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»

Тема: «Марковские случайные процессы»

Студент:	Пронин А. С.
Группа:	ИУ7-72Б
Преподаватель:	Рудаков И. В.
Оценка:	______________

Москва

2022

Задание

Разработать программу для определения времени пребывания сложной системы в каждом из состояний. Количество состояний ≤ 10. Реализовать возможность выбора количества состояний, значений интенсивностей переходов в матрице и отображение результатов работы программы (графики вероятностей состояний, значение и время стабилизации вероятности для каждого состояния) при помощи графического интерфейса.

1 Отчет

1.1 Теория

Для математического описания функционирования устройств, развивающегося в форме случайного процесса, может быть с успехом применен математический аппарат, разработанный в теории вероятности для так называемых Марковских случайных процессов.

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 ее состояния в настоящем и не зависит от того, когда и каким образом система пришла в это состояние.

Для Марковского процесса обычно составляются уравнения Колмогорова, представляющие следующие соотношения:

$F = (p'(t), p(t), \Lambda) = 0$, где λ - набор параметров.

Интегрирование системы дает искомые вероятности состояний, как функций от времени. Начальные условия берутся в зависимости от того, какого было начальное состояние системы.

1.2 Программа

Условием стабилизации вероятности i-ого состояния принимается величина $P_i(t)$, где t - наименьшее время, при котором ${P'}_i(t) < 10^{-8}$.

Также реализована возможность задавать 2 начальных условия:

- в нулевой момент времени система находится в первом состоянии;
- в нулевой момент времени система находится в каждом состоянии с равной вероятностью;

2 Текст программы

В листингах 2.1–2.2 представлена часть кода программы, отвечающего за расчет.

Листинг 2.1: Header файл

```
#ifndef MODEL_H
#define MODEL_H

#include "math.h"

#define MAX_N 10
#define EPS 1e-8

class Model
{
public:
  Model(int _n, bool same = false);

  bool step(double dt);

private:
  bool isStable();
  void Kolmogorov(double res[MAX_N]);
  void SetStableT();

public:
  double prob_mas[MAX_N];
  double time_mas[MAX_N];
  double lambda_mtrx[MAX_N][MAX_N];
  int n = 0;
  double T = 0;
};

#endif // MODEL_H
```

Листинг 2.2: Исходный код

```
#include "Model.h"

Model::Model(int _n, bool same)
{
  n = _n;
  for (int i = 0; i < MAX_N; i++)
  {
    for (int j = 0; j < MAX_N; j++)
      lambda_mtrx[i][j] = 0;
    time_mas[i] = 0;
    prob_mas[i] = 0;
  }
  if (!same)
    prob_mas[0] = 1;
  else
    for (int i = 0; i < n; i++)
      prob_mas[i] = 1.0/n;
}

bool Model::step(double dt)
{
  double newP[MAX_N];
  for (int i = 0; i < n; i++)
  {
    newP[i] = prob_mas[i];
    for (int j = 0; j < n; j++)
    {
      if (i == j) continue;
      newP[i] += dt * (prob_mas[j] * lambda_mtrx[j][i] -
   prob_mas[i] * lambda_mtrx[i][j]);
    }
  }

  bool isSt = isStable();

  for (int i = 0; i < MAX_N; i++)
    prob_mas[i] = newP[i];

  SetStableT();
  T += dt;
```

```
  return isSt;
}

bool Model::isStable()
{
  double res[MAX_N];
  Kolmogorov(res);
  for (int i = 0; i < n; i++)
    if (fabs(res[i]) > EPS/10)
      return false;
  return true;
}

void Model::Kolmogorov(double res[MAX_N])
{
  for (int i = 0; i < n; i++)
  {
    res[i] = 0;
    for (int j = 0; j < n; j++)
    {
      if (i == j) continue;
      res[i] += prob_mas[j] * lambda_mtrx[j][i] - prob_mas[i]
    * lambda_mtrx[i][j];
    }
  }
}

void Model::SetStableT()
{
  double kArr[MAX_N];
  Kolmogorov(kArr);
  for (int i = 0; i < n; i++)
  {
      if (fabs(kArr[i]) < EPS*100 && time_mas[i] <= EPS)
          time_mas[i] = T;
      else if (fabs(kArr[i]) > EPS*100 && time_mas[i] > EPS)
          time_mas[i] = 0;
  }
}
```

3 Графики

Примеры работы программы при различном начальном состоянии системы, количестве состояний и заполнении матрицы интенсивности приведены на рисунках 3.1–3.4.

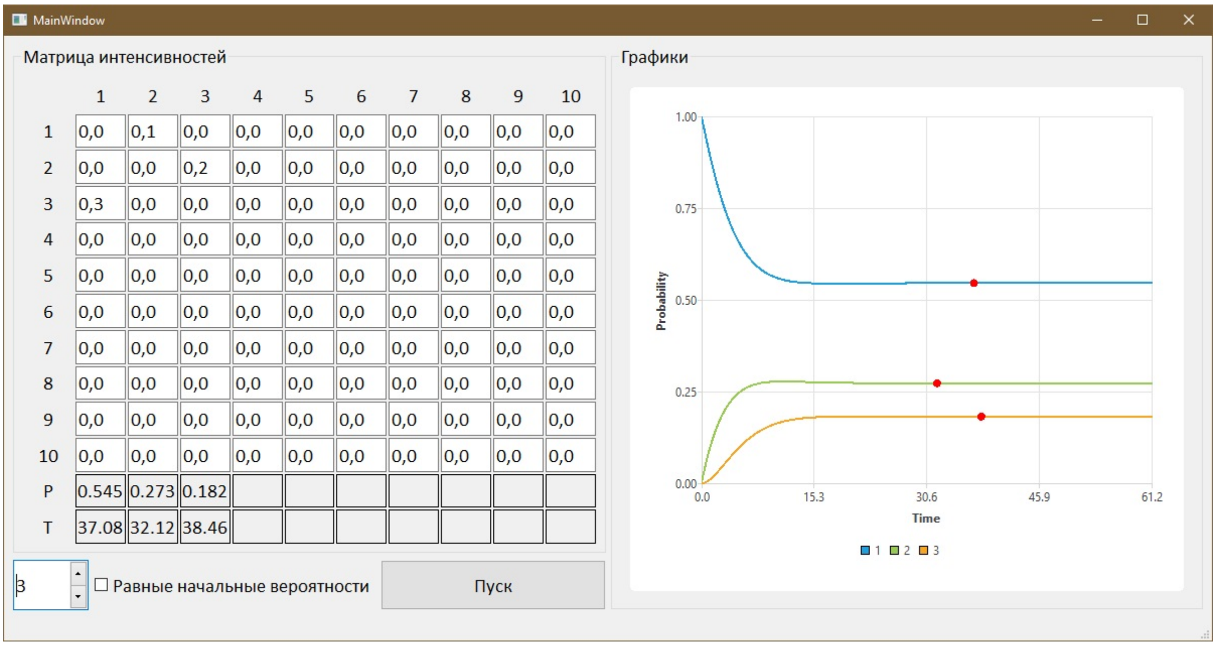

Рис. 3.1: Графики вероятностей трех состояний

Проверка результатов (рисунок 3.1):

$$\begin{cases} -0.1 \cdot 0.545 + 0.3 \cdot 0.182 = 1.0 \cdot 10^{-4} \\ -0.2 \cdot 0.273 + 0.1 \cdot 0.545 = -1.0 \cdot 10^{-4} \\ -0.3 \cdot 0.182 + 0.2 \cdot 0.273 = 0.0 \end{cases} \tag{3.1}$$

Вычислим значения вероятностей аналитически:

$$\begin{cases} -0.1 \cdot P_1 + 0.3 \cdot P_3 = 0 \\ -0.2 \cdot P_2 + 0.1 \cdot P_1 = 0 \\ -0.3 \cdot P_3 + 0.2 \cdot P_2 = 0 \\ P_1 + P_2 + P_3 = 1 \end{cases} \tag{3.2}$$

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

Значения вычисленные при решении системы уравнений 3.2 совпали с результатами программы 3.4.

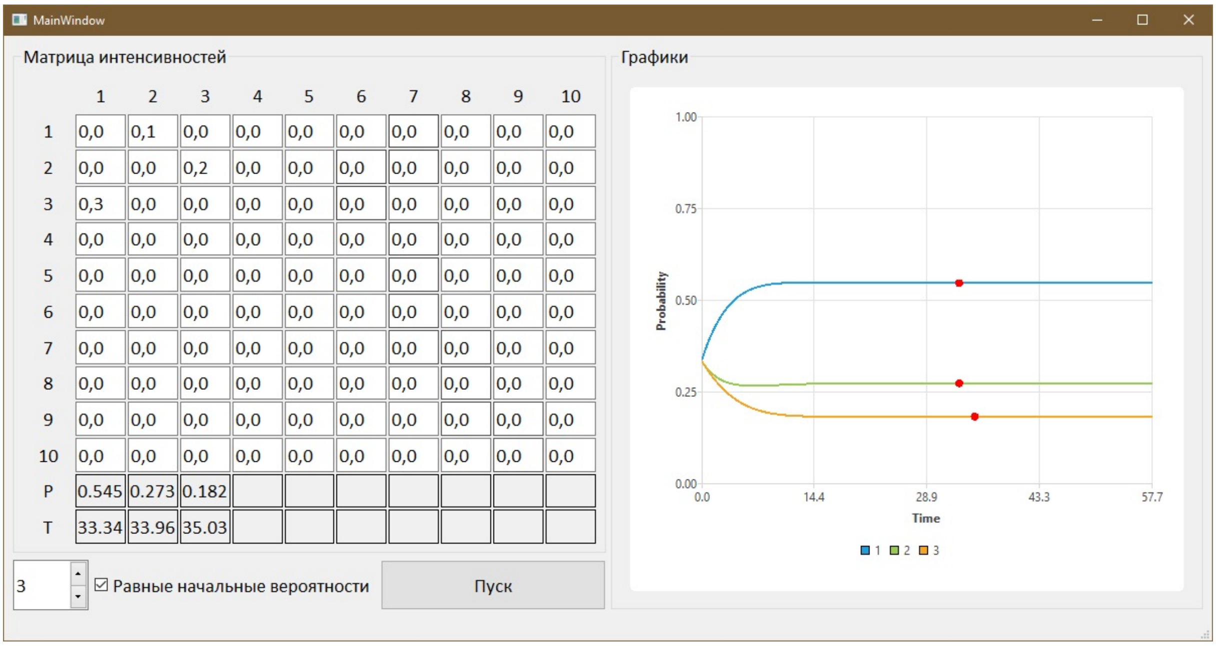

Рис. 3.2: Графики вероятностей при равных начальных вероятностях

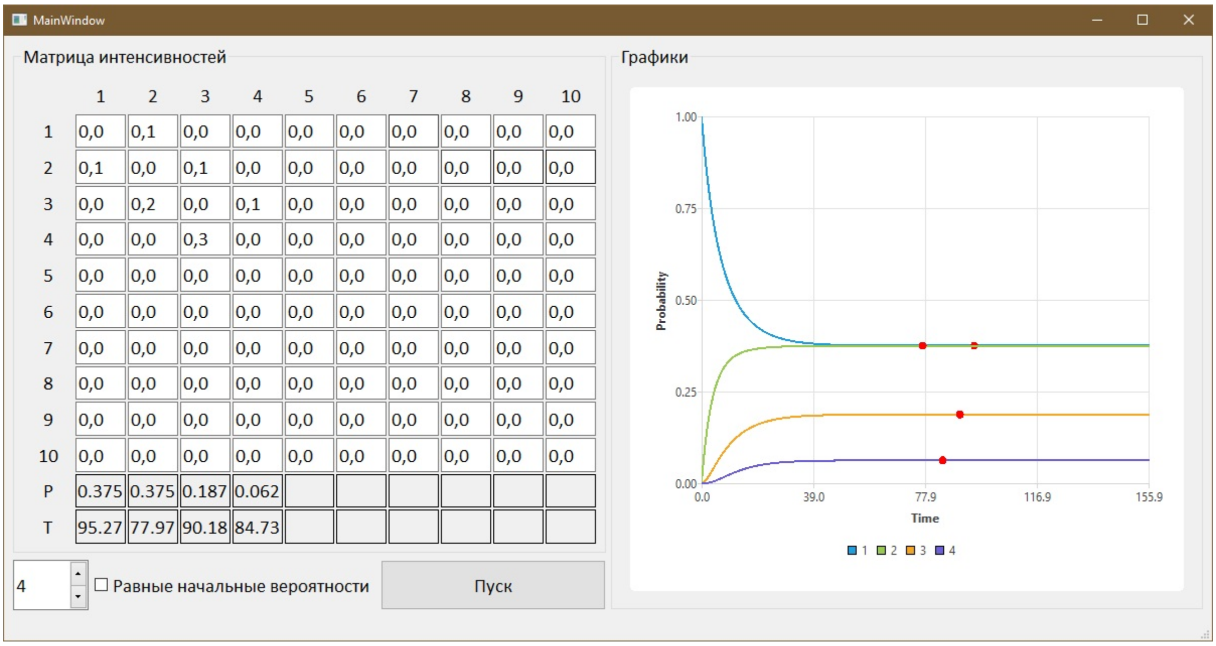

Рис. 3.3: Графики вероятностей четырех состояний

Проверка результатов (рисунок 3.3):

$$\begin{cases} -0.1 \cdot 0.375 + 0.1 \cdot 0.375 = 0.0 \\ -(0.1 + 0.1) \cdot 0.375 + (0.1 \cdot 0.375 + 0.2 \cdot 0.187) = -1.0 \cdot 10^{-4} \\ -(0.2 + 0.1) \cdot 0.187 + (0.1 \cdot 0.375 + 0.3 \cdot 0.062) = 0.0 \\ -0.3 \cdot 0.062 + 0.1 \cdot 0.187 = 1.0 \cdot 10^{-4} \end{cases} \tag{3.3}$$

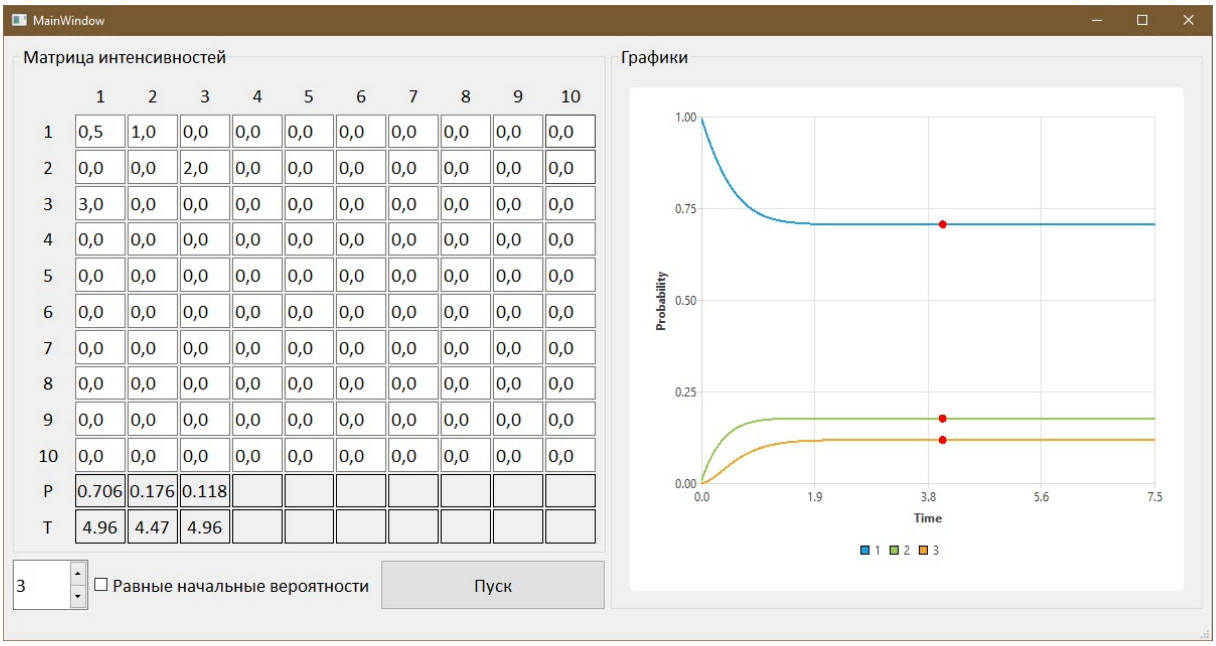

Рис. 3.4: Графики вероятностей с переходом состояния в само себя

Проверка результатов (рисунок 3.4):

$$\begin{cases} (-1.0 + 0.5) \cdot 0.706 + 3.0 \cdot 0.118 = 1.0 \cdot 10^{-3} \\ -2.0 \cdot 0.176 + (1.0 - 0.5) \cdot 0.706 = 1.0 \cdot 10^{-3} \\ -3.0 \cdot 0.118 + 2.0 \cdot 0.176 = -2.0 \cdot 10^{-3} \end{cases} \tag{3.4}$$