Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №2

Название Марковские процессы

Дисциплина Моделирование

Студент Зайцева А. А.

Группа ИУ7-72Б

Оценка (баллы) _____

Преподаватель Рудаков И. В.

Москва — 2022 г.

1 Задание

Написать программу, которая позволяет определить время пребывания сложной системы в каждом из состояний в установившемся режиме работы. Количество состояний ≤ 10.

Реализовать интерфейс, который позволяет указать количество состояний и значения матрицы вероятностей переходов, а также отображает результаты работы программы: графики вероятностей состояний, время стабилизации вероятности каждого состояния, стабилизировавшееся значение вероятности каждого состояния.

2 Теоретические сведения

Случайный процесс, протекающий в некоторой системе S, называется Марковским, если он обладает следующим свойством: для каждого момента времени вероятность любого состояния системы в будущем зависит только от её состояния в настоящем и не зависит от того, когда и каким образом она пришла в это состояние (то есть не зависит от прошлого).

Для марковского процесса обычно составляются уравнения Колмогорова:

$$F = (P'(t), P(t), \lambda) = 0,$$

где λ - некоторый набор коэффициентов.

Интегрирование системы уравнений даёт искомые вероятности как функции времени. Начальное условие берется в зависимости от того, какое было начальное состояние системы. Кроме того, необходимо добавить условие нормировки: $\sum_{i=1}^{n} P_i(t) = 1$ для любого момента t. $P_i(t)$ – вероятность того, что в момент t система будет находиться в i-м состоянии.

Уравнения Колмогорова строятся по следующим правилам:

- В левой части каждого уравнения стоит производная вероятности i-ого состояния, а правая часть содержит столько членов, сколько переходов связано с данным состоянием.

- Если переход осуществляется из этого состояния, то соответствующий член имеет знак минус, если в это состояние, то плюс.

- Каждый член равен произведению плотности вероятности перехода (интенсивности), соответствующей данному переходу, и вероятности того состояния, из которого осуществляется переход.

Для определения предельных вероятностей при $t \to \infty$ (то есть вероятностей в стационарном режиме работы), необходимо приравнять левые части уравнений (то есть производные) к нулю и решить полученную систему линейных уравнений.

Чтобы найти время стабилизации, необходимо найти найти момент времени t_s, когда значение производной $P_i^{'}(t_s)$ меньше заранее заданного ε. Тогда приращение соответствующей вероятности к следующему моменту времени $\Delta P_i = P_i^{'}(t_s)\Delta t$ будет меньше некоторой погрешности.

3 Результаты работы программы

В начале работы система находится в первом состоянии, ε для определения стабилизации вероятности принимается равным 10^{-5}.

На рисунке 1 приведен пример работы программы для 3 состояний.

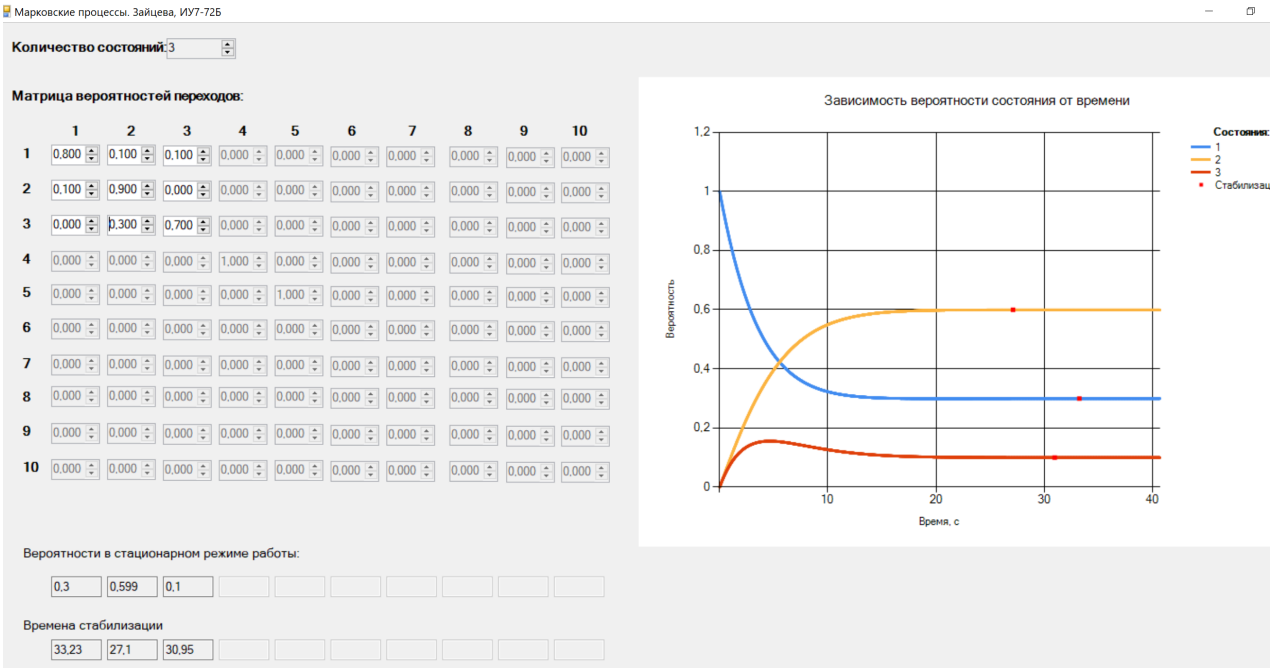

Рисунок 1 – Пример работы программы для 3 состояний

Проверим результаты, приведенные на рисунке выше. Составим систему линейных уравнений для определения вероятностей в стационарном режиме: составим уравнения Колмогорова и приравняем левые части к 0, а также добавим условие нормировки.

$$\begin{cases} 0 = 0.1 \cdot P_2 - 0.1 \cdot P_1 - 0.1 \cdot P_1 \\ 0 = 0.1 \cdot P_1 + 0.3 \cdot P_3 - 0.1 \cdot P2 \\ 0 = 0.1 \cdot P_1 - 0.3 \cdot P_3 \\ P_1 + P_2 + P_3 = 1 \end{cases} \tag{1}$$

$$\begin{cases} P_1 = \frac{3}{10} \\ P_2 = 2 \cdot P_1 = \frac{6}{10} \\ P_3 = \frac{1}{3} \cdot P_1 = \frac{1}{10} \end{cases} \tag{2}$$

Вычисленные значения совпадают с результатами программы.

На рисунке 2 вероятность перехода из второго состояния в любое другое равна 0, поэтому в стабилизировавшемся режиме вероятность этого состояния примерно равна 1, а остальных – 0.

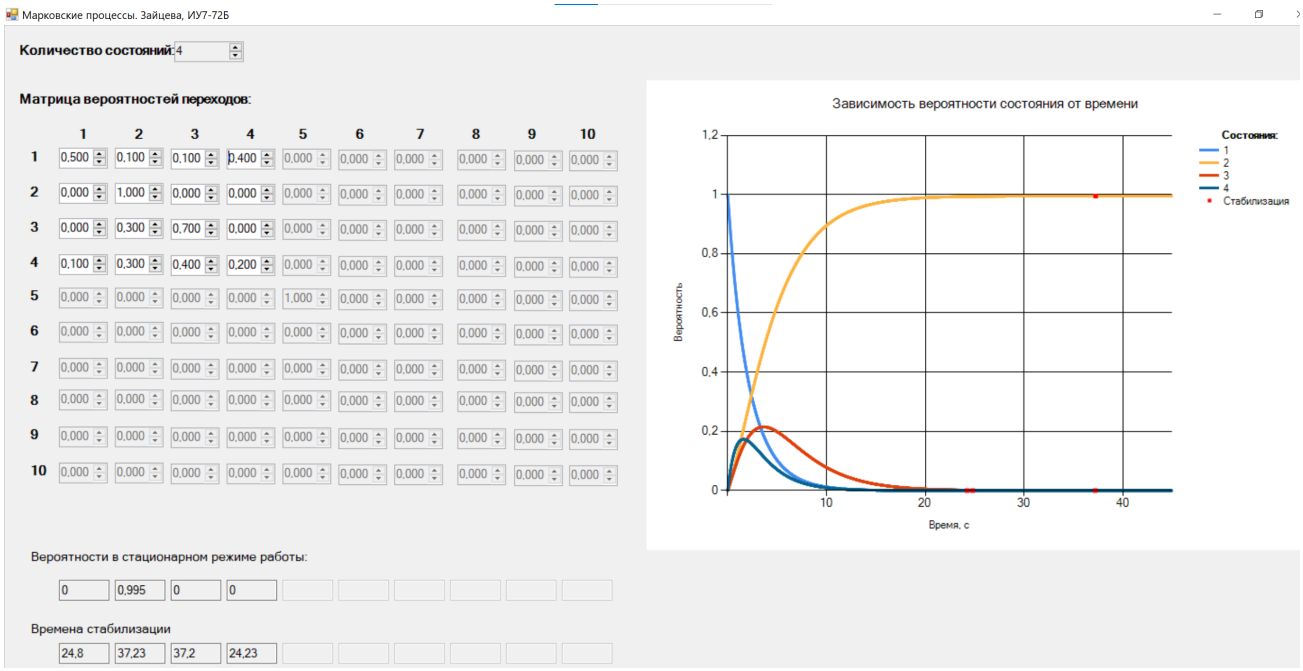

Рисунок 2 – Пример работы программы для 4 состояний

На рисунке 3 из первого состояния есть вероятность перейти в другие, но вероятность попасть из любого состояния в первое равна 0, поэтому и в стабилизировавшемся режиме вероятность первого состояния равна 0.

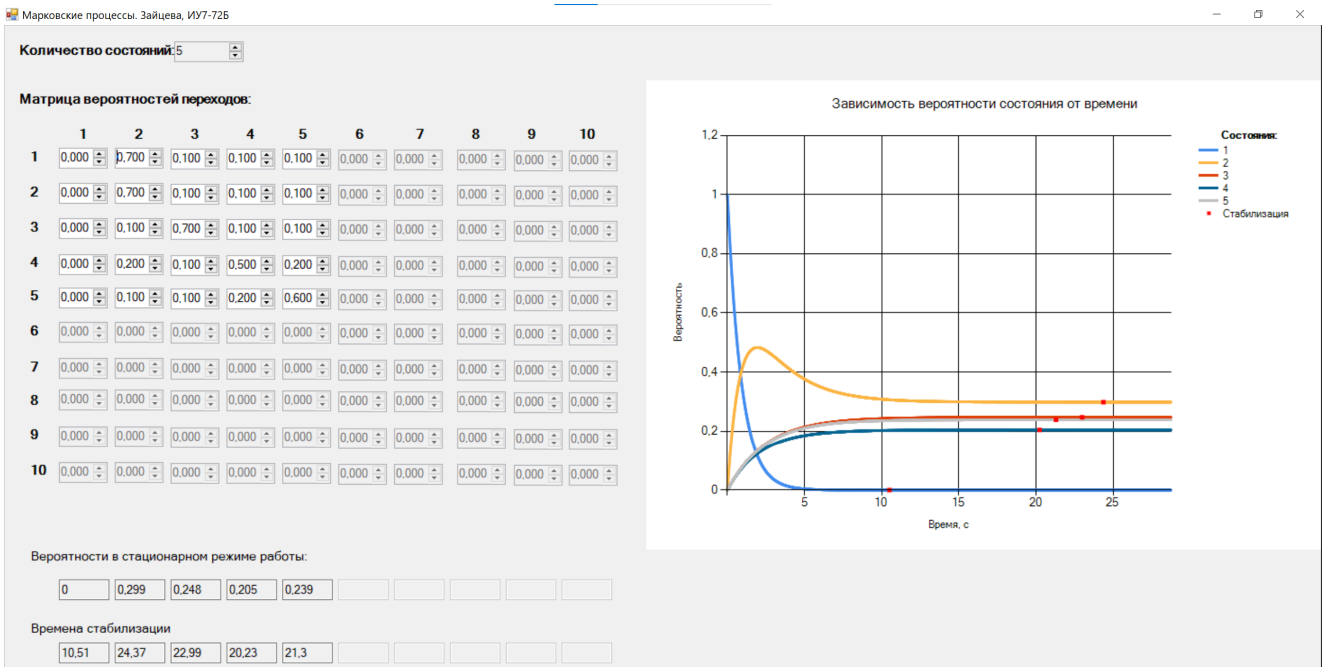

Рисунок 3 – Пример работы программы для 5 состояний

На рисунке 4 приведен пример работы программы для 10 состояний.

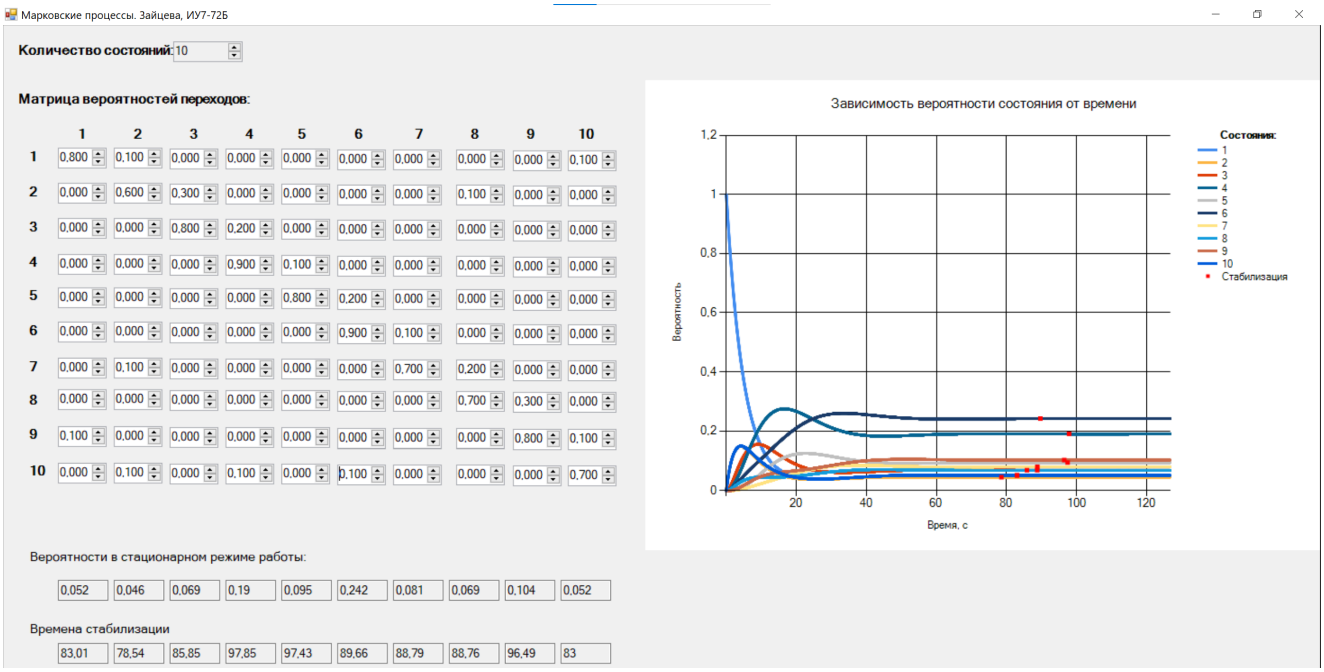

Рисунок 4 – Пример работы программы для 10 состояний

4 Код программы

Класс EmulationModel, используемый для расчетов и построения графиков, приведен в листинге 1 (используемый язык – C#).

Листинг 1 – Класс EmulationModel, используемый для расчетов и построения графиков

```
0     class EmulationModel
1 {
2     public int NStates;
3     public double[,] mtr;
4     public double[] pArr;
5     public double[] tStableArr;
6     public Chart currentChart;
7     readonly double step = 0.01;
8     readonly double stabEpsilon = 1e-5;
9     readonly double zeroEpsilon = 1e-8;
10
```

```
11      public EmulationModel(int nStates, ref Chart chart)
12      {
13          NStates = nStates;
14          pArr = new double[NStates];
15          tStableArr = new double[NStates];
16          mtr = new double[NStates, NStates];
17          currentChart = chart;
18          _initParray();
19      }
20
21      public void Emulate()
22      {
23          _initSeries();
24          double[] deltaProbArray = new double[NStates];
25          deltaProbArray[0] = 2 * stabEpsilon;
26
27          for (double currentT = step; !_checkModelStabelized(deltaProbArray
    ); currentT += step)
28          {
29              _drawArrayOnCurrentT(currentT, pArr);
30
31              deltaProbArray = new double[NStates];
32              double[] PderivativeArr = new double[NStates];
33
34              for (int i = 0; i < NStates; i++)
35              {
36                  for (int j = 0; j < NStates; j++)
37                  {
38                      double probDensityToAdd = mtr[j, i] * pArr[j] - mtr[i,
     j] * pArr[i];
39                      PderivativeArr[i] += probDensityToAdd;
40                      deltaProbArray[i] += probDensityToAdd * step;
41                  }
42                  pArr[i] += deltaProbArray[i];
43              }
44
45              _checkSomeStatesStabelized(currentT, PderivativeArr);
46          }
47          _drawStabelizedParr();
48      }
```

```
49
50      private void _initParray()
51      {
52          pArr[0] = 1;
53          for (int i = 1; i < NStates; i++)
54          pArr[i] = 0;
55      }
56
57      private void _initSeries()
58      {
59          currentChart.Series.Clear();
60          for (int i = 0; i < NStates; i++)
61          {
62              currentChart.Series.Add((i + 1).ToString());
63              currentChart.Series[i].ChartType = SeriesChartType.Line;
64              currentChart.Series[i].BorderWidth = 3;
65          }
66
67          currentChart.Series.Add("Стабилизация");
68          currentChart.Series[NStates].ChartType = SeriesChartType.Point;
69          currentChart.Series[NStates].Color = Color.Red;
70      }
71
72      private bool _checkModelStabelized(double[] arr)
73      {
74          for (int i = 0; i < arr.Length; i++)
75          if (arr[i] > zeroEpsilon)
76          return false;
77          return true;
78      }
79
80      private void _checkSomeStatesStabelized(double currentT, double[]
    klmArr)
81      {
82          for (int i = 0; i < NStates; i++)
83          {
84              if (Math.Abs(klmArr[i]) < stabEpsilon && tStableArr[i] == 0)
85              tStableArr[i] = currentT;
86
87              else if (Math.Abs(klmArr[i]) > stabEpsilon && tStableArr[i] !=
```

```
    0)
88              tStableArr[i] = 0;
89          }
90      }
91
92      private void _drawArrayOnCurrentT(double currentT, double[] arr)
93      {
94          for (int i = 0; i < NStates; i++)
95          {
96              currentChart.Series[i].Points.AddXY(currentT, arr[i]);
97          }
98      }
99
100     private void _drawStabelizedParr()
101     {
102         for (int i = 0; i < NStates; i++)
103         {
104             currentChart.Series[NStates].Points.AddXY(tStableArr[i], pArr[
    i]);
105         }
106     }
107 }
```