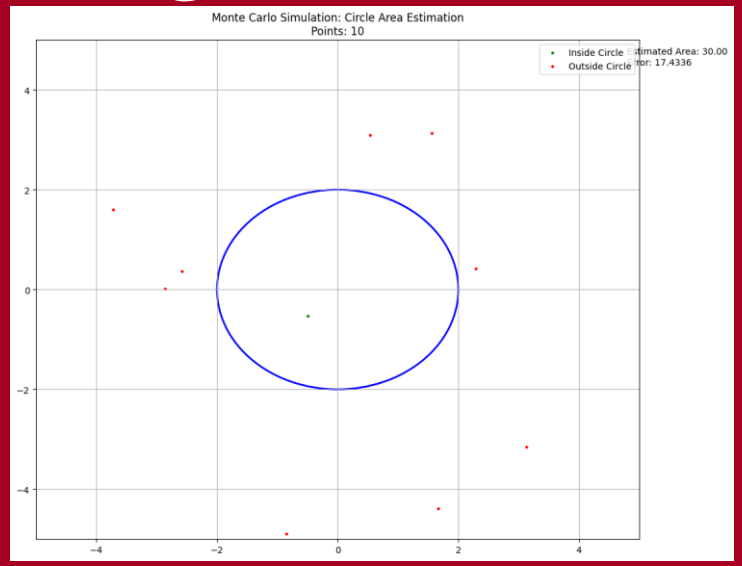
Расчет ошибки

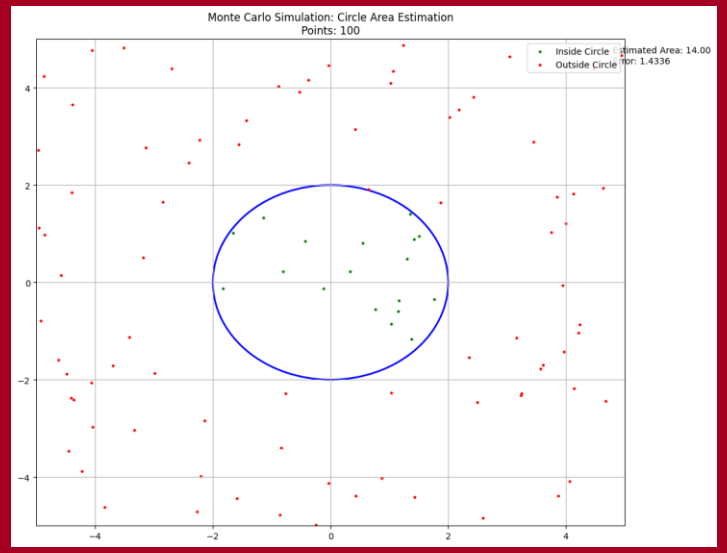
```
3 usages    buljad
def calculate_error(true_area, estimated_area):
    return np.abs(estimated_area - true_area)
```

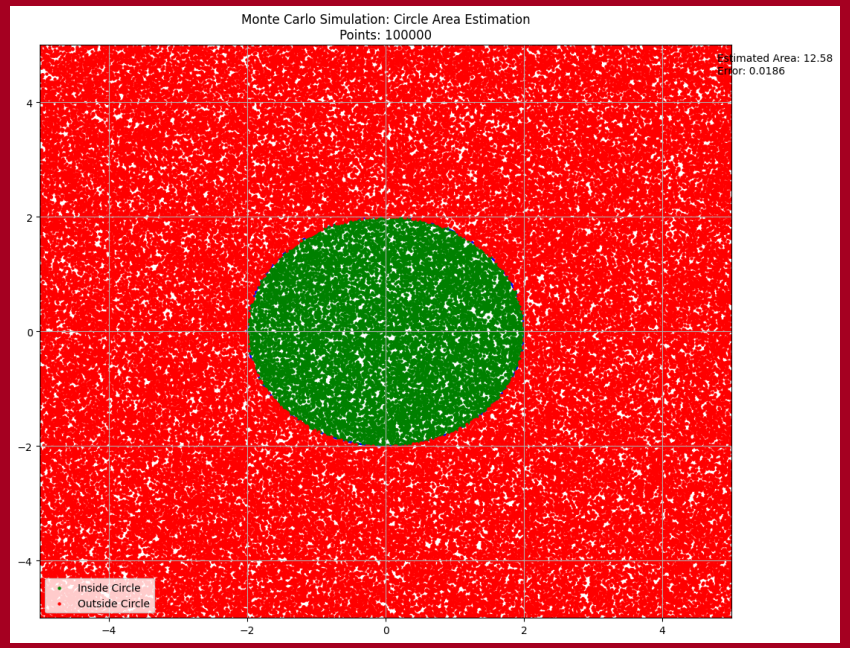
Параметры фигур

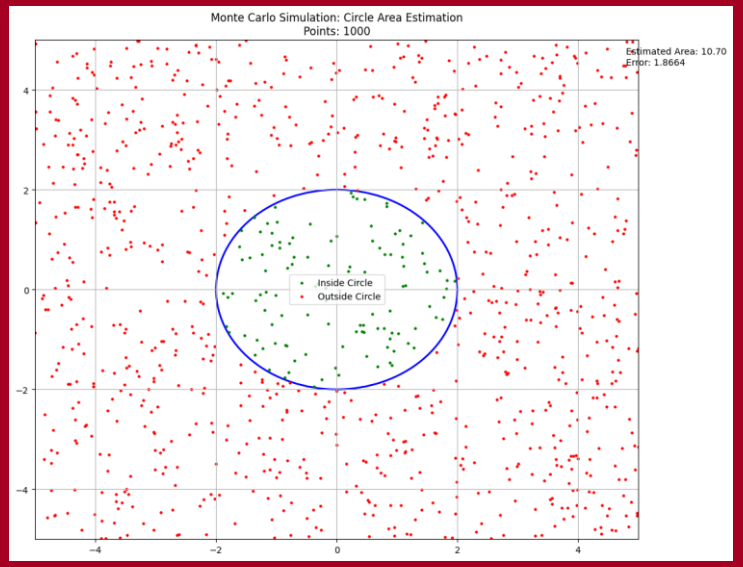
```
# Параметры и создание объектов для круга, квадрата и треугольника
circle = Circle( cx: 0,  cy: 0,  radius: 2)
square = Square(-2,  sx_max: 2, -2,  sy_max: 2)
triangle = Triangle([(0, 0), (3, 0), (1.5, 2)])


# Область измерения
area_domain = (-5, 5, -5, 5)  # (x_min, x_max, y_min, y_max)
```

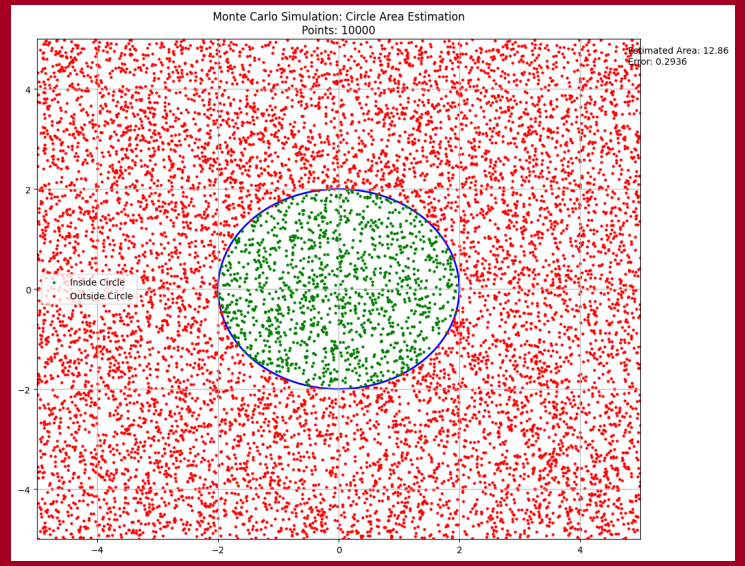
Визуализация проставления точек



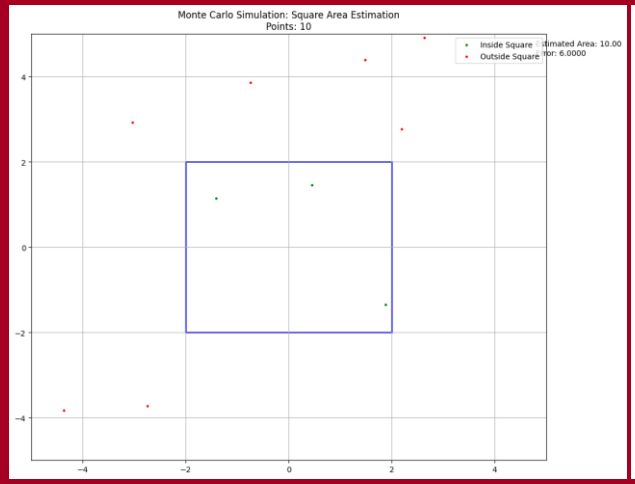


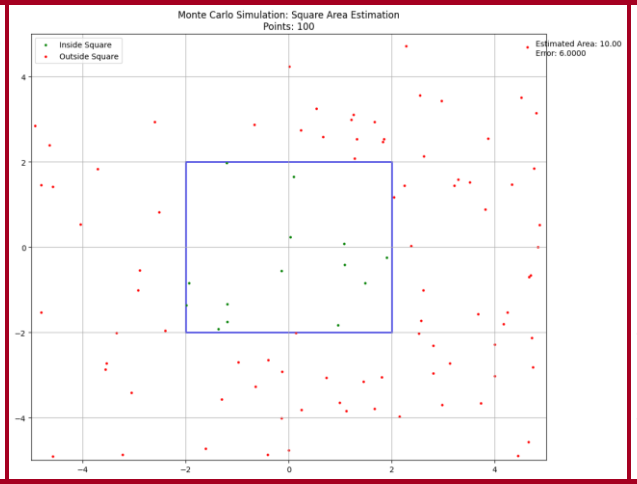


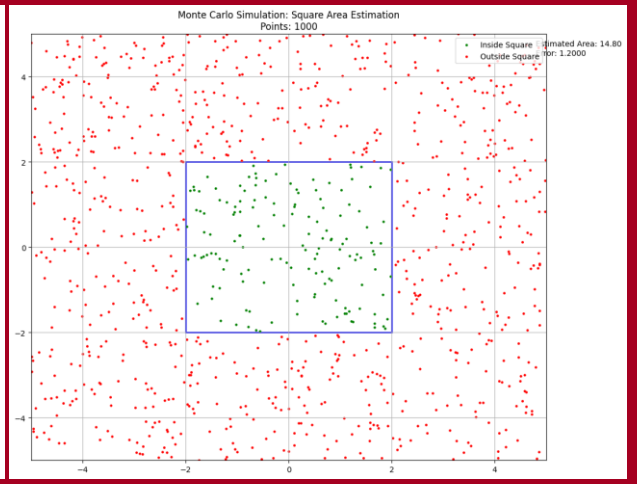


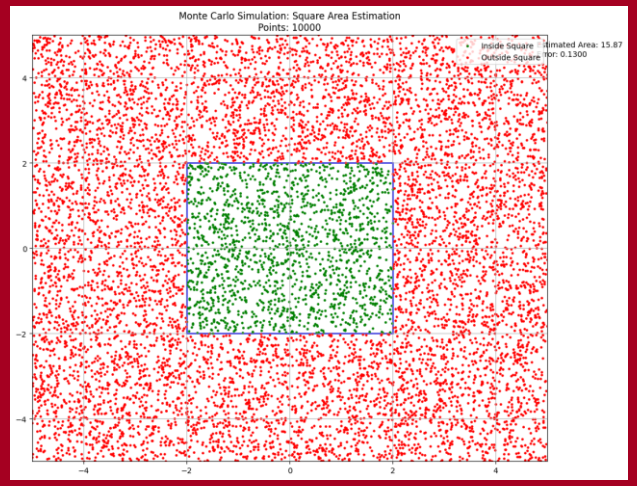


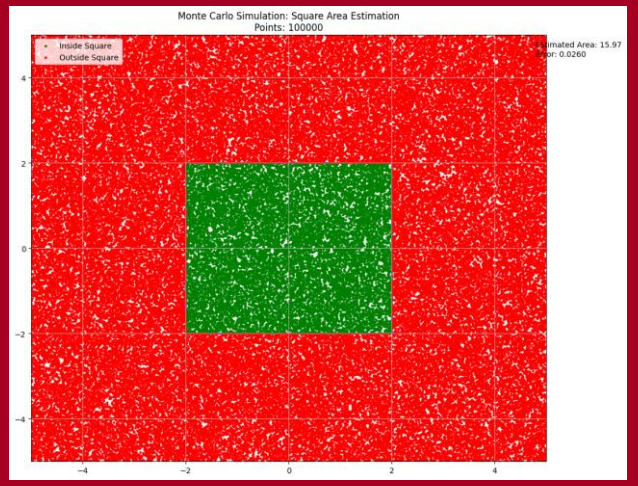
Визуализация проставления точек



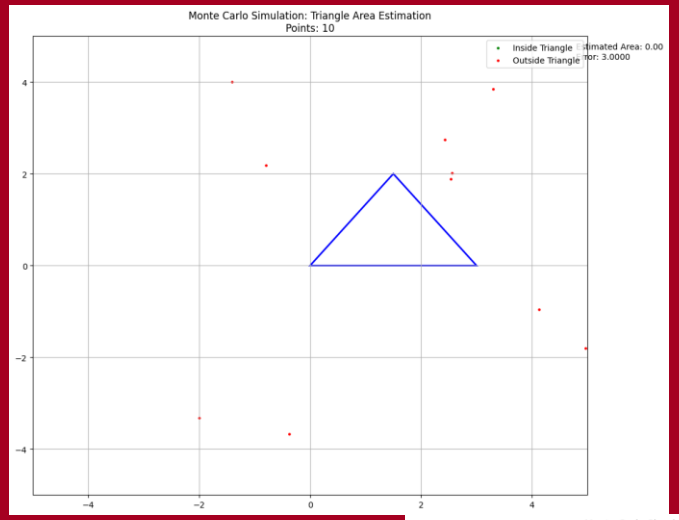


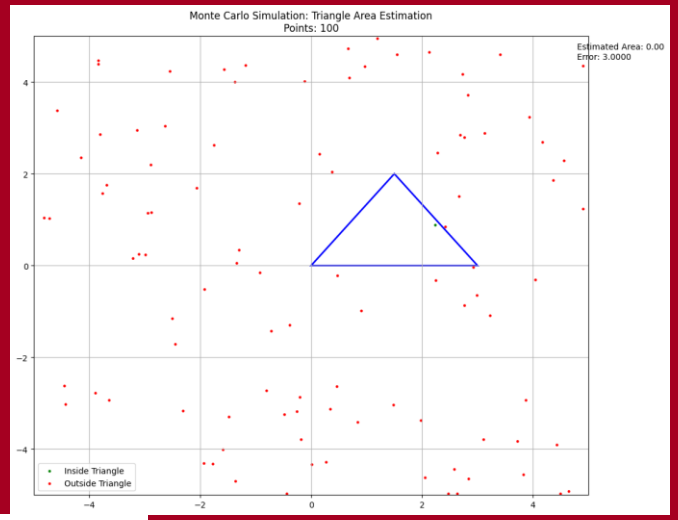


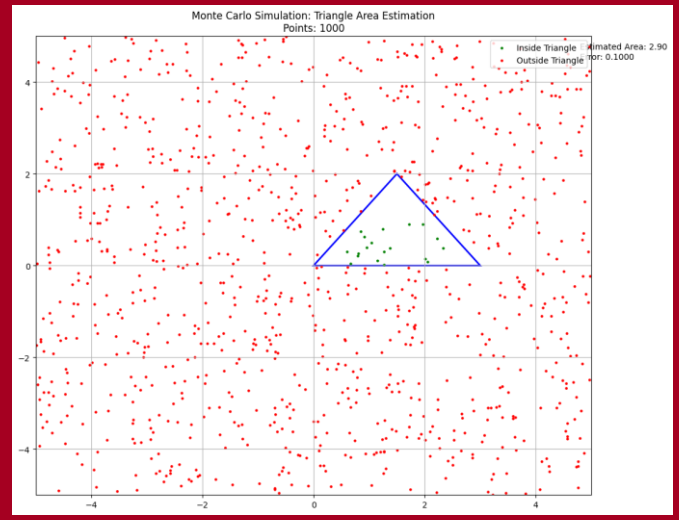


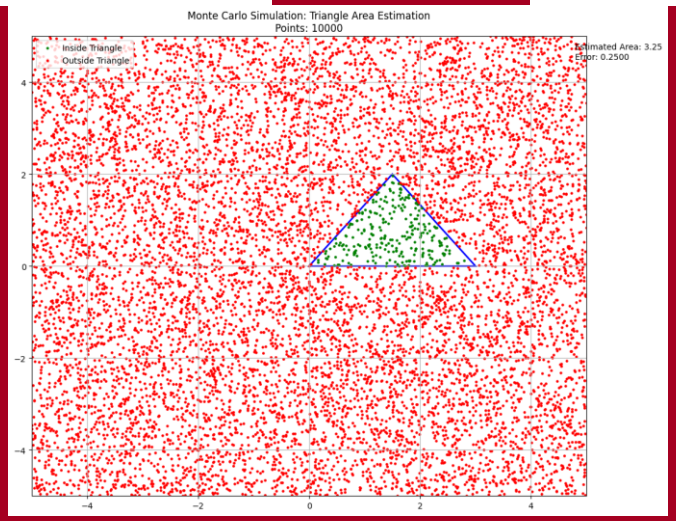


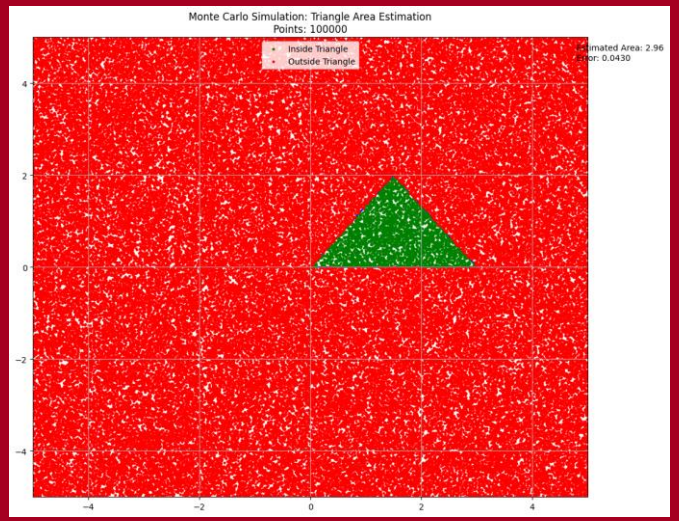
Визуализация проставления точек







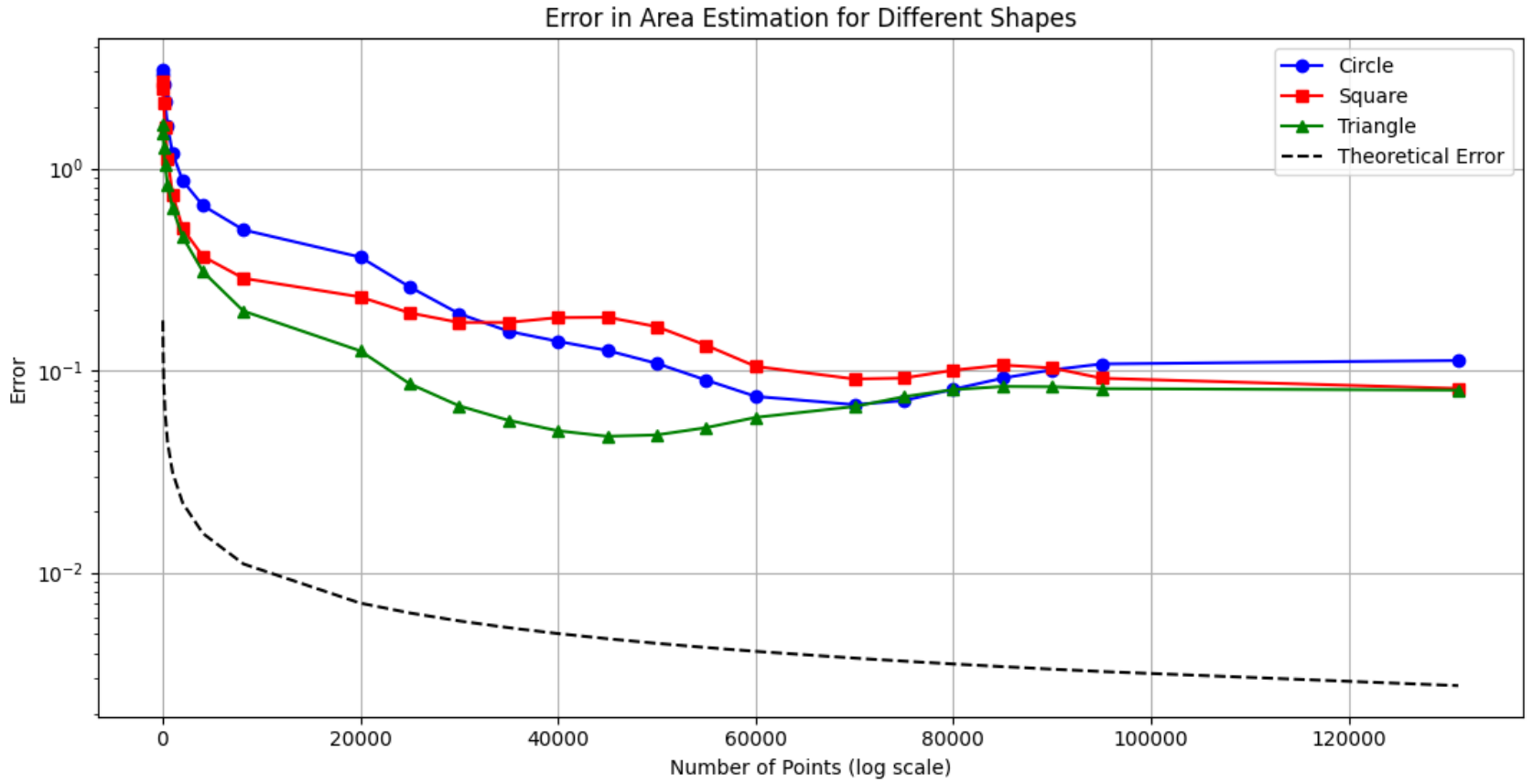




Расчет ошибки

$$Error = \frac{1}{\sqrt{NumberOfPoints}}$$

График ошибки



Итоговое время работы

```
Total time for Circle calculations: 2.4743 seconds
Total time for Square calculations: 1.6556 seconds
Total time for Triangle calculations: 88.0995 seconds

Total elapsed time for the program: 92.2304 seconds
```

Всем Спасибо!
Готов ответить на вопросы!