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» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

О Т Ч Е Т

по лабораторной работе № 2

Название: Цепи Маркова

Дисциплина: Моделирование

Студент ИУ7-72Б _____ Е.В. Брянская
(Группа) (Подпись, дата) (И.О. Фамилия)

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

Москва, 2021

1 Задание

Написать программу, позволяющую определить время прибывания сложной системы в каждом из состояний. Количество состояний не более 10. При t $\to \infty$.

Граф задаётся матрицей. На пересечении вводятся значения интенсивностей. 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.

Общий вид:

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

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. Все уравнения составляются по определённым правилам.

- 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, а, если наоборот, то плюс.
- Каждый член равен произведению плотности вероятности перехода (т.е. интенсивности), соответствующей рассматриваемой стрелки, на вероятность того состояния, из которого исходит стрелка.

3 Результаты работы программы

На Рис. 3.1 представлен пример для 3 состояний.

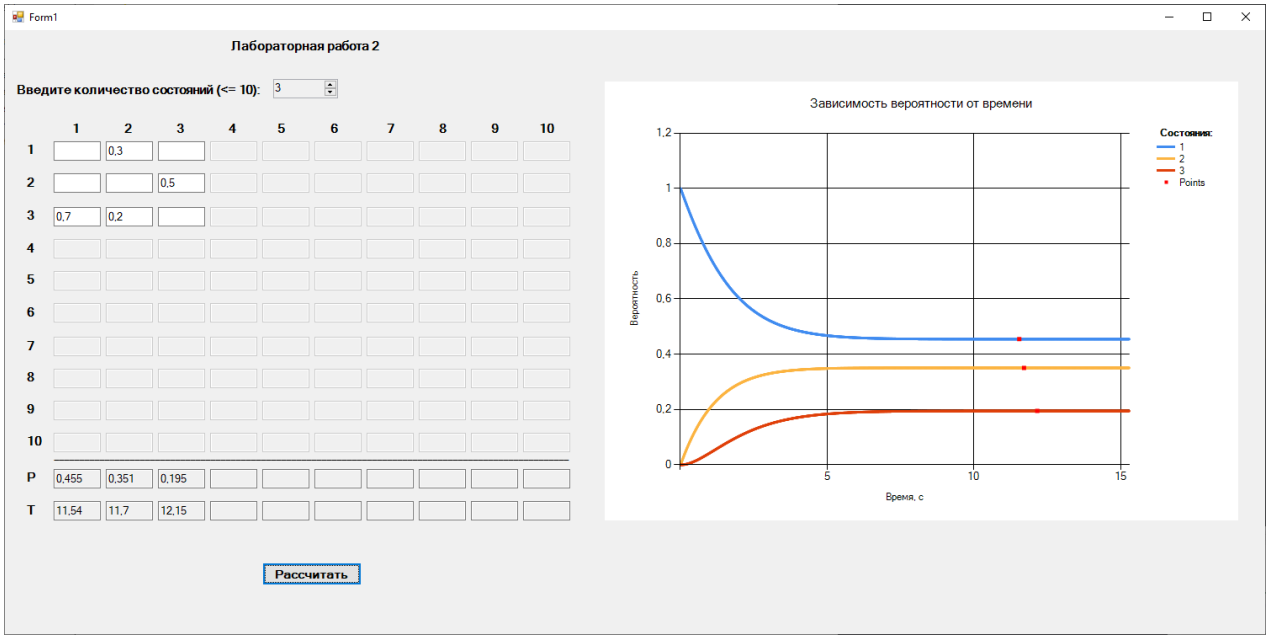

Рисунок 3.1 — Пример 1 (3 состояния)

Посчитаем P_i самостоятельно:

$$\begin{cases} 0 = 0.7 \cdot p_3 - 0.3 \cdot p_1; \\ 0 = 0.3 \cdot p_1 + 0.2 \cdot p_3 - 0.5 \cdot p_2; \\ 0 = 0.5 \cdot p_2 - 0.2 \cdot p_3 - 0.7 \cdot p_3; \\ p_1 + p_2 + p_3 = 1. \end{cases} \tag{1}$$

$$\begin{cases} p_1 = \frac{7}{3} p_3; \\ p_2 = \frac{9}{5} p_3; \\ \frac{7}{3} p_3 + \frac{9}{5} p_3 + p_3 = 1. \end{cases} \tag{2}$$

$$\begin{cases} p_1 \approx 0,4545 \approx 0,455; \\ p_2 \approx 0,3506 \approx 0,351; \\ p_3 \approx 0,1948 \approx 0,195. \end{cases} \tag{3}$$

Результаты приблизительно равны.

На Рис. 3.2 представлен пример из лекции для 8 состояний.

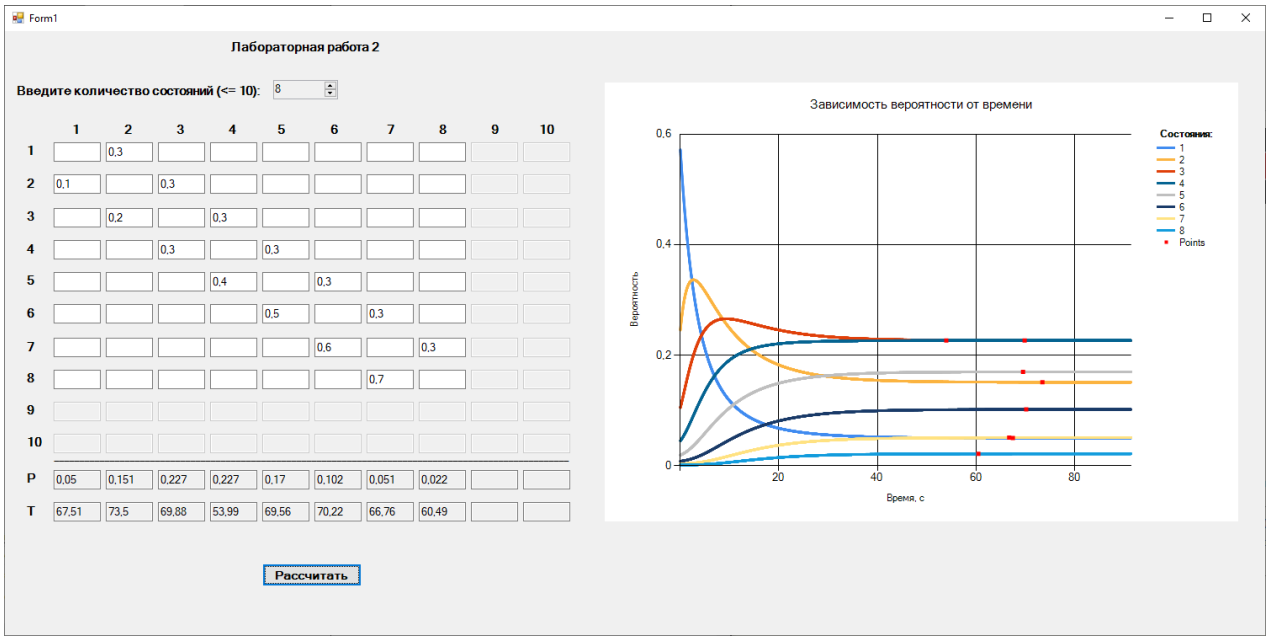

Рисунок 3.2 — Пример 2 (8 состояний)

На Рис. 3.3 представлен пример для 4 состояний, из последнего узла нет переходов в другие состояния.

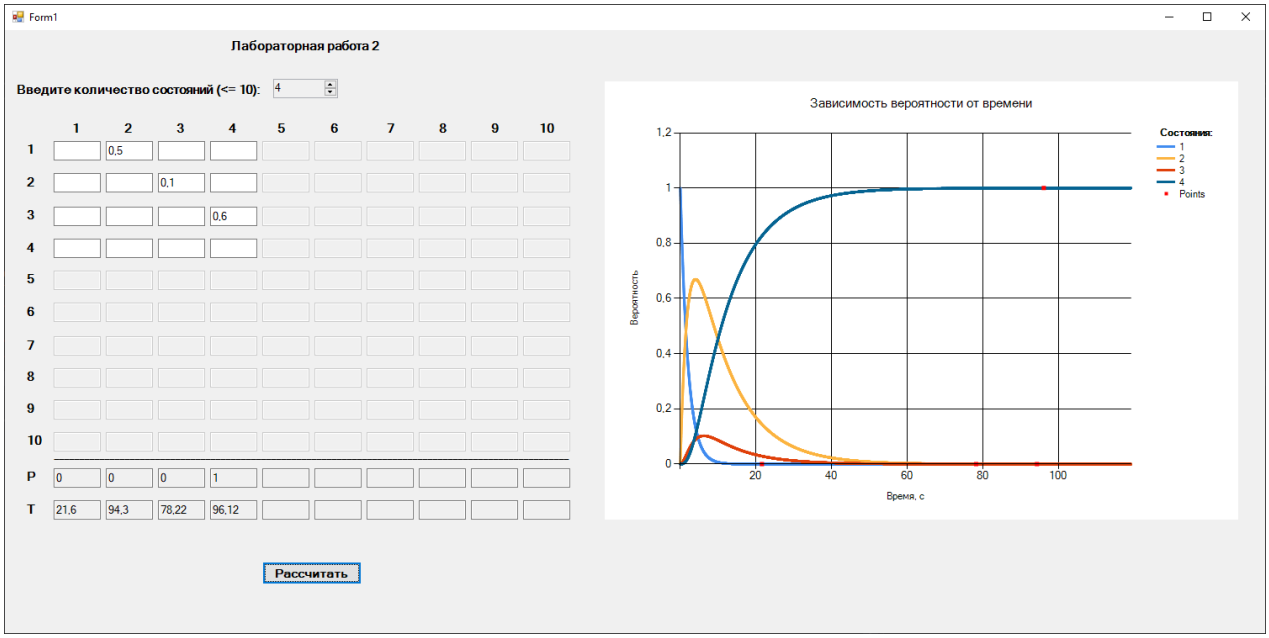

Рисунок 3.3 — Пример 3 (4 состояний)

4 Код программы

На Листинге 1 представлены основные методы.

Листинг 1 — Основные методы

```
class SModel
{
  public int numState { get; }
  public int defaultNum { get; }

  public static int inputNum;
  public double[,] mtr;

  public double[] pArr;
  public double[] timeArr;

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

    _initP();
  }

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

    _initSeries(ref chart);

    while (true) {
      double[] klmArr = new double[inputNum];

      _draw(t, pArr, ref chart);

      for (int i = 0; i < inputNum; i++)
        for (int j = 0; j < inputNum; j++) {
          temp = mtr[j, i] * pArr[j] - mtr[i, j] * pArr[i];
          tempArr[i] += temp * step;
          klmArr[i] += temp;
        }

      for (int i = 0; i < inputNum; i++)
        pArr[i] += tempArr[i];

      _checkStab(t, klmArr, ref timeArr);

```

```
46        if (_allZero(tempArr))
47          break;
48
49        _resetArr(ref tempArr);
50
51        t += step;
52      }
53
54      _drawPoints(timeArr, pArr, ref chart);
55    }
56
57    private void _initP()
58    {
59      pArr[0] = 1;
60      for (int i = 1; i < numState; i++)
61      pArr[i] = 0;
62    }
63
64    private static void _initSeries(ref Chart chart)
65    {
66      chart.Series.Clear();
67      for (int i = 0; i < inputNum; i++) {
68        chart.Series.Add((i + 1).ToString());
69        chart.Series[i].ChartType = SeriesChartType.Line;
70        chart.Series[i].BorderWidth = 3;
71      }
72
73      chart.Series.Add("Points");
74      chart.Series[inputNum].ChartType = SeriesChartType.Point;
75      chart.Series[inputNum].Color = Color.Red;
76    }
77
78    private void _resetArr(ref double[] arr)
79    {
80      for (int i = 0; i < arr.Length; i++)
81        arr[i] = 0;
82    }
83
84    private static bool _allZero(double[] arr)
85    {
86      double temp = 1e-8;
87
88      for (int i = 0; i < arr.Length; i++)
89        if (arr[i] > temp)
90          return false;
91      return true;
92    }
93
94    private static void _checkStab(double t, double[] klmArr, ref double[] timeArr)
95    {
96      double cntr = 1e-5;
```

```
97
98      for (int i = 0; i < inputNum; i++) {
99        if (Math.Abs(klmArr[i]) > cntr && timeArr[i] != 0)
100         timeArr[i] = 0;
101       else if (Math.Abs(klmArr[i]) < cntr && timeArr[i] == 0)
102         timeArr[i] = t;
103     }
104   }
105
106   private static void _draw(double t, double[] arr, ref Chart chart)
107   {
108     for (int i = 0; i < inputNum; i++)
109       chart.Series[i].Points.AddXY(t, arr[i]);
110   }
111
112   private static void _drawPoints(double[] x, double[] y, ref Chart chart)
113   {
114     for (int i = 0; i < inputNum; i++)
115       chart.Series[inputNum].Points.AddXY(x[i], y[i]);
116   }
117
118   private static void _printMtr(double[,] m)
119   {
120     for (int i = 0; i < inputNum; i++) {
121       for (int j = 0; j < inputNum; j++)
122         Console.Write(m[i, j] + " ");
123       Console.WriteLine();
124     }
125     Console.WriteLine();
126   }
127
128   private static void _printArr(double[] arr)
129   {
130     for (int i = 0; i < inputNum; i++)
131       Console.WriteLine($"{i}, {arr[i]}");
132     Console.WriteLine();
133   }
134 }
```