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 **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 2

Тема: Марковские процессы

Дисциплина: Моделирование

Студент	ИУ7-72Б		В.А. Иванов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			И.В. Рудаков
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2021

1. Задание

Определение времени пребывания сложной системы в каждом из состояний. Количество состояний ≤ 10. Рез Реализовать возможность выбора количества состояний и значений матрицы переходов и отображение результатов работы программы (графики вероятностей состояний, значение и время стабилизации вероятности состояния) при помощи графического интерфейса.

2. Результаты

2.1. Теория

Случайный процесс протекающий в некоторой системе S называется марковским если он обладает следующим свойством. Для каждого момента времени t_0 вероятность любого состояния системы в будущем зависит **только** от её состояния в настоящем и не зависит от того как и когда она пришла в это состояние.

Для марковского процесса обычно составляются уравнения Колмогорова:

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

Где λ - набор параметров Интегрирование системы уравнений даёт исходные вероятности как функции времени. Также для решения необходимо условие нормировки $\sum_i^n p_i = 1$

2.2. Работа программы

Условием стабилизации вероятности состояния i принимается величина $P_i(t_{stab})$, где t_{stab} - наименьшее время, при котором $P'_i(t_{stab}) < 10^{-7}$.

Примеры работы программы при различном количестве состояний и заполнении матрицы интенсивности приведены на рисунках 2.1, 2.2 и 2.3.

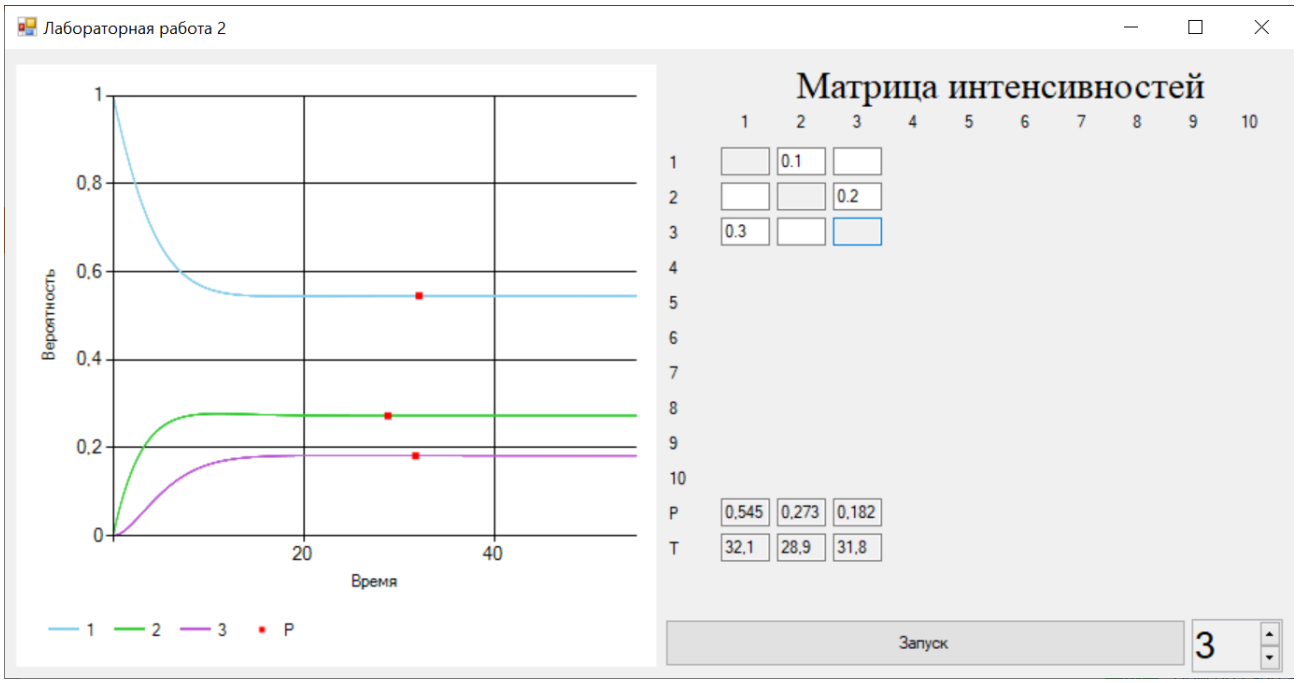

Рис. 2.1 — Графики вероятностей состояний

Проверка результатов:

$$\begin{cases} -0.1 \cdot 0.545 + 0.3 \cdot 0.182 = 1.0 \cdot 10^{-4} \\ -0.2 \cdot 0.273 + 0.1 \cdot 0.545 = -1.0 \cdot 10^{-4} \\ -0.3 \cdot 0.182 + 0.2 \cdot 0.273 = 0.0 \end{cases} \tag{2.1}$$

Вычислим стабилизировавшиеся значения P_1, P_2, P_3:

$$\begin{cases} -0.1 \cdot P_1 + 0.3 \cdot P_3 = 0 \\ -0.2 \cdot P_2 + 0.1 \cdot P_1 = 0 \\ -0.3 \cdot P_3 + 0.2 \cdot P_2 = 0 \\ P_1 + P_2 + P_3 = 1 \end{cases}$$

$$\begin{cases} P_1 = 3 \cdot P_3 \\ P_2 = 1.5 \cdot P_3 \\ 5.5 \cdot P_3 = 1 \end{cases}$$

$$\begin{cases} P_1 = \frac{6}{11} \\ P_2 = \frac{3}{11} \\ P_3 = \frac{2}{11} \end{cases}$$

Вычисленные аналитически значения совпадают со значениями, полученными программно

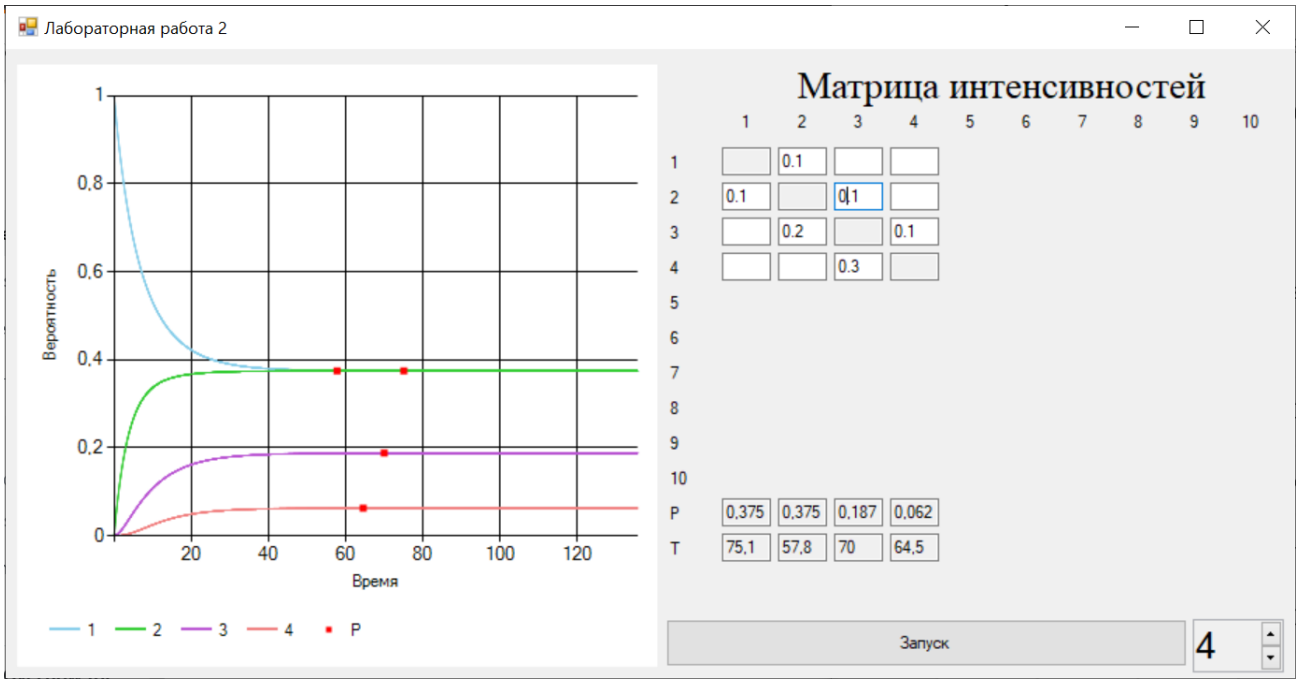

Рис. 2.2 — Графики вероятностей состояний

Проверка результатов:

$$\begin{cases} -0.1 \cdot 0.375 + 0.1 \cdot 0.375 = 0.0 \\ -(0.1 + 0.1) \cdot 0.375 + (0.1 \cdot 0.375 + 0.2 \cdot 0.187) = -1.0 \cdot 10^{-4} \\ -(0.2 + 0.1) \cdot 0.187 + (0.1 \cdot 0.375 + 0.3 \cdot 0.062) = 0.0 \\ -0.3 \cdot 0.062 + 0.1 \cdot 0.187 = 1.0 \cdot 10^{-4} \end{cases} \tag{2.2}$$

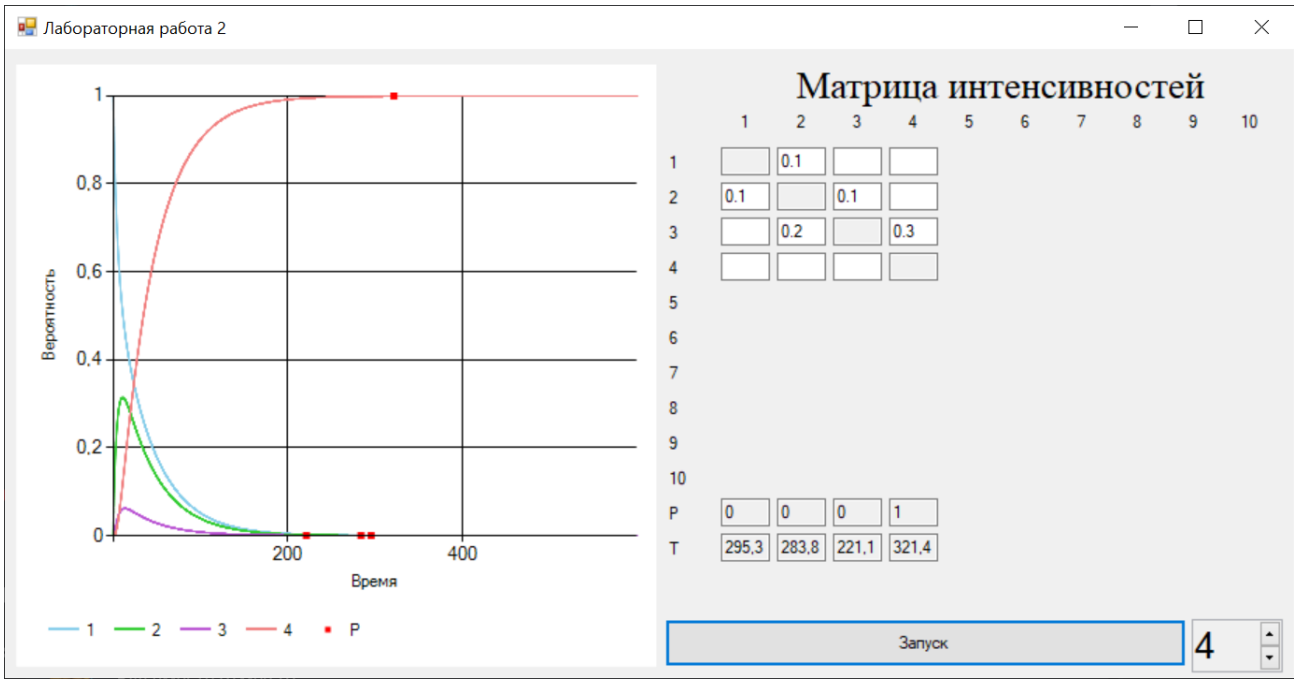

Рис. 2.3 — Графики вероятностей состояний

Проверка результатов: 4-е состояние не имеет исходящее интенсивности, поэтому при $t \to \inf$ вероятность 4-го состояния должна равняться 0.

3. Текст программы

В листинге 3.1 представлен фрагмент кода программы, отвечающий за моделирование цепей Маркова.

Листинг 3.1 — Вычисление функций распредлений

```
1  class Model
2  {
3    public const int maxN = 10;
4    public double[] pArr;
5    public double[] tArr;
6    public double[,] tMatrix;
7    public int n { get; }
8    public double T = 0;
9
10   public Model(int n_)
11   {
12     this.n = n_;
13     this.tMatrix = new double[maxN, maxN];
14     this.tArr = new double[maxN];
15     this.pArr = new double[maxN];
16     pArr[0] = 1;
17   }
18
19   // returns is stabilized
20   public bool Step(double dt)
21   {
22     double[] newP = new double[maxN];
23     for (int i=0; i < n; i++)
24     {
25       newP[i] = pArr[i];
26       for (int j = 0; j < n; j++)
27       {
28         if (i == j) continue;
29         newP[i] += dt * (pArr[j] * tMatrix[j, i] - pArr[i] *
     tMatrix[i, j]);
30       }
```

```
31      }
32
33      pArr = newP;
34      T += dt;
35
36      SetStableT();
37      return IsStable();
38    }
39
40    private bool IsStable()
41    {
42      double[] res = Kolmogorov();
43        for (int i = 0; i < n; i++)
44          if (Math.Abs(res[i]) > 1e-8)
45            return false;
46      return true;
47    }
48
49    public double[] Kolmogorov()
50    {
51      double[] res = new double[maxN];
52
53      for (int i = 0; i < n; i++)
54      {
55        res[i] = 0;
56        for (int j = 0; j < n; j++)
57        res[i] += pArr[j] * tMatrix[j, i] - pArr[i] * tMatrix[i
      , j];
58      }
59
60      return res;
61    }
62
63    private void SetStableT()
```

```
64    {
65      double[] kArr = Kolmogorov();
66      for (int i = 0; i < n; i++)
67      {
68        if (Math.Abs(kArr[i]) < 1e-5 && tArr[i] <= 1e-7)
69        tArr[i] = T;
70        else if (Math.Abs(kArr[i]) > 1e-5 && tArr[i] > 1e-7)
71        tArr[i] = 0;
72      }
73    }
74  }
```