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

Для выполнения работы были выбраны две матрицы с принципиально разными характеристиками: первая матрица демонстрирует сильную склонность к сохранению текущего состояния (высокие диагональные элементы), вторая — более равномерное распределение вероятностей переходов. Такой выбор позволяет сравнить контрастные сценарии поведения РЦМ и оценить их влияние на итоговые характеристики модели. 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:

[0.8, 0.1, 0.1]
[0.1, 0.8, 0.1]
[0.1, 0.1, 0.8]

Эта матрица характеризуется высокими значениями на главной диагонали (0.8), что означает высокую вероятность (80%) остаться в текущем состоянии. Вероятность перехода в другие состояния составляет всего 10% для каждого. Такая структура предполагает "инерционное" поведение цепи, где состояния меняются редко.

Матрица P2:

[0.2, 0.4, 0.4]
[0.4, 0.2, 0.4]
[0.4, 0.4, 0.2]

В отличие от P1, эта матрица демонстрирует более равномерное распределение вероятностей переходов. Вероятность остаться в текущем состоянии составляет 20%, а вероятность перейти в любое из двух других состояний — 40%. Это обеспечивает более динамичное и сбалансированное поведение цепи.

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

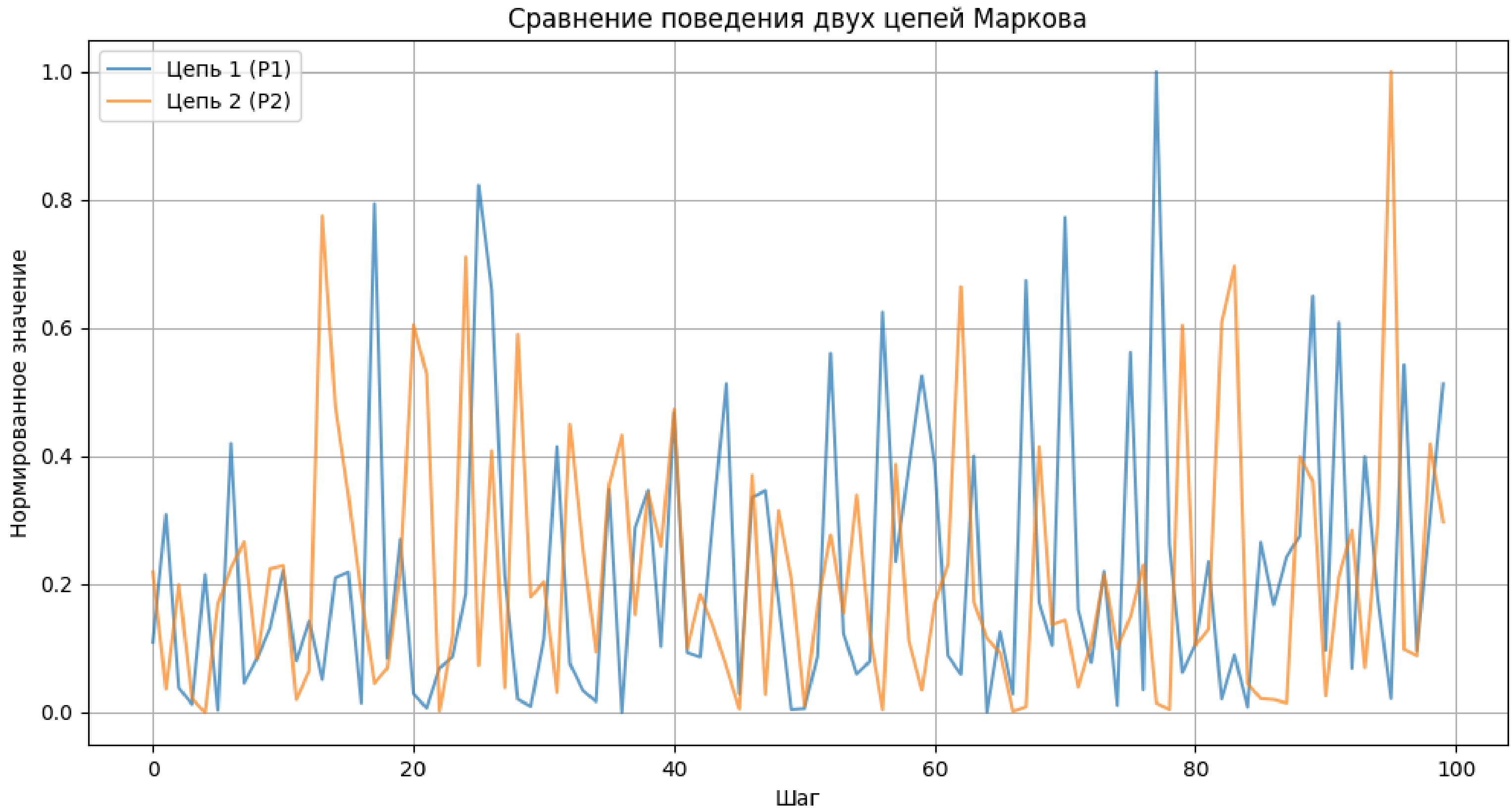
2. Реализация алгоритма

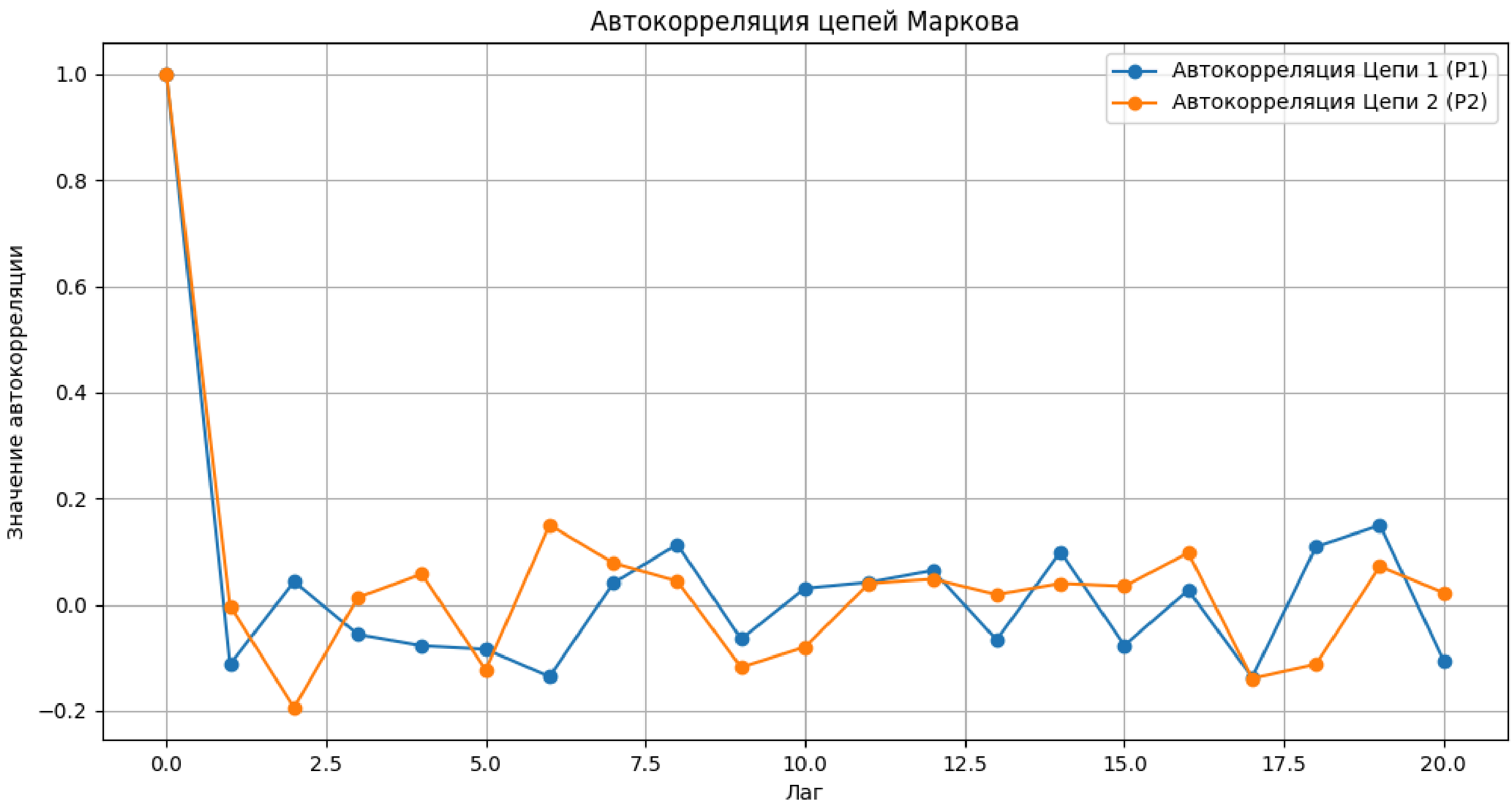
Алгоритм генерации цепи Маркова реализован в функции generate_markov_chain. Начальное состояние задается как 0, а затем на каждом из 100 шагов:

- 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:

```
Проверка P1:
Сумма по строкам: [1.0, 1.0, 1.0]
Сумма по столбцам: [1.0, 1.0, 1.0]
Проверка P2:
Сумма по строкам: [1.0, 1.0, 1.0]
Сумма по столбцам: [1.0, 1.0, 1.0]

Матрица P1:
[[0.8 0.1 0.1]
 [0.1 0.8 0.1]
 [0.1 0.1 0.8]]
Сгенерированные значения: [0.10940915830976777, 0.30935670840951174, 0.03871357013889O985, 0.01281145543024574З, 0.2155349363227671, 0.003600776029133253S, 0
.42017041569253855, 0.04604024738193646, 0.08427895387013175, 0.13212214326820335, 0.22197323361920865, 0.08021189730802905, 0.14236105457211534, 0.051244181
71179484, 0.21044411774400948, 0.21936904478112912, 0.014563292881586809, 0.793934946240117, 0.08440497264173662, 0.270664966649062, 0.02940069044449198, 0.
006949700155679851, 0.06934537851160325, 0.08682533898558624, 0.1853689887349571, 0.822760427635355, 0.6605066150901189, 0.21364093087906988, 0.0205539807891
71797, 0.00961296437868922, 0.11480309582978976, 0.4150101749534347, 0.07650476660353711, 0.034531259304263484, 0.016972621414490918, 0.3477519188622604, 0.0
, 0.2882054059634546, 0.34674466378642355, 0.10342227139660266, 0.4678523902535486, 0.0934956757824928, 0.08657566779139815, 0.3072640080429173, 0.5135577956
449184, 0.02874483563121979, 0.3361697486994986, 0.34643215713101433, 0.17320785519049645, 0.00476828021742216, 0.006227737087030021, 0.08773370928996868, 0.
5605115854288004, 0.12333245452325957, 0.059990034871472635, 0.07941557585847299, 0.6250826799908707, 0.2354504832406649, 0.382784402013113, 0.52501503236771
52, 0.38735662156355477, 0.08899217529778407, 0.05972616536727909, 0.4006824304980585, 0.00033950429499511657, 0.12616160334347634, 0.02881750236755471, 0.67
41970030558334, 0.17126706626358454, 0.10532903732451042, 0.7730264911091334, 0.16093474779729047, 0.07778985255597137, 0.22058623266972652, 0.01116283399577
8403, 0.5623035925133433, 0.035624267123849834, 1.0, 0.2617949049250613, 0.0627098689186705, 0.10687399449654372, 0.2355315583032103, 0.02101964082511682, 0.
08995468035155899, 0.008515357590208427, 0.2657342055489911, 0.1680059543989508, 0.2431489979032266, 0.27572849543349676, 0.6499484916716439, 0.0971086601697
7444, 0.6088600314521098, 0.06907867876028843, 0.3996571492786291, 0.1766526052022117, 0.021750319599018485, 0.5429367099088905, 0.09638059021517535, 0.30410
006900418407, 0.51329370619806]
Переходы: [0, 2, 2, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 0, 0, 2, 2, 0, 0, 0, 2, 2, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Частоты состояний: [0.39603960396039606, 0.46534653465346537, 0.13861386138613863]
```

```
Матрица P2:
[[0.2 0.4 0.4]
 [0.4 0.2 0.4]
 [0.4 0.4 0.2]]
Сгенерированные значения: [0.21974452688501428, 0.03680617742639318, 0.19936229040098297, 0.02190971370100463, 0.0, 0.16995598215838265, 0.22579164836527366,
 0.2666297066034755, 0.08352671558284383, 0.2243583363457326, 0.2293110058018502, 0.020052454639489035, 0.06515267454306296, 0.7754275360493131, 0.4774320675
241842, 0.33937541232768526, 0.18381217737726419, 0.04560267192571649, 0.06975661708335026, 0.22182067415870296, 0.6053419625721076, 0.5284763975673541, 0.00
2257746203659275, 0.11906107487775362, 0.7112447342977315, 0.07388365638445402, 0.40835782791316166, 0.038832089401514475, 0.5903396582233984, 0.180363379858
10133, 0.20409941378972535, 0.031351747637078146, 0.4500355759745059, 0.2555928125409731, 0.09467422389575, 0.35518589258163724, 0.43269684072335224, 0.15284
38154979421, 0.3430566108737036, 0.25913461596559334, 0.4734054091098651, 0.10014387712233996, 0.18451255843426956, 0.13360743833082778, 0.07148778912675709,
 0.005567715448643912, 0.3706583642800561, 0.028120498697219925, 0.31483134113099276, 0.20854679393596973, 0.010316288977762173, 0.1661323215031851, 0.277279
7527548004, 0.15503599837777904, 0.3397685237601815, 0.12315510732563681, 0.0044289510983550615, 0.38751508072663704, 0.11195034247934504, 0.0354790997383639
5, 0.17019089585397698, 0.23093980520103416, 0.6648915095992108, 0.17265005420412424, 0.11585801505747674, 0.0934966024658768, 0.002025748829230859, 0.009032
796238995004, 0.41469420405524465, 0.13708505829826353, 0.1443365363082035, 0.03979429134103369, 0.10818351276023319, 0.2156180340837196, 0.0998100836385049,
 0.14926384273328963, 0.22999061095010354, 0.014640803402214814, 0.004685073398104628, 0.6045191446205602, 0.10452139387623538, 0.13067296322018732, 0.609005
8176386381, 0.6973486506663577, 0.045749141358585835, 0.02174385439547549, 0.020234733849355705, 0.014784225729120677, 0.39950115802175146, 0.361023468873909
, 0.02594099030375521, 0.2120248248179632, 0.2844451685287155, 0.07013420096130989, 0.2974425290547919, 1.0, 0.09892011059718821, 0.08849212988677724, 0.4191
302384626142, 0.29766358147601923]
Переходы: [0, 0, 2, 0, 2, 0, 2, 0, 1, 2, 2, 1, 0, 1, 0, 2, 1, 1, 2, 0, 0, 2, 0, 2, 2, 2, 0, 1, 1, 2, 0, 2, 1, 2, 1, 0, 2, 2, 1, 2, 1, 0, 0, 0, 1, 1, 0, 1, 1,
 0, 0, 1, 2, 2, 0, 1, 0, 2, 1, 0, 1, 2, 0, 2, 1, 0, 2, 0, 0, 2, 0, 1, 1, 2, 0, 2, 2, 0, 2, 1, 1, 2, 1, 0, 2, 0, 0, 2, 0, 0, 1, 2, 2, 2, 0, 2, 1, 2, 2, 1, 2]
Частоты состояний: [0.3465346534653465, 0.27722772277227725, 0.37623762376237624]
```

4. Результаты моделирования

4.1. Поведение цепей

Для матрицы **P1** последовательность состояний демонстрировала длительные периоды пребывания в одном состоянии, например, [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, ...]. Это ожидаемо, учитывая высокую вероятность сохранения состояния. Частоты посещений могли быть неравномерными (например, 0.39, 0.47, 0.14), так как цепь могла "застрять" в одном состоянии из-за случайного характера генерации.

Для матрицы **P2** состояния сменялись чаще и более равномерно, например, [0, 1, 2, 1, 0, 2, 1, ...]. Частоты посещений были близки к равномерному распределению (примерно 0.33 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([
    [0.8, 0.1, 0.1],
    [0.1, 0.8, 0.1],
    [0.1, 0.1, 0.8]
])

P2 = np.array([
    [0.2, 0.4, 0.4],
    [0.4, 0.2, 0.4],
    [0.4, 0.4, 0.2]
])

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]  # Список состояний
    values = []  # Сгенерированные значения

    for _ in range(n_steps):
        # Генерируем случайное значение из экспоненциального распределения
        val = np.random.exponential(scale=1.0)
        values.append(val)
        # Переход в следующее состояние на основе текущего
        current_state = states[-1]
        next_state = np.random.choice([0, 1, 2], p=P[current_state])
        states.append(next_state)

    # Нормируем значения (приводим к диапазону 0-1)
    values = np.array(values)
    values = (values - values.min()) / (values.max() - values.min())
    return states, values

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

freq1 = np.bincount(states1, minlength=3) / len(states1)
freq2 = np.bincount(states2, minlength=3) / len(states2)

print("\nМатрица P1:")
print(P1)
```

```
print("Сгенерированные значения:", values1.tolist())
print("Переходы:", [int(s) for s in states1])
print("Частоты состояний:", freq1.tolist())

print("\nМатрица P2:")
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)", alpha=0.7)
plt.plot(values2, label="Цепь 2 (P2)", 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)", marker='o')
plt.plot(acf2, label="Автокорреляция Цепи 2 (P2)", marker='o')
plt.title("Автокорреляция цепей Маркова")
plt.xlabel("Лаг")
plt.ylabel("Значение автокорреляции")
plt.legend()
plt.grid(True)
plt.show()
```