



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 (ИУ-7)

**ОТЧЕТ**  
***ПО ЛАБОРАТОРНОЙ РАБОТЕ №3***  
***НА ТЕМУ:***

Уравнения Колмогорова

---

---

---

Студент ИУ7-78  
(Группа)  
Вариант 4

А.В. Иванников  
(И.О.Фамилия)

Преподаватель

И.В. Рудаков  
(И.О.Фамилия)

2020 г.

## Выбор языка программирования приложения

В качестве языка программирования приложения был выбран язык C#.

## Цель работы

Найти среднее относительное время пребывания системы в предельном стационарном состоянии. Система  $S$  работает в стационарном режиме. Интенсивности переходов из состояния в состояние задаются в виде матрицы размером  $\leq 10$ .

## Правила составления уравнений Колмогорова

Для решения данной задачи необходимо от заданной матрицы интенсивностей перехода из состояния в состояние перейти к уравнениям Колмогорова:

$$\begin{aligned}\frac{dp_1}{dt} &= \lambda_{21}p_2 - (\lambda_{12} + \lambda_{13})p_1, \\ \frac{dp_2}{dt} &= \lambda_{12}p_1 + \lambda_{32}p_3 - (\lambda_{24} + \lambda_{21})p_2, \\ \frac{dp_3}{dt} &= \lambda_{31}p_1 + \lambda_{43}p_4 - \lambda_{32}p_3, \\ \frac{dp_4}{dt} &= \lambda_{24}p_2 - \lambda_{43}p_4.\end{aligned}$$

В левой части каждого уравнения стоит производная вероятности  $i$ -го состояния. В правой части — сумма произведений вероятностей всех состояний, из которых идут стрелки в данное состояние на интенсивности соответствующих потоков событий, минус суммарная интенсивность всех потоков, выводящих систему из данного состояния, умноженная на вероятность данного  $i$ -го состояния.

Уравнения Колмогорова дают возможность найти все вероятности состояний как функции времени. Особый интерес представляют вероятности системы  $p_i(t)$  в предельном стационарном режиме, т.е. при  $t \rightarrow \infty$ , которые называются предельными (или финальными) вероятностями состояний.

Предельная вероятность состояния  $S_i$  имеет четкий смысл: она показывает среднее относительное время пребывания системы в этом состоянии.

Так как предельные вероятности постоянны, то при замене в уравнениях Колмогорова их производных нулевыми значениями будет получена система линейных алгебраических уравнений, описывающих стационарный режим. В полученной системе независимых уравнений на единицу меньше общего числа уравнений. Поэтому для решения системы необходимо добавить уравнение нормировки (сумма вероятностей всех состояний равна единице).

## Пример работы приложения

Лабораторная работа №3 (Иванников А.В., гр. ИУ7-78)

Размер матрицы интенсивностей

Матрица интенсивностей потоков

	S0	S1	S2	S3
S0	0	2	0	3
S1	2	0	3	0
S2	3	3	0	1
S3	0	0	2	0
*				

Расширенная матрица системы уравнений Колмогорова

	-5	2	3	0	0	
	2	-5	3	0	0	
	0	3	-7	2	0	
	3	0	1	-2	0	
	1	1	1	1	1	
*						

Среднее относительное время пребывания системы в предельном стационарном состоянии

	S0	S1	S2	S3
T	0,2	0,2	0,2	0,4
*				

Рисунок 1 – Пример работы приложения

## Листинг программного кода приложения

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Процесс_Маркова
{
    public partial class Form1 : Form
    {
        public void Calculate()
        {
            int n = Convert.ToInt32(textBox1.Text);

            /*Двумерный массив для матрицы интенсивностей*/
            double[,] matrIntens = new double[n, n];

            /*Заполнение DataGridView столбцами*/
            DataGridViewColumn column;

            for(int i = 0; i < n; i++)
            {
                column = new DataGridViewTextBoxColumn();

                /*Название столбца*/
                column.HeaderCell.Value = "S" + i.ToString();

                dataGridView1.Columns.Add(column);

                dataGridView1.Columns[i].Width = 30;
            }

            /*Заполнение DataGridView строками*/
            for (int i = 0; i < n; i++)
            {
                dataGridView1.Rows.Add();

                dataGridView1.Rows[i].HeaderCell.Value = "S" + i.ToString();
            }

            Random rnd = new Random();

            for (int i = 0; i < n; i++)
            {
                for(int j = 0; j < n; j++)
                {
                    if(i == j)
                    {
                        matrIntens[i, j] = 0;
                    }
                    else
                    {
                        matrIntens[i, j] = rnd.Next(0, 4);
                    }
                }
            }
        }
    }
}
```

```

for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        dataGridView1[i, j].Value = Convert.ToString(matrIntens[i, j]);
    }
}

double[,] resMatr = new double[n + 1, n + 1];

for (int i = 0; i < n + 1; i++)
{
    column = new DataGridViewTextBoxColumn();

    dataGridView2.Columns.Add(column);

    dataGridView2.Columns[i].Width = 30;
}

for (int i = 0; i < n + 1; i++)
{
    dataGridView2.Rows.Add();
}

/*Левая часть уравнений Колмогорова: сумма произведений
интенсивностей потоков, входящих в i-е состояние, на
вероятности тех состояний, из которых эти потоки исходят*/
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        resMatr[i, i] = resMatr[i, i] - matrIntens[j, i];
    }
}

/*Правая часть уравнений Колмогорова: предельная вероятность i-го
состояния, умноженная на суммарную интенсивность всех потоков,
ведущих из данного состояния*/
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        if (i != j)
        {
            resMatr[i, j] = matrIntens[j, i];
        }
    }
}

/*Сумма предельных вероятностей состояний равна единице*/
for (int i = 0; i < n + 1; i++)
{
    resMatr[i, n] = 1;
}

for (int j = 0; j < n + 1; j++)
{
    for (int i = 0; i < n + 1; i++)
    {
        resMatr[j, i] = resMatr[j, i];
    }
}

for (int j = 0; j < n + 1; j++)

```

```

{
    for (int i = 0; i < n + 1; i++)
    {
        dataGridView2[j, i].Value = Convert.ToString(resMatr[j, i]);
    }
}

double[,] tempMatr = new double[n + 1, n];

for (int i = 0; i < n + 1; i++)
{
    for(int j = 0; j < n; j++)
    {
        tempMatr[i, j] = resMatr[j, i];
    }
}

double[,] reducedMatr = new double[n, n];

int index = 0;

for (int j = 0; j < n; j++)
{
    tempMatr[0, j] = tempMatr[0, j] - tempMatr[1, j];
}

for (int i = 0; i < n + 1; i++)
{
    if (i == 1)
    {
        continue;
    }
    else
    {
        for (int j = 0; j < n; j++)
        {
            reducedMatr[index, j] = tempMatr[i, j];
        }
        index++;
    }
}

double[] tempMas = new double[n + 1];

for (int i = 0; i < n + 1; i++)
{
    tempMas[i] = resMatr[n, i];
}

double[] reducedMas = new double[n];

index = 0;

reducedMas[0] = reducedMas[0] - reducedMas[1];

for (int i = 0; i < n + 1; i++)
{
    if (i == 1)
    {
        continue;
    }
    else
    {
        reducedMas[index] = tempMas[i];
    }
}

```

```

        index++;
    }
}

/*Метод Гаусса*/
for (int k = 0; k < n - 1; k++)
{
    for (int i = k + 1; i < n; i++)
    {
        for (int j = k + 1; j < n; j++)
        {
            reducedMatr[i, j] = reducedMatr[i, j] - reducedMatr[k, j] *
(reducedMatr[i, k] / reducedMatr[k, k]);
        }
        reducedMas[i] = reducedMas[i] - reducedMas[k] * reducedMatr[i, k] /
reducedMatr[k, k];
    }
}

double c = 0;

double[] resMas = new double[n];

for (int k = n - 1; k >= 0; k--)
{
    c = 0;

    for (int j = k + 1; j < n; j++)
    {
        c = c + reducedMatr[k, j] * resMas[j];
    }
    resMas[k] = (reducedMas[k] - c) / reducedMatr[k, k];
}

for (int i = 0; i < n; i++)
{
    column = new DataGridViewTextBoxColumn();

    column.HeaderCell.Value = "S" + i.ToString();

    dataGridView3.Columns.Add(column);

    dataGridView3.Columns[i].Width = 200;
}

for (int i = 0; i < 1; i++)
{
    dataGridView3.Rows.Add();

    dataGridView3.Rows[i].HeaderCell.Value = "T";
}

for (int j = 0; j < n; j++)
{
    for (int i = 0; i < 1; i++)
    {
        dataGridView3[j, i].Value = Convert.ToString(resMas[j]);
    }
}
}

public Form1()
{
    InitializeComponent();
}

```

```

private void button1_Click(object sender, EventArgs e)
{
    /*Очистка DataGridView*/
    int k = dataGridView1.ColumnCount;

    if (k != 0)
    {
        for (int i = 0; i < k; i++)
        {
            dataGridView1.Columns.Clear();
            dataGridView2.Columns.Clear();
            dataGridView3.Columns.Clear();
        }

        Calculate();
    }
}
}

```