МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 52

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

кандидат техн. наук, доцент		Марковская Н.В.
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

ИССЛЕДОВАНИЕ ИНТЕНСИВНОСТИ ОТКАЗОВ ДЛЯ НЕВОССТАНАВЛИВАЕМЫХ СИСТЕМ

по курсу: Надежность инфокоммуникационных систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 5912		Нам Д.О.
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

1. Цель работы

Исследовать интенсивность отказов для невосстанавливаемых систем путем проведения имитационного моделирования функционирования невосстанавливаемой системы.

2. Ход работы

$\lambda(t) = \frac{n_t - n_{t+\Delta t}}{n_t} * \frac{1}{\Delta}$ - формула интенсивности отказов

$R(t) = e^{-\lambda t}$ - формула надежности

В текущей работе, $\lambda_1 = 0.8$, $\lambda_2 = 1.2$, $p_1 = 0.45$, $p_2 = 0.55$, $n = 1000000$

1. Интенсивность отказов в первом периоде

$R(t) = e^{\lambda_1 t} p_1 + e^{\lambda_2 t} p_2$ - теоретическая формула надежности для первого периода

$R'(t) = -\lambda_1 e^{-\lambda_1 t} p_1 - \lambda_2 e^{-\lambda_2 t} p_2$

$\hat{\lambda}(t) = -\frac{-\lambda_1 e^{-\lambda_1 t} p_1 - \lambda_2 e^{-\lambda_2 t} p_2}{e^{\lambda_1 t} p_1 + e^{\lambda_2 t} p_2}$ - теоретическая формула интенсивности отказов

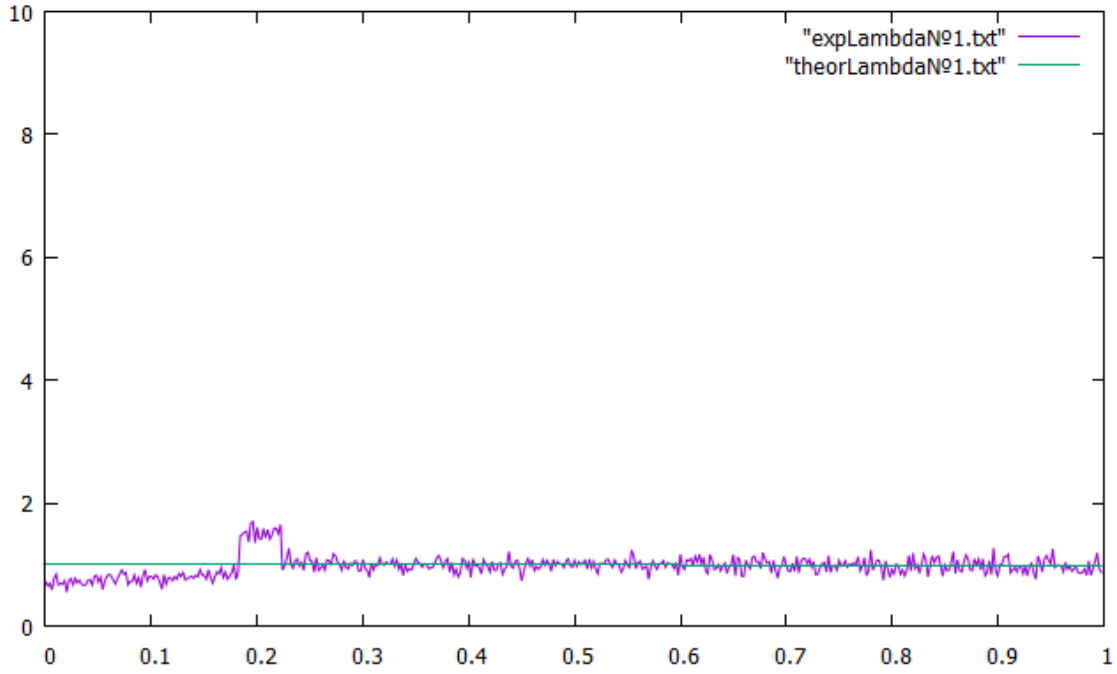

Рисунок 1 - График теоретической и экспериментальной зависимости λ от t

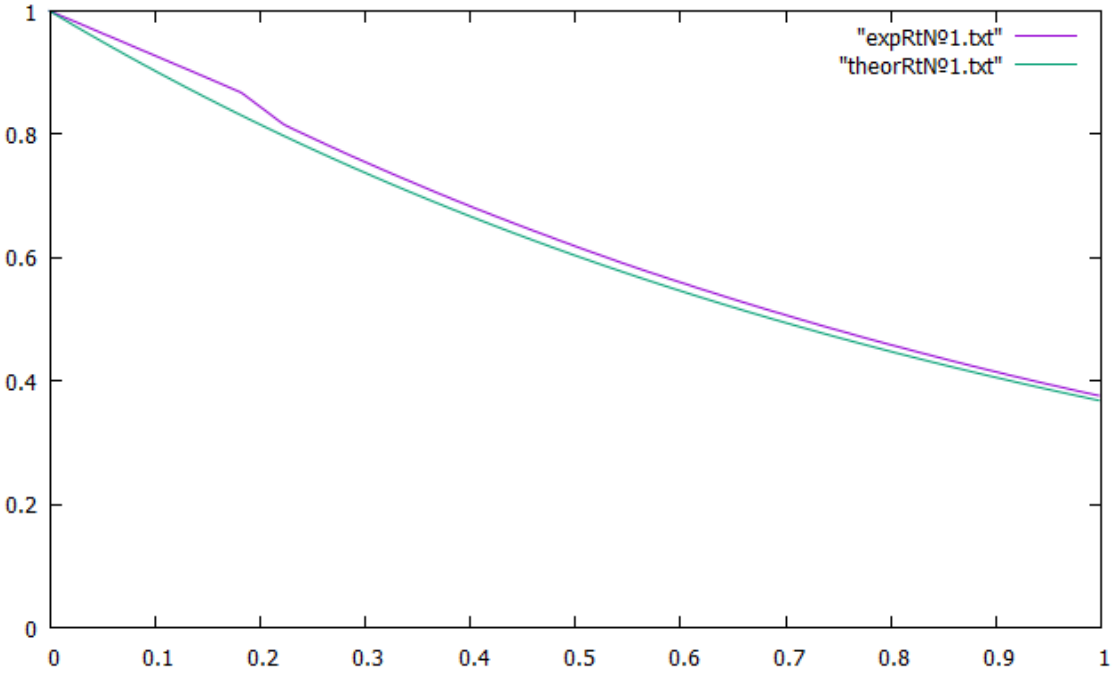

Рисунок 2 - График теоретической и экспериментальной зависимости R от t

2. Интенсивность отказов во втором периоде

$R(t) = e^{-\lambda_1 t} e^{-\lambda_2 t} = e^{-(\lambda_1+\lambda_2)t}$ - теоретическая формула надежности

$R'(t) = -(\lambda_1 + \lambda_2) e^{-(\lambda_1+\lambda_2)t}$

$\lambda(t) = -\frac{-(\lambda_1 + \lambda_2) e^{-(\lambda_1+\lambda_2)t}}{e^{-(\lambda_1+\lambda_2)t}} = (\lambda_1 + \lambda_2)$ - теоретическая формула интенсивности отказов

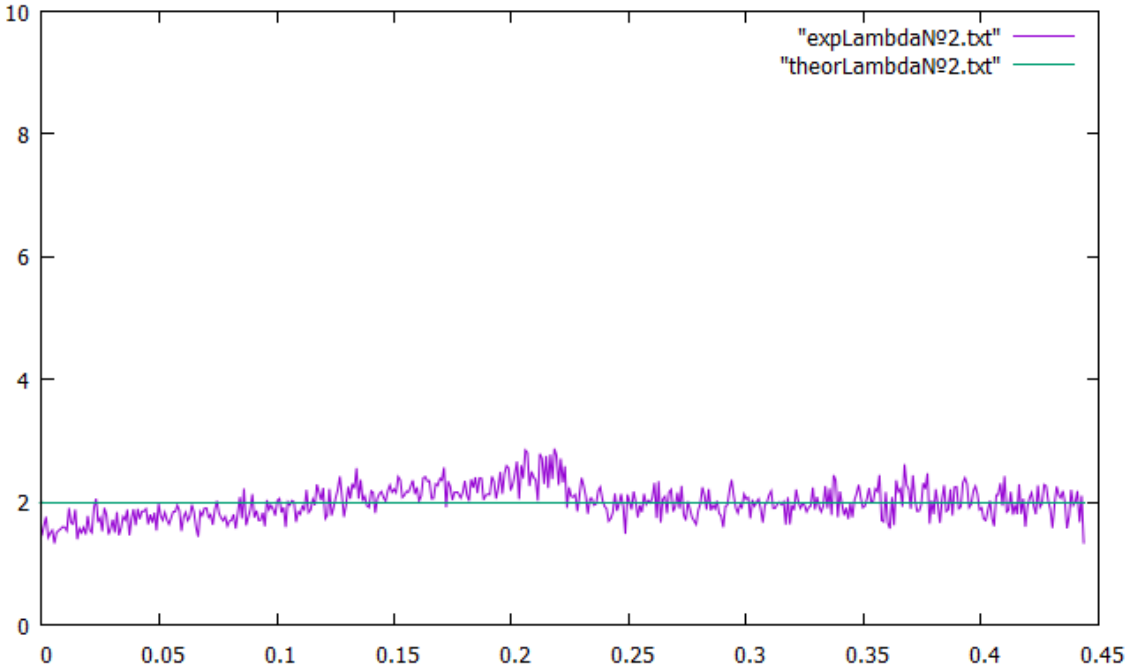

Рисунок 3 - График теоретической и экспериментальной зависимости λ от t

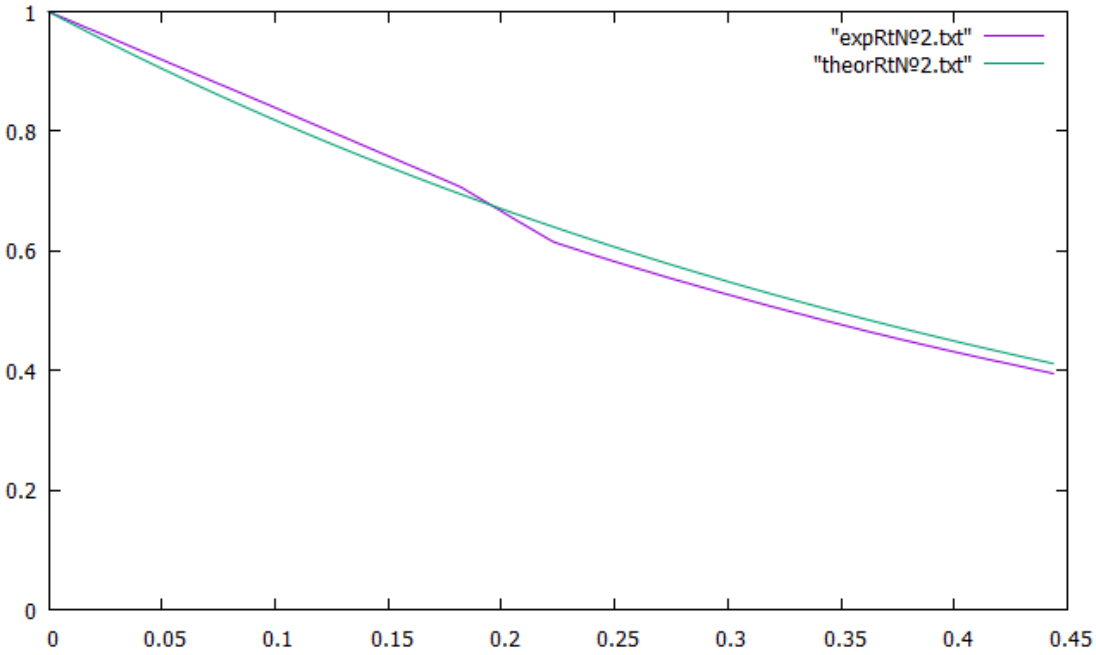

Рисунок 4 - График теоретической и экспериментальной зависимости R от t

3. Интенсивность отказов в третьем периоде

$R(t) = e^{-\lambda_1 t} + e^{-\lambda_2 t} - e^{-(\lambda_1+\lambda_2)t}$ - теоретическая формула надежности

$R'(t) = -\lambda_1 e^{-\lambda_1 t} - \lambda_2 e^{-\lambda_2 t} + (\lambda_1 + \lambda_2) e^{-(\lambda_1+\lambda_2)t}$

$\lambda(t) = -\dfrac{-\lambda_1 e^{-\lambda_1 t} - \lambda_2 e^{-\lambda_2 t} + (\lambda_1 + \lambda_2) e^{-(\lambda_1+\lambda_2)t}}{e^{-\lambda_1 t} + e^{-\lambda_2 t} - e^{-(\lambda_1+\lambda_2)t}}$ - теоретическая формула интенсивности отказов

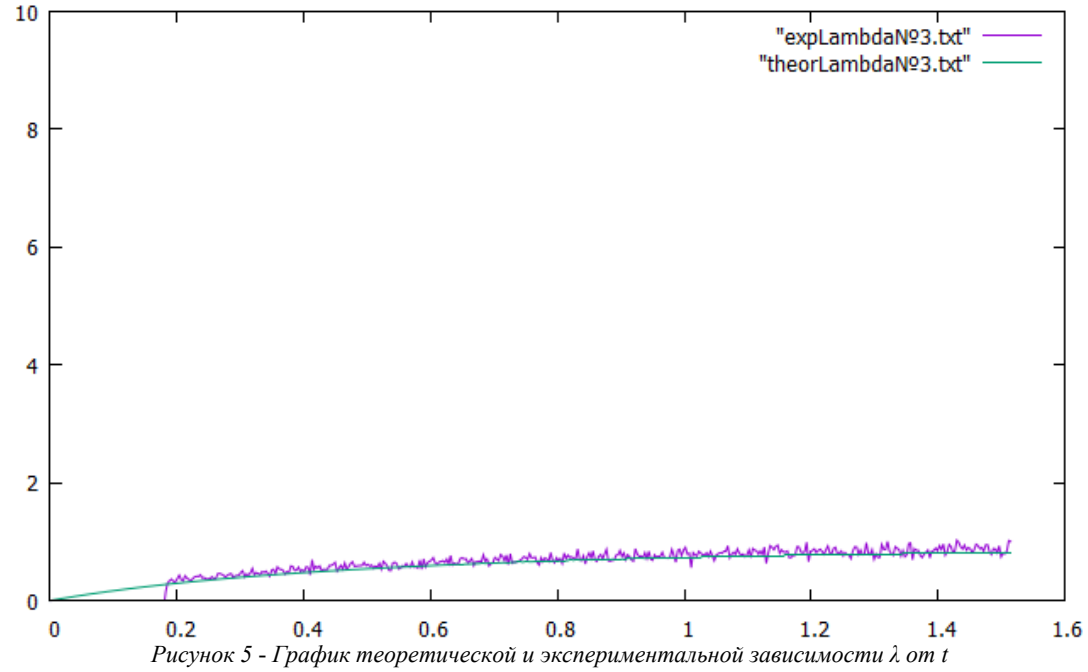

Рисунок 5 - График теоретической и экспериментальной зависимости λ от t

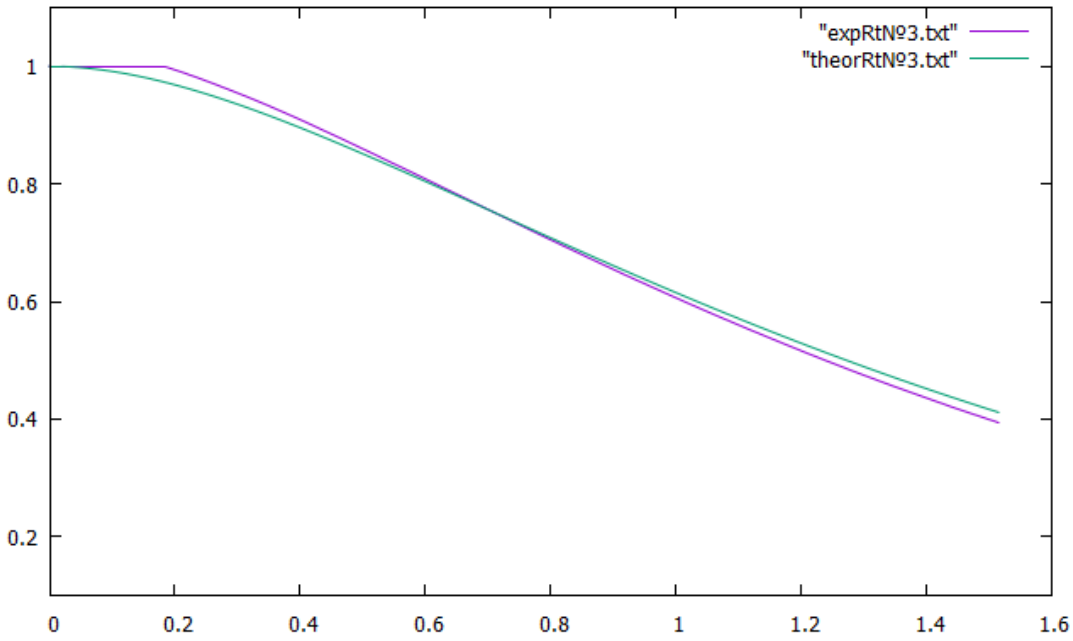

Рисунок 6 - График теоретической и экспериментальной зависимости R от t

3. Вывод

Была исследована интенсивность отказов для невосстанавливаемых систем с 3 периодами жизни, путем проведения имитационного моделирования каждого период, а также построены теоретические и экспериментальные графики $\lambda(t)$ и $R(t)$. В данном случае теоретические и экспериментальные значения имеют некоторую разницу, что отображено на графиках.

Приложение А

Листинг программы

```
public class RIS3 {
    int N = 500;

    public void Period(int period, double[] lambda, double p, int n) throws
IOException {
        double[] T = T(lambda, p, period, n);

        LinkedList<Double> expLambda = new LinkedList<>();
        LinkedList<Double> expRt = new LinkedList<>();

        LinkedList<Double> theorLambda = new LinkedList<>();
        LinkedList<Double> theorRt = new LinkedList<>();

        double Tm = 0;
        for (double t : T) {
            Tm += t;
        }

        Tm /= T.length;
        double dt = Tm / N;
        double nt, nt_dt, R1, R2, Rt, RDt;

        for (double t = 0; t < Tm; t += dt) {
            nt = nt(T, t);
            nt_dt = nt(T, t + (0.1 * dt));

            expLambda.add((nt - nt_dt) / (nt * dt * 0.1));
            expRt.add(nt / n);

            R1 = Ri(lambda[0], t);
            R2 = Ri(lambda[1], t);
            Rt = 0;

            if (period == 1) {
                Rt = R1 * p + R2 * (1 - p);
                RDt = (-lambda[0] * R1 * p - lambda[1] * R2 * (1 - p));
                theorLambda.add(-RDt / Rt);
            }
            else if (period == 2) {
                Rt = R1 * R2;
                theorLambda.add(lambda[0] + lambda[1]);
            }
            else if (period == 3) {
                Rt = R1 + R2 - R1 * R2;
                RDt = ((-lambda[0] * R1 - lambda[1] * R2) + (lambda[0] +
lambda[1]) * R1 * R2);
                theorLambda.add(-RDt / Rt);
            }
            theorRt.add(Rt);
        }
        writeIntoFile("expLambda№" + period + ".txt", expLambda, dt);
        writeIntoFile("expRt№" + period + ".txt", expRt, dt);
        writeIntoFile("theorLambda№" + period + ".txt", theorLambda, dt);
        writeIntoFile("theorRt№" + period + ".txt", theorRt, dt);
    }

    public double Ri(double lambda, double t) {
        return exp(-lambda * t);
    }
```

```
    public double[] T(double[] lambda, double p, int period, int n) {
        double[] T = new double[n];
        for (int i = 0; i < n; i++) {
            if (period == 1)
                T[i] = -log(random() / ((i < p * n) ? lambda[0] :
lambda[1]));
            else if (period == 2)
                T[i] = min(-log(random() / lambda[0]), -log(random() /
lambda[1]));
            else if (period == 3)
                T[i] = max(-log(random() / lambda[0]), -log(random() /
lambda[1]));
        }
        return T;
    }

    public int nt(double[] T, double t) {
        int nt = 0;
        for (double i : T) {
            if (t < Math.abs(i))
                nt++;
        }
        return nt;
    }

    public double rnd() {
        return new Fibonacci().ZRand().get(5);
    }

    public void writeIntoFile(String filename, LinkedList<Double> arr, double
dt) throws IOException {
        File file = new File(filename);
        FileWriter writer = new FileWriter(file);
        double t = 0;
        for (Double a : arr) {
            writer.append(Double.toString(t)).append("
").append(Double.toString(a)).append("\n");
            t += dt;
        }
        writer.flush();
        writer.close();
    }
}
```