



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 ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение эвм и информационные технологии»

ОТЧЕТ

По лабораторной работе №2

«Марковские процессы»

По курсу «Моделирование»

Студент

Группа

Преподаватель

Т. А. Казаева

ИУ7-76Б

И. В. Рудаков

2022 г.

1. ЗАДАНИЕ

Для сложной системы S , имеющей не более 10 состояний, определить время нахождения системы в предельных состояниях, то есть при установившемся режиме работы.

2. МАТЕМАТИЧЕСКАЯ ФОРМАЛИЗАЦИЯ

Для математической формализации функционирования устройств процесс в котором развивается в форме случайного процесса может быть с успехом применен аппарат из теории вероятностей (теории массового обслуживания) для так называемых марковских случайных процессов.

Случайный процесс протекающий в некоторой системе S , называют марковским если он обладает следующим свойством:

для каждого момента времени вероятность любого состояния в будущем зависит только от его состояния в настоящем и не зависит от того, когда и каким образом система пришла в это состояние.

Решив уравнение Колмогорова, определить вероятность нахождения системы в этом состоянии. Уравнение Колмогорова в общем виде можно представить следующим образом (2.1):

$$F = (p'(t), P(t), \Lambda) = 0, \quad (2.1)$$

Где Λ — это коэффициенты.

Уравнение Колмогорова строится по следующим правилам:

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

Для получения предельных вероятностей, то есть вероятностей в стационарном режиме работы при $t \rightarrow \inf$, необходимо приравнять левые части уравнений к нулю. Получается система линейных уравнений. Необходимо соблюдение условия нормировки: (2.2):

$$\sum_i^n p_i = 1 \quad (2.2)$$

3. РЕЗУЛЬТАТ

На рисунке 3.1 представлен графический интерфейс приложения.

Лабораторная работа №2, Казаева Татьяна ИУ7-76Б

Размер матрицы: 3

Построить

Заполнить случайными числами Обнулить

Вычислить

Матрица

	1	2	3
1	0.0	0.4148	0.2784
2	0.8158	0.0	0.4363
3	0.981	0.899	0.0

Результат

	1	2	3
1	Состояние	Предельная вероятность	Время
2	0	0.5570865139195444	11.651999999998981
3	1	0.2925288038492662	2.625999999999822
4	2	0.1503846822311894	9.734000000000044

Рис. 3.1: Графический интерфейс приложения

Пользователь может как редактировать матрицу вручную, так и заполнить её случайными числами, нажав на соответствующую кнопку. Также пользователь может заполнить таблицу нулями, нажав на кнопку «занулить».

4. ПРОГРАММНЫЙ КОД

Для реализации программы был выбран язык Python.

Листинг 4.1: Класс для уравнений Колмогорова

```
1 class Kolmogorov:
2     def __init__(self, intensityMatrix: list[list[float]]):
3         self.N: int = len(intensityMatrix[0])
4         self.intensityMatrix: npt.NDArray =
5             np.array(intensityMatrix, dtype=float)
6         self.coefMatrix: npt.NDArray = self.InitCoefMatrix()
7
8     def InitCoefMatrix(self) -> npt.NDArray:
9         res: npt.NDArray = np.zeros((self.N, self.N), dtype=float)
10        for i in range(self.N):
11            for j in range(self.N):
12                res[i, i] -= self.intensityMatrix[i, j]
13                res[i, j] += self.intensityMatrix[j, i]
14        return res
15
16    def InitAugmMatrix(self) -> npt.NDArray:
17        result: npt.NDArray = np.zeros(self.N, dtype=float)
18        result[-1] = 1
19        return result
20
21    def computeStaionaryProbabilites(self) -> npt.NDArray:
22        base: npt.NDArray = self.coefMatrix.copy()
23        base[-1] = np.ones(self.N)
24        augm: npt.NDArray = self.InitAugmMatrix()
25        return np.linalg.solve(base, augm)
26
27    def getProbIncrement(self, start_probabilities: npt.NDArray)
28        -> npt.NDArray:
29        result: npt.NDArray = np.zeros(self.N)
30        for i, curr_prob in enumerate(start_probabilities):
31            for j in range(self.N):
32                if i == j:
33                    result[i] += (-sum(self.intensityMatrix[i]) +
```

```

32         self.intensityMatrix[i][j]) * curr_prob
33     else:
34         result[i] += self.intensityMatrix[j][i] *
35             start_probabilities[j]
36     return result * TIME_DELTA
37
38 def getLimitTimes(self, start_probabilities: npt.NDArray) ->
39     npt.NDArray:
40     limit_probabilities: npt.NDArray =
41         self.computeStaionaryProbabilites()
42     curProbs: npt.NDArray = start_probabilities.copy()
43     times: npt.NDArray = np.zeros(self.N)
44     cur = 0
45
46     while not all(times):
47         delta_p = self.getProbIncrement(curProbs)
48         for i in range(self.N):
49             if not times[i] and abs(curProbs[i] -
50                 limit_probabilities[i]) <= EPS:
51                 times[i] = cur
52                 curProbs[i] += delta_p[i]
53                 cur += TIME_DELTA
54
55     return times

```

4.1 ПРОВЕРКА УСЛОВИЯ НОРМИРОВКИ

Расчеты проводились для данных, представленных на рисунке 3.1:

$$\begin{bmatrix} 0 & 0.4148 & 0.2784 \\ 0.8158 & 0.0 & 0.4363 \\ 0.981 & 0.899 & 0.0 \end{bmatrix} \quad (4.1)$$

Выполним проверку условия нормировки для полученных вероятностей(4.2):

$$\sum_{i=1}^3 p_i = 0.55708651 + 0.2925288 + 0.15038468 = 1 \quad (4.2)$$

Условие нормировки выполняется.