

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, к.т.н			Н.В. Марковская
должность, уч. степень, звание		подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

по курсу: НАДЕЖНОСТЬ ИНФОКОММУНИКАЦИОННЫХ СЕТЕЙ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	5911	13.05.2022	М.С. Собакин
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

## 1 Цель работы

Имитационное моделирование функционирования системы со сложной схемой резервирования. Построение временной зависимости, отражающей изменение коэффициента готовности восстанавливаемой □□ системы. Проверка того, что установившееся значение □□ находится в пределах границ.

## 2 Исходные данные

Имитационное моделирование необходимо провести для схемы, изображенной ниже:

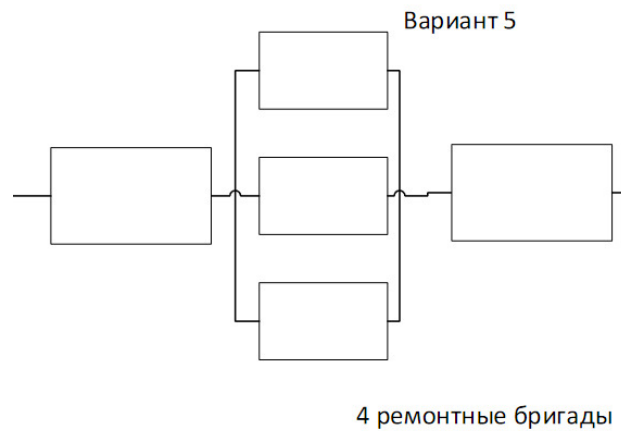


Схема 1 - Исходная схема

Для каждого элемента схемы задается  $\lambda$  и  $\mu$  ( $\lambda = 1$  и  $\mu = 2$  для рассматриваемого примера). Число экспериментов  $N = 30\,000$ . Моделирование будет продолжаться  $T = 100$  с шагом  $\Delta t = 0.01$ .

## 3 Верхняя и нижняя граница

### 3.1 Верхняя граница

Для подсчета верхней границы необходимо увеличить количество бригад до количества элементов системы. Таким образом, коэффициент готовности каждого элемента

$$K_{1,1} = \frac{\mu}{\lambda + \mu}$$

Чтобы заданная система работала в момент времени, необходимо, чтобы работали элементы 1, 5 и любой из элементов 2, 3, 4, т.е. коэффициент готовности будет равен:

$$K^+ = (1 - (1 - K_{1,1})^3) * K_{1,1}^2$$

Для заданных значений  $\square^+ = 0.427$

### 3.2 Нижняя граница через распределение бригад

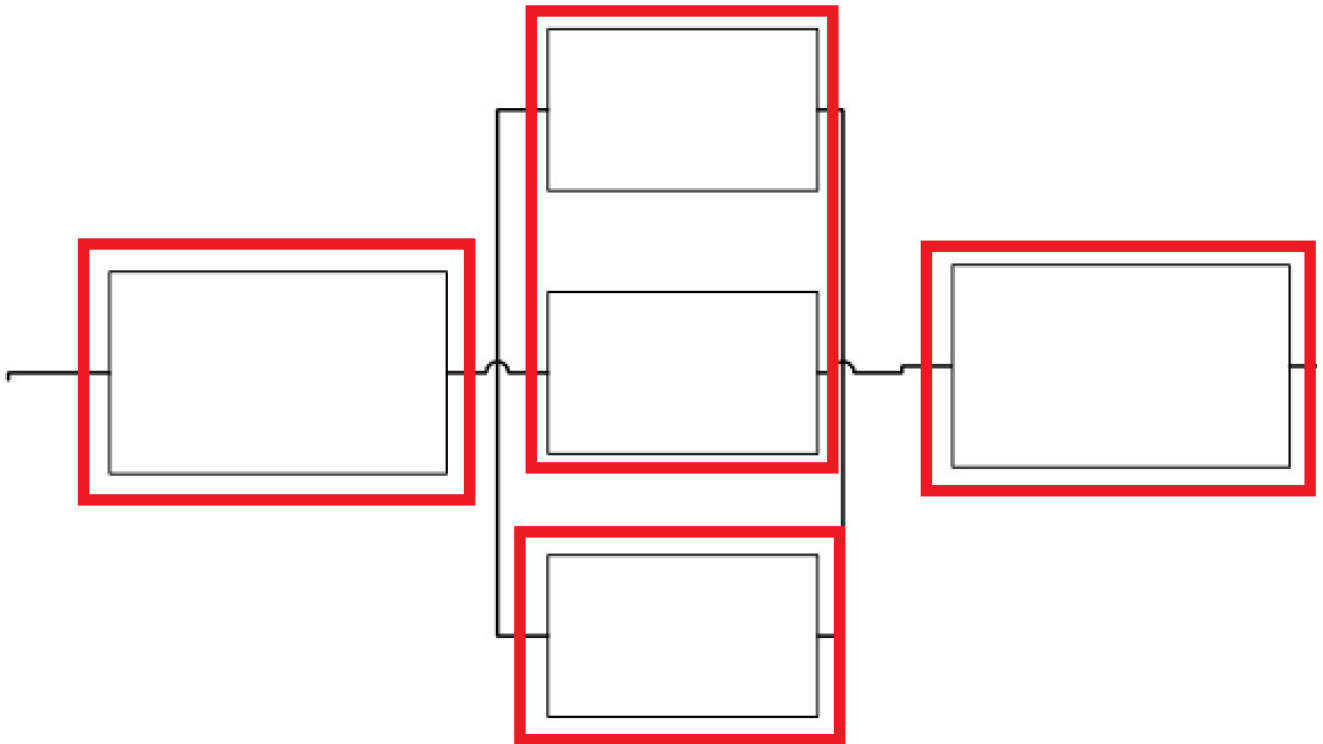


Схема 2 - Распределение бригады

Коэффициент готовности  $\square_{2,1}$  равен:

$$K_{2,1} = \frac{2\lambda\mu + \mu^2}{2\lambda^2 + 2\lambda\mu + \mu^2}$$

Чтобы заданная система работала в момент времени, необходимо, чтобы одновременно работали элементы 1, 5 и любая из двух групп в параллельном соединении, т.е. коэффициент готовности системы будет равен:

$$\square^- = (1 - (1 - K_{2,1})(1 - K_{1,1})) * K_{11}^2$$

Для заданных значений  $\square^- = 0,414$ .

### 3.3 Нижняя граница через исключение систем

Для подсчета нижней границы будем считать коэффициент готовности системы, исключив дублирующие элементы из параллельного соединения:

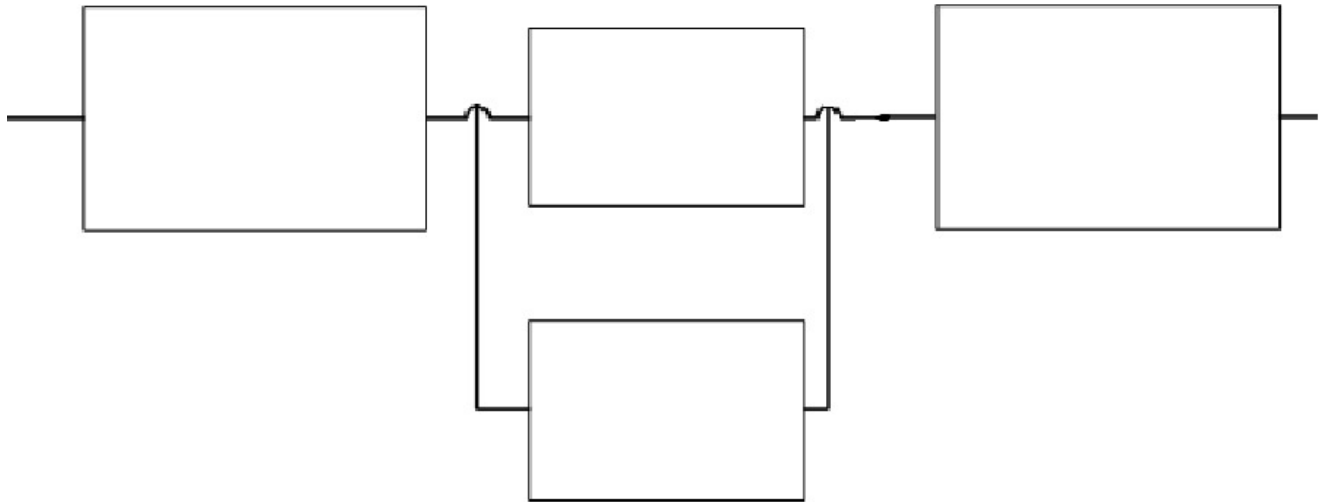


Схема 3 - С исключением дублирующих систем

Необходимо, чтобы одновременно работали все три элемента. Тогда коэффициент готовности будет равен:

$$K_{г,2} = K_{2,2} * K_{1,1}^2 = (1 - (1 - K_{1,1})^2) * K_{1,1}^2$$

Для заданных значений  $K_{г,2} = 0,395$ .

## 4 Имитационное моделирование

В каждый момент времени происходит оценка работоспособности системы в целом. Если один из элементов ломается, то одна из свободных бригад приступает к ремонту. Если все бригады заняты, то время ремонта элемента увеличивается на  $\Delta t_{ожид}$  (элемент дожидается, пока бригада освободится).

Для каждого из элементов системы необходимо случайным образом сгенерировать время работы  $T_w$  и время ремонта  $T_r$  по следующим формулам:

$$T_w = \frac{-\ln [0,1]}{\lambda} \quad T_r = \frac{-\ln [0,1]}{\mu}$$

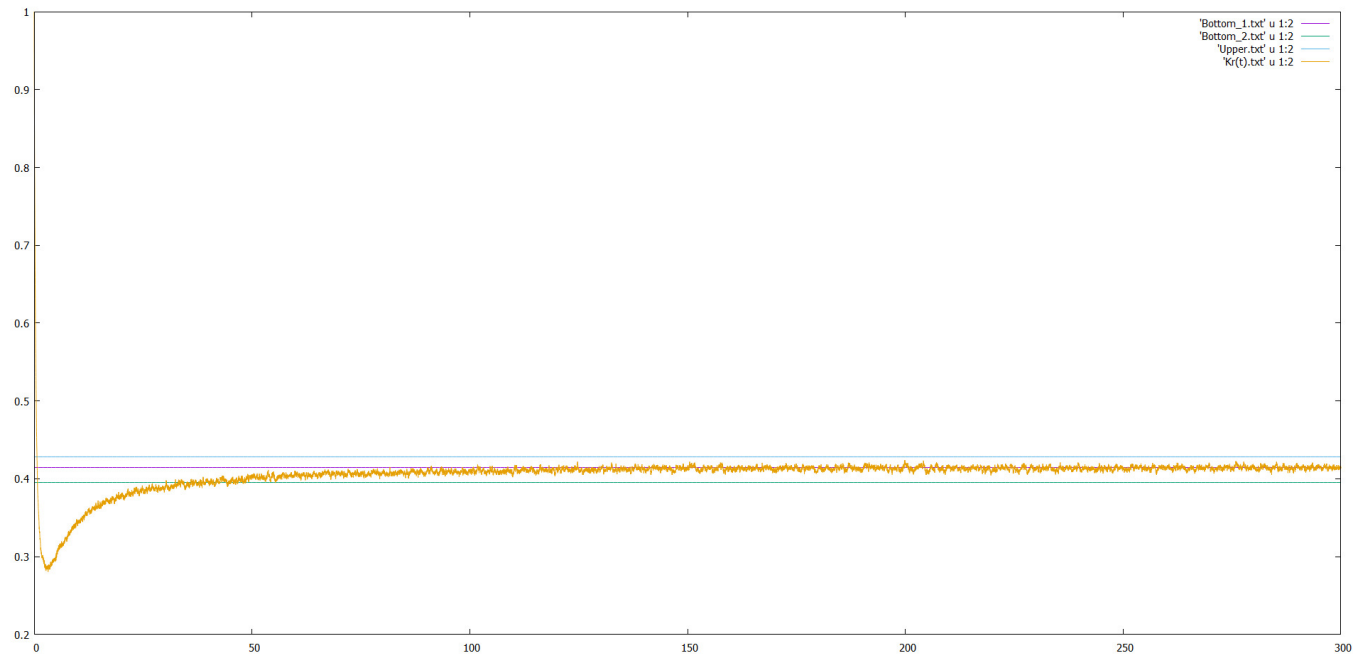
Затем в каждый момент времени функция проверки работоспособности системы возвращается  $\chi(t) = \{0,1\}$ , где 0 означает, что система не работает, иначе 1.

Таким образом моделируется  $N = 30\,000$  экспериментов. Коэффициент готовности

определяется как:

$$\bar{K}_i(K) = \left( \sum_{j=1}^N K_j(K) \Delta t \right) / N, \text{ где } i = 1, 2, \dots, N$$

Результаты имитационного моделирования сравниваются с результатами аналитического расчета границ коэффициента готовности:



### Выводы

Были получены верхняя и нижняя оценка коэффициента готовности системы. Также были получены экспериментальные значения коэффициента готовности, которые, как видно из графика 1 в установившемся режиме находятся в пределах заданных границ, что говорит о корректности работы программы.

### Листинг программы

```
import java.io.FileWriter;

public class Modeling {
    private class Repair {
        private boolean status = true;
        private double ending;
        private int repairNum = n;

        public void createRepair(boolean s, double e, int num) {
            status = s;
            ending = e;
            repairNum = num;
        }

        public void end() {
            status = true;
            ending = 0;
            repairNum = n;
        }

        public String toString() {
            StringBuilder str = new StringBuilder();
            str.append("is free = ").append(status).append("
").append(ending).append(" ").append(repairNum);
            return str.toString();
        }
    }

    private class Element {
        private double Tw = 0;
        private double Tr = 0;
        private int waitNumber = 0;
        private boolean isWorking = false;
        private int index = 0;

        public String toString() {
            StringBuilder str = new StringBuilder();
            str.append(index).append(": ");
            str.append("Tw = ").append(Tw).append(" Tr =
").append(Tr).append(" ").append(isWorking);
            str.append("
");
            return str.toString();
        }
    }

    private int N = 30000;
    private int n = 5;
```

```

private int r = 4;
private double l = 0.5;
private double m = 1;
private int T = 100;
private double step = 0.01;
private Element[] elements = new Element[n];
private int[] E = new int[(int) ((double) T / step) + 1];

public void BottomBound() throws Exception {
    double K21 = ((2 * m * l) + Math.pow(m, 2)) / (2 * Math.pow(l, 2) + 2 * m * l + Math.pow(m,
2));
    double K11 = m / (m + l);
    double K3 = 1 - (1 - K21)*(1 - K11);

    double K_d = K3 * K11 * K11;

    double K_elim = (1 - Math.pow(1 - K11, 2)) * K11 * K11;

    FileWriter file1 = new FileWriter("Bottom_1.txt");
    FileWriter file2 = new FileWriter("Bottom_2.txt");

    for (int t = 0; t <= T; t += 1) {
        StringBuilder str1 = new StringBuilder();
        StringBuilder str2 = new StringBuilder();
        str1.append(t).append(" ").append(K_d).append("\n");
        file1.write(str1.toString());
        file1.flush();
        str2.append(t).append(" ").append(K_elim).append("\n");
        file2.write(str2.toString());
        file2.flush();
    }
}

public void TopBound() throws Exception {
    double K11 = m / (m + l);

    System.out.println(K11);
    System.out.println(Math.pow(1 - K11, 3));
    System.out.println(1 - Math.pow(1 - K11, 3));
    System.out.println(K11 * K11);
    double K = (1 - Math.pow(1 - K11, 3)) * K11 * K11;
    System.out.println(K);
    FileWriter file = new FileWriter("Upper.txt");
    for (int t = 0; t <= T; t += 1) {
        StringBuilder str = new StringBuilder();
        str.append(t).append(" ").append(K).append("\n");
        file.write(str.toString());
        file.flush();
    }
}

```



```
    }  
}
```

```
public void timeModeling(Element e) {  
    double Tw = (-1) * ((Math.log(Math.random())) / l);  
    double Tr = (-1) * ((Math.log(Math.random())) / m);  
    e.Tw = Tw;  
    e.Tr = Tr;  
}
```

```
private void isWorking(Element e, double t, Repair[] brigades) {  
    int tmp = (int) Math.floor(t / (e.Tw + e.Tr));  
    if ((t - (tmp * (e.Tw + e.Tr))) <= e.Tw) {  
        e.isWorking = true;  
    } else {  
        e.isWorking = false;  
  
        for (int i = 0; i < r; i++) {  
            if (brigades[i].repairNum == e.index) {  
                if (brigades[i].ending <= (t + step)) {  
                    e.Tr -= step * e.waitNumber;  
                    e.waitNumber = 0;  
                    brigades[i].end();  
                }  
                return;  
            }  
  
            if (brigades[i].status) {  
                brigades[i].createRepair(false, (tmp + 1) * (e.Tw + e.Tr), (byte) e.index);  
                return;  
            }  
        }  
  
        e.Tr += step;  
        e.waitNumber++;  
        return;  
    }  
}
```

```
private void simulate() {  
    elements = new Element[n];  
    for (int i = 0; i < n; i++) {  
        elements[i] = new Element();  
        elements[i].index = i;  
    }  
    for (int i = 0; i < n; i++) {  
        timeModeling(elements[i]);  
    }  
}
```

```

Repair[] brigades = new Repair[r];
for (int i = 0; i < r; i++)
    brigades[i] = new Repair();

int index = 0;
for (double t = 0; t < T; t += step) {
    for (int i = 0; i < n; i++) {
        isWorking(elements[i], t, brigades);
    }
    if (elements[0].isWorking && (elements[1].isWorking || elements[2].isWorking ||
elements[3].isWorking) && elements[4].isWorking) {
        E[index]++;
    }
    index++;
    clear();
}
}

public void modeling() throws Exception {
    TopBound();
    BottomBound();
    for (int i = 0; i < N; i++) {
        simulate();
    }
    FileWriter file = new FileWriter("Kr(t).txt");
    int index = 0;
    for (double t = 0; t < T; t += step) {
        StringBuilder str = new StringBuilder();
        str.append(t).append(" ").append(((double) E[index]) / N).append("\n");
        file.write(str.toString());
        file.flush();
        index++;
    }
}

public void clear() {
    for (int i = 0; i < n; i++)
        elements[i].isWorking = false;
}

public void clearFile() {
    try {
        FileWriter file = new FileWriter("Upper.txt");
        file.close();
        FileWriter file1 = new FileWriter("Bottom_1.txt");
        file1.close();
        FileWriter file2 = new FileWriter("Bottom_2.txt");
        file2.close();
    }
}

```

```
        FileWriter file3 = new FileWriter("Kr(t).txt");
        file3.close();
    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }
}

public static void main() {
    Modeling m = new Modeling();

    try {
        m.modeling();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
```