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. _____ /Наумов А.А./

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

В рамках данной работы была реализована модель рандомизированной цепи Маркова (РЦМ) с использованием двух различных двухстохастических матриц размером 10x10. Для выполнения задания применялись библиотеки Python: NumPy для работы с массивами и генерации случайных чисел, а также Matplotlib для визуализации результатов. Основные этапы работы включали:

1. **Создание и проверка двухстохастических матриц.**
2. **Генерация цепей Маркова и случайных значений.**
3. **Нормировка сгенерированных значений.**
4. **Подсчет частот посещений состояний.**
5. **Визуализация поведения цепей и их автокорреляции.**

1. Создание и проверка двухстохастических матриц

Были созданы две двухстохастические матрицы размером 10x10, где двухстохастическая матрица характеризуется тем, что сумма элементов в каждой строке и каждом столбце равна 1. Проверка выполнялась с помощью функции is_doubly_stochastic, использующей метод np.allclose с допуском 1e-3 для учета вычислительных погрешностей.

```
alexeynaumov@Lenovo-Legion-5:~/4course/model$ python3 5_fix.py

Проверка матриц:

Матрица 1 (10x10):
0.119 0.079 0.037 0.170 0.145 0.150 0.054 0.032 0.034 0.181
0.063 0.007 0.110 0.087 0.045 0.147 0.060 0.293 0.166 0.022
0.146 0.114 0.054 0.006 0.175 0.029 0.156 0.060 0.092 0.168
0.187 0.088 0.095 0.066 0.191 0.073 0.057 0.069 0.114 0.060
0.055 0.112 0.139 0.079 0.061 0.136 0.111 0.212 0.090 0.007
0.021 0.143 0.215 0.077 0.023 0.126 0.043 0.001 0.144 0.208
0.082 0.116 0.002 0.139 0.103 0.082 0.196 0.137 0.102 0.041
0.147 0.117 0.071 0.146 0.053 0.054 0.114 0.011 0.100 0.188
0.058 0.156 0.149 0.132 0.130 0.108 0.106 0.001 0.126 0.033
0.123 0.068 0.128 0.099 0.074 0.095 0.102 0.185 0.033 0.092
• Двустохастическая: True
```

```
Матрица 2 (10x10):
0.118 0.095 0.014 0.121 0.176 0.091 0.052 0.102 0.160 0.071
0.092 0.006 0.227 0.061 0.012 0.161 0.140 0.099 0.153 0.049
0.139 0.067 0.054 0.096 0.169 0.118 0.043 0.081 0.222 0.010
0.135 0.183 0.055 0.068 0.007 0.067 0.151 0.161 0.084 0.090
0.164 0.052 0.115 0.108 0.160 0.020 0.134 0.064 0.038 0.146
0.131 0.005 0.157 0.103 0.124 0.112 0.106 0.054 0.077 0.132
0.015 0.290 0.058 0.008 0.137 0.091 0.132 0.026 0.167 0.074
0.017 0.112 0.167 0.167 0.140 0.067 0.018 0.116 0.055 0.142
0.101 0.089 0.096 0.133 0.008 0.146 0.123 0.110 0.022 0.172
0.087 0.101 0.057 0.135 0.067 0.126 0.102 0.187 0.022 0.116
• Двустохастическая: True
```

Обе матрицы подтвердили свою двухстохастичность, что обеспечило корректность дальнейших вычислений.

2. Генерация цепей Маркова и случайных значений

Для каждой матрицы была сгенерирована цепь Маркова длиной 100 шагов. Начальное состояние выбиралось случайным образом, а последующие состояния определялись на основе вероятностей переходов из соответствующей матрицы. Последовательности состояний для матрицы 1:

```
Сгенерированные значения: [1.45755386 1.22287716 0.84555803 0.28203633 0.18189975 0.30431505
 0.91346763 1.40008686 0.84073373 3.65798043 3.61417029 0.62187257
 0.82607449 1.00736845 3.50414732 1.48501622 0.21758928 1.17899002
 0.10346406 0.51189045 1.00871592 0.15047395 0.72995631 0.01978422
 1.36287103 1.48445966 0.13435201 0.58311522 1.45165525 3.02588704
 0.42295911 2.48713269 1.11086756 2.89500568 0.48395751 1.34730783
 3.56941254 0.53300825 0.06753374 0.77066293 0.37525152 0.15746628
 1.36350629 0.86651225 0.93690187 0.32751832 0.25409117 0.65404141
 0.82626311 0.13531463 1.46917877 1.01680434 0.01347692 0.26661251
 2.1722444  2.15789883 1.32794849 0.27535869 5.7014334  1.99844536
 2.5808002  0.34847545 2.32604536 0.07787698 0.76121514 1.26591967
 0.80738631 0.71979116 0.59418994 1.470912   0.04936204 0.06188491
 1.361672   1.06210638 0.05697466 1.14823046 0.62279622 0.33344469
 0.2826883  0.2830378  2.34479279 1.63370132 0.4418913  0.45060075
 0.18988358 0.60355829 2.19959024 1.95929556 2.02235992 4.3193872
 2.5661226  0.20674193 0.01321034 1.09158626 0.46316634 0.22896703
 0.28778299 0.73191643 3.60877463 1.50829296]
```

Для матрицы 2:

```
Сгенерированные значения: [3.56644652 3.10418067 1.46671656 0.36923306 0.1719072  0.18432073
 0.63474852 0.85506569 0.22066552 3.29040165 0.45464052 1.00182621
 0.02230496 2.28455109 0.50787875 0.77703056 0.01764835 0.91075662
 0.18531264 3.1191766  1.55903548 0.29465567 0.9031053  1.67035649
 0.47175225 0.35327893 0.7979104  0.40394311 0.78053864 0.38221441
 0.07060106 2.5642702  0.08113217 0.98340999 0.68746329 0.01136018
 0.20342884 0.70467073 3.22878706 0.50657471 0.3398745  1.72729425
 0.37092789 0.75541771 1.77477608 0.30746619 0.09965562 0.23275503
 0.27474651 0.03474024 0.82279757 0.5903223  1.01703969 0.49835734
 1.66635544 0.7299711  0.26332198 1.10459367 5.14924696 0.27866608
 0.42127348 0.61150229 0.7288218  0.12074574 2.0903651  0.30982839
 1.146349   0.35343359 0.6987704  1.04194193 0.2508879  0.2893725
 0.51231548 0.56630764 0.26901806 0.93411994 0.46014636 0.69239682
 0.58097414 0.22976214 1.55108547 0.2490415  1.13228673 0.54347153
 0.27070273 2.45624903 1.28007911 0.60020737 1.55241722 0.14143118
 0.59380913 0.40236302 0.29589262 0.32725125 2.12126971 0.34732747
 0.53957622 0.28138384 2.64667529 1.24713447]
```

Для каждого состояния в цепи генерировались случайные значения из экспоненциального распределения с параметром scale=1.0. Значения варьировались от 0.01321034 до 5.7014334 для матрицы 1 и от 0.01136018 до 5.14924696 для матрицы 2, что демонстрирует широкий диапазон случайных величин.

3. Нормировка сгенерированных значений

Сгенерированные значения были приведены к диапазону [0, 1] с использованием min-max нормализации. Это позволило унифицировать значения для сравнения между матрицами и упростило их визуализацию.

4. Подсчет частот посещений состояний

Переходы матрицы 1:

```
Переходы матрицы: [0 4 6 1 7 0 9 0 3 0 0 9 2 9 8 6 7 9 2 2 6 7 9 0 3 8 0 2 0 4 2 4 8 9 2 9 1
 7 6 4 1 7 0 2 4 0 0 6 7 1 8 6 8 5 6 8 3 8 3 0 0 9 7 0 7 9 5 6 1 7 9 9 3 3
 4 1 5 1 5 5 9 7 3 1 7 3 2 6 3 3 5 2 4 6 3 2 6 1 3 2]
```

Частоты посещений каждого состояния вычислялись как доля шагов, проведённых в данном состоянии, с использованием функции np.bincount.

Для матрицы 1:

```
Частоты  матрицы:  [0.14 0.09 0.11 0.12 0.07 0.06 0.11 0.11 0.07 0.12]
```

Переходы матрицы 2:

```
Переходы матрицы: [2 7 2 3 3 3 6 6 6 4 2 8 3 7 3 7 7 5 9 0 1 0 8 5 4 4 0 4 0 3 7 9 9 7 9 2 7
 7 3 7 6 1 2 3 7 7 9 7 4 4 0 1 8 7 3 1 5 8 3 5 4 1 8 1 8 1 3 2 0 8 7 6 8 1
 7 9 5 4 9 3 1 2 8 5 0 1 2 8 5 6 4 0 5 2 4 2 5 2 0 4]
```

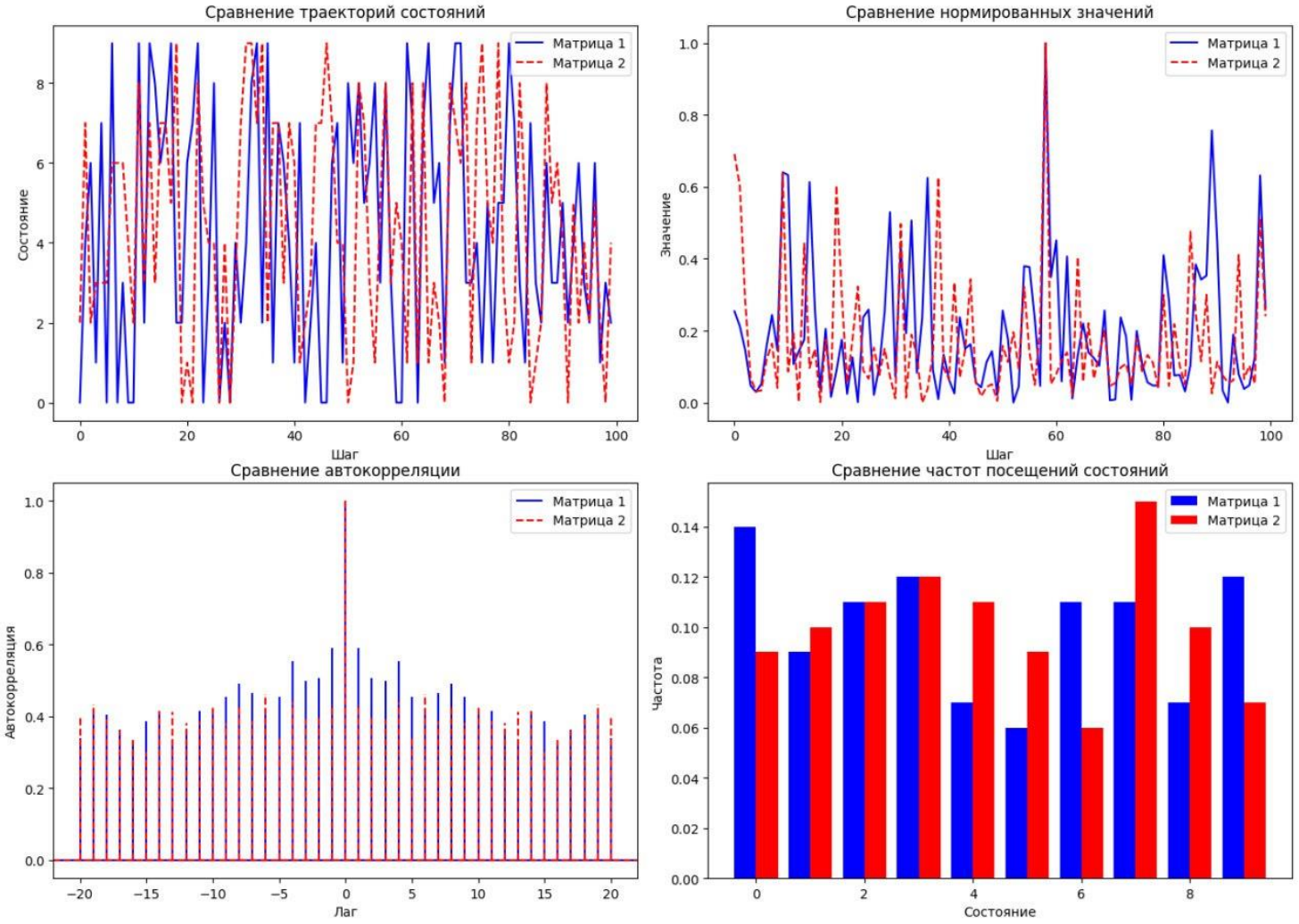
Для матрицы 2:

```
 7 9 5 4 9 3 1 2 8 5 0 1 2 8 5 6 4 0 5 2 4 2 5 2 0 4]
Частоты матрицы:  [0.09 0.1  0.11 0.12 0.11 0.09 0.06 0.15 0.1  0.07]
alexeynaumov@Lenovo-Legion-5:~/4course/model$
```

Эти данные показывают распределение времени пребывания цепи в каждом из 10 состояний.

5. Визуализация поведения цепей и их автокорреляции

Для анализа поведения цепей Маркова были построены четыре графика, отражающие различные аспекты их динамики:



- **Сравнение траекторий состояний**:
  График демонстрирует переходы между состояниями для матрицы 1 (синяя линия) и матрицы 2 (красная пунктирная линия) на 100 шагах. Ось X — "Шаг" (0–100), ось Y — "Состояние" (0–9).
  Наблюдения: Для матрицы 1 цепь активно переключается между состояниями, с частыми скачками (например, 0 → 4 → 6 → 1). Для матрицы 2 заметно большее количество посещений

состояния 7 (частота 0.15), что отражено в траектории. Обе цепи демонстрируют случайное блуждание без признаков сходимости за 100 шагов.

- **Сравнение нормированных значений**:
  График отображает нормированные экспоненциальные значения (ось Y: 0.0–1.0) для обеих матриц на 100 шагах (ось X).
  Наблюдения: Для матрицы 1 значения колеблются с пиками до 1.0 около шагов 10, 60 и 90, что соответствует большим исходным значениям (например, 5.7014334). Для матрицы 2 пики наблюдаются на шагах 20 и 60 (максимум 5.14924696). Высокая вариативность указывает на разнообразие сгенерированных значений.
- **Сравнение автокорреляции**:
  График показывает автокорреляцию нормированных значений с максимальным лагом 20 (ось X: -20–20, ось Y: 0.0–1.0).
  Наблюдения: Для матрицы 1 автокорреляция резко падает после лага 0 (1.0) до значений близких к 0, с небольшими всплесками (например, около лагов 5 и 15). Для матрицы 2 на лаге 1 наблюдается пик около 0.2, что указывает на чуть более выраженную краткосрочную зависимость. В целом, обе цепи демонстрируют слабую корреляцию между шагами.
- **Сравнение частот посещений состояний**:
  Столбчатая диаграмма сравнивает частоты посещений (ось Y: 0.00–0.16) для состояний 0–9 (ось X).
  Наблюдения: Для матрицы 1 состояние 0 самое частое (0.14), для матрицы 2 — состояние 7 (0.15). Минимальные частоты: 0.06 (состояние 5 для матрицы 1 и состояние 6 для матрицы 2). Это подчеркивает различия в вероятностях переходов между матрицами.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была успешно реализована модель рандомизированной цепи Маркова (РЦМ) с использованием двух различных двухстохастических матриц переходов размером 10x10. Анализ поведения цепей Маркова и связанных с ними случайных процессов позволил сделать следующие ключевые выводы:

- **Динамика траекторий состояний**:
  Обе цепи Маркова продемонстрировали случайное поведение с частыми переходами между состояниями. При этом матрица 2 чаще посещала состояние 7 (частота 0.15), что напрямую связано с особенностями её структуры переходов, в отличие от матрицы 1, где состояние 0 имело частоту 0.14. Это подтверждает, что структура матрицы переходов играет определяющую роль в формировании динамики цепи.

- **Вариативность нормированных значений**:
  Сгенерированные экспоненциальные величины показали значительную вариативность, отражая широкий диапазон значений. Пики нормированных значений для каждой матрицы возникали в разные моменты времени, что подчеркивает различия в поведении случайных процессов, обусловленные спецификой каждой матрицы переходов.

- **Автокорреляция**:
  Анализ автокорреляции выявил быстрое падение корреляции до нуля после лага 1 для обеих цепей, что указывает на слабую зависимость значений между последовательными шагами. Однако матрица 2 продемонстрировала немного большую краткосрочную корреляцию, что может быть связано с её структурой переходов.

- **Влияние структуры матриц**:
  Различия в частотах посещений состояний (0.14 для состояния 0 в матрице 1 и 0.15 для состояния 7 в матрице 2) наглядно иллюстрируют, как структура матриц переходов определяет долгосрочное поведение цепей Маркова. Это согласуется с теоретическими ожиданиями о роли вероятностей переходов в формировании стационарного распределения.

Работа успешно продемонстрировала реализацию РЦМ, показав, что структура двухстохастических матриц переходов существенно влияет на динамику цепей Маркова и связанные с ними случайные процессы. Визуализация результатов обеспечила наглядное представление этих закономерностей, что позволило лучше понять зависимость поведения модели

от её параметров. Полученные выводы могут служить основой для дальнейших исследований свойств цепей Маркова и их применения в различных областях.

Листинг программы

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

# Параметры
n_states = 10  # Количество состояний
n_steps = 100  # Длина цепи Маркова

# Функция проверки двустохастичности
def is_doubly_stochastic(matrix, tolerance=1e-3):
    row_sums = np.sum(matrix, axis=1)
    col_sums = np.sum(matrix, axis=0)
    return (np.allclose(row_sums, 1.0, atol=tolerance) and
            (np.allclose(col_sums, 1.0, atol=tolerance)))

# Функция для создания двухстохастической матрицы
def create_doubly_stochastic_matrix(n):
    matrix = np.random.rand(n, n)
    for _ in range(10):  # Увеличили число итераций для лучшей сходимости
        matrix /= matrix.sum(axis=1, keepdims=True)
        matrix /= matrix.sum(axis=0, keepdims=True)
    return matrix

def print_matrix(matrix, name, max_size=10):
    print(f"\n{name} ({matrix.shape[0]}x{matrix.shape[1]}):")
    n = min(max_size, matrix.shape[0])
    for row in matrix[:n]:
        print(' '.join([f"{x:.3f}" for x in row[:n]]))
        if n < matrix.shape[0]: print("...")
    if n < matrix.shape[0]: print("...")

# Создаем и проверяем матрицы
matrix1 = create_doubly_stochastic_matrix(n_states)
matrix2 = create_doubly_stochastic_matrix(n_states)

# Функция для генерации цепи Маркова
def generate_markov_chain(matrix, n_steps):
    states = [np.random.randint(0, matrix.shape[0])]
    for _ in range(n_steps - 1):
        current_state = states[-1]
        next_state = np.random.choice(matrix.shape[0], p=matrix[current_state])
        states.append(next_state)
    return np.array(states)

def generate_exponential_values(states, scale=1.0):
    return np.random.exponential(scale, size=len(states))

def normalize_values(values):
    min_val = np.min(values)
    max_val = np.max(values)
    return (values - min_val) / (max_val - min_val)

def compute_frequencies(states, n_states):
    return np.bincount(states, minlength=n_states) / len(states)

try:
    states1 = generate_markov_chain(matrix1, n_steps)
    states2 = generate_markov_chain(matrix2, n_steps)
except ValueError as e:
    print(f"Ошибка: {e}")
    exit()

values1 = generate_exponential_values(states1)
normalized_values1 = normalize_values(values1)
frequencies1 = compute_frequencies(states1, n_states)

values2 = generate_exponential_values(states2)
normalized_values2 = normalize_values(values2)
frequencies2 = compute_frequencies(states2, n_states)


print("\nПроверка матриц:")
print_matrix(matrix1, "Матрица 1")
print(f"• Двустохастическая: {is_doubly_stochastic(matrix1)}")
```

```
print("Сгенерированные значения:", values1)
print("Переходы матрицы:", states1)
print("Частоты  матрицы: ",frequencies1)

print_matrix(matrix2, "Матрица 2")
print(f"• Двустохастическая: {is_doubly_stochastic(matrix2)}")
print("Сгенерированные значения:", values2)
print("Переходы матрицы:", states2)
print("Частоты матрицы: ",frequencies2)

# Построение графиков
plt.figure(figsize=(14, 12))

plt.subplot(2, 2, 1)
plt.plot(states1[:100], label='Матрица 1', color='blue')
plt.plot(states2[:100], label='Матрица 2', color='red', linestyle='--')
plt.title("Сравнение траекторий состояний")
plt.xlabel("Шаг")
plt.ylabel("Состояние")
plt.legend()

plt.subplot(2, 2, 2)
plt.plot(normalized_values1[:100], label='Матрица 1', color='blue')
plt.plot(normalized_values2[:100], label='Матрица 2', color='red', linestyle='--')
plt.title("Сравнение нормированных значений")
plt.xlabel("Шаг")
plt.ylabel("Значение")
plt.legend()

plt.subplot(2, 2, 3)
plt.acorr(normalized_values1, maxlags=20, color='blue', label='Матрица 1')
plt.acorr(normalized_values2, maxlags=20, color='red', linestyle='--', label='Матрица 2')
plt.title("Сравнение автокорреляции")
plt.xlabel("Лаг")
plt.ylabel("Автокорреляция")
plt.legend()

plt.subplot(2, 2, 4)
bar_width = 0.4
x = np.arange(n_states)
plt.bar(x - bar_width/2, frequencies1, width=bar_width, label='Матрица 1', color='blue')
plt.bar(x + bar_width/2, frequencies2, width=bar_width, label='Матрица 2', color='red')
plt.title("Сравнение частот посещений состояний")
plt.xlabel("Состояние")
plt.ylabel("Частота")
plt.legend()

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