Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»
Мегафакультет компьютерных технологий и управления
Факультет программной инженерии и компьютерной техники

Лабораторная работа № 3
по дисциплине «Теория систем»

Выполнил: студент
Чайкин Вадим Константинович
группа P3324
Принял: преподаватель
Русак Алёна Викторовна

г. Санкт-Петербург
2025

Задание лабораторной работы

Цель работы

Получить знания о принципах функционирования и назначении цепей Маркова.

Этапы выполнения работы

Для выполнения лабораторной работы необходимо выполнить следующие пункты:

1. Придумать эргодическую марковскую цепь, содержащую не менее 4-х состояний.
2. Нарисовать диаграмму переходов и записать матрицу переходных вероятностей данной цепи.
3. Промоделировать марковскую цепь пошагово с несколькими различными начальными векторами вероятностей состояний и получить конечные вектора, к которым привело моделирование.
4. Построить графики изменения компонентов финальных векторов, а также графики изменения среднеквадратического отклонения на каждом шаге моделирования для всех начальных векторов.
5. Найти стационарное распределение аналитически (см. пример в презентации).
6. Сравнить вектора из пункта 3 и вектор, рассчитанный аналитически, между собой.

Выполнение лабораторной работы

1. Построение цепи

Составим эргодическую цепь Маркова. Эргодическая цепь должна быть:

- **неразложимой**. Из одного состояния должна быть возможность перейти в любое другое состояние.
- **нециклической**.

Пользовательский вариант: Депрессивный студент Арсентий может в любой час находиться в одном из 4 состояний:

- сон (Sleep, S),
- учёба (sTudy, T),
- прокрастинация (Procrastinate, P),
- приём пищи (Eat, E).

По наблюдениям было установлено: если Арсентий спит, с вероятностью 60% в следующий час он продолжит спать. Ему трудно заставить себя проснуться, поэтому с вероятностью 10% он будет учиться, с вероятностью 10% — есть, с вероятностью 10% — прокрастинировать.

From \To	S	T	E	P
S	0.6	0.1	0.1	0.2
T	0.2	0.6	0.1	0.1
E	0.3	0.1	0.2	0.4
P	0.1	0.2	0.2	0.5

Таблица 1: Таблица вероятностей переходов между состояниями

2. Диаграмма и матрица переходов

Вероятности перехода из одного состояния можно задать матрицей переходов:

$$P = \begin{pmatrix} 0.6 & 0.1 & 0.1 & 0.2 \\ 0.2 & 0.6 & 0.1 & 0.1 \\ 0.3 & 0.1 & 0.2 & 0.4 \\ 0.1 & 0.2 & 0.2 & 0.5 \end{pmatrix}$$

Также можно визуализировать модель марковского процесса с помощью диаграммы (графа):

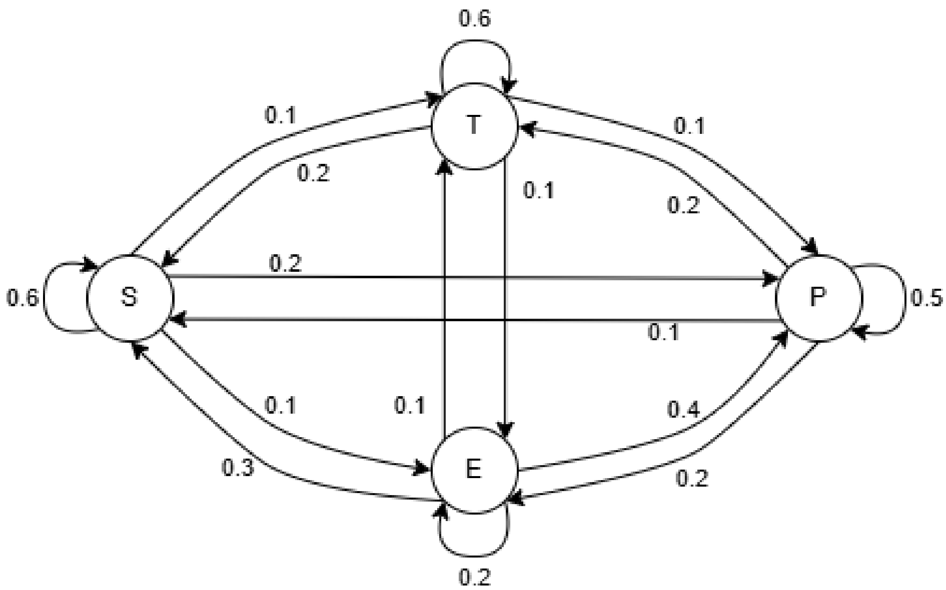

Рис. 1: Диаграмма переходов

3. Моделирование с разными начальными векторами

Выберем несколько начальных векторов вероятностей состояний:

1. В начале дня Арсентий спит:
$$\pi^{(0)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

2. В начале дня Арсентий учится:
$$\pi^{(0)} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

3. В начале дня Арсентий может с равной вероятностью находится в любом состоянии:
$$\pi^{(0)} = \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix}$$

Опишем эти шаги на Python:

```
import numpy as np

prob_matrix = np.array([
    [0.6, 0.1, 0.1, 0.2],
    [0.2, 0.6, 0.1, 0.1],
    [0.3, 0.1, 0.2, 0.4],
    [0.1, 0.2, 0.2, 0.5]
])

state_names = ['Sleep', 'sTudy', 'Eat', 'Procrastinate']

vec_1_start = np.array([1, 0, 0, 0])  # Arsentiy is sleeping
vec_2_start = np.array([0, 1, 0, 0])  # Arsentiy is studying
vec_3_start = np.array([.25, .25, .25, .25])  # Arsentiy can be in any state with equal probability

vec_1 = vec_1_start
vec_2 = vec_2_start
vec_3 = vec_3_start
```

Для каждого из начальных векторов рассчитаем конечный вектор. Попробуем рассчитать стационарные вероятности, используя такое количество шагов, по прошествии которого максимальное изменение не будет ниже ε.
Т. е.
$$\pi^{(i)} = P \cdot \pi^{(i-1)},$$

с условием остановки:

$$\max_j |\pi_j^{(i)} - \pi_j^{(i-1)}| < \varepsilon,$$

где:

- ε — заранее выбранная малая величина (например, $\varepsilon = 10^{-10}$);
- $\pi_j^{(i)}$ — j-ая компонента вектора вероятностей на i-м шаге моделирования.

Данное условие гарантирует, что моделирование прекращается, как только изменения компонент вектора становятся пренебрежимо малыми и цепь приблизительно достигает своего стационарного состояния.

```python
epsilon = 1e-10

def compute_steady_state(
        vec: np.ndarray,
        matrix: np.ndarray,
        print_iterations: bool = False,
        precision: float = epsilon,
        max_iterations: int = None,
        vector_list_adder: Callable[[np.ndarray], None] = None,
        stddev_list_adder: Callable[[float], None] = None,
):
    current_vec = vec
    iteration = 0

    while True:
        next_vec = current_vec @ matrix
        diff = np.abs(next_vec - current_vec)
        max_diff = np.max(diff)

        # Calculating standard deviation of difference
        stddev = np.sqrt(np.mean(diff**2))

        # Optional data capturing
        if vector_list_adder is not None:
            vector_list_adder(next_vec)

        if stddev_list_adder is not None:
            stddev_list_adder(stddev)

        if print_iterations:
            print(f"Iteration {iteration}: vector={next_vec}, max_diff={max_diff}, stddev={stddev}")

        # Stopping condition based on precision
        if max_diff < precision:
            break

        # Optional stopping condition based on max iterations
        iteration += 1
        if max_iterations is not None and iteration >= max_iterations:
            print("Reached maximum iterations.")
            break

        current_vec = next_vec

    return next_vec
```

```

vec_1_records = [vec_1_start,]
vec_2_records = [vec_2_start,]
vec_3_records = [vec_3_start,]
vec_1_stddev = []
vec_2_stddev = []
vec_3_stddev = []

vec_1_steady = compute_steady_state(
    vec_1,
    prob_matrix,
    vector_list_adder=vec_1_records.append,
    stddev_list_adder=vec_1_stddev.append,
)

vec_2_steady = compute_steady_state(
    vec_2,
    prob_matrix,
    vector_list_adder=vec_2_records.append,
    stddev_list_adder=vec_2_stddev.append,
)

vec_3_steady = compute_steady_state(
    vec_3,
    prob_matrix,
    vector_list_adder=vec_3_records.append,
    stddev_list_adder=vec_3_stddev.append,
)
```

Проверим значения конечных векторов $\pi = \lim\limits_{k\to\infty} \pi^{(k)}$:

```
print(vec_1_steady)  # [0.3089172  0.25796178 0.1433121  0.28980892]
print(vec_2_steady)  # [0.3089172  0.25796178 0.1433121  0.28980892]
print(vec_3_steady)  # [0.3089172  0.25796178 0.1433121  0.28980892]
```

Видим, что они почти равны. Это, в принципе, верно, так как при стремлении числа шагов к ∞ система проводит в каждом состоянии некоторый процент времени, который не зависит от начального состояния.
Проверим также, что $\sum\limits_j \pi_j = 1$:

```
print(np.sum(vec_1_steady))  # 1.0000000000000007
print(np.sum(vec_2_steady))  # 1.0000000000000004
print(np.sum(vec_3_steady))  # 1.0000000000000004
```

Все суммы практически равны 1, небольшое отклонение обусловлено численной погрешностью вычислений с плавающей точкой.

4. Графики изменения компонентов векторов и СКО

Построим графики изменения компонентов финальных векторов и среднеквадратичного отклонения с увеличением числа шагов.

```
import matplotlib.pyplot as plt

vec_1_array = np.array(vec_1_records)
vec_2_array = np.array(vec_2_records)
vec_3_array = np.array(vec_3_records)

state_names = ['Sleep', 'sTudy', 'Eat', 'Procrastinate']

fig, axs = plt.subplots(3, 2, figsize=(14, 12))
fig.suptitle('Vector components and STD change', fontsize=16)

for i, name in enumerate(state_names):
    axs[0, 0].plot(vec_1_array[:, i], label=name)
axs[0, 0].set_title('Vector 1 components')
axs[0, 0].legend()
axs[0, 0].grid(True)

axs[0, 1].plot(vec_1_stddev, label="stddev", color="black")
axs[0, 1].set_title('Vector 1 standard deviation')
axs[0, 1].grid(True)

for i, name in enumerate(state_names):
    axs[1, 0].plot(vec_2_array[:, i], label=name)
axs[1, 0].set_title('Vector 2 components')
axs[1, 0].legend()
axs[1, 0].grid(True)

axs[1, 1].plot(vec_2_stddev, label="stddev", color="black")
axs[1, 1].set_title('Vector 2 standard deviation')
axs[1, 1].grid(True)

for i, name in enumerate(state_names):
    axs[2, 0].plot(vec_3_array[:, i], label=name)
axs[2, 0].set_title('Vector 3 components')
axs[2, 0].legend()
axs[2, 0].grid(True)

axs[2, 1].plot(vec_3_stddev, label="stddev", color="black")
axs[2, 1].set_title('Vector 3 standard deviation')
axs[2, 1].grid(True)

plt.tight_layout()
plt.show()
```

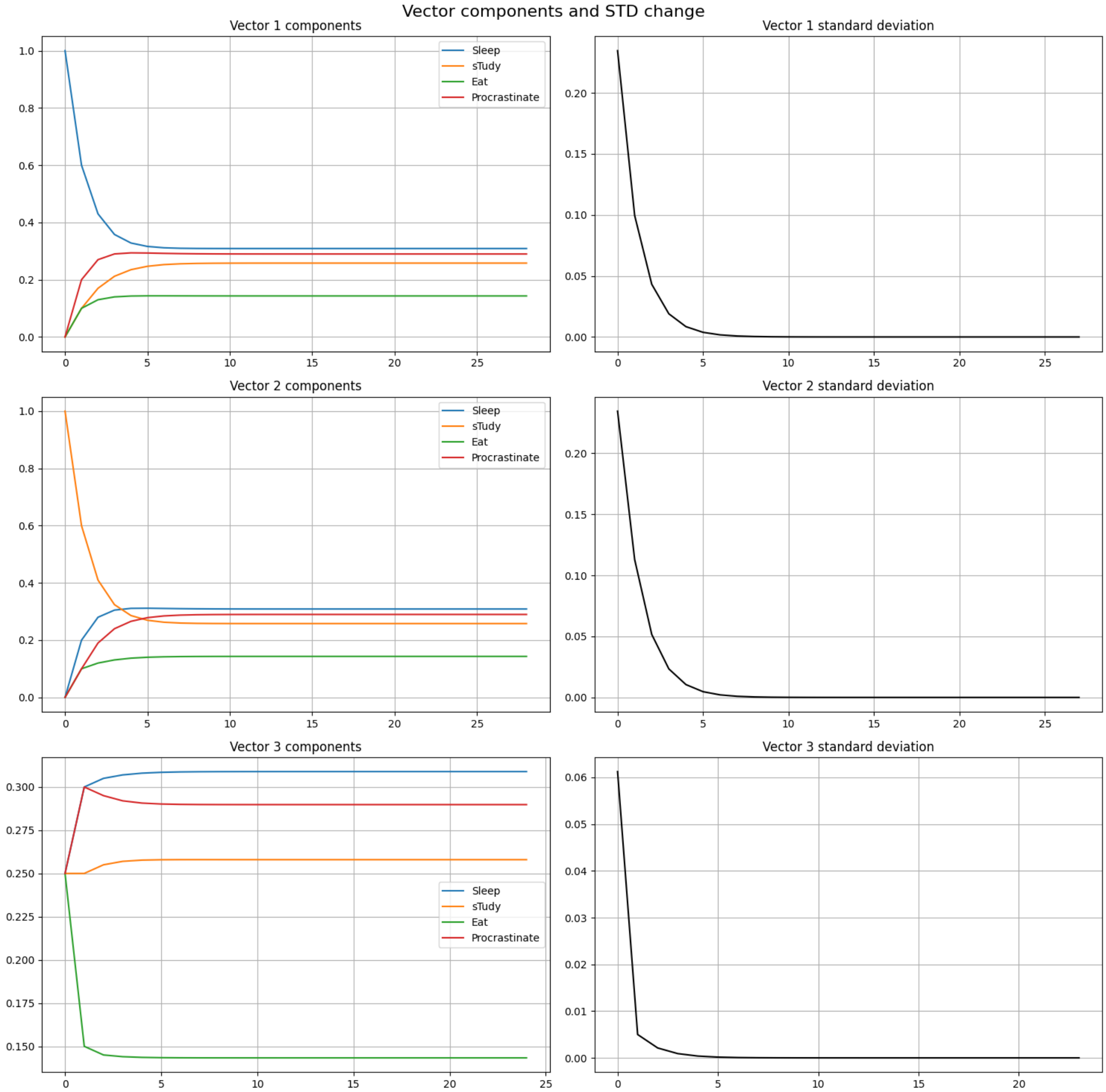

Рис. 2: Графики изменения компонентов и СКО

5. Аналитическое нахождение стационарного распределения

Найдём такое π, что $\pi P = \pi$, т. е. при переходе в следующее состояние не меняется и также удовлетворяется условие $\sum\limits_j \pi_j = 1$.

Для этого составим систему линейных алгебраических уравнений:

$$\begin{cases} P_1 = 0.6P_1 + 0.2P_2 + 0.3P_3 + 0.1P_4 \\ P_2 = 0.1P_1 + 0.6P_2 + 0.1P_3 + 0.2P_4 \\ P_3 = 0.1P_1 + 0.1P_2 + 0.2P_3 + 0.2P_4 \\ P_4 = 0.2P_1 + 0.1P_2 + 0.4P_3 + 0.5P_4 \\ P_1 + P_2 + P_3 + P_4 = 1 \end{cases}$$

Решим систему при помощи метода наименьших квадратов:

- $\pi P = \pi I$, где I — единичная матрица.
- $\pi(P - I) = 0$.
- $(P - I)^T \pi^T = 0$.
- Добавляем строку из единиц, чтобы задать ограничение $\sum\limits_j \pi_j = 1$.

```python
prob_t = prob_matrix.T
n = prob_t.shape[0]

a = prob_t - np.eye(n)
a = np.vstack([a, np.ones(n)])
b = np.zeros(n + 1)
b[-1] = 1

pi_analytical = np.linalg.lstsq(a, b, rcond=None)[0]
pi_analytical  # array([0.3089172 , 0.25796178, 0.1433121 ,
    0.28980892])
```

6. Сравнение стационарных векторов

Как мы можем видеть, разница между решениями, найденными пошагово и аналитически, пренебрежимо мала:

```
np.max(np.abs(pi_analytical - vec_1_steady))  # np.float64(7.455058792515956e-11)
```

Она может быть обусловлена тем, что в пошаговом решении точность конечная, а также погрешностью вычислений с плавающей точкой.
Аналитически найденный стационарный вектор:

- Сон (Sleep): 0.3089172
- Учёба (sTudy): 0.25796178
- Приём пищи (Eat): 0.1433121
- Прокрастинация (Procrastinate): 0.28980892

Он почти полностью совпадает с результатами, полученными с помощью моделирования, что подтверждает корректность обоих подходов.

Выводы

В ходе лабораторной работы была построена и исследована эргодическая марковская цепь с четырьмя состояниями, моделирующая поведение студента Арсентия в течение дня. Были заданы вероятности переходов между состояниями, построена диаграмма переходов и матрица переходных вероятностей.

Моделирование динамики цепи было выполнено пошагово для трёх различных начальных распределений. Полученные результаты показали, что независимо от начального состояния, система сходится к одному и тому же стационарному распределению, что подтверждает эргодичность цепи.

Построенные графики наглядно продемонстрировали, как изменяются вероятности состояний на каждом шаге моделирования и как быстро снижается среднеквадратическое отклонение, указывая на достижение устойчивого состояния.

Также было выполнено аналитическое вычисление стационарного распределения с помощью решения системы линейных уравнений. Полученное аналитическое решение совпало с результатами моделирования, что подтверждает корректность всех проведённых вычислений.

Таким образом, цель работы была достигнута: на практическом примере были усвоены принципы построения, анализа и применения марковских цепей.

Исходный код блокнота доступен по ссылке.

Список использованных источников

1. Google Classroom