МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего профессионального образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №52

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Доцент. |  |  |  | Марковская Н.В. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ |
| Исследование интенсивности отказов  для невосстанавливаемых систем |
| по курсу: СЕТИ С СИСТЕМЫ ПЕРЕДАЧИ ИНФОРМАЦИИ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 5512 |  |  |  | К.А.Абдулжамилов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2019

1. Цель работы:

Смоделировать периоды жизни невосстанавливаемой системы.

1. Исходные данные:

Система состоит из двух групп. Для этих групп определены интенсивности:

Также определены вероятности того, что выбрана конкретная система для первого периода:

1. Имитационное моделирование первого периода:

Всего моделируется N систем. Из них N\* имеют время жизни и оставшиеся системы имеют время жизни , где случайная равномерно распределенная величина. Затем для сгенерированных N систем считаются .

Далее выбирается некая промежуточная точка между данный промежуток разбивается на k отрезков. В каждый момент времени считается, сколько систем находится в рабочем состоянии и высчитывается функция надежности , где – количество работающих систем. Также для модели рассчитывается интенсивность отказов . Полученные результаты сравниваются с теоритическими.

.

Для заданной модели

Результат моделирования и расчетов представлен на графиках

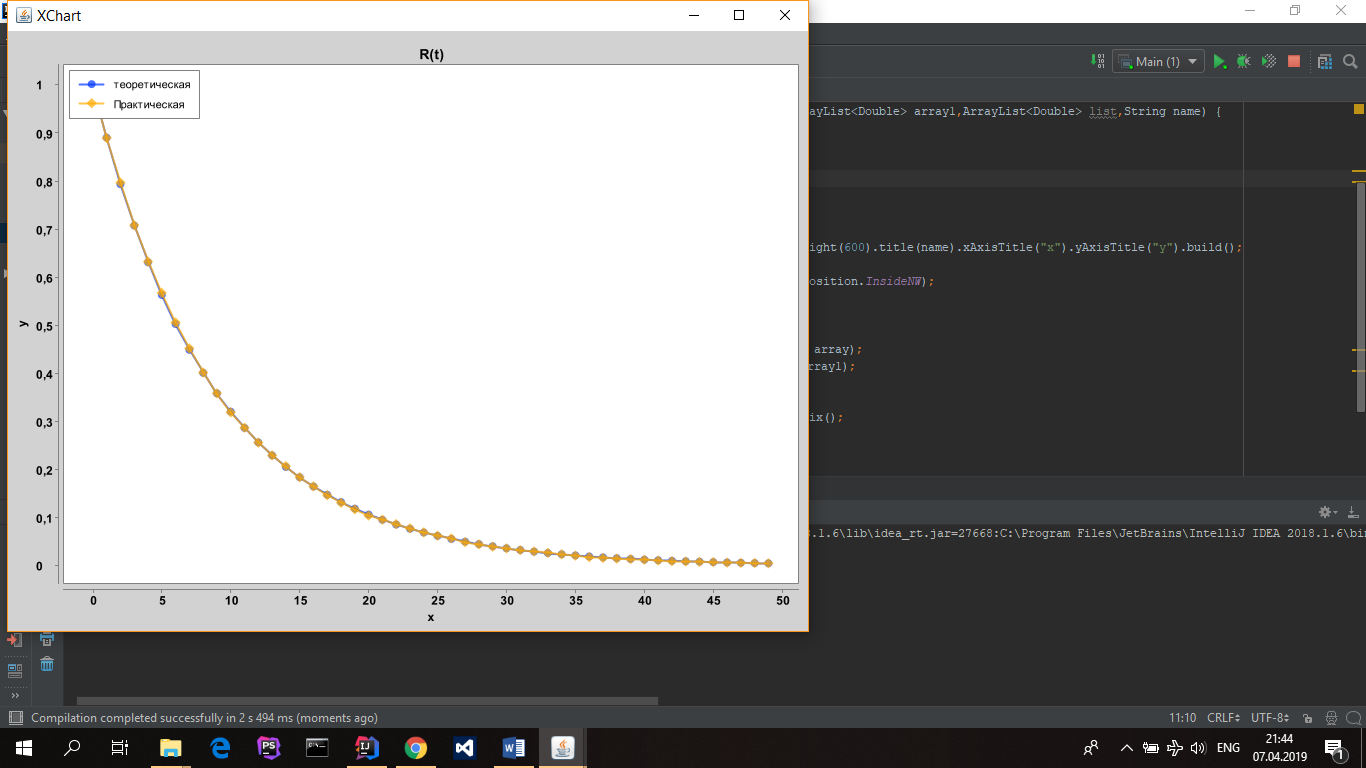


Рисунок 1 - R(t) для первой модели

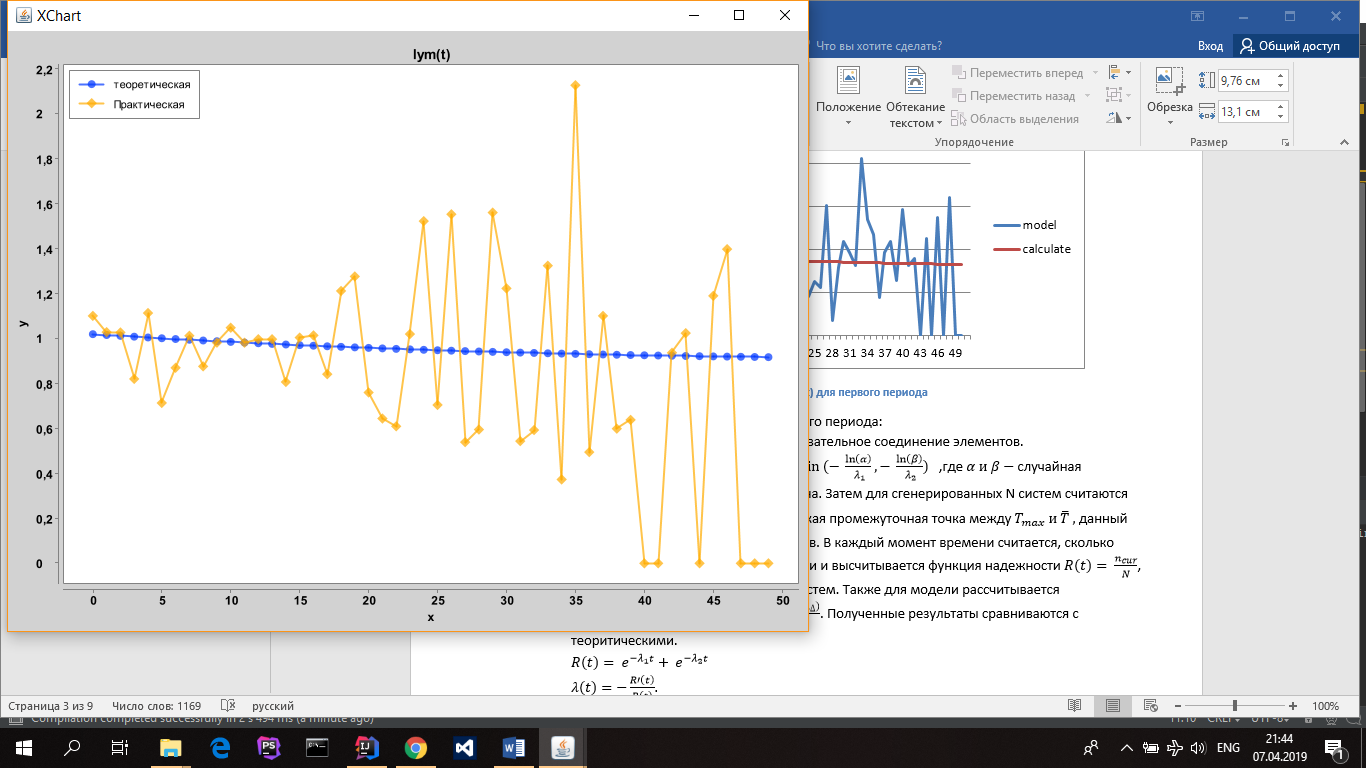


Рисунок 2 - λ(t) для первого периода

1. Имитационное моделирование второго периода:

Система представляет собой последовательное соединение элементов.

Всего моделируется N систем. ,где случайная равномерно распределенная величина. Затем для сгенерированных N систем считаются . Далее выбирается некая промежуточная точка между данный промежуток разбивается на k отрезков. В каждый момент времени считается, сколько систем находится в рабочем состоянии и высчитывается функция надежности , где – количество работающих систем. Также для модели рассчитывается интенсивность отказов . Полученные результаты сравниваются с теоритическими.

.

Для заданной модели

Результат моделирования и расчетов представлен на графиках

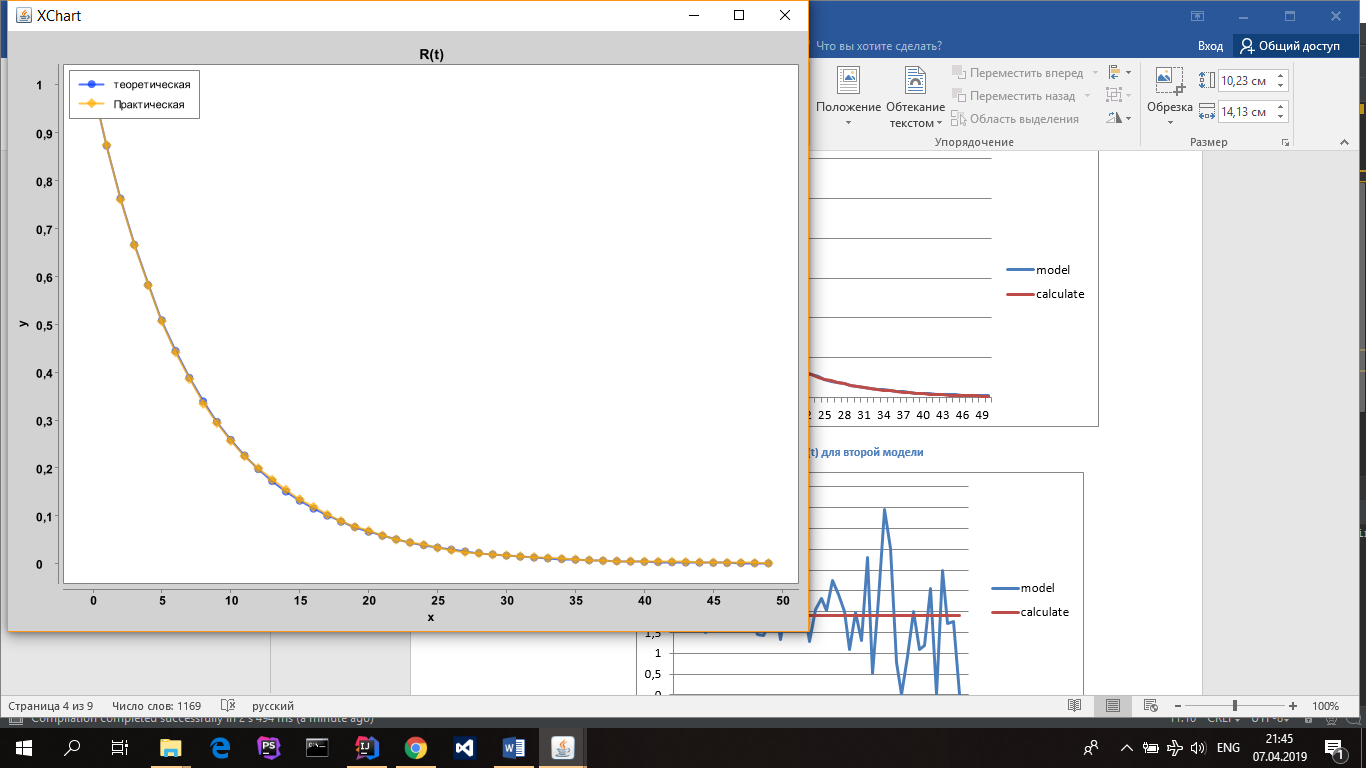


Рисунок 3 - R(t) для второй модели

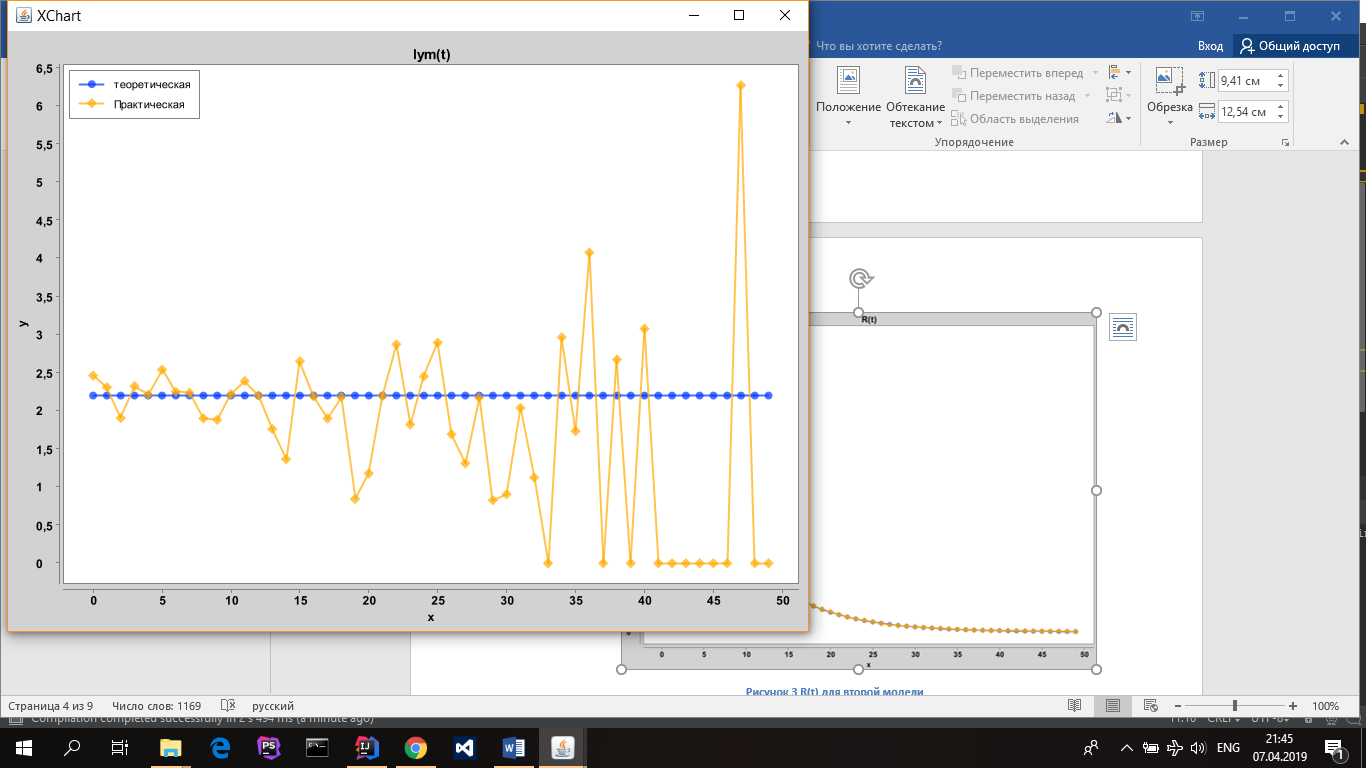


Рисунок 4 - λ(t) для второго периода

1. Имитационное моделирование третьего периода:

Система представляет собой параллельное соединение элементов.

Всего моделируется N систем. ,где случайная равномерно распределенная величина. Затем для сгенерированных N систем считаются . Далее выбирается некая промежуточная точка между данный промежуток разбивается на k отрезков. В каждый момент времени считается, сколько систем находится в рабочем состоянии и высчитывается функция надежности , где – количество работающих систем. Также для модели рассчитывается интенсивность отказов . Полученные результаты сравниваются с теоритическими.

,

.

Для заданной модели

Результат моделирования и расчетов представлен на графиках

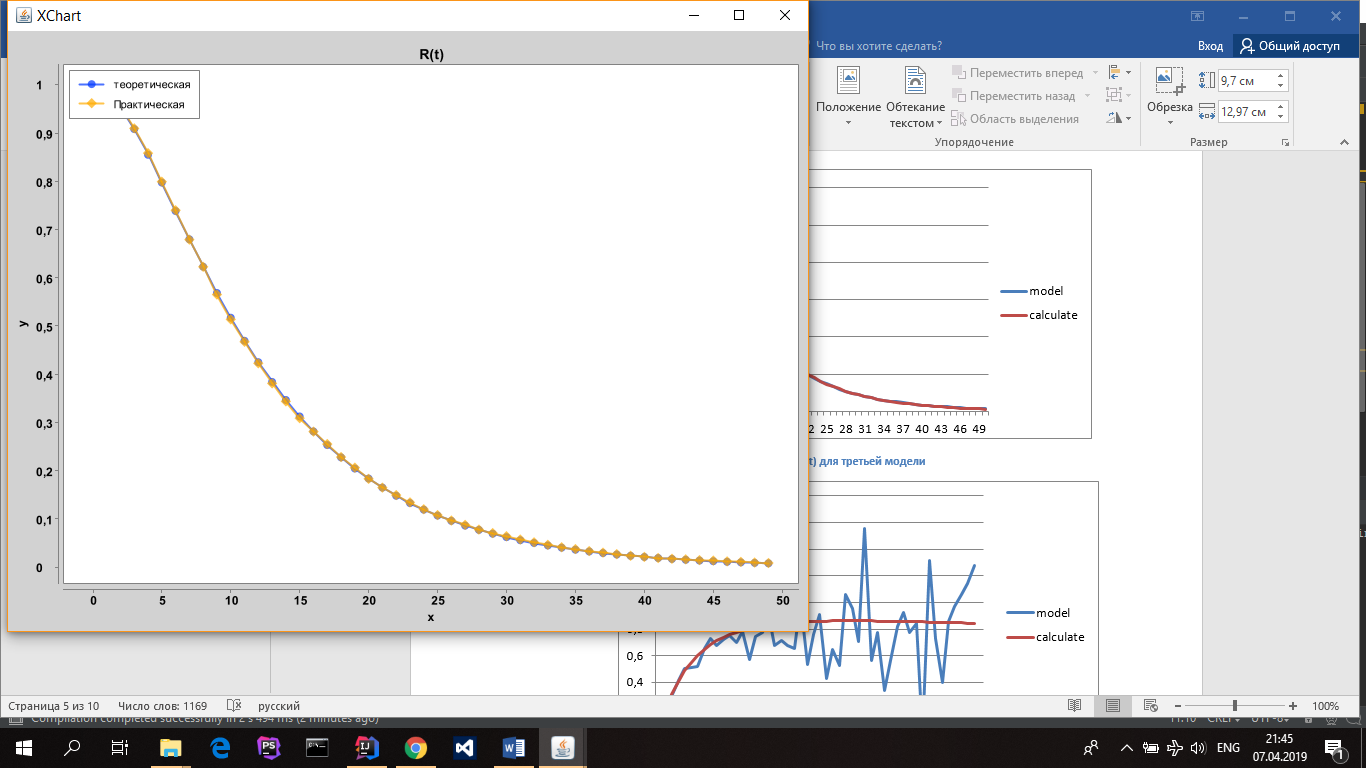


Рисунок 5 - R(t) для третьей модели

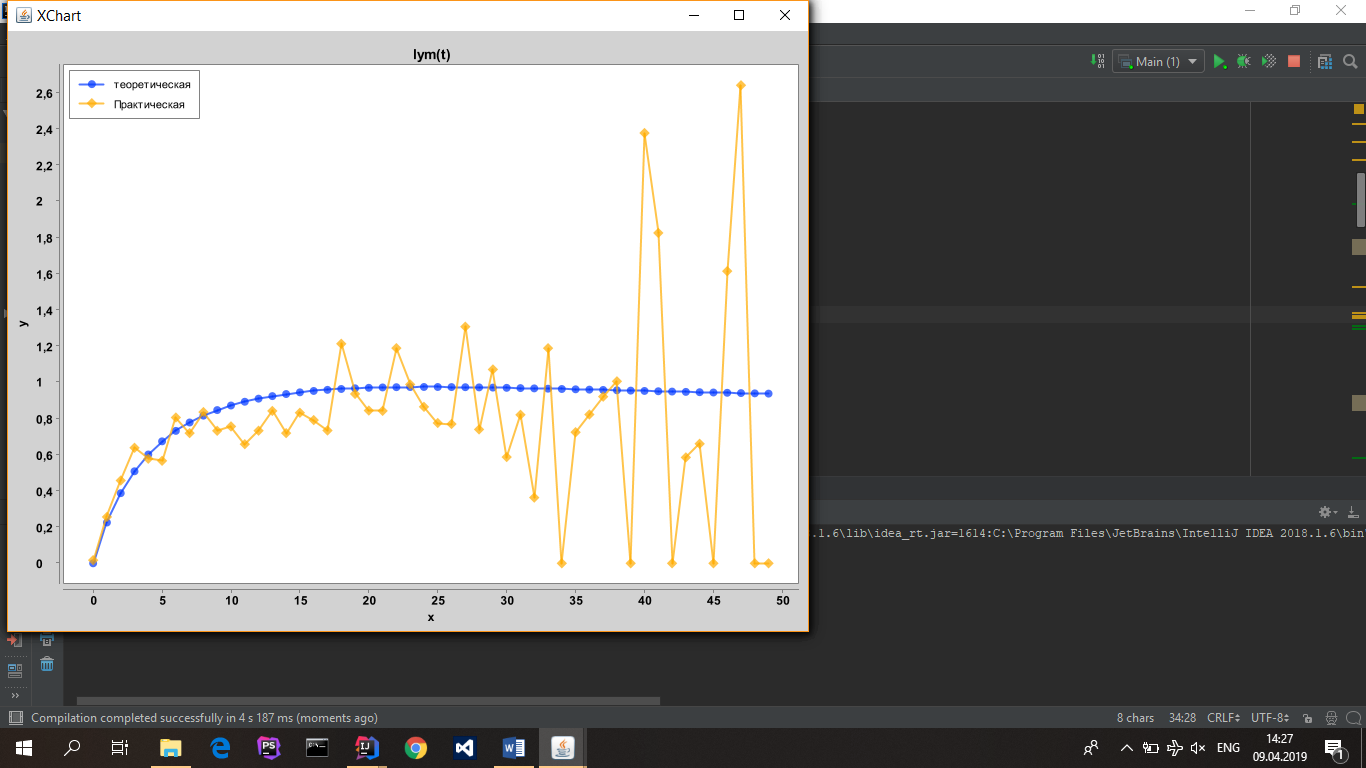


Рисунок 6- λ(t) для третьего периода

Вывод: в данной лабораторной работе было выполнено имитационное моделирование для моделей разного периода. Также для каждой модели были построены графики теоритических значений.

Листинг программы

import java.util.ArrayList;  
  
public class Lab3 {  
 private int N;  
 private double lymda[];  
 private double prob[];  
 private double step;  
 private double delta;  
  
 public Lab3(int n,double[] l, double []pr, double s, double d) {  
 N = n;  
 lymda = l;  
 prob = pr;  
 step = s;  
 delta = d;  
 }  
 public void firstStage()  
 {  
 ArrayList<Double>R = new ArrayList<>();  
 ArrayList<Double>curs = new ArrayList<>();  
 ArrayList<Double>R\_teor = new ArrayList<>();  
 ArrayList<Double>lym = new ArrayList<>();  
 ArrayList<Double>lym\_teor = new ArrayList<>();  
 double T\_average = 0;  
 double T\_max = 0;  
 double T[] = new double[N];  
 int index = 0;  
 for(int j = 0; j < N; j++)  
 {  
 if(j ==(int)(N \* prob[index]))  
 {  
 index++;  
 }  
 T[j] = getLymda(index);  
 T\_average += T[j];  
 if(T[j] > T\_max)  
 T\_max = T[j];  
 }  
 T\_average = T\_average / N;  
 double Tmidle = (T\_average + T\_max) / 2;  
 double st = Tmidle/step;  
 double delta\_S = st / delta;  
 double cur = 0;  
 for(int i = 0; i < step; i++)  
 {  
 int counter = 0;  
 int counter\_delta = 0 ;  
 for(int j = 0; j < N; j++)  
 {  
 if(T[j] > cur)  
 counter++;  
 if(T[j] > cur + delta\_S)  
 counter\_delta++;  
 }  
 R.add((double)counter/N);  
 lym.add((counter-counter\_delta)/(counter \* delta\_S));  
 R\_teor.add((Math.*pow*(Math.*exp*(1), -lymda[0]\*cur) \* prob[0] + Math.*pow*(Math.*exp*(1), -lymda[1]\*cur) \* prob[1]));  
 lym\_teor.add(-(-lymda[0] \* Math.*pow*(Math.*exp*(1), -lymda[0] \* cur) \* prob[0] - lymda[1] \* Math.*pow*(Math.*exp*(1), -lymda[1] \* cur) \* prob[1]) / (Math.*pow*(Math.*exp*(1), -lymda[0]\*cur) \* prob[0] + Math.*pow*(Math.*exp*(1), -lymda[1]\*cur) \* prob[1]));  
 curs.add(cur);  
 cur += st;  
 }  
 Schedule.*print*(R\_teor,R,curs,"R(t)");  
 Schedule.*print*(lym\_teor,lym,curs,"lym(t)");  
  
 }  
 public void secondAndThirdStage(boolean stage)  
 {  
 ArrayList<Double>R = new ArrayList<>();  
 ArrayList<Double>R\_teor = new ArrayList<>();  
 ArrayList<Double>curs = new ArrayList<>();  
 ArrayList<Double>lym = new ArrayList<>();  
 ArrayList<Double>lym\_teor = new ArrayList<>();  
 double T\_average = 0;  
 double T\_max = 0;  
 double T[] = new double[N];  
 double T1 = 0;  
 double T2 = 0;  
 int index = 0;  
 for(int j = 0; j < N; j++)  
 {  
 T1 = getLymda(0);  
 T2 = getLymda(1);  
 if(!stage) {  
 if (T1 < T2)  
 T[j] = T1;  
 else  
 T[j] = T2;  
 }  
 else {  
 if (T1 > T2)  
 T[j] = T1;  
 else  
 T[j] = T2;  
 }  
 T\_average += T[j];  
 if(T[j] > T\_max)  
 T\_max = T[j];  
 }  
 T\_average = T\_average / N;  
 double Tmidle = (T\_average + T\_max) / 2;  
 double st = Tmidle/step;  
 double delta\_S = st / delta;  
 double cur = 0;  
 for(int i = 0; i < step; i++)  
 {  
 int counter = 0;  
 int counter\_delta = 0 ;  
 for(int j = 0; j < N; j++)  
 {  
 if(T[j] > cur)  
 counter++;  
 if(T[j] > cur + delta\_S)  
 counter\_delta++;  
 }  
 R.add((double)counter/N);  
 lym.add((counter-counter\_delta)/(counter \* delta\_S));  
 if(!stage) {  
 R\_teor.add((Math.*pow*(Math.*exp*(1), (-1) \* lymda[0] \* cur)) \* (Math.*pow*(Math.*exp*(1), (-1) \* lymda[1] \* cur)));  
 lym\_teor.add(lymda[1] + lymda[0]);  
 }  
 else {  
 R\_teor.add(Math.*pow*(Math.*exp*(1), (-1) \* lymda[0] \* cur) + Math.*pow*(Math.*exp*(1), (-1) \* lymda[1] \* cur) - Math.*pow*(Math.*exp*(1), (-1) \* (lymda[1] + lymda[0]) \* cur));  
 lym\_teor.add(((lymda[0] \* Math.*pow*(Math.*exp*(1),(-1)\*lymda[0]\*cur)) + (lymda[1] \* Math.*pow*(Math.*exp*(1),(-1)\*lymda[1]\*cur)) - (lymda[1] + lymda[0])\* Math.*pow*(Math.*exp*(1),(-1)\*(lymda[1]+ lymda[0])\*cur))/((Math.*pow*(Math.*exp*(1),(-1)\*lymda[0]\*cur)) + (Math.*pow*(Math.*exp*(1),(-1)\*lymda[1]\*cur)) - Math.*pow*(Math.*exp*(1),(-1)\*(lymda[1]+ lymda[0])\*cur)));  
 }  
 curs.add(cur);  
 cur += st;  
 }  
 Schedule.*print*(R\_teor,R,curs,"R(t)");  
 Schedule.*print*(lym\_teor,lym,curs,"lym(t)");  
  
 }  
 private double getLymda(int index)  
 {  
 return Math.*log*(Math.*random*())/lymda[index]\*(-1);  
 }

public class Main {  
 public static void main(String[] arg)  
 {  
 Lab3 lab3 = new Lab3(10000,new double[]{0.9,1.3},new double[]{0.7,0.3},50,10);  
 lab3.firstStage();  
 lab3.secondAndThirdStage(false);  
 lab3.secondAndThirdStage(true);  
 }  
  
}

import java.util.ArrayList;  
import java.util.List;  
import org.knowm.xchart.\*;  
import org.knowm.xchart.style.Styler;  
  
public class Schedule {  
 public static void print (ArrayList<Double> array, ArrayList<Double> array1,ArrayList<Double> list,String name) {  
  
 ArrayList<Integer> list1 = new ArrayList<>();  
 for (int i = 0; i < 50;i++)  
 {  
 list1.add(i);  
 }  
  
 XYChart chart = new XYChartBuilder().width(800).height(600).title(name).xAxisTitle("x").yAxisTitle("y").build();  
  
 chart.getStyler().setLegendPosition(Styler.LegendPosition.*InsideNW*);  
 chart.getStyler().setHasAnnotations(false);  
 chart.getStyler().setPlotGridLinesVisible(false);  
  
 chart.addSeries("теоретическая", list1, array);  
 chart.addSeries("Практическая", list1,array1);  
 List<XYChart> charts = new ArrayList<XYChart>();  
 charts.add(chart);  
 new SwingWrapper<XYChart>(charts).displayChartMatrix();  
  
 }  
}