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»

О Т Ч Е Т

по лабораторной работе № 2

Название: «Цепи Маркова»

Дисциплина: Моделирование

Студент ИУ7-76Б (Группа) А. А. Петрова (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.

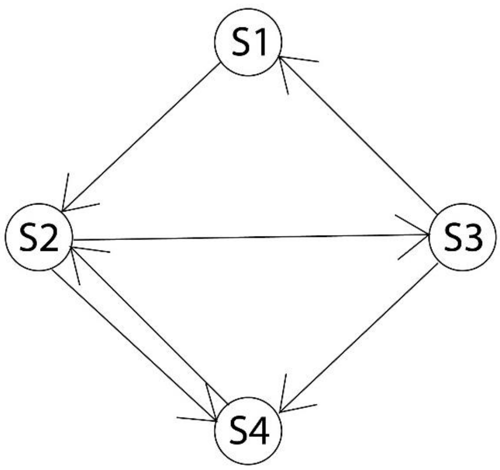
Математическая формализация

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:

$$p'_1(t) = -\lambda_{12}p_1(t) + \lambda_{31}p_3(t)$$

$$p'_2(t) = -\lambda_{24}p_2(t) - \lambda_{23}p_2(t) + \lambda_{42}p_4(t) + \lambda_{12}p_1(t)$$

$$p'_3(t) = -\lambda_{31}p_3(t) - \lambda_{34}p_3(t) + \lambda_{23}p_2(t)$$

$$p'_4(t) = -\lambda_{42}p_4(t) + \lambda_{24}p_2(t) + \lambda_{34}p_3(t)$$

Для получения предельных вероятностей, то есть вероятностей в стационарном режиме работы при $t \to \infty$, необходимо приравнять левые части уравнений к нулю. Таким образом получается система линейных уравнений. Для решения полученной системы необходимо добавить условие нормировки $\sum_{i=1}^{n} p_i = 1$.

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

$$\Delta p_i = p'_i(t) * \Delta t.$$

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.

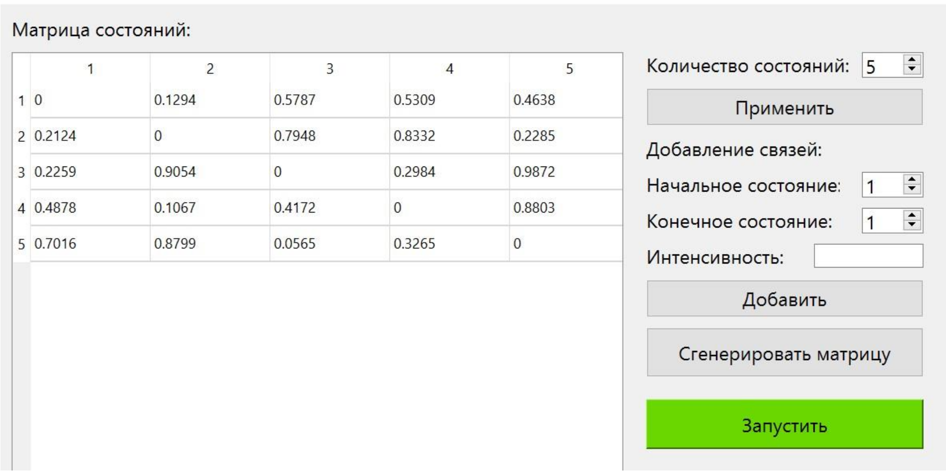


Рисунок 1: интерфейс программы

Предельные вероятности:

	1	2	3	4	5
1	0.2035	0.1934	0.1539	0.2081	0.2411

Время стабилизации:

	1	2	3	4	5
1	2.47	2.17	0.97	1.77	1.77

Рисунок 2: результаты работы программы на сгенерированной выше матрице

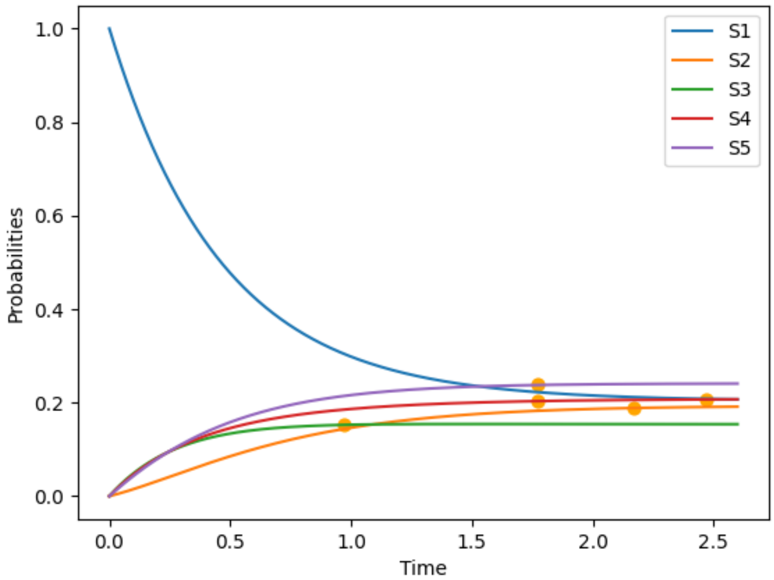


Рисунок 3: графики зависимости вероятностей пребывания системы в соответствующих состояниях от времени (точки – моменты стабилизации)