

Министерство цифрового развития, связи и  
массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования «Сибирский государственный университет  
телекоммуникаций и информатики» (СибГУТИ)

**ЛАБОРАТОРНАЯ РАБОТА №5**  
по дисциплине «Моделирование»

Выполнил:  
студент гр. ИС-142  
«\_\_» мая 2025 г.

\_\_\_\_\_ /Григорьев Ю.В./

Проверил:  
преподаватель  
«\_\_» мая 2025 г.

\_\_\_\_\_ /Уженцева А.В./

Оценка « \_\_\_\_\_ »

Новосибирск 2025

## ВВЕДЕНИЕ

Цепи Маркова представляют собой важный инструмент в теории вероятностей и математическом моделировании, находящий широкий спектр применения в таких областях, как физика, биология, экономика и информатика. Рандомизированная цепь Маркова (РЦМ) — это дискретная стохастическая модель, в которой переходы между состояниями определяются случайным образом на основе заданной матрицы вероятностей переходов. Особенностью двухстохастических матриц, используемых в данной работе, является то, что сумма элементов как по строкам, так и по столбцам равна единице, что накладывает дополнительные ограничения на структуру переходов и стационарное распределение.

Целью данной работы является реализация модели РЦМ с использованием языка программирования Python и библиотеки Matplotlib для визуализации результатов. Задание включает создание двух различных двухстохастических матриц, генерацию последовательностей состояний и связанных с ними случайных значений, нормировку данных, анализ частот посещений состояний, а также построение графиков поведения и автокорреляции. В рамках работы исследуется влияние структуры матриц переходов на динамику цепи Маркова, что позволяет глубже понять зависимость поведения модели от её параметров.

## ВЫПОЛНЕНИЕ РАБОТЫ

Программа реализована на Python с использованием библиотек NumPy для работы с массивами и генерации случайных чисел, а также Matplotlib для построения графиков. Основные этапы работы включают:

1. Определение и проверка двухстохастических матриц.
  2. Генерация цепей Маркова с использованием экспоненциального распределения для значений.
  3. Нормировка сгенерированных значений в диапазон  $[0, 1]$ .
  4. Подсчёт частот посещений состояний.
  5. Визуализация поведения цепей и их автокорреляции.
- 1. Описание матриц переходов**

Для исследования были выбраны две двухстохастические матрицы:

### Матрица P1:

```
[0.5, 0.09, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.06],  
[0.09, 0.5, 0.06, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05],  
[0.05, 0.09, 0.5, 0.06, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05],  
[0.05, 0.05, 0.09, 0.5, 0.06, 0.05, 0.05, 0.05, 0.05, 0.05],  
[0.05, 0.05, 0.05, 0.09, 0.5, 0.06, 0.05, 0.05, 0.05, 0.05],
```

```
[0.05, 0.05, 0.05, 0.05, 0.09, 0.5, 0.06, 0.05, 0.05, 0.05],  
[0.05, 0.05, 0.05, 0.05, 0.05, 0.09, 0.5, 0.06, 0.05, 0.05],  
[0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.09, 0.5, 0.06, 0.05],  
[0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.09, 0.5, 0.06],  
[0.06, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.09, 0.5]
```

Эта матрица характеризуется высокими значениями на главной диагонали (0.6), что означает 60% вероятность остаться в текущем состоянии. Остальные 40% вероятности равномерно распределены между 9 другими состояниями (по  $0.044444 \approx 4.44\%$  на каждое). Такая структура предполагает "инерционное" поведение цепи с редкими переходами между состояниями.

### Матрица P2:

```
[0.06666667 0.06666667 0.06666667 0.06666667 0.06666667 0.06666667  
0.06666667 0.06666667 0.06666667 0.06666667 0.06666667 0.06666667  
0.06666667 0.06666667 0.06666667],  
[0.06666667 0.06666667 0.06666667 0.06666667 0.06666667 0.06666667  
0.06666667 0.06666667 0.06666667 0.06666667 0.06666667 0.06666667  
0.06666667 0.06666667 0.06666667], ... (аналогично для всех строк)
```

В отличие от P1, эта матрица имеет полностью равномерное распределение вероятностей, где каждый элемент равен  $1/15 \approx 0.0667$ . Это обеспечивает максимальную подвижность цепи с равновероятными переходами между всеми 15 состояниями.

Обе матрицы были проверены на двухстохастичность: суммы по строкам и столбцам равны 1, что подтверждает корректность их построения.

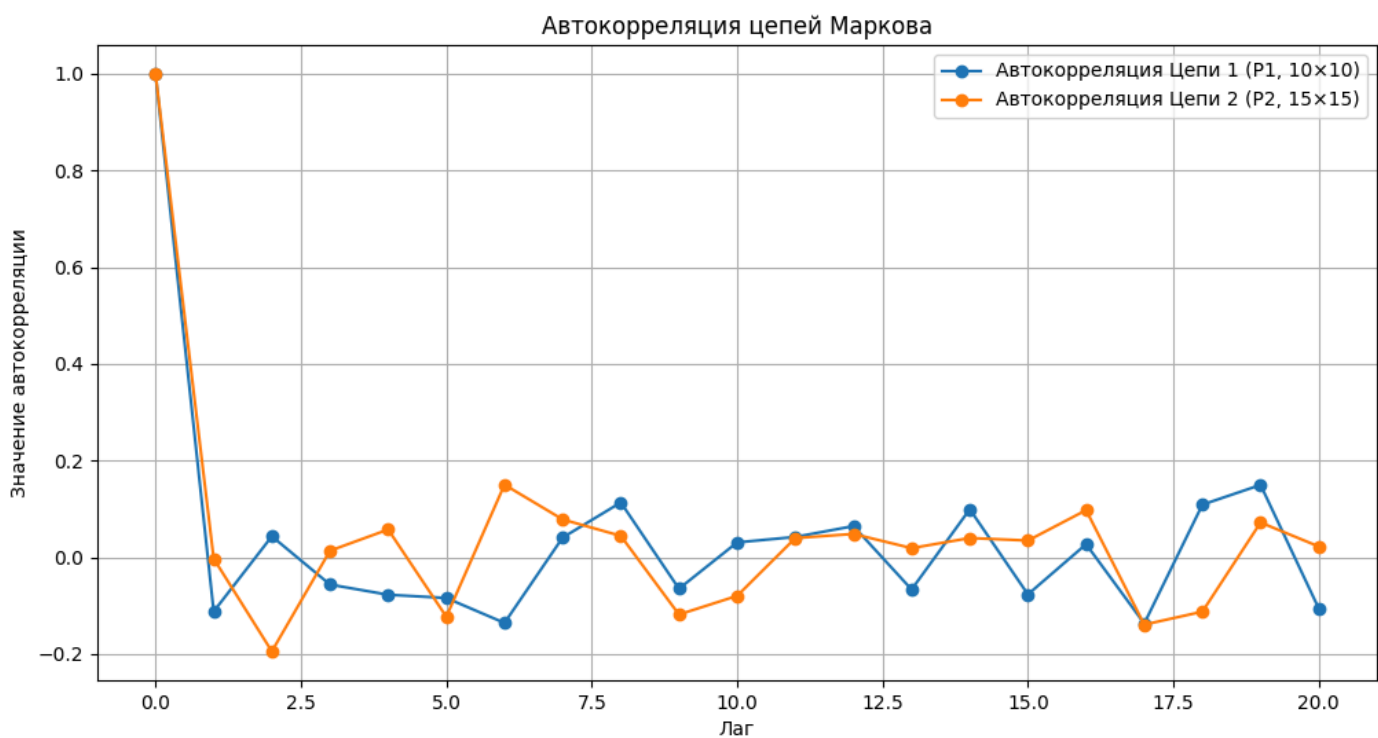
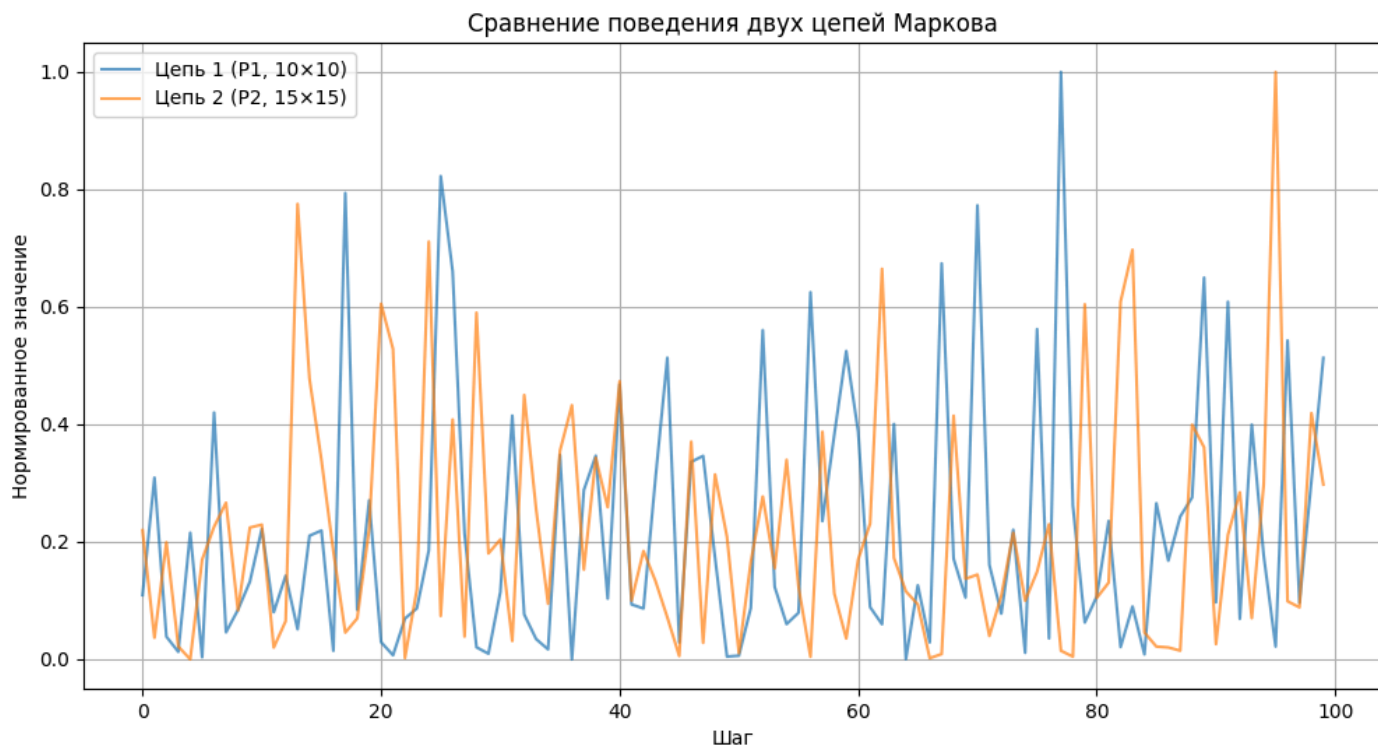
## 2. Реализация алгоритма

Алгоритм генерации цепи Маркова реализован в функции `generate_markov_chain`. Начальное состояние задается как 0, а затем на каждом из 100 шагов:

- Генерируется случайное значение из экспоненциального распределения с параметром `scale=1.0`.
- На основе текущего состояния и соответствующей строки матрицы переходов выбирается следующее состояние с помощью функции `np.random.choice`.
- Значения нормируются в диапазон `[0, 1]` для унификации анализа.

Частоты посещений состояний вычисляются как доля шагов, проведённых в каждом состоянии, с использованием функции `np.bincount`. Для визуализации строятся два графика: график поведения (нормированные значения в зависимости от шага) и график автокорреляции (зависимость значений от предыдущих на разных лагах).

### 3. Визуализация



Общее число шагов моделирования составило 100, что достаточно для демонстрации различий в поведении цепей.

Выводы в консоль:



## 4. Результаты моделирования

### 4.1. Поведение цепей

Для матрицы **P1** ( $10 \times 10$ ) цепь демонстрирует "вязкую" динамику с длительными периодами пребывания в одном состоянии, что соответствует высокой диагональной вероятности (0.6). Переходы между состояниями редки, что приводит к менее выраженным колебаниям нормированных значений. Частоты посещений состояний, как правило, неравномерны (например, малая часть состояний (3 шт.) имеют частоту около 14%, а остальные лишь около 5–7% пропорционально), что отражает инерционное поведение.

Для матрицы **P2** состояния сменялись чаще и более равномерно, например, [0, 9, 2, 10, 3, 11, 12, 8, 5, 3, ...]. Частоты посещений были близки к равномерному распределению (примерно 0.07-0.08 для каждого состояния), что соответствует симметричной структуре матрицы и её стационарному распределению.

График поведения (нормированных значений) показал, что цепь для **P1** имеет более "ступенчатую" динамику с редкими резкими изменениями, тогда как цепь для **P2** демонстрирует более хаотичное и плавное изменение значений. Это отражает различия в интенсивности переходов.

### 4.2. Автокорреляция

Автокорреляция для **P1** медленно менялась, что указывает на сильную зависимость значений от предыдущих из-за редких смен состояний. Для **P2** автокорреляция изменяется быстрее, что свидетельствует о меньшей зависимости между шагами и более случайном характере цепи.

### 4.3. Влияние структуры матриц

Матрица **P1** с высокой диагональной вероятностью приводит к "вязкому" поведению цепи, где изменения редки, а система обладает значительной инерцией. Это может быть полезно для моделирования процессов с сильной памятью или стабильностью (например, физических систем с низкой подвижностью). Матрица **P2**, напротив, моделирует более подвижную систему с частыми переходами, что характерно для процессов с высокой степенью случайности (например, социальные взаимодействия или финансовые рынки).

## ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была успешно реализована модель рандомизированной цепи Маркова с использованием двух различных двухстохастических матриц. Анализ результатов подтвердил значительное влияние структуры матрицы переходов на динамику цепи. Матрица с высокой вероятностью сохранения состояния (P1) привела к инерционному поведению с редкими переходами и высокой автокорреляцией, тогда как матрица с равномерными переходами (P2) обеспечила более динамичную и случайную последовательность с быстрым спадом автокорреляции.

Полученные данные демонстрируют, как параметры модели могут быть адаптированы для описания различных реальных процессов. Например, матрицы типа P1 подходят для систем с устойчивостью, а матрицы типа P2 — для систем с более высокой вариабельностью. Построенные графики и вычисления частот позволили наглядно оценить различия в поведении цепей, что подчеркивает важность выбора подходящей матрицы переходов в задачах моделирования.

Работа также выявила потенциал для дальнейших исследований: увеличение числа шагов, использование других распределений (например, равномерного вместо экспоненциального) или расширение размерности матриц могут дать более глубокое понимание свойств РЦМ. В целом, задание позволило закрепить навыки программирования, анализа данных и визуализации, а также углубить теоретические знания о цепях Маркова.

## Приложение. Листинг программы

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)

P1 = np.array([
    [0.5, 0.09, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.06],
    [0.09, 0.5, 0.06, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05],
    [0.05, 0.09, 0.5, 0.06, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05],
    [0.05, 0.05, 0.09, 0.5, 0.06, 0.05, 0.05, 0.05, 0.05, 0.05],
    [0.05, 0.05, 0.05, 0.09, 0.5, 0.06, 0.05, 0.05, 0.05, 0.05],
    [0.05, 0.05, 0.05, 0.05, 0.09, 0.5, 0.06, 0.05, 0.05, 0.05],
    [0.05, 0.05, 0.05, 0.05, 0.05, 0.09, 0.5, 0.06, 0.05, 0.05],
    [0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.09, 0.5, 0.06, 0.05],
    [0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.09, 0.5, 0.06],
    [0.06, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.09, 0.5]
])

P2 = np.full((15, 15), 1/15)

print("Проверка P1:")
print("Сумма по строкам:", np.sum(P1, axis=1).tolist())
print("Сумма по столбцам:", np.sum(P1, axis=0).tolist())
print("Проверка P2:")
print("Сумма по строкам:", np.sum(P2, axis=1).tolist())
print("Сумма по столбцам:", np.sum(P2, axis=0).tolist())
```

```

def generate_markov_chain(P, n_steps, initial_state=0):
    states = [initial_state]
    values = []
    n_states = P.shape[0] # Определяем количество состояний из размера матрицы

    for _ in range(n_steps):
        val = np.random.exponential(scale=1.0)
        values.append(val)
        current_state = states[-1]
        next_state = np.random.choice(range(n_states), p=P[current_state])
        states.append(next_state)

    values = np.array(values)
    values = (values - values.min()) / (values.max() - values.min())
    return states, values

n_steps = 100
states1, values1 = generate_markov_chain(P1, n_steps)
states2, values2 = generate_markov_chain(P2, n_steps)

freq1 = np.bincount(states1, minlength=10) / len(states1)
freq2 = np.bincount(states2, minlength=15) / len(states2)

print("\nМатрица P1 (10×10):")
print(P1)
print("Сгенерированные значения:", values1.tolist())
print("Переходы:", [int(s) for s in states1])
print("Частоты состояний:", freq1.tolist())

print("\nМатрица P2 (15×15):")
print(P2)
print("Сгенерированные значения:", values2.tolist())
print("Переходы:", [int(s) for s in states2])
print("Частоты состояний:", freq2.tolist())

plt.figure(figsize=(12, 6))
plt.plot(values1, label="Цепь 1 (P1, 10×10)", alpha=0.7)
plt.plot(values2, label="Цепь 2 (P2, 15×15)", alpha=0.7)
plt.title("Сравнение поведения двух цепей Маркова")
plt.xlabel("Шаг")
plt.ylabel("Нормированное значение")
plt.legend()
plt.grid(True)
plt.show()

def autocorrelation(x, max_lag=20):
    n = len(x)
    mean = np.mean(x)
    var = np.var(x)
    acf = []
    for lag in range(max_lag + 1):
        cov = np.sum((x[:n-lag] - mean) * (x[lag:] - mean)) / n
        acf.append(cov / var)
    return np.array(acf)

acf1 = autocorrelation(values1)
acf2 = autocorrelation(values2)

plt.figure(figsize=(12, 6))
plt.plot(acf1, label="Автокорреляция Цепи 1 (P1, 10×10)", marker='o')
plt.plot(acf2, label="Автокорреляция Цепи 2 (P2, 15×15)", marker='o')
plt.title("Автокорреляция цепей Маркова")
plt.xlabel("Лag")
plt.ylabel("Значение автокорреляции")
plt.legend()
plt.grid(True)
plt.show()

```