|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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** |

**Название:**

«Цепи Маркова»

**Дисциплина:** Моделирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-76Б |  |  | А. А. Петрова |
|  | (Группа) |  |  | (И.О. Фамилия) |
|  |  |  |  |  | |  |
| Преподаватель |  |  |  | И. В. Рудаков |
|  |  |  |  | (И.О. Фамилия) |

2022 г.

**Задание**

Реализовать программу, которая позволяет определить время пребывания системы массового обслуживания в каждом состоянии в установившемся режиме работы. Количество состояний не больше 10. Граф состояний задается матрицей.

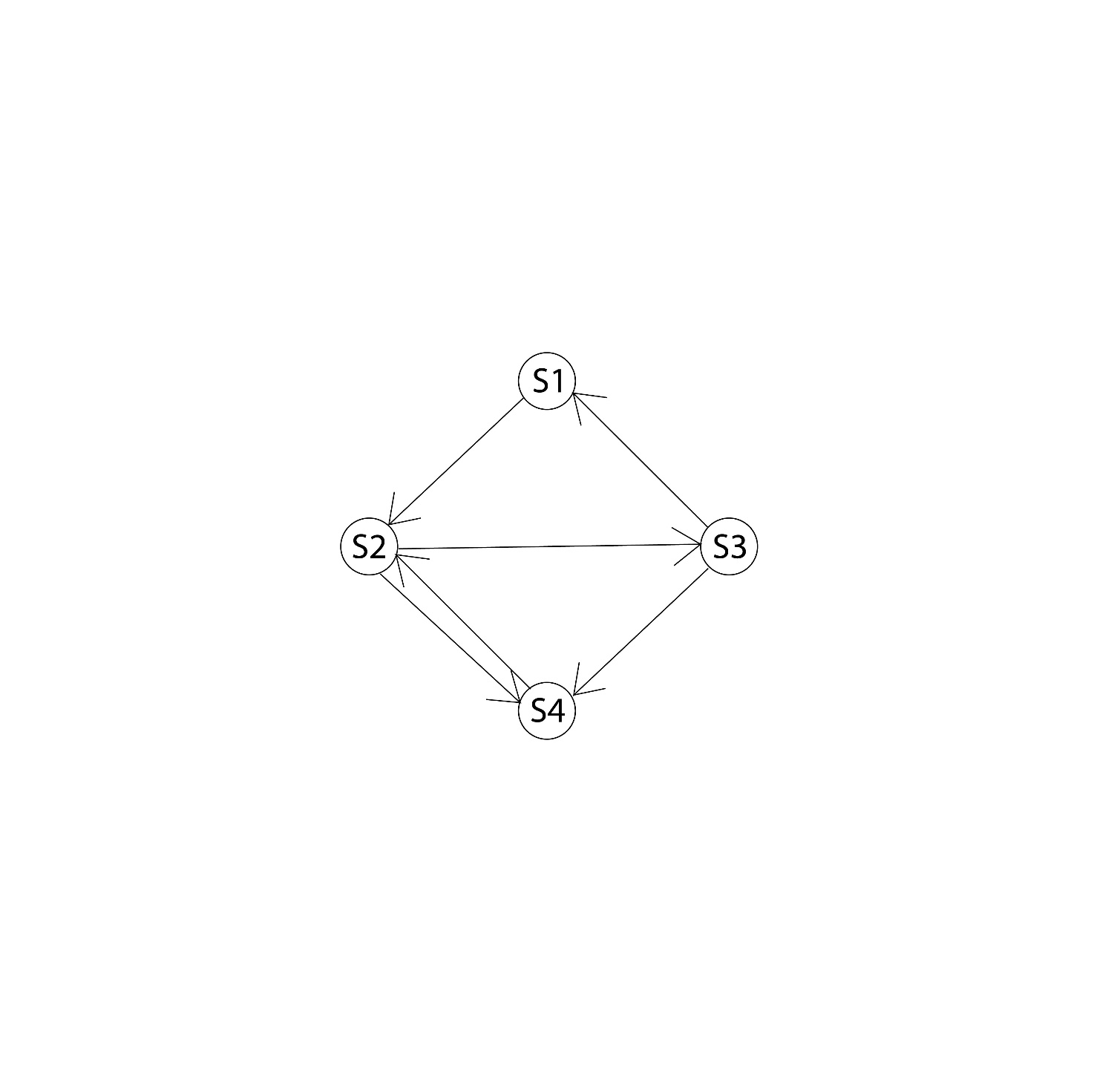
**Математическая формализация**

Для решения поставленной задачи, необходимо составить систему уравнений Колмогорова по следующим принципам:

* в левой части каждого из уравнений стоит производная вероятности этого состояния, а правая часть содержит столько членов, сколько стрелок связано с этим состоянием;
* если стрелка направлена из состояния, то соответствующий член имеет знак минус. Если в состояние - знак плюс;
* каждый член равен произведению плотности вероятности перехода (интенсивности), соответствующей данной стрелке, умноженной на вероятность того состояния, из которого исходит стрелка.

Пример:

Пусть система имеет 4 возможных состояния.



Тогда уравнения Колмогорова для неё будут иметь вид:

Для получения предельных вероятностей, то есть вероятностей в стационарном режиме работы при , необходимо приравнять левые части уравнений к нулю. Таким образом получается система линейных уравнений. Для решения полученной системы необходимо добавить условие нормировки .

После того, как предельные вероятности будут найдены, необходимо найти время. Для этого надо найти каждую вероятность в момент времени . На каждом шаге необходимо вычислять приращения для каждой вероятности (как функции):

.

Когда найденная вероятность будет равна соответствующей предыдущей с точностью до заданной погрешности, тогда можно завершить вычисления.

**Реализация**

В листинге ниже представлена реализация расчёта времени и вероятности пребывания системы в состояниях.

Листинг 1: расчёт вероятностей пребывания системы в каждом состоянии

|  |
| --- |
| matrix **=** self**.**get\_table**()**  matrix **=** numpy**.**array**(**matrix**)**  n **=** **len(**matrix**)**  coeff\_matrix **=** numpy**.**zeros**((**n**,** n**))**  **for** state **in** **range(**n **-** 1**):**  **for** col **in** **range(**n**):**  coeff\_matrix**[**state**,** state**]** **-=** **float(**matrix**[**state**,** col**])**  **for** row **in** **range(**n**):**  coeff\_matrix**[**state**,** row**]** **+=** **float(**matrix**[**row**,** state**])**  **for** state **in** **range(**n**):**  coeff\_matrix**[**n **-** 1**,** state**]** **=** 1  res **=** **[**0 **for** i **in** **range(**n**)]**  res**[**n **-** 1**]** **=** 1  augmentation\_matrix **=** numpy**.**array**(**res**)**  probs **=** numpy**.**linalg**.**solve**(**coeff\_matrix**,** augmentation\_matrix**)** |

Листинг 2: получение вероятностей и времени в стационарном режиме работы

|  |
| --- |
| **def** \_dp**(**self**,** matrix**,** probabilities**):**  res **=** **[]**  n **=** **len(**matrix**)**  **for** i **in** **range(**n**):**  summ **=** 0  **for** j **in** **range(**n**):**  **if** i **==** j**:**  sum\_i **=** 0  **for** t **in** **range(**n**):**  sum\_i **+=** **float(**matrix**[**i**][**t**])**  summ **+=** probabilities**[**j**]** **\*** **(-**1 **\*** sum\_i **+** **float(**matrix**[**i**][**i**]))**  **else:**  summ **+=** probabilities**[**j**]** **\*** **float(**matrix**[**j**][**i**])**  res**.**append**(**TIME\_DELTA **\*** summ**)**  **return** res  **def** get\_stab\_time**(**self**,** matrix**,** start\_probabilities**):**  n **=** **len(**matrix**)**  current\_time **=** 0  current\_probabilities **=** start\_probabilities**.**copy**()**  stabilization\_times **=** **[**0 **for** i **in** **range(**n**)]**  stabilization\_p **=** **[**0 **for** i **in** **range(**n**)]**  prev\_probabilities **=** **[]**  **for** i **in** **range(**n**):**  prev\_probabilities**.**append**([])**  x **=** **[]**  counter **=** 0  prev\_dp **=** self**.**\_dp**(**matrix**,** current\_probabilities**)**  **while** **not** **all(**stabilization\_times**):**  **while** counter **<** 100**:**  curr\_dp **=** self**.**\_dp**(**matrix**,** current\_probabilities**)**  **for** i **in** **range(**n**):**  prev\_probabilities**[**i**].**append**(**current\_probabilities**[**i**])**  current\_probabilities**[**i**]** **+=** curr\_dp**[**i**]**  counter **+=** 1  x**.**append**(**current\_time**)**  current\_time **+=** TIME\_DELTA  **for** i **in** **range(**n**):**  **if** **not** stabilization\_times**[**i**]** **and** **abs(**prev\_dp**[**i**]** **-** curr\_dp**[**i**])** **<** EPS **and** **abs(**curr\_dp**[**i**])** **<** EPS**:**  stabilization\_times**[**i**]** **=** current\_time **-** TIME\_DELTA **\*** 30  stabilization\_p**[**i**]** **=** current\_probabilities**[**i**]**  counter **=** 0  prev\_dp **=** curr\_dp |

**Результаты работы**

На рисунках ниже приведены интерфейс программы и результаты её работы.

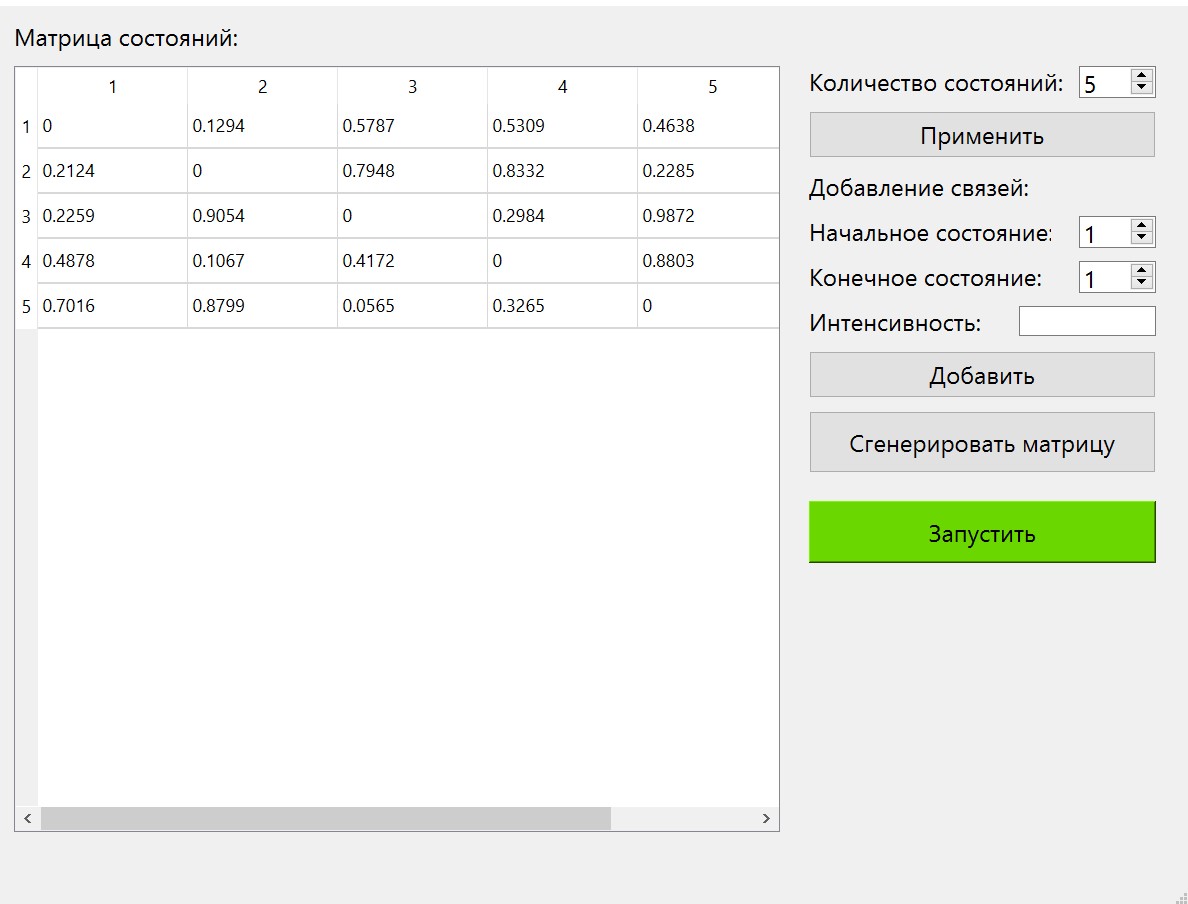
****

Рисунок 1: интерфейс программы

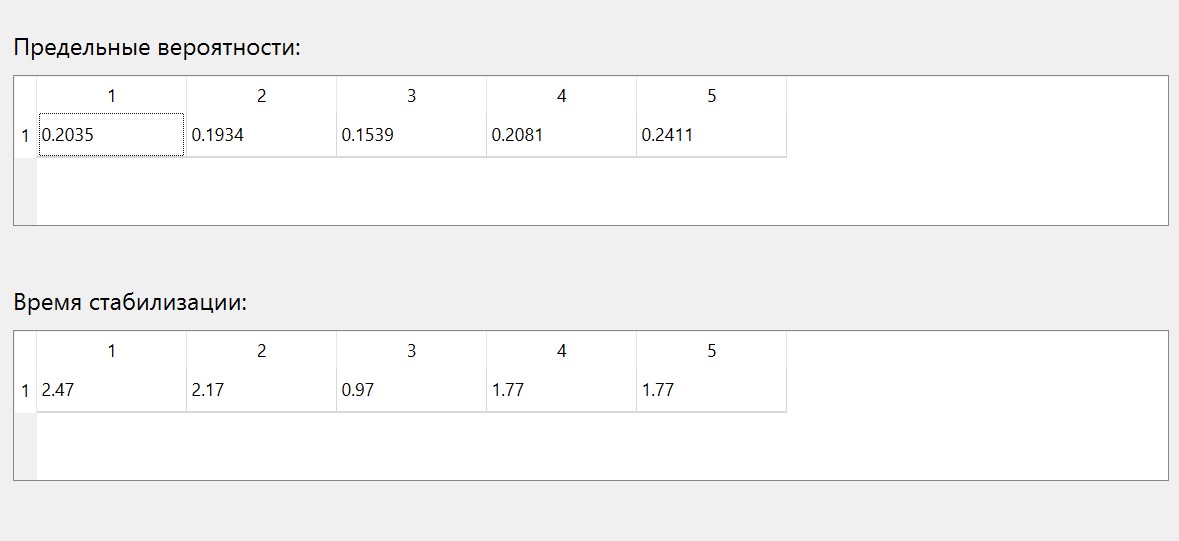


Рисунок 2: результаты работы программы на сгенерированной выше матрице

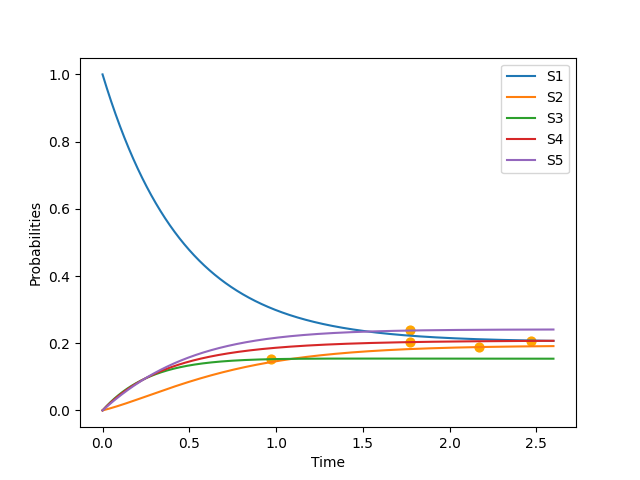


Рисунок 3: графики зависимости вероятностей пребывания системы в соответствующих состояниях от времени (точки – моменты стабилизации)