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	Фам Минь Хиеу
Группа	ИУ7-72Б
Преподаватель	И. В. Рудаков

2024 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 обсуживания) 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):

$$F = (p'(t), P(t), \Lambda) = 0, \tag{2.1}$$

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 $t \to \inf$, необходимо приравнять левые части уравнений к нулю. Получается система линейных уравнений. Необходимо соблюдение условия нормировки: (2.2):

$$\sum_{i}^{n} p_i = 1 \tag{2.2}$$

3. Результат

На рисунке 3.1 представлен графический интерфейс приложения.

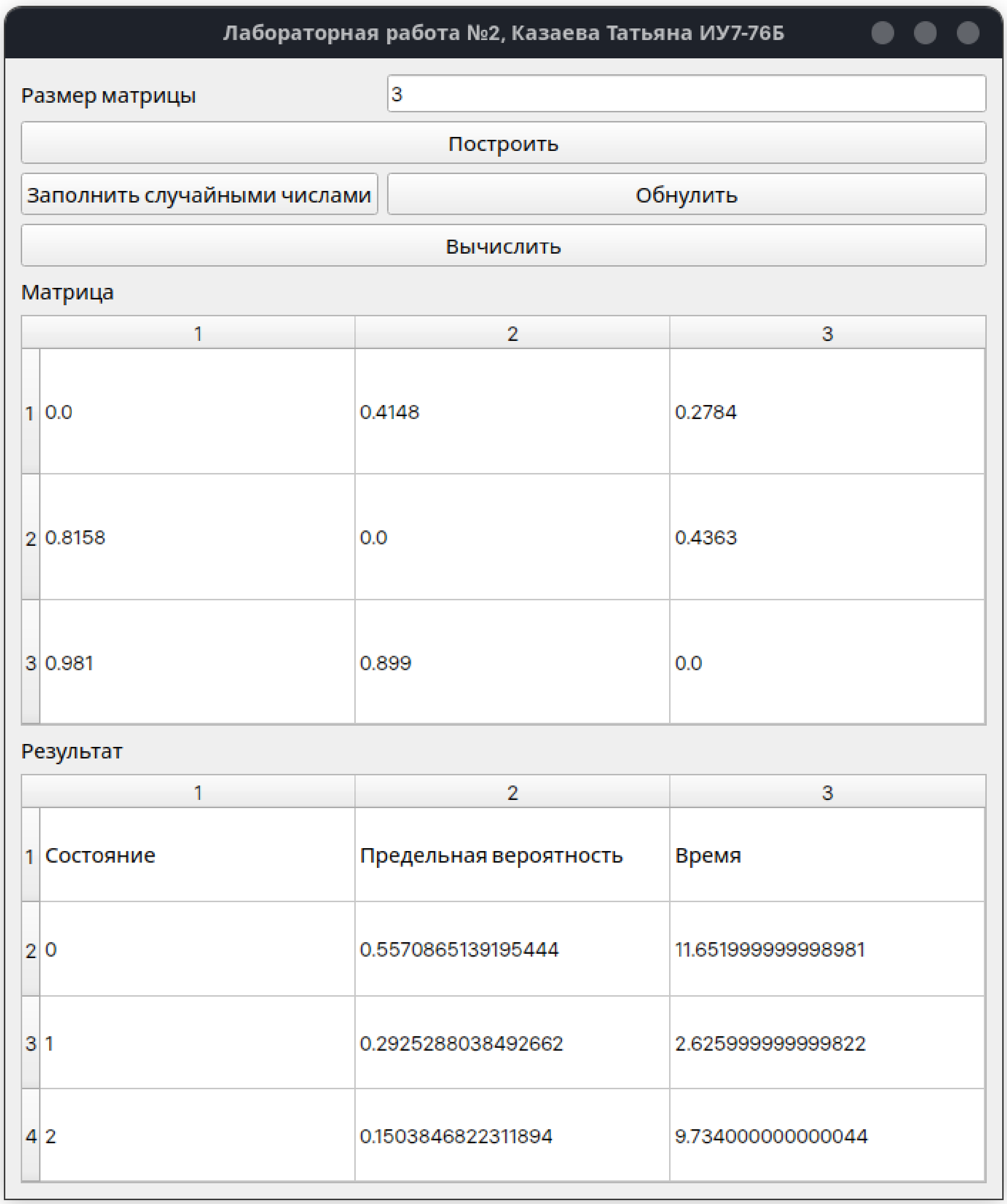

Рис. 3.1: Графический интерфейс приложения

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

```python
class Kolmogorov:
    def __init__(self, intensityMatrix: list[list[float]]):
        self.N: int = len(intensityMatrix[0])
        self.intensityMatrix: npt.NDArray = np.array(intensityMatrix, dtype=float)
        self.coefMatrix: npt.NDArray = self.InitCoefMatrix()

    def InitCoefMatrix(self) -> npt.NDArray:
        res: npt.NDArray = np.zeros((self.N, self.N), dtype=float)
        for i in range(self.N):
            for j in range(self.N):
                res[i, i] -= self.intensityMatrix[i, j]
                res[i, j] += self.intensityMatrix[j, i]
        return res

    def InitAugmMatrix(self) -> npt.NDArray:
        result: npt.NDArray = np.zeros(self.N, dtype=float)
        result[-1] = 1
        return result

    def computeStaionaryProbabilites(self) -> npt.NDArray:
        base: npt.NDArray = self.coefMatrix.copy()
        base[-1] = np.ones(self.N)
        augm: npt.NDArray = self.InitAugmMatrix()
        return np.linalg.solve(base, augm)

    def getProbIncrement(self, start_probabilities: npt.NDArray) -> npt.NDArray:
        result: npt.NDArray = np.zeros(self.N)
        for i, curr_prob in enumerate(start_probabilities):
            for j in range(self.N):
                if i == j:
                    result[i] += (-sum(self.intensityMatrix[i]) +
```

```
                        self.intensityMatrix[i][j]) * curr_prob
                else:
                    result[i] += self.intensityMatrix[j][i] * start_probabilities[j]
        return result * TIME_DELTA

    def getLimitTimes(self, start_probabilities: npt.NDArray) -> npt.NDArray:
        limit_probabilities: npt.NDArray = self.computeStaionaryProbabilites()
        curProbs: npt.NDArray = start_probabilities.copy()
        times: npt.NDArray = np.zeros(self.N)
        cur = 0

        while not all(times):
            delta_p = self.getProbIncrement(curProbs)
            for i in range(self.N):
                if not times[i] and abs(curProbs[i] - limit_probabilities[i]) <= EPS:
                    times[i] = cur
                curProbs[i] += delta_p[i]
                cur += TIME_DELTA

        return times
```

4.1 Проверка условия нормировки

Расчеты проводились для данных, представленных на рисунке 3.1:

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

Выполним проверку условия нормировки для полученных вероятностей(4.2):

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

Условие нормировки выполняется.