

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение  
высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ  
УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

---

КАФЕДРА №51

ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

Доцент, КТН

должность, уч. степень, звание

подпись, дата

Н.В.Марковская

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

по курсу: НАДЕЖНОСТЬ ИНФОКОММУНИКАЦИОННЫХ СИСТЕМ

СТУДЕНТ ГР. №

5912

номер группы

подпись, дата

Б.А.Карханин

инициалы, фамилия

Санкт-Петербург  
2022

## Оглавление

Цель работы .....	3
Исходные данные .....	3
Ход работы.....	3
Имитационное моделирование процесса функционирования невосстанавливаемых систем. .....	3
1. Моделирование первого периода .....	3
2. Моделирование второго периода .....	5
3. Моделирование третьего периода.....	6
Выводы.....	8
Листинг программы .....	9

## Цель работы

Исследование интенсивности отказов и функции надежности для невосстанавливаемых систем путем имитационного моделирования процесса функционирования невосстанавливаемой системы для трех периодов жизни системы. Построение зависимости  $\lambda(t)$  оценки интенсивности отказов от времени и функции надежности  $R(t)$ .

## Исходные данные

Моделирование осуществляется для  $N = 35\,000$  систем, которые поделены на  $k = 2$  групп, причем вероятность попадания в первую группу  $p_1 = 0.3$ , а во вторую  $p_2 = 0.7$ . Интенсивность отказа для первой системы  $\lambda_1 = 0.7$  и для второй  $\lambda_2 = 1.3$ .

## Ход работы

### Имитационное моделирование процесса функционирования невосстанавливаемых систем.

При имитационном моделировании необходимо провести  $N$  экспериментов. В каждом эксперименте моделируется процесс функционирования одного экземпляра системы. Для  $i$ -ой системы моделирование состоит в вычислении значения случайной величины  $T_i$  – времени работы этой системы до момента отказа. Для каждого периода жизни системы случайная величина  $T_i$  вычисляется по своему алгоритму.

### 1. Моделирование первого периода

Для  $i$ -ой системы первоначально надо определить, к какому подмножеству она относится. Для этого используется распределение  $p_i$ . После того, как номер  $i$  подмножества определен, необходимо сгенерировать значение  $T_i$ , как случайной величины, распределенной

поэкспоненциальному закону с параметром  $\lambda_i$ , который определяется по номеру подмножества. Определение  $T_i$  осуществляется по формуле:

$$T_i = \frac{-\ln [0,1]}{\lambda_i}$$

Среднее время работы  $i$ -ой системы определяется по формуле:

$$\bar{T} = \sum_{i=0}^n T_i$$

Затем для каждого  $t$  до  $\bar{T}$  с шагом  $\Delta t = 0.01$  происходит подсчет систем, которые работают.

Теоретическое значение функции надежности:

$$R(t) = R_1(t)p_1 + R_2(t)p_2 = e^{-\lambda_1 t}p_1 + e^{-\lambda_2 t}p_2$$

Экспериментальное значение функции надежности:

$$\hat{R}(t) = \frac{n_t}{n}$$

где,  $n_t$  – число систем, работающих в момент времени  $t$ ,  $n$  – общее число систем.

Теоретическое значение интенсивности отказа:

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

Экспериментальное значение интенсивности отказа:

$$\hat{\lambda}(t) = \frac{n_t - n_{t+\Delta t}}{n_t \Delta t}$$

где  $n_t$  – число работоспособных систем в момент  $t$ ,  $n_{t+\Delta t}$  – число систем, работающих в момент  $t + \Delta t$ , где  $\Delta t = 0.001$ .

Графики:

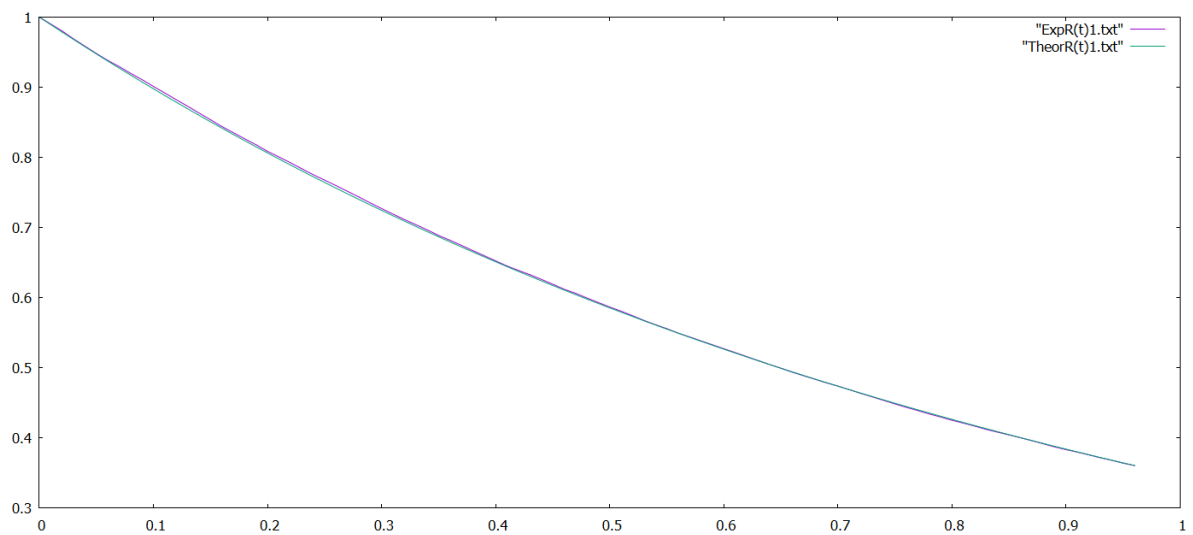


Рисунок 1. Функция надежности для 1-го периода

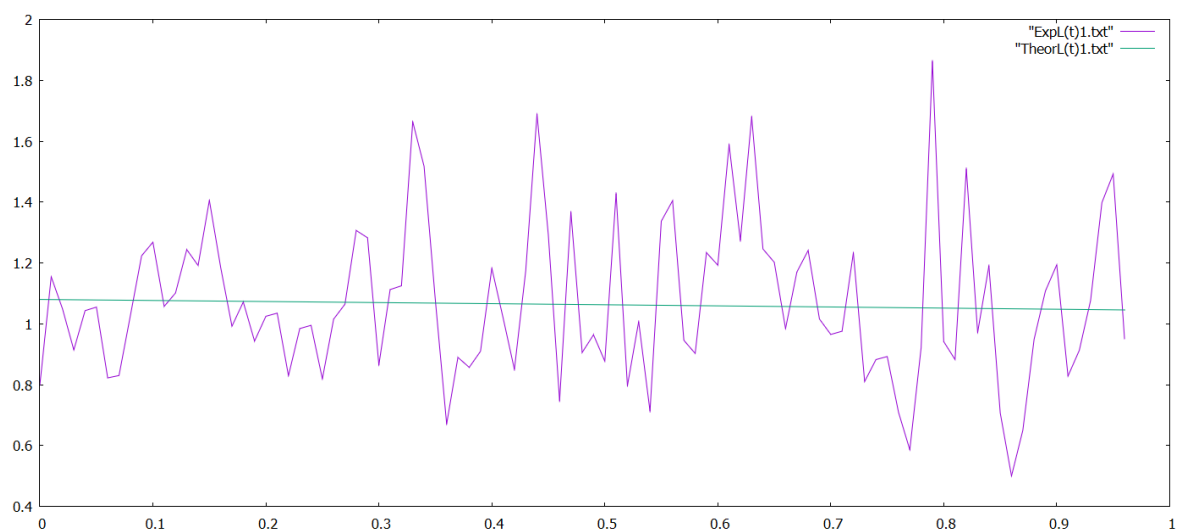


Рисунок 2. Интенсивность отказа для 1-го периода

## 2. Моделирование второго периода

Для  $i$ -ой системы первоначально надо определить, к какому подмножеству она относится. Для этого используется распределение  $p_i$ . После того, как номер  $i$  подмножества определен, необходимо сгенерировать значение  $T_i$ , как случайной величины, распределенной по экспоненциальному закону с параметром  $\lambda_i$ , который определяется по номеру подмножества. Значение времени  $i$ -ой системы определяется как

$$T = \min T_i$$

Среднее время работы  $i$ -ой системы определяется по формуле:

$$\bar{T} = \sum_{i=0}^n T_i$$

Затем для каждого  $t$  до  $\bar{T}$  с шагом  $\Delta t = 0.01$  происходит подсчет систем, которые работают.

Теоретическое значение функции надежности:

$$R(t) = R_1(t)R_2(t) = e^{-(\lambda_1 + \lambda_2)t}$$

Экспериментальное значение функции надежности:

$$\hat{R}(t) = \frac{n_t}{n}$$

где,  $n_t$  – число систем, работающих в момент времени  $t$ ,  $n$  – общее число систем.

Теоретическое значение интенсивности отказа:

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

Экспериментальное значение интенсивности отказа:

$$\hat{\lambda}(t) = \frac{n_t - n_{t+\Delta t}}{n_t \Delta t}$$

где  $n_t$  – число работоспособных систем в момент  $t$ ,  $n_{t+\Delta t}$  – число систем, работающих в момент  $t + \Delta t$ , где  $\Delta t = 0.001$ .

Графики:

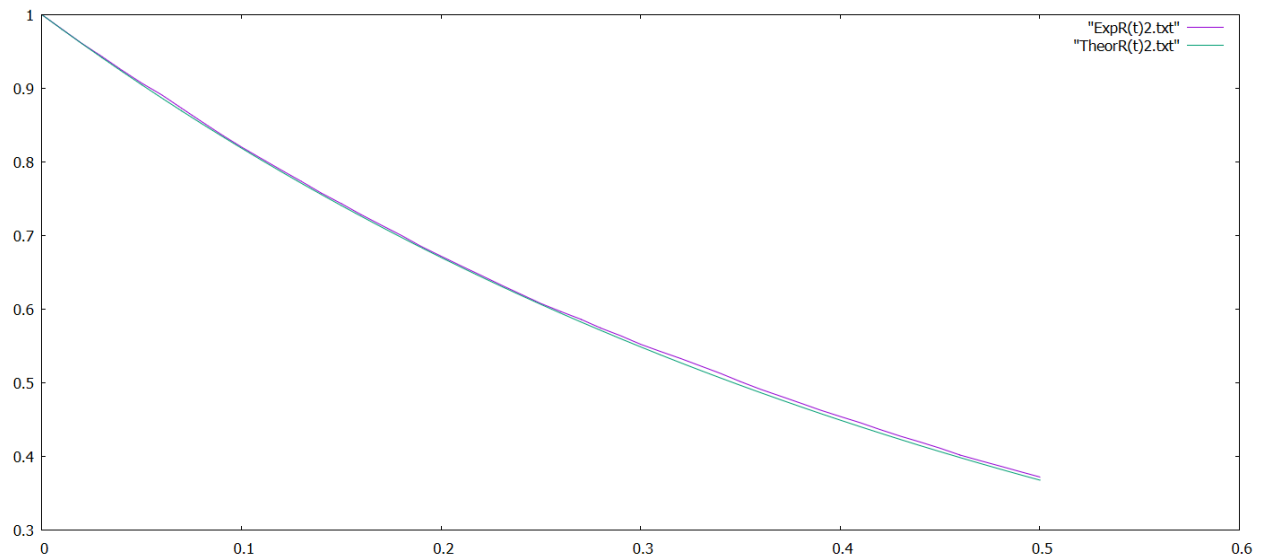


Рисунок 1. Функция надежности для 2-го периода

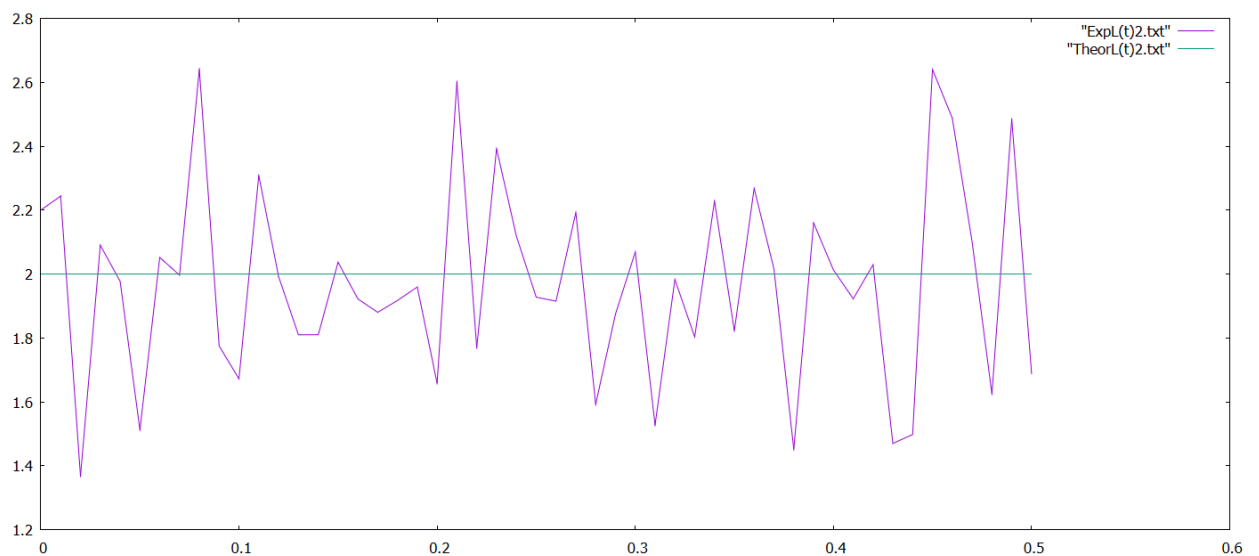


Рисунок 2. Интенсивность отказа для 2-го периода

### 3. Моделирование третьего периода

Для  $i$ -ой системы первоначально надо определить, к какому подмножеству она относится. Для этого используется распределение  $p_i$ . После того, как номер  $i$  подмножества определен, необходимо сгенерировать значение  $T_i$ , как случайной величины, распределенной по экспоненциальному закону с параметром  $\lambda_i$ , который определяется по номеру подмножества. Значение времени  $i$ -ой системы определяется как

$$T = \max T_i$$

Среднее время работы  $i$ -ой системы определяется по формуле:

$$\bar{T} = \sum_{i=0}^n T_i$$

Затем для каждого  $t$  до  $T$  с шагом  $\Delta t = 0.01$  происходит подсчет систем, которые работают.

Теоретическое значение функции надежности:

$$R(t) = R_1(t) + R_2(t) = e^{-\lambda_1 t} + e^{-\lambda_2 t} - e^{-(\lambda_1 + \lambda_2)t}$$

Экспериментальное значение функции надежности:

$$\hat{R}(t) = \frac{n_t}{n}$$

где,  $n_t$  – число систем, работающих в момент времени  $t$ ,  $n$  – общее число систем.

Теоретическое значение интенсивности отказа:

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

Экспериментальное значение интенсивности отказа:

$$\hat{\lambda}(t) = \frac{n_t - n_{t+\Delta t}}{n_t \Delta t}$$

где  $n_t$  – число работоспособных систем в момент  $t$ ,  $n_{t+\Delta t}$  – число систем, работающих в момент  $t + \Delta t$ , где  $\Delta t = 0.001$

Графики:

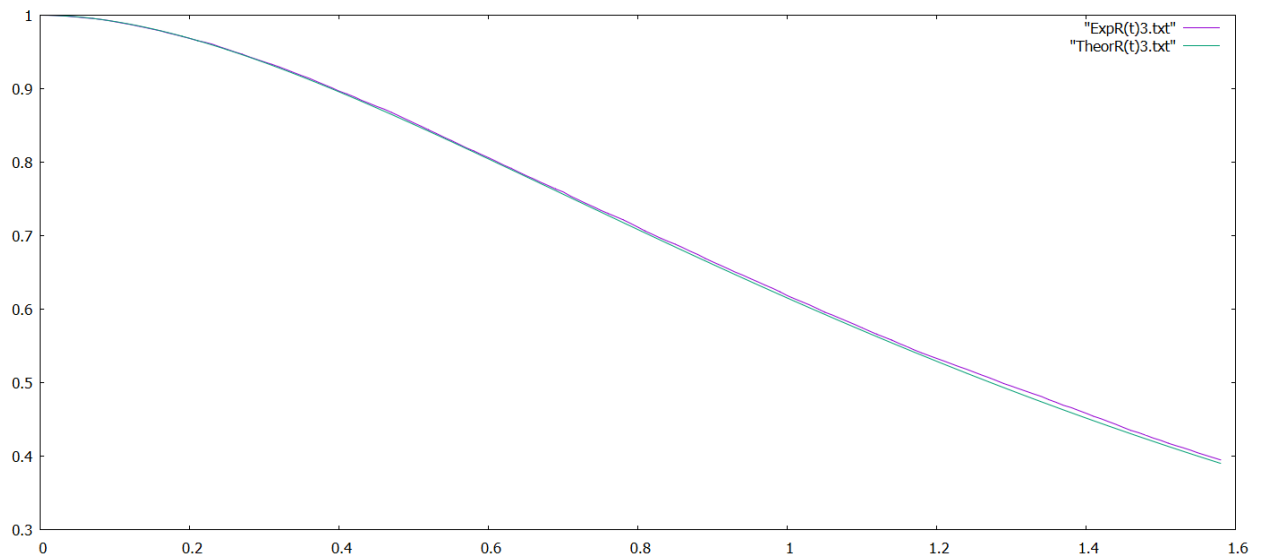


Рисунок 3. Функция надежности для 3-го периода

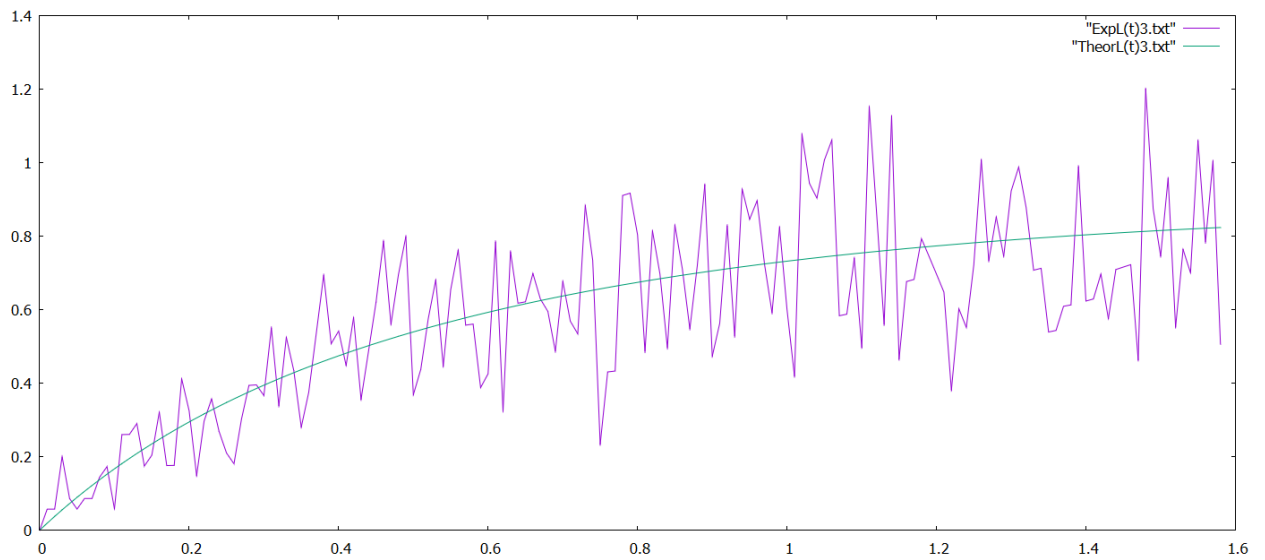


Рисунок 4. Интенсивность отказа для 3-го периода

## Выводы

В ходе выполнения лабораторной работы были смоделированы работы трех периодов невосстанавливаемых систем, а также, были получены экспериментальные значения функции надежности  $R(t)$  и интенсивности отказа  $\lambda(t)$ . Экспериментальные значения функции надежности  $R(t)$  и интенсивности отказа  $\lambda(t)$  близки по значениям с теоретическими значениями, что свидетельствует о корректной работе программы.



## Листинг программы

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;

public class Modeling {
    private int n = 35000;
    private double l1 = 0.7;
    private double l2 = 1.3;
    private double p1 = 0.3;
    private double p2 = 0.7;

    private double T;
    private ArrayList Ti = new ArrayList();

    private void timeModeling1() {
        for (int i = 0; i < n; i++) {
            double tmp = Math.random();
            if (((double) i / n - (p1 + p2) < 0.01)) {
                Ti.add((Math.log(tmp) / l1) * (-1));
                continue;
            }
            if (((double) i / n - (p1 + p2)) >= 0.01 & (((double) i / n - (p1 + p2)) < 0.02)) {
                Ti.add((Math.log(tmp) / l2) * (-1));
                continue;
            } else {
                Ti.add((Math.log(tmp) / l3) * (-1));
                //System.out.println("r");
            }
        }
    }

    private void timeModeling2() {
        for (int i = 0; i < n; i++) {
            double tmp1 = (-1) * Math.log(Math.random());
            double tmp2 = (-1) * Math.log(Math.random());
            if (tmp1 / l1 <= tmp2 / l2)
                Ti.add(tmp1 / l1);
            else
                Ti.add(tmp2 / l2);
        }
    }

    private void timeModeling3() {
        for (int i = 0; i < n; i++) {
            double tmp1 = (-1) * Math.log(Math.random());
            double tmp2 = (-1) * Math.log(Math.random());
```

```

        if (tmp1 / l1 >= tmp2 / l2)
            Ti.add(tmp1 / l1);
        else
            Ti.add(tmp2 / l2);
    }
}

public void AverageT() {
    for (int i = 0; i < Ti.size(); i++) {
        T += (double) Ti.get(i);
    }
    T /= n;
}

public void Theoretical1() {
    for (double t = 0; t - T <= 0.0; t += 0.01) {
        double R = 0;
        double l = 0;
        R = Math.exp(-l1 * t) * p1 + Math.exp(-l2 * t) * p2 + Math.exp(-l3 * t) * p3;
        l = (l1 * p1 * Math.exp((-1) * l1 * t) + l2 * p2 * Math.exp((-1) * l2 * t) + l3 * p3 *
Math.exp((-1) * l3 * t)) / R;
        writeStringToFile(R, t, "TheorR(t)1.txt");
        writeStringToFile(l, t, "TheorL(t)1.txt");
    }
}

public void Theoretical2() {
    double l = l1 + l2;
    for (double t = 0; t - T <= 0.0; t += 0.01) {
        double R = 0;
        R = Math.exp(-l1 * t) * Math.exp(-l2 * t);
        writeStringToFile(R, t, "TheorR(t)2.txt");
        writeStringToFile(l, t, "TheorL(t)2.txt");
    }
}

public void Theoretical3() {
    for (double t = 0; t - T <= 0.0; t += 0.01) {
        double R = 0;
        double l = 0;
        R = Math.exp(-l1 * t) + Math.exp(-l2 * t) - Math.exp(-l1 * t) * Math.exp(-l2 * t);
        writeStringToFile(R, t, "TheorR(t)3.txt");
        l = (-1) * ((-1) * l1 * Math.exp((-1) * l1 * t) - l2 * Math.exp((-1) * l2 * t) + (l1 + l2) *
Math.exp((-
        1) * (l1 + l2) * t)) / R;
        writeStringToFile(l, t, "TheorL(t)3.txt");
    }
}

```

```

public int findN(double t) {
    int nT = 0;
    for (int i = 0; i < Ti.size(); i++) {
        if (t <= (double) Ti.get(i)) {
            nT++;
        }
    }
    return nT;
}

public void Experimental(String filename1, String filename2) {
    for (double t = 0; t < T; t += 0.01) {
        int nT = findN(t);
        double R = (double) nT / n;
        writeStringToFile(R, t, filename1);
        double l = (double) (nT - findN(t + 0.001)) / (nT * 0.001);
        writeStringToFile(l, t, filename2);
    }
}

public void writeStringToFile(double R, double t, String filename) {
    try {
        FileWriter file = new FileWriter(filename, true);
        StringBuilder str = new StringBuilder();
        str.append(t).append(" ").append(R).append("\n");
        file.write(str.toString());
        file.flush();
    } catch (IOException ex) {
        System.out.println(ex.getMessage());
        ex.printStackTrace();
    }
}

public void firstModeling() {
    Ti.clear();
    T = 0;
    timeModeling1();
    AverageT();
    Experimental("ExpR(t)1.txt", "ExpL(t)1.txt");
    Theoretical1();
}

public void secondModeling() {
    Ti.clear();
    T = 0;
    timeModeling2();
    AverageT();
}

```

```

        Experimental("ExpR(t)2.txt", "ExpL(t)2.txt");
        Theoretical2();
    }

    public void thirdModeling() {
        Ti.clear();
        T = 0;
        timeModeling3();
        AverageT();
        Experimental("ExpR(t)3.txt", "ExpL(t)3.txt");
        Theoretical3();
    }

    public static void main(String[] args) {
        Modeling m = new Modeling();
        m.firstModeling();
        m.secondModeling();
        m.thirdModeling();
    }
}

```