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

по курсу «Моделирование»

на тему: «Цепи Маркова»

Студент ИУ7-72Б _____ Т. М. Сучкова
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ И. В. Рудаков
(Подпись, дата) (И. О. Фамилия)

2022 г.

1 Задание

Необходимо определить время пребывания системы в каждом состоянии в установившемся режиме работы системы массового обслуживания.

Количество состоянииий системы не больше 10 (вводится перед началом рассчета). Граф задается матрицей. На пересечении строк и столбцов – значения интенсивности перехода состояния. После нажатия кнопки «рассчитать» получаем вероятность пребывания в каждом состоянии и соответствующее время. Разработать соответствующий интерфейс.

2 Теоретическая часть

Случайный процесс, протекающий в некоторой системе S, называется **марковским**, если он обладает свойством: для каждого момента времени t_0 вероятность любого состояния системы в будущем (при $t > t_0$) зависит только от её состояния в настоящем, и не зависит от того, когда и каким образом система пришла в это состояние, т. е. не зависит от того, как процесс развивался в прошлом.

Для Марковских процессов обычно составляют уравнения Колмогорова, представленные в общем виде следующим соотношением:

$$F = (p'(t), p(t), \Lambda) = 0,$$

где $\Lambda = \lambda_1, \lambda_2, ..., \lambda_n$ - набор коэффициентов.

Вероятностью i-го состояния называется вероятность $p_i(t)$ того, что в момент t система будет находиться в состоянии S_i. К системе может быть добавлено условие нормировки: для любого момента t сумма вероятностей всех состояний равна единице:

$$\sum_{i=1}^{n} p_i = 1.$$

Для того, чтобы решить поставленную задачу, необходимо составить систему уравнений Колмогорова. Каждое уравнение Колмогорова строится по следующему правилу.

- В левой части каждого уравнения стоит производная вероятности i-ого состояния.
- В правой части содержится столько членов, сколько переходов, связанных с данным состоянием. Если переход из состояния, то соответствующий член имеет знак минус, если в состояние, то плюс.
- Каждый член равен произведению плотности вероятности перехода (т.е. интенсивности), соответствующей данной стрелке, на вероятность того состояния, из которого исходит стрелка.

3 Результат

На рисунках 3.1 и 3.2 представлены примеры результата работа программы для 3 и 8 состояний соответственно.

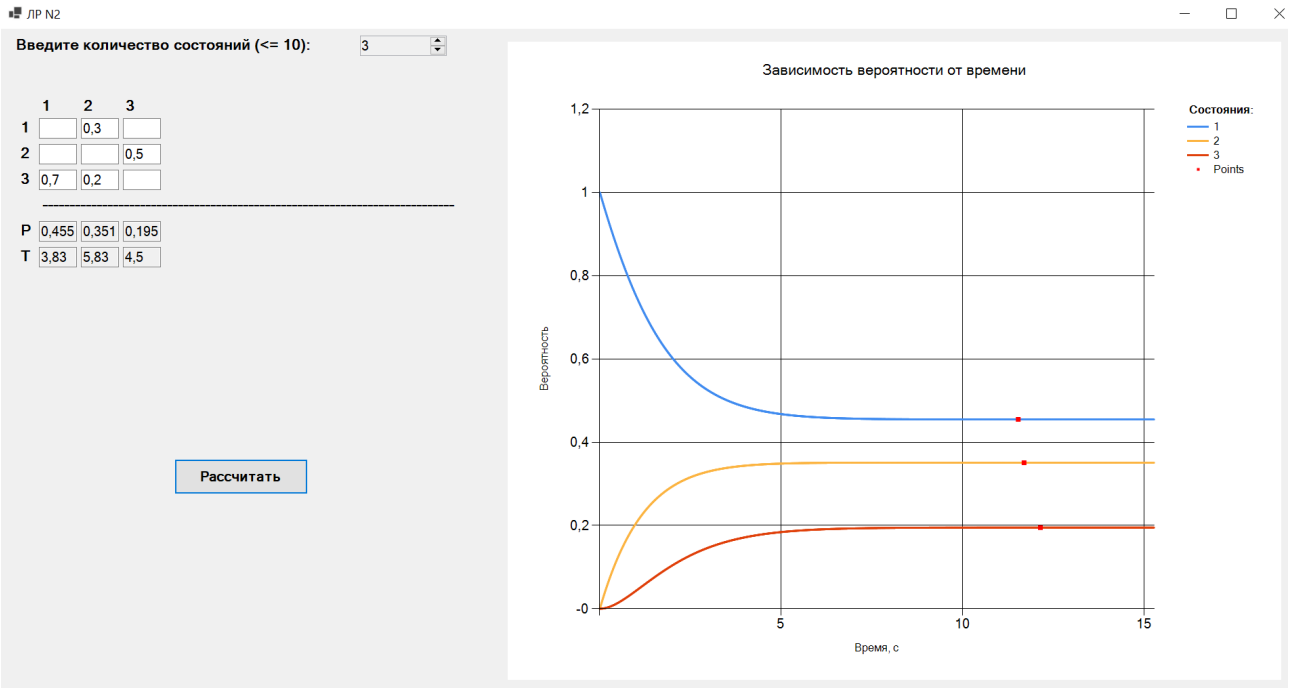

Рисунок 3.1 – Пример 1 (3 состояния)

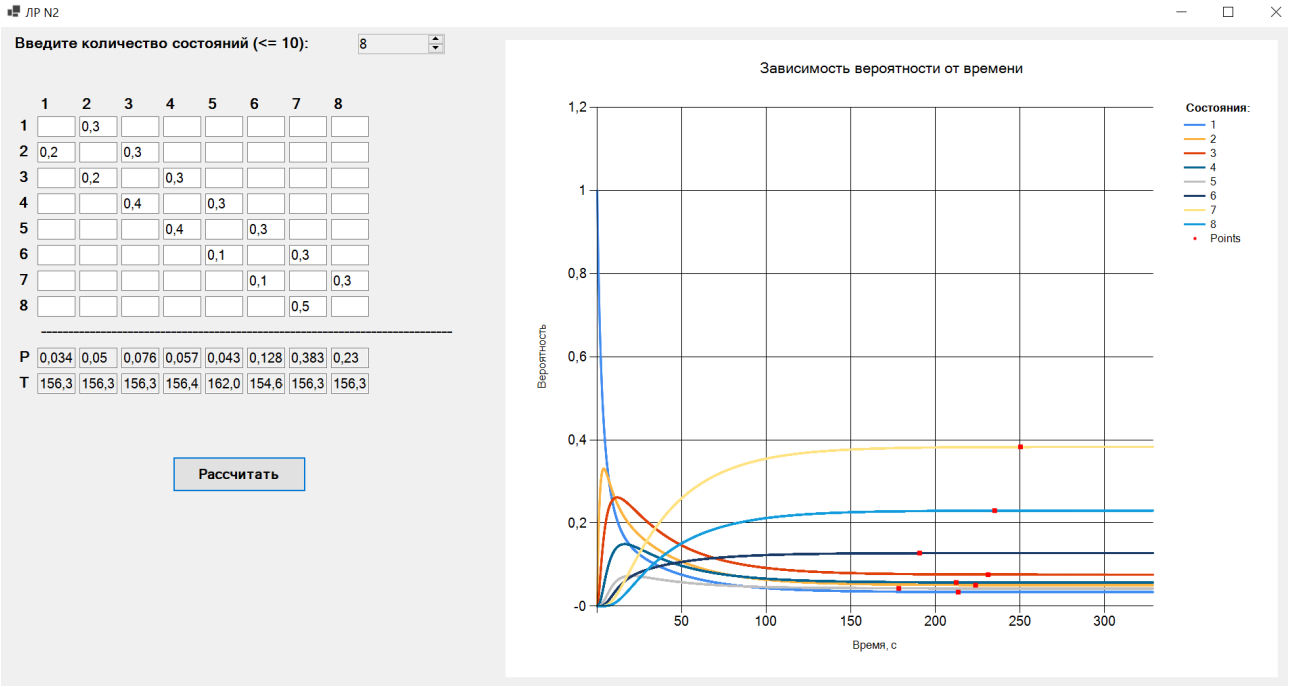

Рисунок 3.2 – Пример 2 (8 состояний)

4 Код программы

В листингах 4.1–4.5 представлен основной код программы.

Листинг 4.1 – Класс SModel (часть 1)

```
class SModel
{
    public int MaxStateNum { get; }
    public int defaultNum { get; }
    public static int inputNum;
    public double[,] mtr;
    public double[] pArr;
    public double[] tArr;
    public double[] steadyTimeArr;

    public SModel()
    {
        MaxStateNum = 10;
        defaultNum = 5;
        pArr = new double[MaxStateNum];
        inputNum = defaultNum;
        mtr = new double[MaxStateNum, MaxStateNum];

        _initP();
    }

    public void Emulate(ref Chart chart)
    {
        double[] tempArr = new double[inputNum];
        tArr = new double[inputNum];
        double step = 0.01, t = step, temp;

        List<double> tArrAll = new List<double>();
        List<List<double>> pArrAll = new List<List<double>>();

        for (int i = 0; i < inputNum; i++)
        {
            pArrAll.Add(new List<double>());
        }

        _initSeries(ref chart);
```

Листинг 4.2 – Класс SModel (часть 2)

```
37
38          while (true)
39          {
40              double[] klmArr = new double[inputNum];
41              tArrAll.Add(t);
42              for (int i = 0; i < inputNum; i++)
43              {
44                  pArrAll[i].Add(pArr[i]);
45              }
46
47              _draw(t, pArr, ref chart);
48
49              for (int i = 0; i < inputNum; i++)
50              {
51                  for (int j = 0; j < inputNum; j++)
52                  {
53                      temp = mtr[j, i] * pArr[j] - mtr[i, j] * pArr[i
                            ];
54                      tempArr[i] += temp * step;
55                      klmArr[i] += temp;
56                  }
57              }
58
59              for (int i = 0; i < inputNum; i++)
60                  pArr[i] += tempArr[i];
61
62              _checkStab(t, klmArr, ref tArr);
63
64              if (__isZeroArr(tempArr))
65                  break;
66
67              _resetArr(ref tempArr);
68
69              t += step;
70          }
71
72          _getSteadyTime(tArrAll, pArrAll, tArr, ref steadyTimeArr);
73
74          _drawPoints(tArr, pArr, ref chart);
75      }
```

Листинг 4.3 – Класс SModel (часть 3)

```
77      private void _initP()
78      {
79          pArr[0] = 1;
80          for (int i = 1; i < MaxStateNum; i++)
81              pArr[i] = 0;
82      }
83
84      private static void _initSeries(ref Chart chart)
85      {
86          chart.Series.Clear();
87
88          for (int i = 0; i < inputNum; i++)
89          {
90              chart.Series.Add((i + 1).ToString());
91              chart.Series[i].ChartType = SeriesChartType.Line;
92              chart.Series[i].BorderWidth = 3;
93          }
94          chart.Series.Add("Points");
95          chart.Series[inputNum].ChartType = SeriesChartType.Point;
96          chart.Series[inputNum].Color = Color.Red;
97      }
98
99      private void _resetArr(ref double[] arr)
100     {
101         for (int i = 0; i < arr.Length; i++)
102             arr[i] = 0;
103     }
104
105     private static bool _isZeroArr(double[] arr)
106     {
107         double eps = 1e-8;
108         for (int i = 0; i < arr.Length; i++)
109             if (arr[i] > eps)
110                 return false;
111         return true;
112     }
```

Листинг 4.4 – Класс SModel (часть 4)

```
114     private static void _checkStab(double t, double[] klmArr, ref
            double[] tArr)
115     {
116         double eps = 1e-5;
117 
118         for (int i = 0; i < inputNum; i++)
119         {
120             if (Math.Abs(klmArr[i]) > eps && tArr[i] != 0)
121                 tArr[i] = 0;
122             else if (Math.Abs(klmArr[i]) < eps && tArr[i] == 0)
123                 tArr[i] = t;
124         }
125     }
126 
127     private static void _getSteadyTime(List<double> tArrAll, List<
            List<double>> pArrAll, double[] tArr, ref double[]
            steadyTimeArr)
128     {
129         double eps = 1e-5;
130         bool flag = true;
131 
132         steadyTimeArr = new double[inputNum];
133 
134         for (int i = 0; i < inputNum; i++)
135         {
136             flag = true;
137             double next_p = pArrAll[i][tArrAll.Count - 1];
138 
139             for (int j = tArrAll.Count - 2; j > -1; j--)
140             {
141                 double cur_p = pArrAll[i][j];
142                 if (Math.Abs(cur_p - next_p) > eps)
143                 {
144                     steadyTimeArr[i] = Math.Abs(tArr[i] - tArrAll[j
                            ]);
145                     flag = false;
146                     break;
147                 }
148                 next_p = cur_p;
149             }
```

Листинг 4.5 – Класс SModel (часть 5)

```
150
151             if (flag)
152             {
153                 steadyTimeArr[i] = 0;
154             }
155         }
156     }
157
158     private static void _draw(double t, double[] arr, ref Chart
           chart)
159     {
160         for (int i = 0; i < inputNum; i++)
161         {
162             chart.Series[i].Points.AddXY(t, arr[i]);
163         }
164     }
165
166     private static void _drawPoints(double[] x, double[] y, ref
           Chart chart)
167     {
168         for (int i = 0; i < inputNum; i++)
169         {
170             chart.Series[inputNum].Points.AddXY(x[i], y[i]);
171         }
172     }
173 }
```