Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №1

Название Изучение функций распределения и функций плотности распределения

Дисциплина Моделирование

Студент Зайцева А. А.

Группа ИУ7-72Б

Оценка (баллы) ______________________

Преподаватель Рудаков И. В.

Москва — 2022 г.

1 Задание

Разработать программу для построения графиков функции распределения и функции плотности распределения для следующих распределений:

- равномерное распределение;
- распределение Пуассона (вариант 3).

Разработать графический интерфейс, предоставляющий возможность выбора закона распределения и указания его параметров.

2 Теоретические сведения

2.1 Равномерное распределение

Функция плотности распределения $f(x)$ случайной величины X, имеющей равномерное распределение на отрезке $[a, b]$ ($X \sim R(a, b)$), где $a, b \in R$, имеет следующий вид:

$$f(x) = \begin{cases} \frac{1}{b-a}, & x \in [a, b] \\ 0, & \text{иначе.} \end{cases} \tag{1}$$

Соответствующая функция распределения $F(x) = \int_{-\infty}^{x} f(t)dt$ принимает вид:

$$F(x) = \begin{cases} 0, & x < a, \\ \frac{x-a}{b-a}, & x \in [a, b] \\ 1, & x > b \end{cases} \tag{2}$$

2.2 Распределение Пуассона

Биномиальное распределение с параметрами n и p – это распределение количества «успехов» в последовательности из n независимых случайных экспериментов, таких, что вероятность «успеха» в каждом из них постоянна и равна p.

Распределение Пуассона – это частный случай биномиального распределения при $n \gg 0$ и $p \to 0$. Распределение Пуассона также называют законом «редких» событий, так как оно всегда проявляется там, где производится большое число испытаний, в каждом из которых с малой вероятностью происходит «редкое» событие.

Дискретная случайная величина X имеет закон распределения Пуассона с параметром λ ($X \sim \Pi(\lambda)$), где $\lambda > 0$, если она принимает значения $0, 1, 2, ...$ с вероятностями:

$$P(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}, \quad k \in \{0, 1, 2, ...\} \tag{3}$$

Параметр λ распределения Пуассона – это среднее количество успешных испытаний в заданной области возможных исходов.

Соответствующая функция распределения принимает вид:

$$F(x) = P(X < x) = \sum_{k=0}^{x-1} P(X = k) = e^{-\lambda}\sum_{k=0}^{x-1}\frac{\lambda^k}{k!} \tag{4}$$

Для дискретной случайной величины не существует функции плотности распределения вероятностей.

3 Результаты работы программы

3.1 Равномерное распределение

На рисунках 1 и 2 приведены результаты построения графиков функций плотности $f(x)$ и распределения $F(x)$ для случайных величин $X \sim R(-4, 4)$ и $X \sim R(1, 3)$, соответственно.

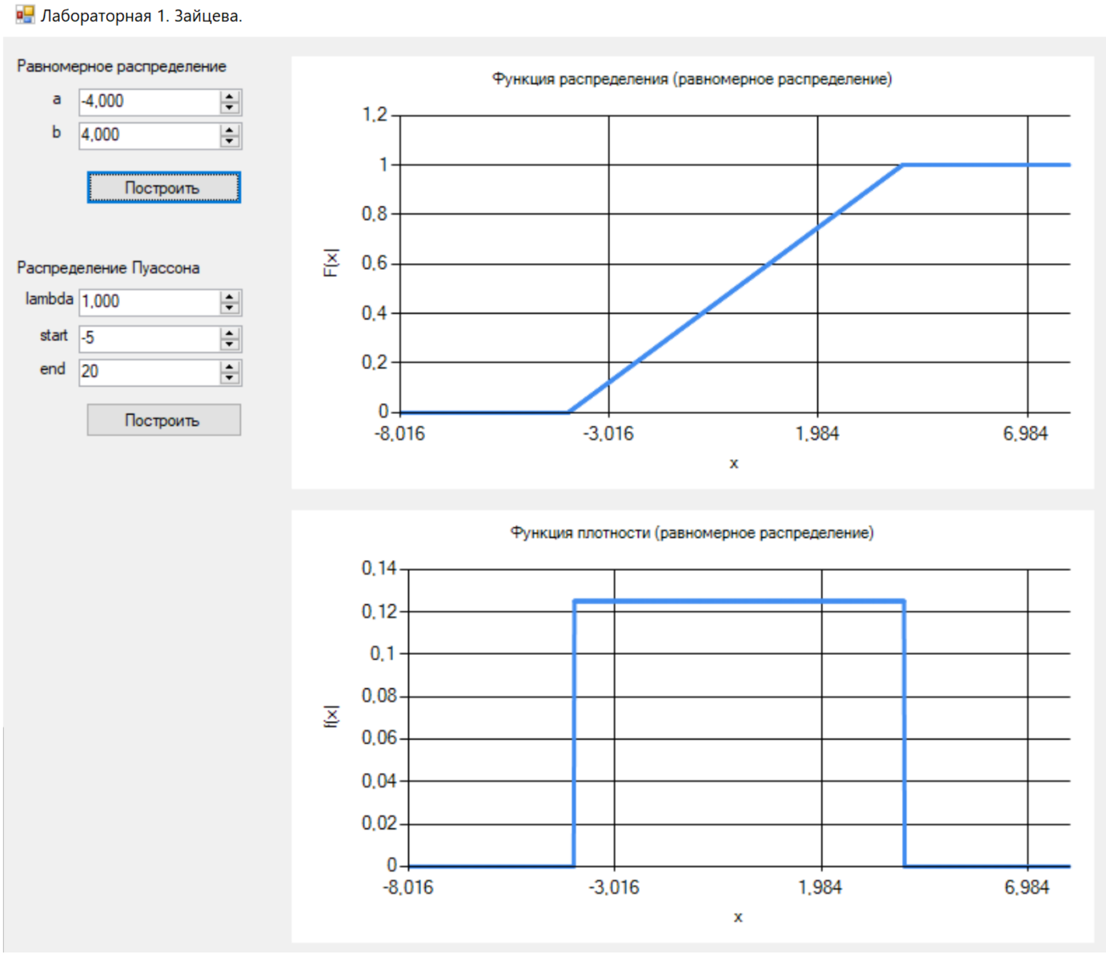

Рисунок 1 – Графики функций плотности $f(x)$ и распределения $F(x)$ для случайной величины $X \sim R(-4, 4)$.

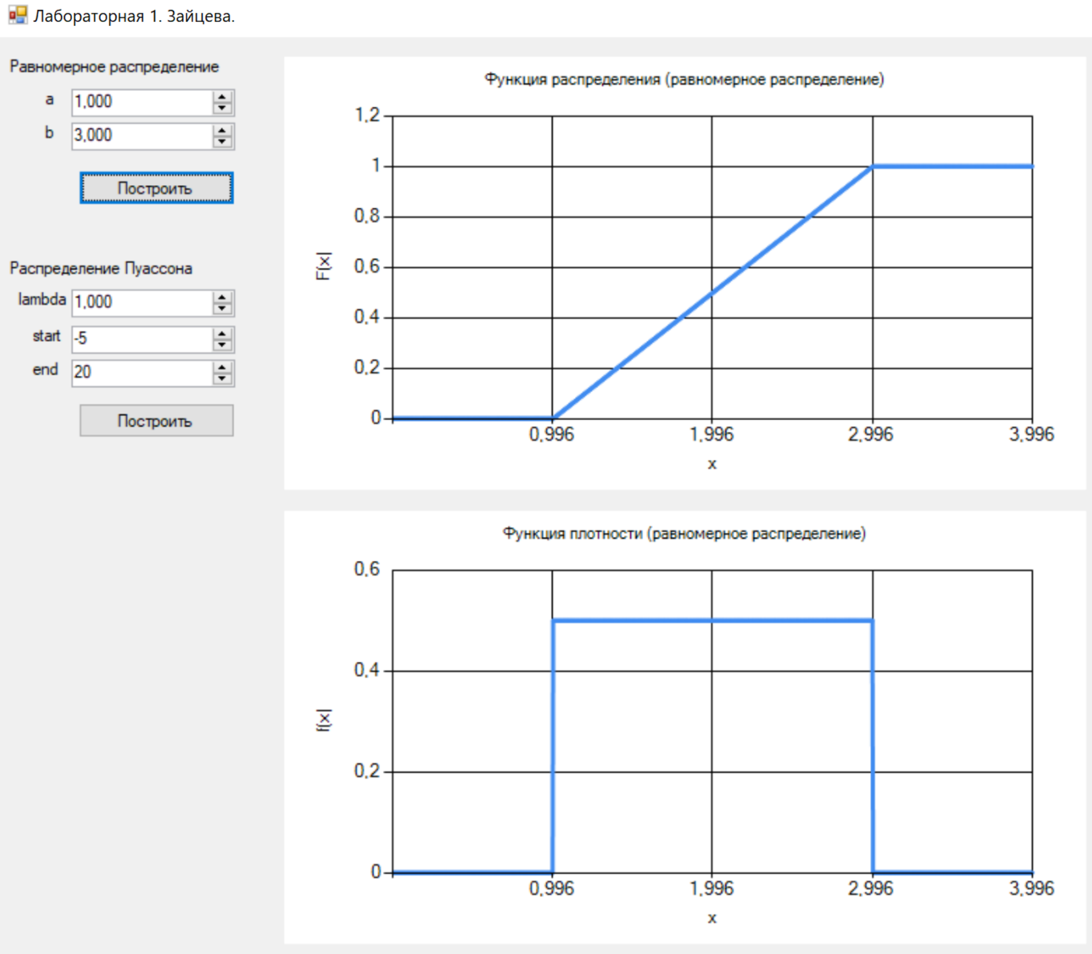

Рисунок 2 – Графики функций плотности $f(x)$ и распределения $F(x)$ для случайной величины $X \sim R(1,3)$.

3.2 Распределение Пуассона

На рисунках 3, 4 и 5 приведены результаты построения графиков функции вероятности $P(x)$ и распределения $F(x)$ на отрезке $x \in [-3, 20]$ для случайных величин $X \sim \Pi(1)$, $X \sim \Pi(3)$ и $X \sim \Pi(10)$, соответственно.

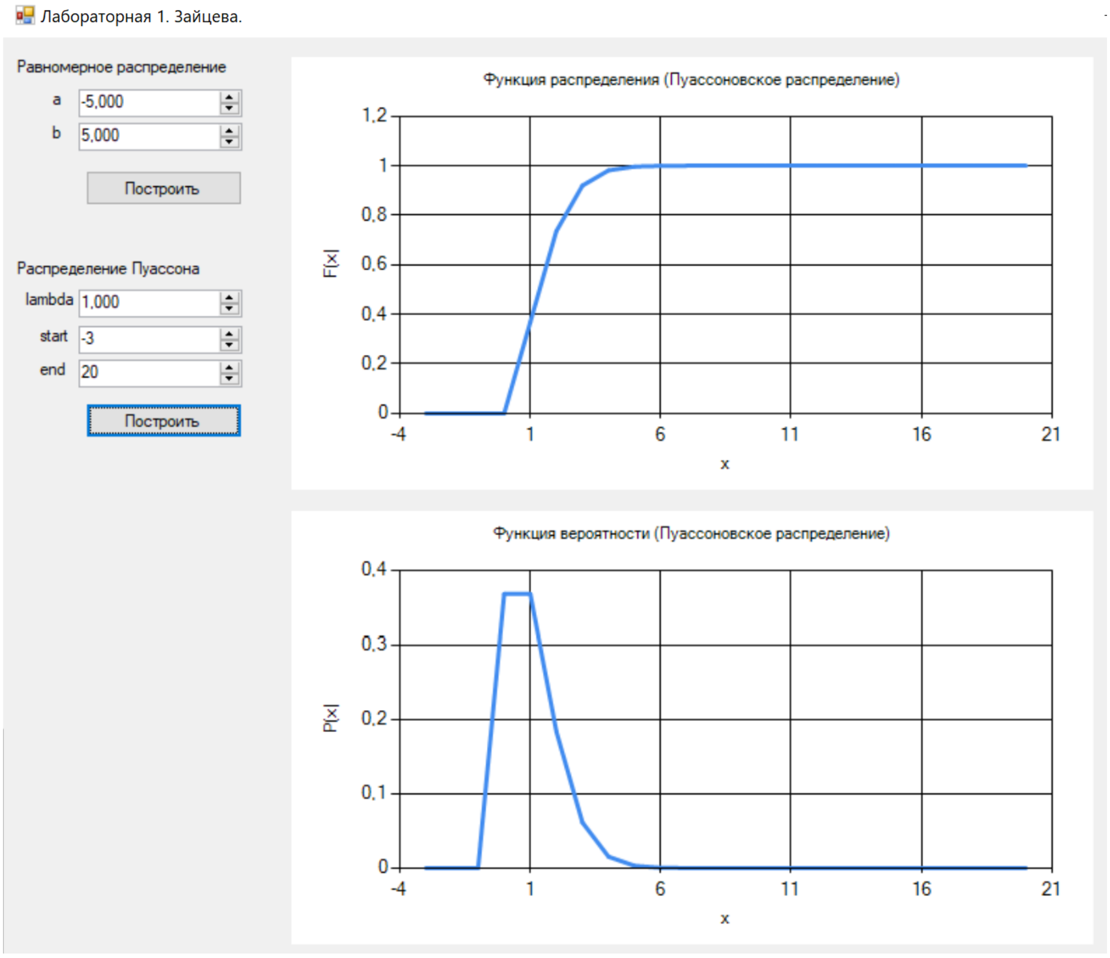

Рисунок 3 – Графики функций вероятности $P(x)$ и распределения $F(x)$ для случайной величины $X \sim \Pi(1)$.

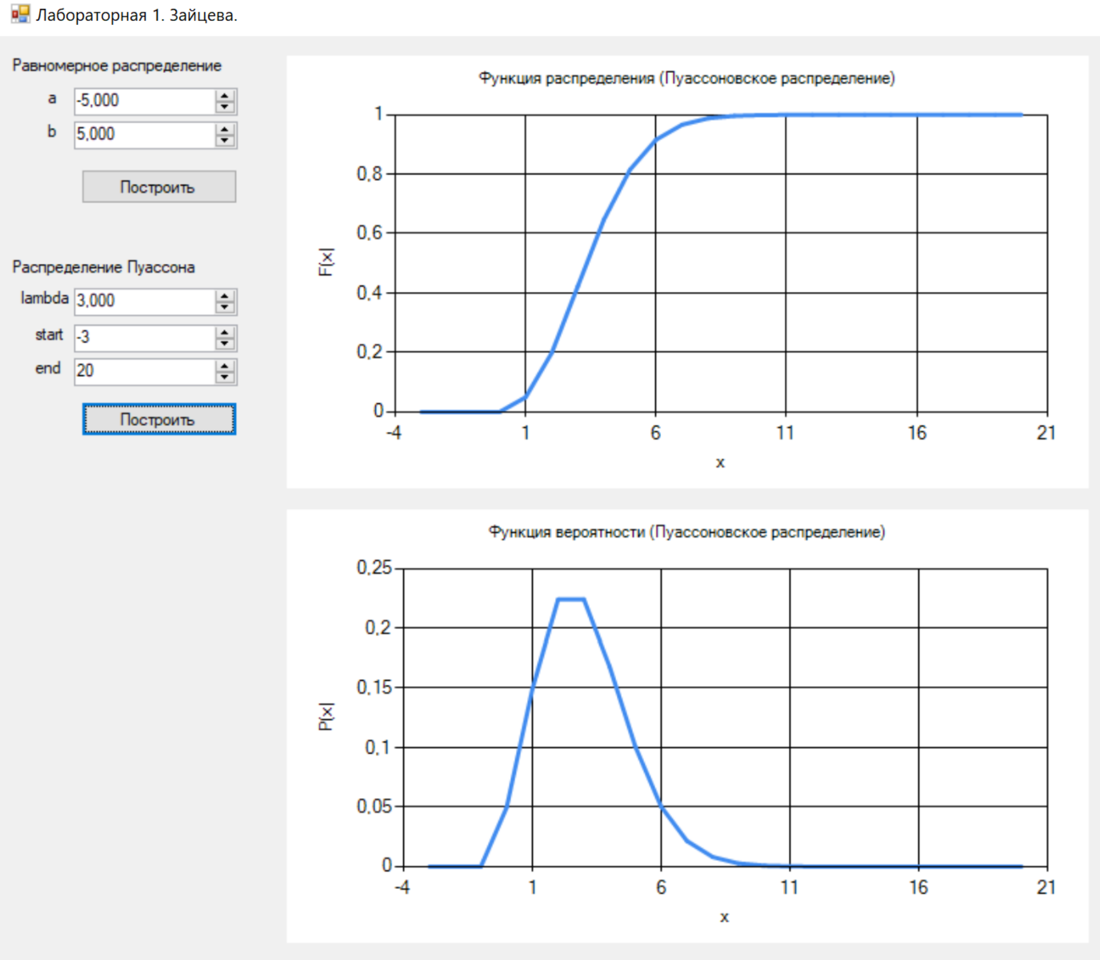

Рисунок 4 – Графики функций вероятности $P(x)$ и распределения $F(x)$ для случайной величины $X \sim \Pi(3)$.

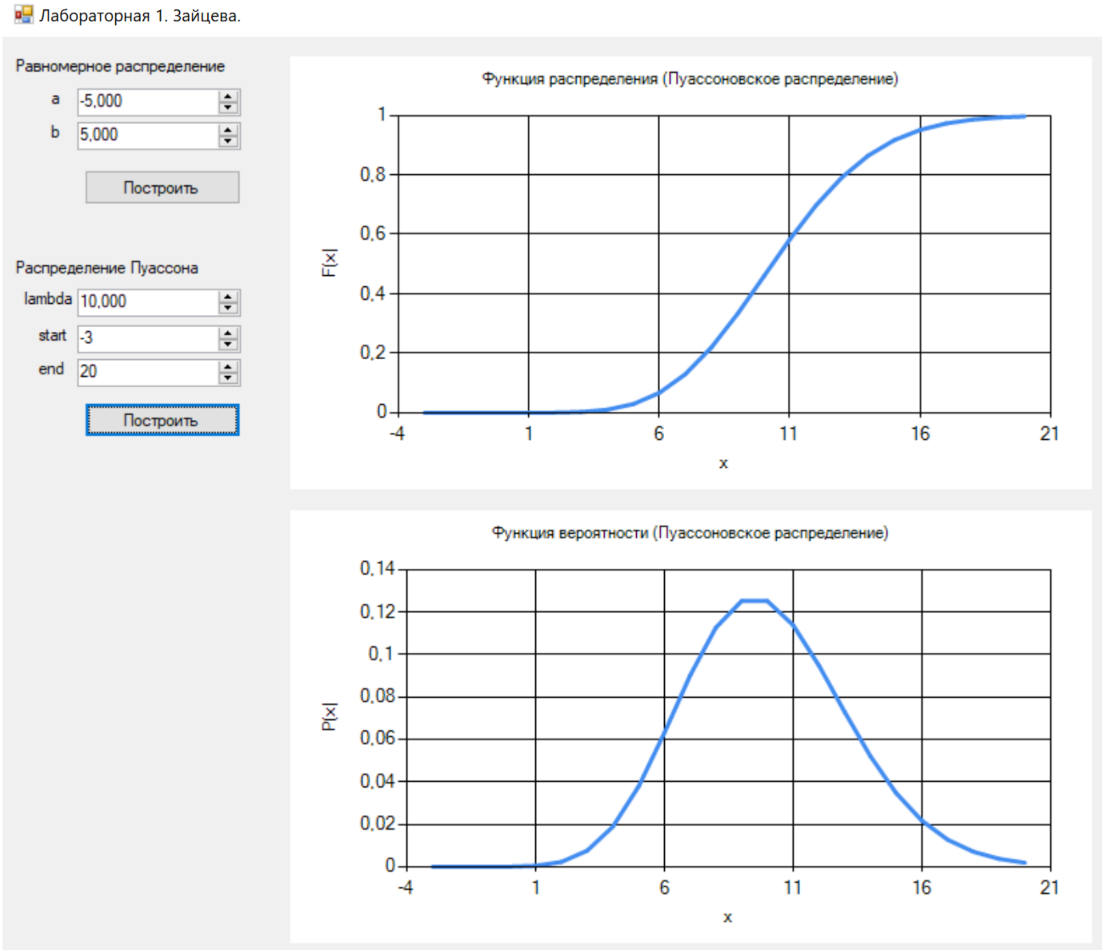

Рисунок 5 – Графики функций вероятности $P(x)$ и распределения $F(x)$ для случайной величины $X \sim \Pi(10)$.

4 Код программы

В листинге 1 приведены классы и их методы, используемые для построения графиков функций (использованный язык программирования – C#).

Листинг 1 – Реализация построения графиков функций

```
public class EqualDistribution
{
    private double a;
    private double b;
    private double p;

    public EqualDistribution(double a, double b)
```

```
    {
        this.a = a;
        this.b = b;
        this.p = 1 / (b - a);
    }

    private  double f(double x)
    {
        if ((x < a) || (x > b))
            return 0;
        else
            return p;
    }

    private double F(double x)
    {
        if (x < a)
            return 0;
        else if (x < b)
            return (x - a) * p;
        else
            return 1;
    }

    public void buildPlots(Chart chartDistr, Chart chartDens, double
   GraphStep = 1000)
    {
        var range = (b - a) * 2;
        var begin = (a + b - range) / 2;
        var end = (a + b + range) / 2;
        var step = range / GraphStep;

        for (double x = begin; x <= end; x += step)
        {
            chartDistr.Series[0].Points.AddXY(x, F(x));
            chartDens.Series[0].Points.AddXY(x, f(x));
        }
    }
}

```

```
public class PuassonDistribution
{
    private double lambda;
    private double exp_lambda;
    private int begin;
    private int end;

    public PuassonDistribution(double lambda, int begin, int end)
    {
        this.lambda = lambda;
        this.begin = begin;
        this.end = end;
        this.exp_lambda = Math.Exp(-this.lambda);
    }

    public double P(int x)
    {
        if (x < 0)
            return 0;
        else
            return exp_lambda * Math.Pow(lambda, x) / factorial(x);
    }

    public double F(int x)
    {
        if (x < 0)
            return 0;

        double sum = 0;
        for (int k = 0; k < x; k++)
            sum += P(k);

        return sum;
    }
    private long factorial(int n)
    {
        if (n < 2) return 1;

        return n * factorial(n - 1);
    }
```

```
    public void buildPlots(Chart chartDistr, Chart chartDens, double
   GraphStep = 1000)
    {
        int step = 1;

        for (int x = begin; x <= end; x += step)
        {
            chartDistr.Series[0].Points.AddXY(x, F(x));
            chartDens.Series[0].Points.AddXY(x, P(x));
        }
    }
}
```