Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Теория Систем

Лабораторная работа №3

Моделирование цепей Маркова

Выполнил:

Маликов Глеб Игоревич

Группа № P3324

Преподаватель:

Русак Алена Викторовна

Санкт-Петербург

2025

**Содержание**

[Задание 3](#_Toc193898099)

[Реализация 4](#_Toc193898100)

[Код 10](#_Toc193898101)

[Вывод 12](#_Toc193898102)

# Задание

Для выполнение 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.

**Замечания:**

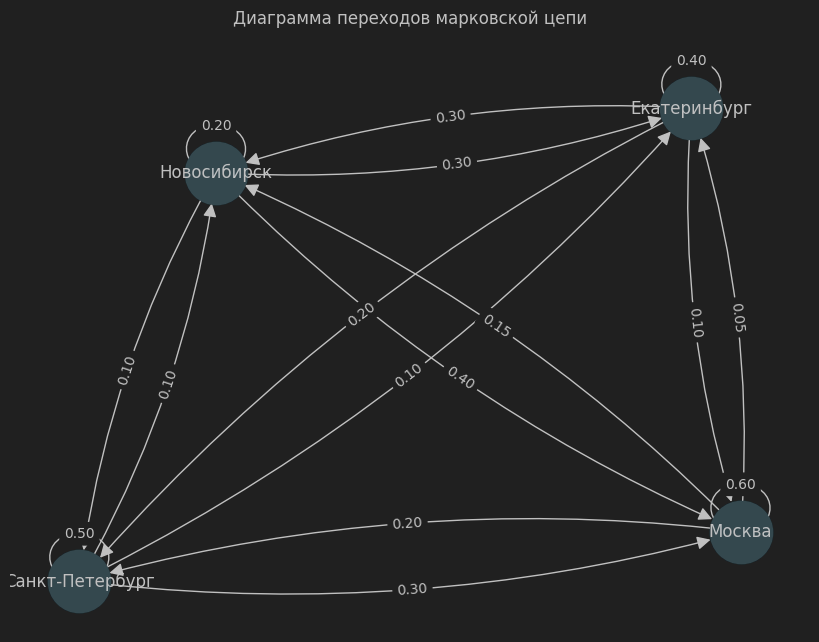
1. При пошаговом моделировании необязательно хранить все значения стационарного вектора, можно иметь два вектора вероятностей – старый и новый, главное, после применения формулы переписать значение из нового в старое и далее можно совершить еще один шаг моделирования (вычислить новый вектор). Вычисления продолжаются до тех пор, пока среднеквадратичное отклонение между старым и новым вектором не будет меньше заданного значения ошибки e.
2. Для нахождения аналитического решения вектора стационарного распределения можно использовать функцию numpy.linalg.solve.

# Реализация

Смоделированная цепь Маркова симулирует вероятности поездки пассажира между городами России.

A screen shot of a computer

AI-generated content may be incorrect.



Для получения конечных векторов была выполнена симуляция, которая останавливается при среднеквадратическом отклонении меньше 1e-6:

Начало в Москва: достигнуто за 16 шагов, финальный вектор: [0.40644418 0.26146235 0.16976435 0.16232911]

Начало в Санкт-Петербург: достигнуто за 15 шагов, финальный вектор: [0.40644428 0.26146241 0.1697643 0.16232901]

Начало в Новосибирск: достигнуто за 16 шагов, финальный вектор: [0.40644306 0.26146205 0.16976477 0.16233012]

Начало в Екатеринбург: достигнуто за 17 шагов, финальный вектор: [0.40644304 0.26146205 0.16976478 0.16233013]

A graph with colorful lines and numbers

AI-generated content may be incorrect.A graph with red lines

AI-generated content may be incorrect.A graph with different colored lines

AI-generated content may be incorrect.A graph with red lines

AI-generated content may be incorrect.

A graph with colorful lines and numbers

AI-generated content may be incorrect.A graph with a line graph

AI-generated content may be incorrect.

A graph with colorful lines and numbers

AI-generated content may be incorrect.A graph with a red line

AI-generated content may be incorrect.

Аналитическое распределение дало схожие значения с результатами симуляции.

Аналитическое стационарное распределение:

Москва: 0.4064

Санкт-Петербург: 0.2615

Новосибирск: 0.1698

Екатеринбург: 0.1623

Сравнение финальных векторов моделирования с аналитическим стационарным распределением:

Начало в Москва:

Москва: моделирование = 0.4064, аналитика = 0.4064, разница = 5.6570e-07

Санкт-Петербург: моделирование = 0.2615, аналитика = 0.2615, разница = 1.4715e-07

Новосибирск: моделирование = 0.1698, аналитика = 0.1698, разница = 2.1125e-07

Екатеринбург: моделирование = 0.1623, аналитика = 0.1623, разница = 5.0161e-07

Начало в Санкт-Петербург:

Москва: моделирование = 0.4064, аналитика = 0.4064, разница = 6.6100e-07

Санкт-Петербург: моделирование = 0.2615, аналитика = 0.2615, разница = 2.0827e-07

Новосибирск: моделирование = 0.1698, аналитика = 0.1698, разница = 2.6160e-07

Екатеринбург: моделирование = 0.1623, аналитика = 0.1623, разница = 6.0767e-07

Начало в Новосибирск:

Москва: моделирование = 0.4064, аналитика = 0.4064, разница = 5.6061e-07

Санкт-Петербург: моделирование = 0.2615, аналитика = 0.2615, разница = 1.5545e-07

Новосибирск: моделирование = 0.1698, аналитика = 0.1698, разница = 2.1326e-07

Екатеринбург: моделирование = 0.1623, аналитика = 0.1623, разница = 5.0280e-07

Начало в Екатеринбург:

Москва: моделирование = 0.4064, аналитика = 0.4064, разница = 5.7413e-07

Санкт-Петербург: моделирование = 0.2615, аналитика = 0.2615, разница = 1.5365e-07

Новосибирск: моделирование = 0.1698, аналитика = 0.1698, разница = 2.1615e-07

Екатеринбург: моделирование = 0.1623, аналитика = 0.1623, разница = 5.1164e-07

Ниже показан результат симуляции цепи Маркова на несколько дней и с разными начальными состояниями.

Начатьное состояние: Москва

Последовательность посещённых городов: ['Москва', 'Екатеринбург', 'Новосибирск', 'Новосибирск', 'Екатеринбург', 'Новосибирск']

Конечное состояние через 5 дней: Новосибирск

Вероятность выбранной последовательности: 0.00027

Начатьное состояние: Санкт-Петербург

Последовательность посещённых городов: ['Санкт-Петербург', 'Санкт-Петербург', 'Екатеринбург', 'Екатеринбург', 'Москва', 'Москва', 'Екатеринбург', 'Екатеринбург', 'Екатеринбург', 'Екатеринбург', 'Екатеринбург']

Конечное состояние через 10 дней: Екатеринбург

Вероятность выбранной последовательности: 1.536000000000001e-06

## Код

states = ["Москва", "Санкт-Петербург", "Новосибирск", "Екатеринбург"]  
  
# Матрица переходных вероятностей  
P = np.array([  
 [0.6, 0.2, 0.15, 0.05],  
 [0.3, 0.5, 0.1, 0.1],  
 [0.4, 0.1, 0.2, 0.3],  
 [0.1, 0.2, 0.3, 0.4]  
])

def simulate\_chain(P, initial\_vector, epsilon=1e-6, max\_steps=1000):  
 current = initial\_vector.copy()  
 history = [current.copy()]  
 rms\_errors = []  
  
 for step in range(max\_steps):  
 new = np.dot(current, P)  
 # Вычисление среднеквадратичного отклонения между итерациями  
 rms = np.sqrt(np.mean((new - current) \*\* 2))  
 rms\_errors.append(rms)  
 history.append(new.copy())  
 if rms < epsilon:  
 break  
 current = new  
 return np.array(history), rms\_errors

# Зададим несколько различных начальных векторов:  
initial\_vectors = {f"Начало в {state}": np.eye(len(states))[i] for i, state in enumerate(states)}  
  
# Для хранения результатов моделирования  
sim\_results = {}  
  
# Моделирование для каждого начального вектора  
for label, init\_vec in initial\_vectors.items():  
 history, rms\_errors = simulate\_chain(P, init\_vec, epsilon=1e-6, max\_steps=1000)  
 sim\_results[label] = {  
 "history": history,  
 "rms\_errors": rms\_errors,  
 "final\_vector": history[-1]  
 }  
 print(f"{label}: достигнуто за {len(history) - 1} шагов, финальный вектор: {history[-1]}")

# Графики для каждого начального вектора: компоненты вероятностного вектора по шагам  
for label, result in sim\_results.items():  
 history = result["history"]  
 steps = np.arange(history.shape[0])  
  
 plt.figure(figsize=(8, 6))  
 for i, state in enumerate(states):  
 plt.plot(steps, history[:, i], marker='o', label=state)  
 plt.xlabel("Шаг моделирования")  
 plt.ylabel("Вероятность")  
 plt.title(f"Изменение компонентов вероятностного вектора\n{label}")  
 plt.legend()  
 plt.grid(True)  
 plt.show()  
  
 # График изменения среднеквадратичного отклонения  
 plt.figure(figsize=(8, 6))  
 plt.plot(np.arange(len(result["rms\_errors"])), result["rms\_errors"], marker='o', color='red')  
 plt.xlabel("Шаг моделирования")  
 plt.ylabel("Среднеквадратичное отклонение")  
 plt.title(f"Изменение RMS отклонения\n{label}")  
 plt.grid(True)  
 plt.show()

def city\_forecast(days, start):  
 print("Начатьное состояние: ", start)  
 current\_city = start  
 path = [current\_city] # Список для хранения посещённых городов  
 prob = 1.0 # Произведение вероятностей переходов  
  
 for i in range(days):  
 # Определяем индекс текущего города  
 index = states.index(current\_city)  
 # Выбираем следующий город согласно вероятностям из соответствующей строки матрицы  
 next\_city = str(np.random.choice(states, replace=True, p=P[index]))  
 # Вероятность данного перехода  
 transition\_prob = P[index, states.index(next\_city)]  
 prob \*= transition\_prob # обновляем совокупную вероятность пути  
 path.append(next\_city)  
 current\_city = next\_city  
  
 print("Последовательность посещённых городов: " + str(path))  
 print("Конечное состояние через " + str(days) + " дней: " + current\_city)  
 print("Вероятность выбранной последовательности: " + str(prob))  
  
city\_forecast(5, states[0])  
city\_forecast(10, states[1])

# Вывод

В ходе выполнения лабораторной работы была разработана и реализована модель эргодической марковской цепи для имитации пассажиропотока между четырьмя городами России. Моделирование, проведённое с использованием различных начальных векторов, показало, что независимо от исходного распределения вероятностей система сходится к единому стационарному состоянию.

Аналитическое вычисление стационарного распределения подтвердило корректность результатов моделирования, поскольку полученные значения практически идентичны симулированным, что свидетельствует о правильности построенной модели и алгоритма.