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

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

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

ИССЛЕДОВАНИЕ ИНТЕНСИВНОСТИ ОТКАЗОВ ДЛЯ НЕВОССТАНАВЛИВАЕМЫХ СИСТЕМ

по курсу: НАДЕЖОСТЬ ИНФОКОММУНИКАЦИОННЫХ СИСТЕМ

СТУДЕНТ ГР. № 5922		К. Д. Камышников
номер группы	подпись, дата	инициалы, фамилия

Санкт-Петербург
2022

1. Цели работы:

- Выбор периода жизни системы и соответствующей ему статистической модели;
- Имитационное моделирование процесса функционирования невосстанавливаемой системы для выбранного периода жизни системы;
- Построение зависимости $\lambda(t)$ оценки интенсивности отказов от времени.

2. Ход работы

Процесс функционирования невосстанавливаемой системы можно разделить на 3 периода:

1) Период приработки

   Время жизни системы:

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

   $[0;1]$ – случайное число в интервале от 0 до 1.

   Теоретическая функция надежности:

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

   Теоретическая функция интенсивности отказа:

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

В результате моделирования системы получаем графики зависимости надежности и интенсивности отказов от времени функционирования системы (эти и последующие графики построены при входных данных: $p_1 = 0.6$; $p_2 = 0.4$; $\lambda_1 = 0.7$; $\lambda_2 = 1.3$).

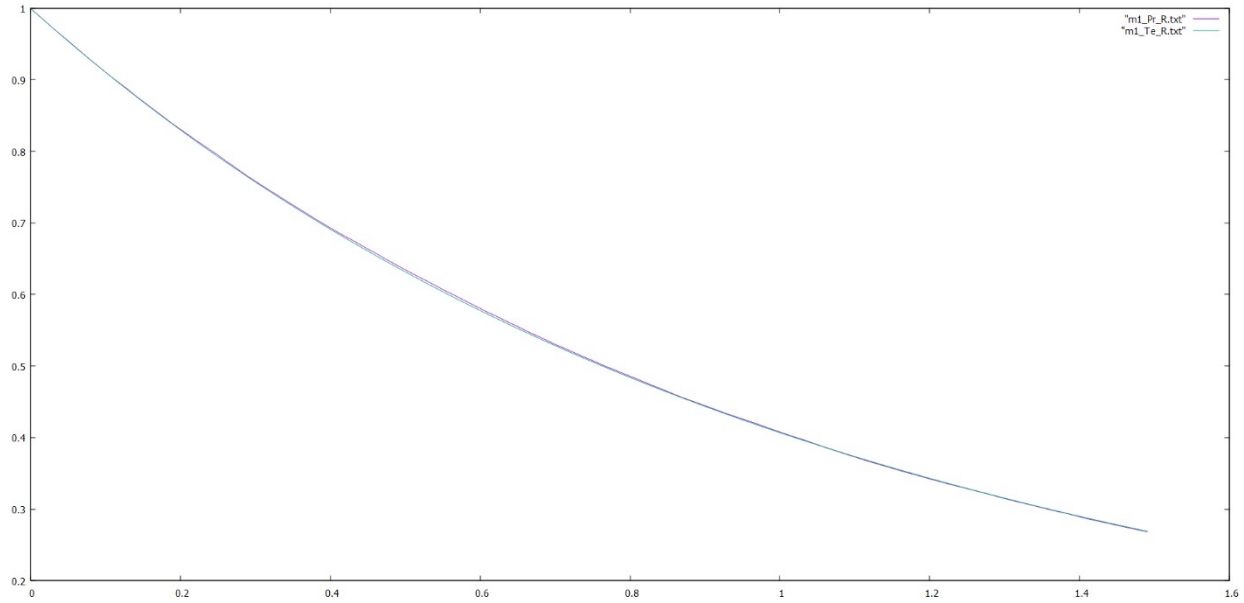

Рис.1 Теоретический и практический графики функции надежности

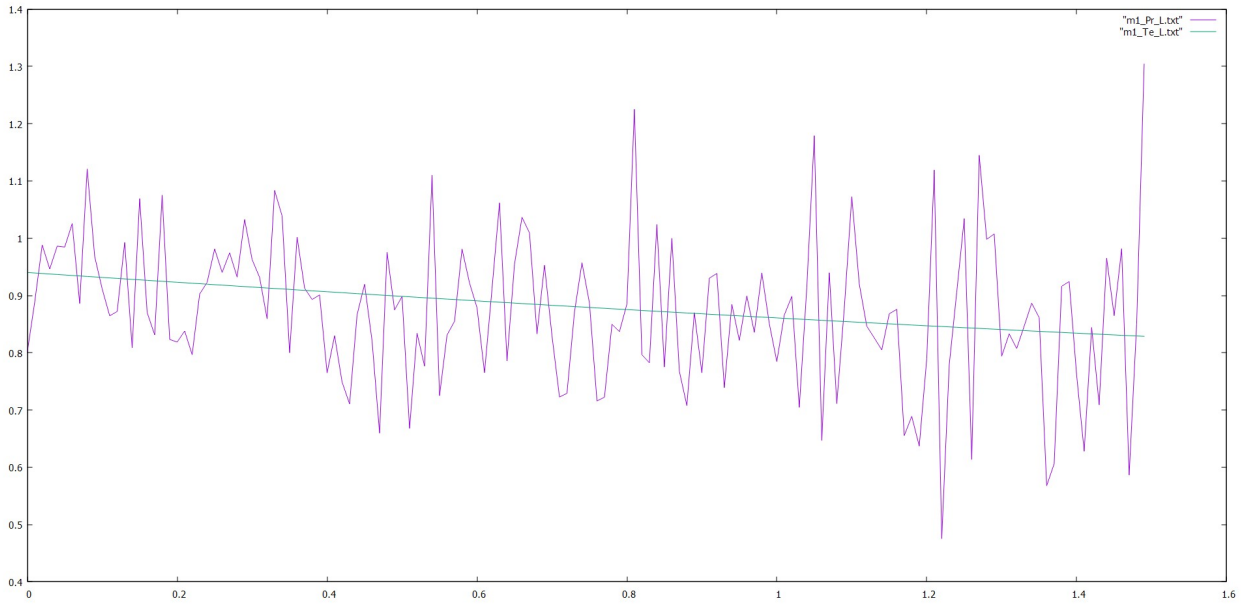

Рис.2 Теоретический и практический графики интенсивности отказов

2) Период нормального функционирования
Время жизни системы:

$$T_1 = \frac{-\ln[0;1]}{\lambda_1}$$

$$T_2 = \frac{-\ln[0;1]}{\lambda_2}$$

$$T_i = min\ (T_1, T_2)$$

Теоретическая функция надежности:

$$R(t) = e^{-\lambda_1 t} \cdot e^{-\lambda_2 t}$$

Теоретическая функция интенсивности отказа:

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

В результате моделирования системы получаем следующие графики:

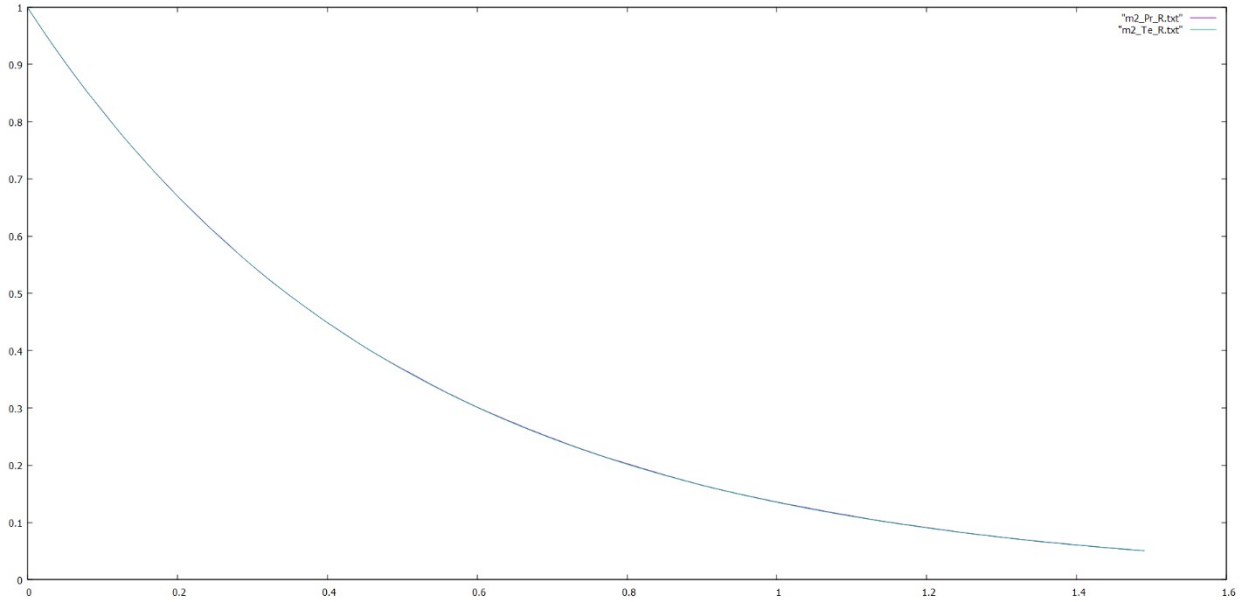

Рис.3 Теоретический и практический графики функции надежности

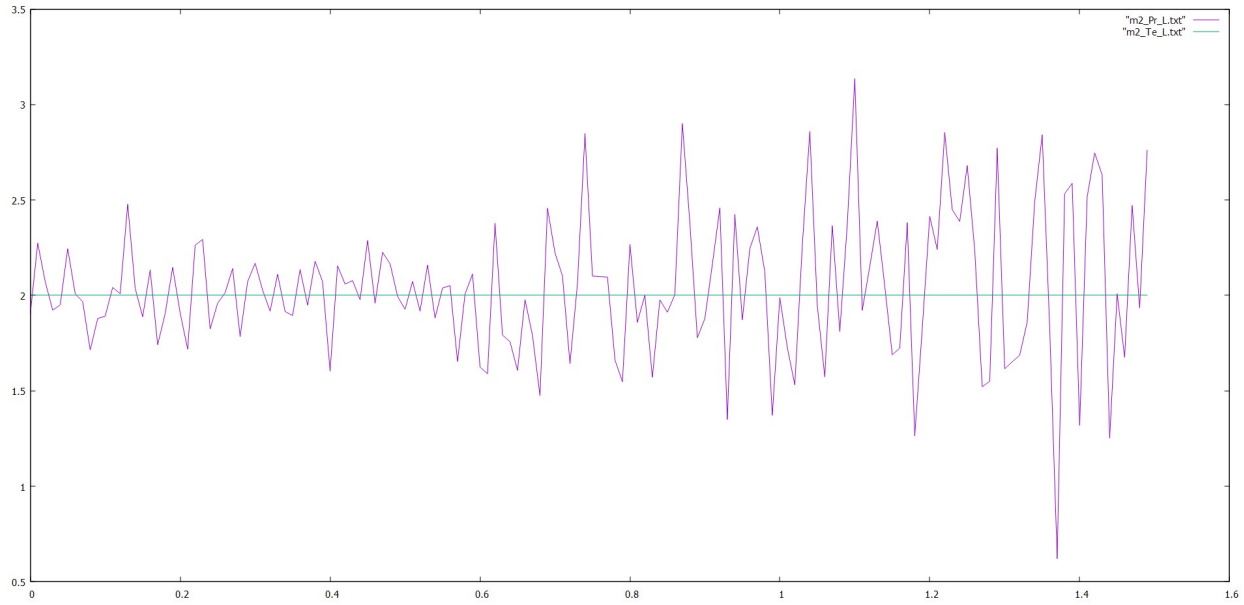

Рис.4 Теоретический и практический графики интенсивности отказов

3) Период старения

Время жизни системы:

$$T_1 = \frac{-\ln[0;1]}{\lambda_1}$$

$$T_2 = \frac{-\ln[0;1]}{\lambda_2}$$

$$T_i = max\ (T_1, T_2)$$

Теоретическая функция надежности:

$$R(t) = 1 - \left(1 - e^{-\lambda_1 t}\right) \cdot \left(1 - e^{-\lambda_2 t}\right)$$

Теоретическая функция интенсивности отказа:

$$\lambda(t) = \frac{-R'(t)}{R(t)}$$

В результате моделирования системы получаем следующие графики:

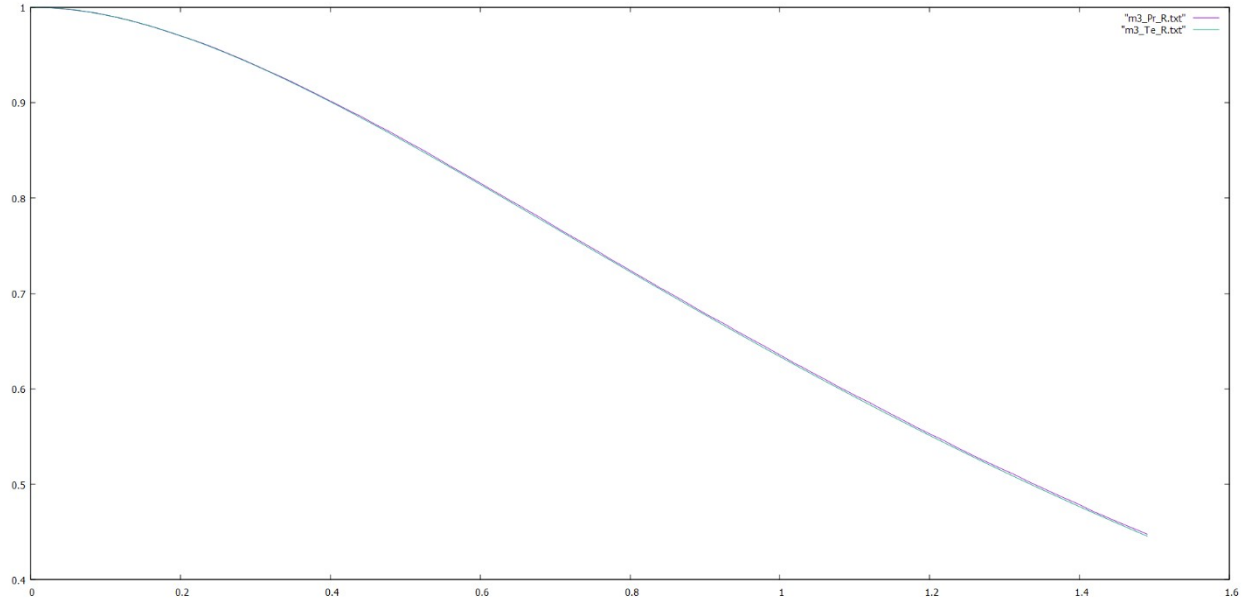

Рис.5 Теоретический и практический графики функции надежности

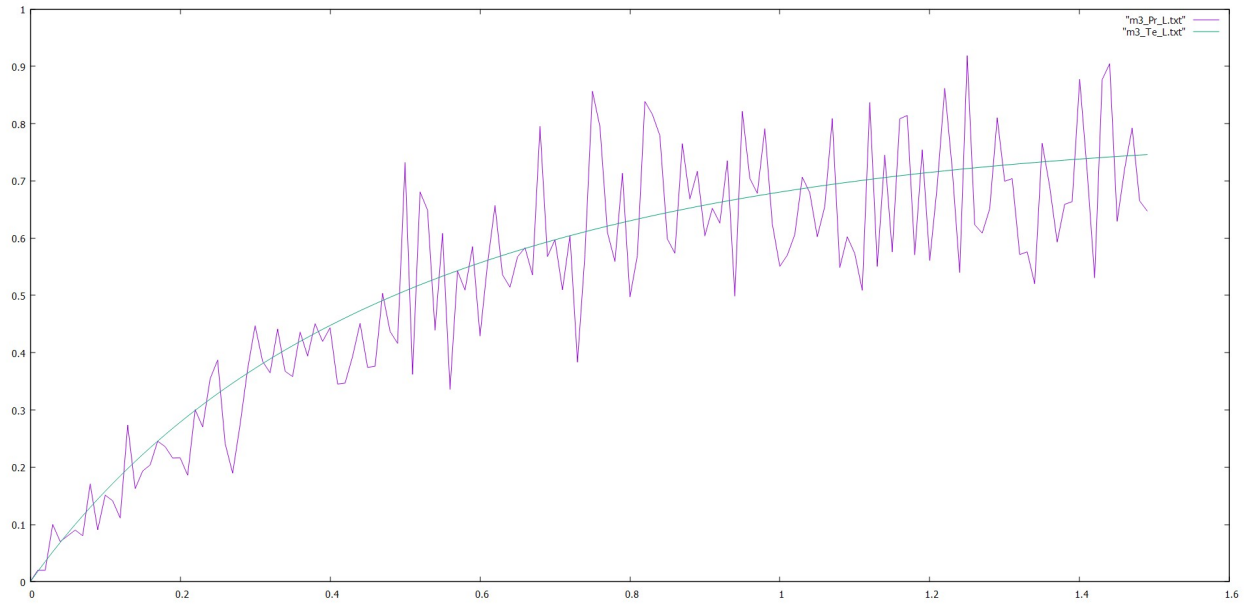

Рис.6 Теоретический и практический графики интенсивности отказов

3. Вывод

В ходе лабораторной работы было реализовано имитационное моделирование процесса функционирования невосстанавливаемой системы для всех периодов жизни. Были построены графики зависимости надежности и интенсивности отказов от времени функционирования системы. Графики функции надежности не сильно отличаются друг от друга. Практические графики интенсивности отказов имеют колебательную форму, но если провести среднюю линию, то она будет схожа с теоретическим графиком. Таким образом, моделирование системы выполнено правильно.

4. Листинг

Приложение 1

```
#include <iostream>
#include <vector>
#include <fstream>

using namespace std;

double p[] = { 0.6, 0.4 };
double lambda[] = { 0.7, 1.3 };
int N = 10000;

void write(string filename, vector<double> Ti, int num)
{
    vector<double> R;
    vector<double> Lambda;
    vector<double> R_T;
    vector<double> Lambda_T;
    int Nt = 0;
    int Ntdt = 0;
    fstream outR(filename + "_Pr_R.txt", ios::out);
    fstream outL(filename + "_Pr_L.txt", ios::out);
    for (double t = 0; t < 3; t += 0.01)
    {
        for (int i = 0; i < N; ++i)
        {
            if (Ti[i] >= t)
            {
                Nt++;
            }
            if (Ti[i] >= (t + 0.001))
            {
                Ntdt++;
            }
        }
        R.push_back((double)Nt / (double)N);
        Lambda.push_back(((double)Nt - (double)Ntdt) / ((double)Nt * 0.001));
        outR << t << " " << R[R.size() - 1] << endl;
        outL << t << " " << Lambda[Lambda.size() - 1] << endl;
        Nt = 0;
        Ntdt = 0;
    }
    outR.close();
    outL.close();
    outR.open(filename + "_Te_R.txt", ios::out);
    outL.open(filename + "_Te_L.txt", ios::out);
    for (double t = 0; t < 3; t += 0.01)
    {
        if (num == 1)
        {
            R_T.push_back(exp(-lambda[0] * t) * p[0] + exp(-lambda[1] * t) *
p[1]);
            Lambda_T.push_back(-((exp(-lambda[0] * t) * p[0] * (-lambda[0]) +
exp(-lambda[1] * t) * p[1] * (-lambda[1])) / R_T[R_T.size() - 1]));
        }
```

```
        if (num == 2)
        {
            R_T.push_back(exp(-lambda[0] * t) * exp(-lambda[1] * t));
            Lambda_T.push_back(lambda[0] + lambda[1]);
        }
        if (num == 3)
        {
            R_T.push_back(exp(-lambda[0] * t) + exp(-lambda[1] * t) - exp(-
(lambda[0] + lambda[1]) * t));
            Lambda_T.push_back(-(((-lambda[0] * exp(-lambda[0] * t)) - (lambda[1]
* exp(-lambda[1] * t)) + (lambda[0] + lambda[1]) * exp(-(lambda[0] + lambda[1]) *
t)) / (R_T[R_T.size() - 1])));
        }
        outR << t << " " << R_T[R_T.size() - 1] << endl;
        outL << t << " " << Lambda_T[Lambda_T.size() - 1] << endl;
    }
    outR.close();
    outL.close();
}

void model1()
{
    vector<double> Ti;
    srand(time(0));
    for (int i = 0; i < N; ++i)
    {
        double tmp = ((double)rand() + 1) / RAND_MAX;
        if ((rand() / RAND_MAX) > p[0])
        {
            Ti.push_back(-log(tmp) / lambda[1]);
        }
        else
        {
            Ti.push_back(-log(tmp) / lambda[0]);
        }
    }
    write("m1", Ti, 1);
}

void model2()
{
    vector<double> Ti;
    for (int i = 0; i < N; ++i)
    {
        double first = -log((double)rand() / RAND_MAX) / lambda[0];
        double second = -log((double)rand() / RAND_MAX) / lambda[1];
        Ti.push_back(min(first, second));
    }
    write("m2", Ti, 2);
}

void model3()
{
    vector<double> Ti;
    double first = 0, second = 0;
    for (int i = 0; i < N; ++i)
```

```
    {
        double tmp = ((double)rand() + 1) / RAND_MAX;
        first = -log(tmp) / lambda[0];
        tmp = ((double)rand() + 1) / RAND_MAX;
        second = -log(tmp) / lambda[1];
        Ti.push_back(max(first, second));
    }
    write("m3", Ti, 3);
}

int main()
{
    model1();
    model2();
    model3();
}
```