Лабораторная работа №3
ЦЕПИ МАРКОВА И СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ

Выполнил работу: Студент группы ИА-231

Зырянов Иван

2024

Цель лабораторной работы – изучить методы создания и анализа цепей Маркова.

Цепь Маркова – последовательность случайных событий с конечным или счётным числом исходов, характеризующаяся тем свойством, что, говоря нестрого, при фиксированном настоящем будущее независимо от прошлого.

Коэффициенты в матрице переходных вероятностей

Вариант	p_{11}	p_{12}	p_{13}	p_{14}	p_{21}	p_{22}	p_{23}	p_{24}	p_{31}	p_{32}	p_{33}	p_{34}	p_{41}	p_{42}	p_{43}	p_{44}
4	0	3	3	3	3	0	2	1	3	2	0	1	3	2	1	0

Характеристики системы массового обслуживания

Вариант	λ	μ	n	m
4	20	20	4	10

1-4

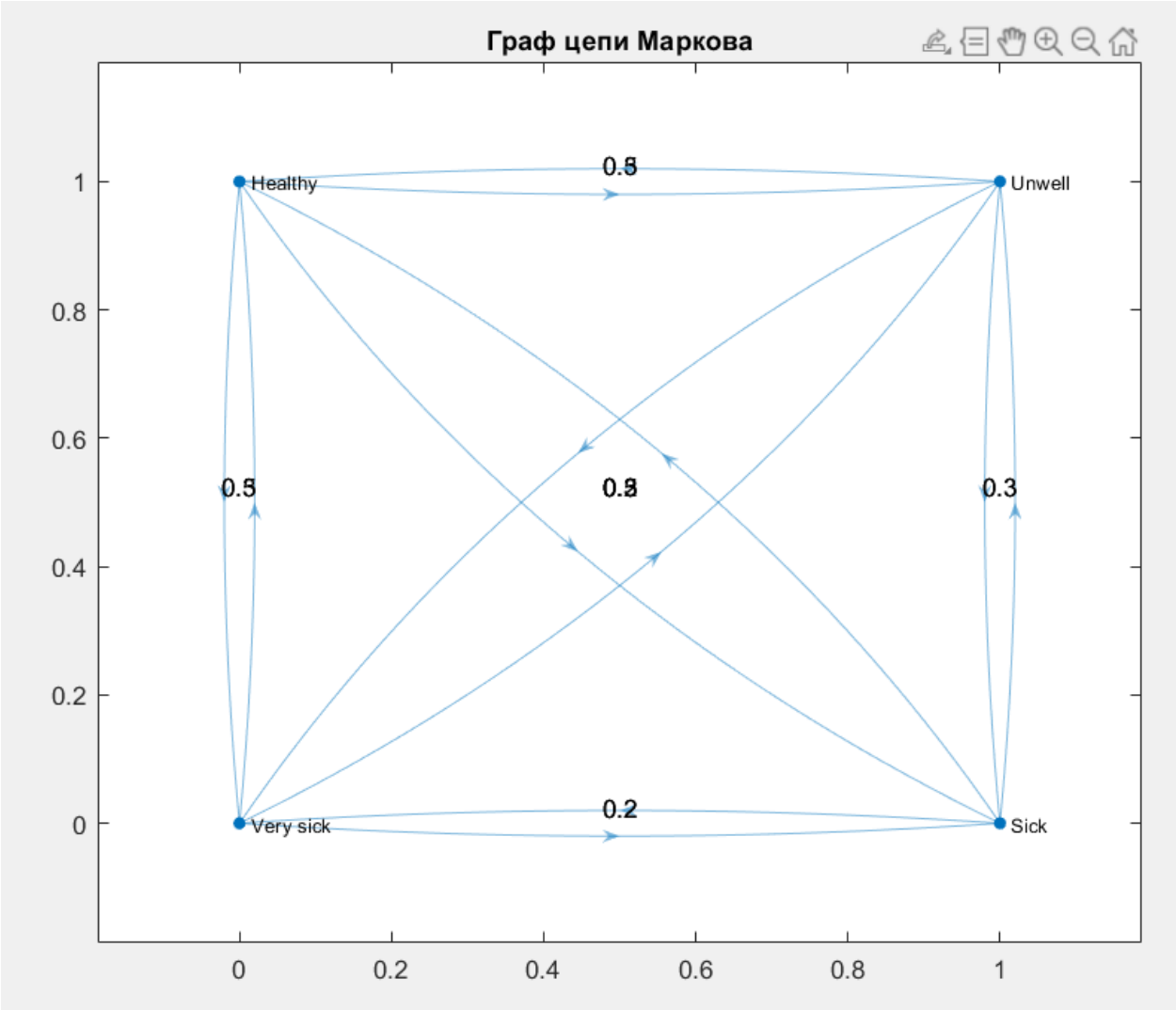
Шаг 1: Мы создали .m файл в MATLAB и определили в нем матрицу переходов P, которая описывает вероятности перехода между состояниями "Healthy", "Unwell", "Sick" и "Very sick".

Шаг 2: Используя функцию dtmc() и матрицу P, мы создали объект цепи Маркова MC, присвоив имена состояниям.

Шаг 3: Далее мы вывели матрицу переходов MC.P в консоль, убедившись, что сумма элементов в каждой строке равна 1, что подтверждает свойство стохастической матрицы.

Шаг 4: Наконец, мы визуализировали граф цепи Маркова с помощью функции plot(), на котором узлы представляют состояния, ребра — возможные переходы, а подписи на ребрах отображают соответствующие вероятности.

```
Матрица переходов:
         0    0.3333    0.3333    0.3333
    0.5000         0    0.3333    0.1667
    0.5000    0.3333         0    0.1667
    0.5000    0.3333    0.1667         0

Суммы строк матрицы переходов:
    1.0000
    1.0000
    1.0000
    1.0000
```



5-6

Шаг 5: На основе нормированной матрицы переходов мы построили кумулятивную матрицу P_cum, где каждое значение в последующем столбце являлось суммой предыдущих значений в той же строке.

Шаг 6: Используя кумулятивную матрицу P_cum и выражение, мы провели симуляцию поведения цепи Маркова на протяжении 200 итераций, определив последовательность состояний, которые посещала система.

$$z_{t+1} = \sum_{k} (r > P\_cum(z_t, k)) + 1,$$

Результат: В результате симуляции мы получили последовательность из 200 чисел, каждое из которых представляет состояние, в котором находилась система на соответствующей итерации.

```
Состояния цепи Маркова:
  Columns 1 through 21

     1     4     2     1     4     2     1     4     3     2     3     2     1     2     1     2     4     1     2     1     2

  Columns 22 through 42

     1     2     4     1     2     4     3     1     2     1     3     2     1     3     2     1     2     1     2     1     3

  Columns 43 through 63

     1     2     3     1     4     2     1     3     1     3     4     2     3     1     3     2     3     1     3     4     1

  Columns 64 through 84

     4     3     2     1     2     1     4     1     4     1     3     1     4     2     4     3     1     4     1     2     3

  Columns 85 through 105

     2     1     4     2     3     4     2     3     1     2     4     1     3     1     4     2     3     1     2     3     1

Columns 106 through 126

    2     3     1     4     2     3     1     3     2     4     2     3     1     3     2     1     2     1     3     2     3

Columns 127 through 147

    1     3     2     1     3     2     1     3     1     4     1     2     1     2     1     3     4     3     1     4     1

Columns 148 through 168

    3     2     4     1     4     1     4     2     3     2     3     1     2     4     1     2     3     4     2     1     3

Columns 169 through 189

    1     4     1     4     2     1     2     1     3     1     3     1     3     2     1     2     3     1     4     3     2

Columns 190 through 200

    1     3     1     4     3     2     1     3     1     3     1
```

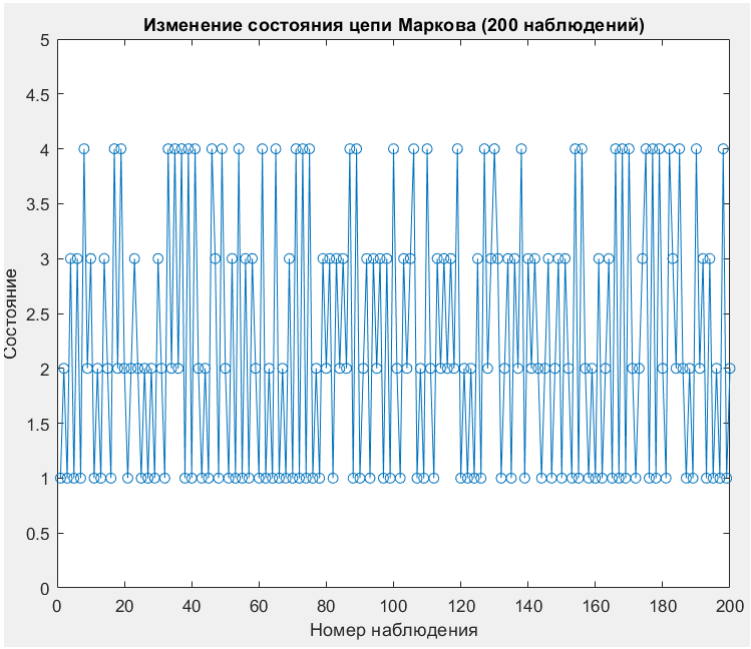
7-8

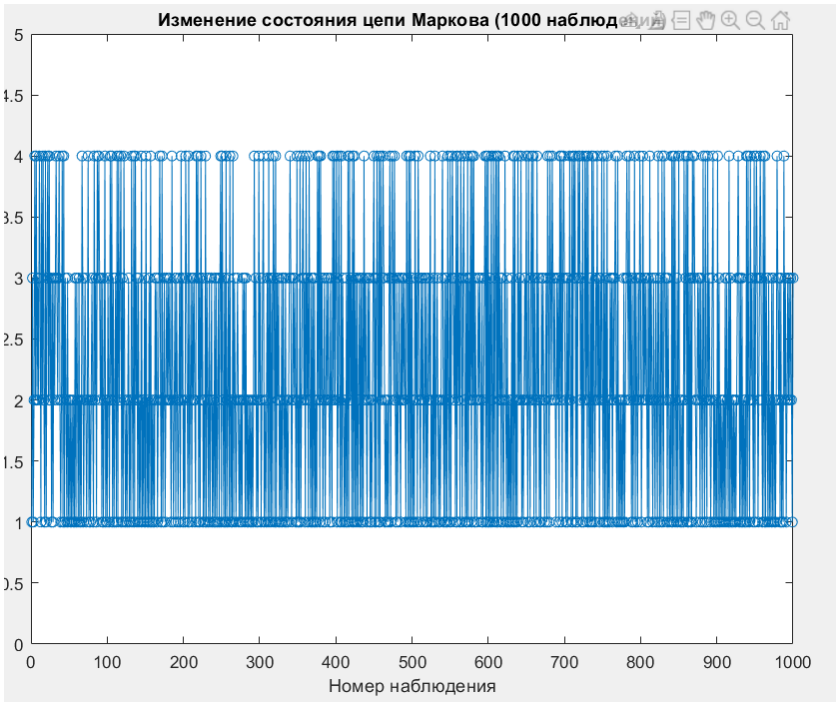
Шаг 7: Мы построили график, используя функцию plot(), для визуализации изменения состояний цепи Маркова в течение 200 итераций симуляции, получив наглядное представление о динамике переходов между состояниями.

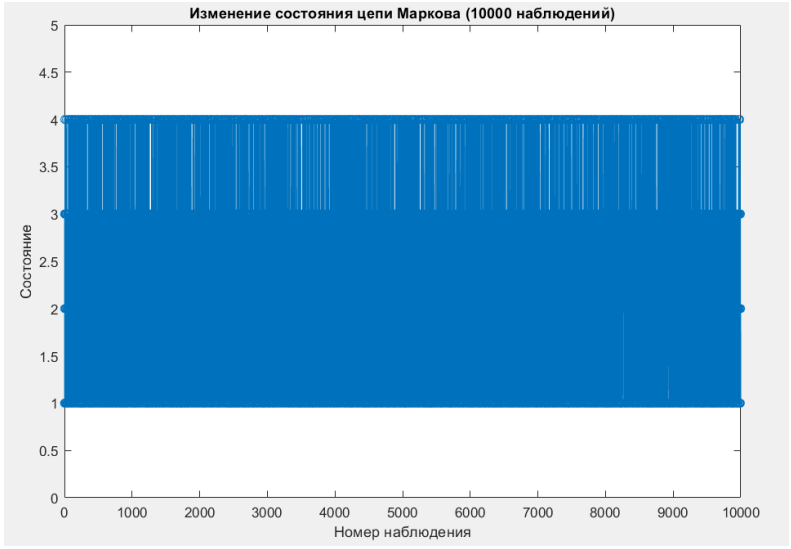
Шаг 8: Далее мы повторили симуляцию (шаг 6) для большего количества итераций: 1000 и 10000.

Анализ: На графиках можно заметить, что с увеличением количества итераций (от 200 до 10000) траектория состояний становится все более плотной, визуально демонстрируя свойство эргодичности цепи Маркова: с течением времени система стремится посещать все доступные состояния, а частота посещения каждого состояния стабилизируется.

Теперь у нас есть наглядное представление о поведении цепи Маркова на различных временных интервалах, что позволяет судить о ее динамике и делать выводы о долгосрочных тенденциях.


Изменение состояния цепи Маркова (200 наблюдений)
Состояние
Номер наблюдения
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
20
40
60
80
100
120
140
160
180
200


Изменение состояния цепи Маркова (1000 наблюдений)
Номер наблюдения
0
100
200
300
400
500
600
700
800
900
1000


Изменение состояния цепи Маркова (10000 наблюдений)
Состояние
Номер наблюдения
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

9-

Шаг 9: Мы рассчитали оценку цепи Маркова, используя смоделированные траектории состояний для 200, 1000 и 10000 итераций. Для этого мы подсчитали частоту переходов между состояниями на основе полученных последовательностей и нормализовали эти значения, чтобы получить оцененные вероятности перехода.

Анализ: Сравнивая оцененные матрицы переходов с теоретической матрицей (из шага 3), можно заметить, что:

- **Сходимость:** С увеличением количества итераций (от 200 до 10000) оцененные матрицы становятся все более похожими на теоретическую. Это подтверждает, что при достаточном количестве наблюдений экспериментальные данные хорошо аппроксимируют истинные вероятности перехода.
- **Влияние случайности:** Оценка для 200 итераций заметно отличается от остальных, что связано с большей ролью случайности при малом объеме данных.
- **Стабилизация:** Оценки для 1000 и 10000 итераций выглядят более стабильными и близкими к теоретической матрице, что говорит о стабилизации оценок с увеличением объема данных.

Мы также визуализировали полученные оценки в виде графов, которые демонстрируют ту же тенденцию сходимости к теоретическому графу (из шага 4) с ростом числа наблюдений.

Этот шаг подчеркивает важность достаточного количества данных для точной оценки параметров модели, а также демонстрирует на практике свойство сходимости цепей Маркова.

```
Матрица переходов (200 итераций):
         0    0.3438    0.3281    0.3281
    0.4130         0    0.4130    0.1739
    0.5400    0.2600         0    0.2000
    0.4615    0.2821    0.2564         0

Суммы строк:
     1
     1
     1
     1
```

Матрица переходов (1000 итераций):

0	0.3795	0.3735	0.2470
0.4846	0	0.3462	0.1692
0.5187	0.3154	0	0.1660
0.4880	0.3494	0.1627	0

Суммы строк:

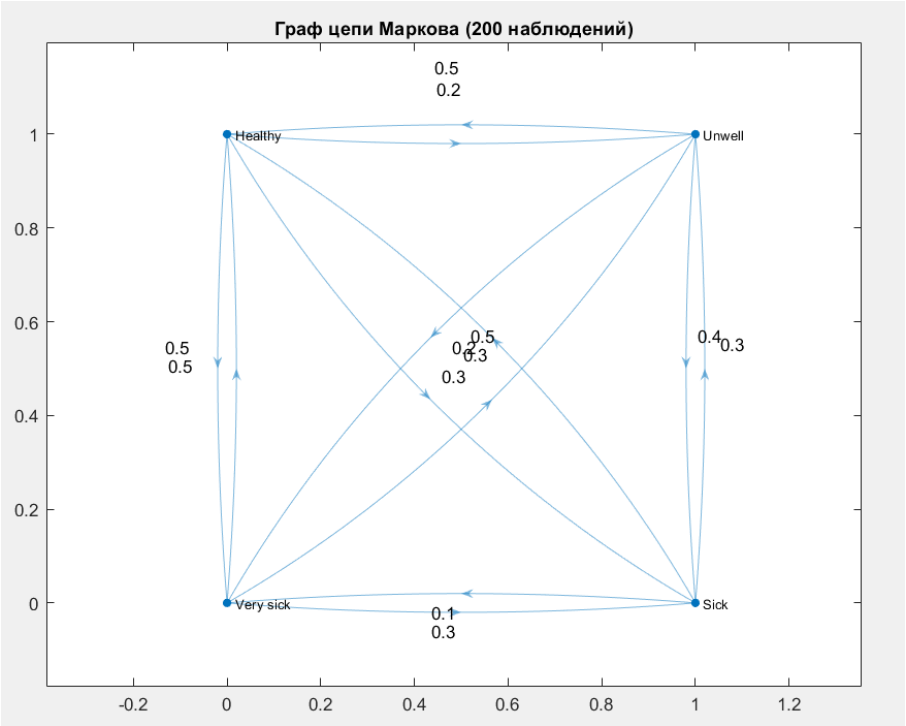
1
1
1
1

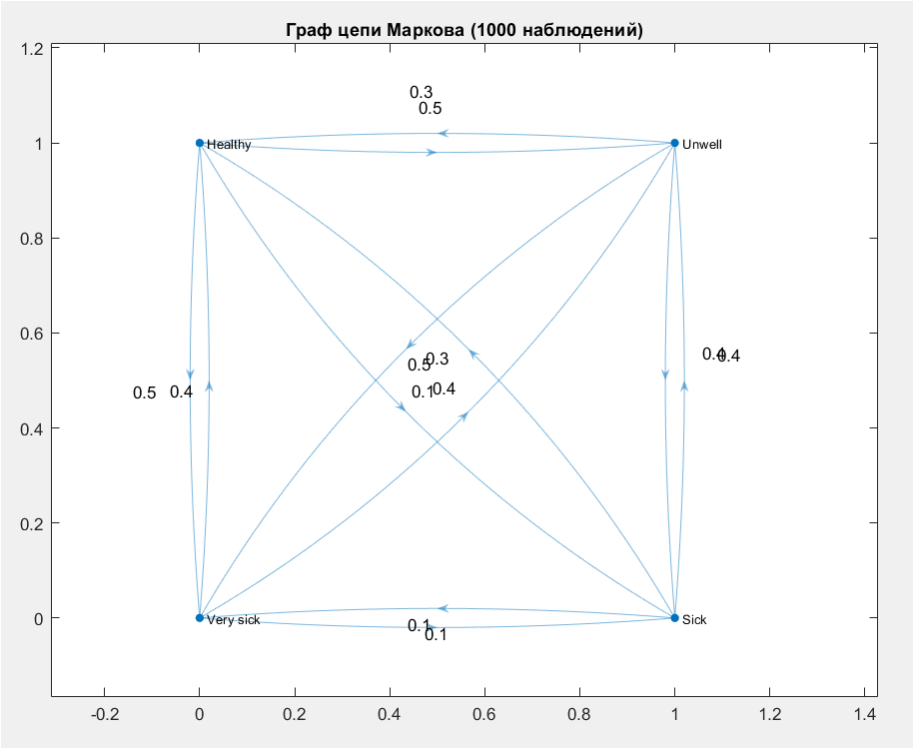
Матрица переходов (10000 итераций):

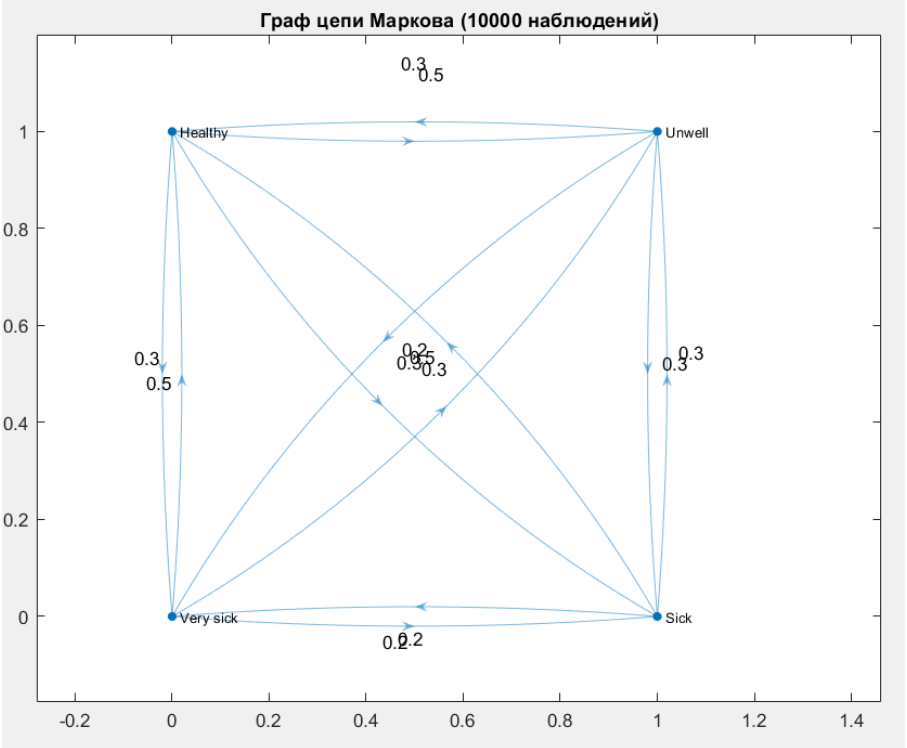
0	0.3308	0.3201	0.3492
0.5012	0	0.3475	0.1513
0.5196	0.3258	0	0.1547
0.5053	0.3316	0.1632	0

Суммы строк:

1
1
1
1







10-Глядя на скриншоты графов цепей Маркова из пунктов 4 и 9, можно сделать следующие выводы:

1. **Сходимость к теоретической матрице:** С увеличением числа наблюдений (200, 1000, 10000) оцененная матрица переходов, полученная в пункте 9, становится все более похожей на теоретическую матрицу из пункта 4. Это демонстрирует свойство сходимости цепей Маркова: при достаточном количестве наблюдений экспериментальные данные начинают хорошо аппроксимировать теоретические вероятности.

2. **Влияние случайности при малом числе наблюдений:**

- Граф для 200 наблюдений значительно отличается от остальных. Это связано с тем, что при малом числе наблюдений на оценку матрицы переходов сильнее влияет случайность.
- Некоторые переходы, имеющие ненулевую вероятность в теоретической матрице, могут вообще не наблюдаться в симуляции с 200 наблюдениями (например, переход из "Unwell" в "Very sick").
- Наоборот, некоторые переходы могут встречаться чаще, чем ожидается, из-за случайных флуктуаций.

3. **Стабилизация при большом числе наблюдений:**
   - При 1000 и 10000 наблюдениях графики становятся более стабильными и похожими друг на друга и на теоретический граф.
   - Вероятности на ребрах становятся ближе к теоретическим значениям.

Общий вывод: Чем больше количество наблюдений, тем точнее оценка цепи Маркова. При малом числе наблюдений оценка может быть неточной из-за случайных флуктуаций, но с ростом количества данных она сходится к теоретической матрице переходов, отражая истинное поведение системы.

11-

Шаг 11: Мы вернулись к симуляции цепи Маркова (шаг 6), но на этот раз использовали не теоретическую матрицу переходов, а оцененную матрицу, полученную на основе 200 наблюдений (из шага 9). Провели симуляцию на 200 итераций и построили график изменения.

Анализ:

Сравнивая график, построенный на основе оцененной матрицы (шаг 11), с графиком, построенным на основе теоретической матрицы (шаг 7, 200 наблюдений), можно заметить как сходства, так и различия.

Сходства:

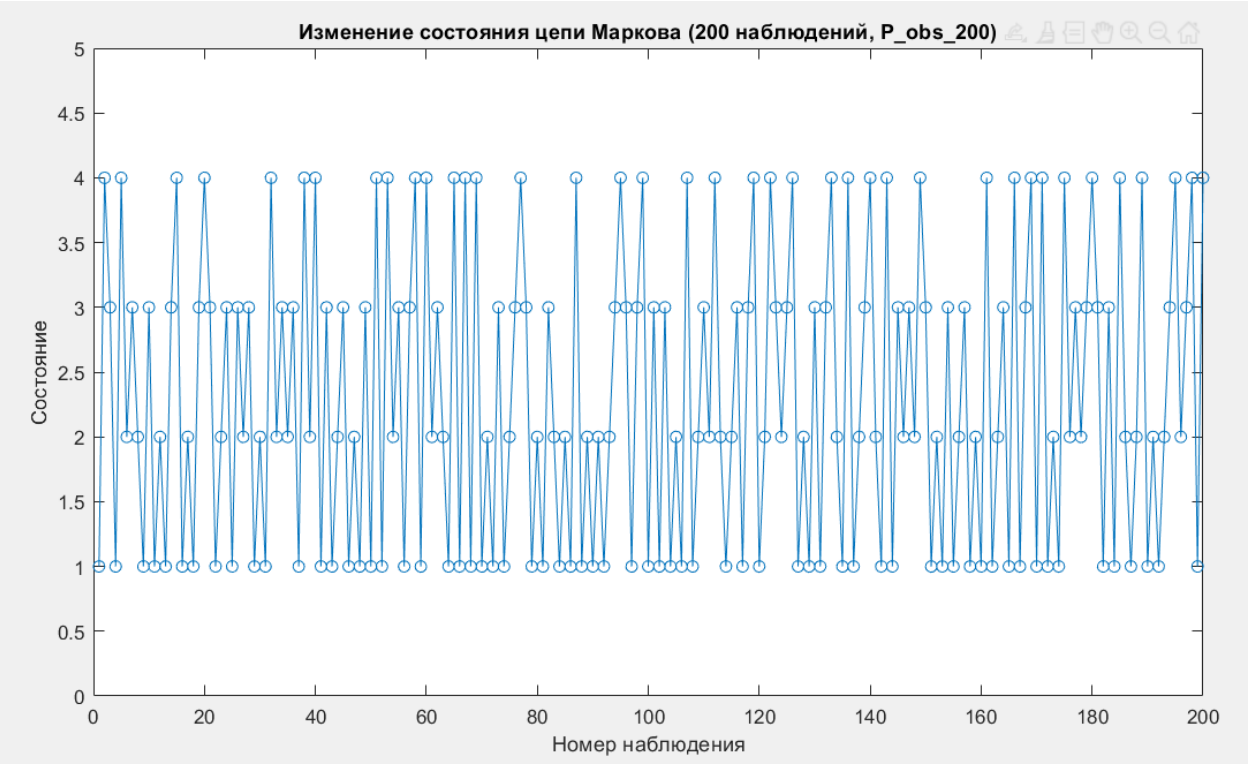
- Оба графика демонстрируют частые переходы между состояниями, что характерно для данной цепи Маркова.
- В обоих случаях система не застревает надолго в одном состоянии.

Различия:

- Могут наблюдаться отличия в частоте посещения отдельных состояний. Например, на графике, построенном на основе оцененной матрицы, какое-то состояние может встречаться чаще, чем на графике, построенном на основе теоретической матрицы.

Выводы:

- Симуляция с использованием оцененной матрицы может давать несколько иные результаты по сравнению с симуляцией, основанной на теоретической матрице, особенно при небольшом количестве наблюдений, использованных для оценки.
- Это связано с тем, что оценка матрицы, полученная на основе ограниченного числа наблюдений, не идеально отражает истинные вероятности перехода.
- С ростом количества наблюдений, использованных для оценки матрицы, различия между симуляциями должны уменьшаться.



12-

Pokazatel	Value
{'lambda'}	20
{'mu' }	20
{'n' }	4
{'m' }	10
{'Ro' }	1
{'P0' }	0.36181
{'Potk' }	1.4377e-08
{'Q' }	1
{'A' }	20
{'kzan' }	1
{'Loch' }	0.0024242
{'Toch' }	0.00012121
{'Lsist' }	1.0024
{'Tsist' }	0.050121

Система работоспособна (ro < n)

Интерпретация:

- **Работоспособность:** Система работоспособна, если ro < n.

- **Пропускная способность:** A и Q показывают реальную производительность системы.
- **Очереди:** Loч и Tоч характеризуют задержки в обслуживании.
- **Загрузка:** kзан показывает среднее количество занятых каналов.

Результаты выглядят вполне правдоподобно для СМО с заданными параметрами (λ=20, μ=20, n=4, m=10). Давайте проанализируем некоторые ключевые показатели:

- **Ro (коэффициент загрузки) = 1**: Это означает, что система загружена на 100%. В среднем, заявки поступают с той же интенсивностью, с которой обслуживаются.
- **P0 (вероятность простоя системы) = 0.36181**: Это довольно высокая вероятность простоя, учитывая 100% загрузку. Скорее всего, это связано с ограничением на размер очереди (m=10). Когда очередь заполняется, система начинает отказывать новым заявкам.
- **Potk (вероятность отказа) = 1.4377e-08**: Крайне низкая вероятность отказа подтверждает предположение о том, что система практически не отказывает заявкам из-за занятости всех каналов, а скорее из-за заполнения очереди.
- **Lsist (среднее количество заявок в системе) = 1.0024**: В системе находится в среднем чуть больше одной заявки.
- **Loch (средняя длина очереди) = 0.0024242**: В среднем очередь очень короткая, что говорит о том, что заявки обслуживаются достаточно быстро.
- **Tsist (среднее время пребывания заявки в системе) = 0.050121**: Заявка проводит в системе в среднем 0.05 единиц времени (скорее всего, часов, если интенсивность измеряется в заявках в час).

Полученные результаты указывают на то, что СМО работает достаточно эффективно:

- Низкая вероятность отказа.
- Небольшие очереди.
- Относительно короткое время пребывания заявки в системе.

Однако высокая вероятность простоя системы (P0) при 100% загрузке (Ro) может свидетельствовать о некотором дисбалансе. Возможно, стоит рассмотреть увеличение числа каналов обслуживания (n) или размера очереди (m) для повышения пропускной способности системы, если это допустимо с точки зрения затрат и ограничений.

Заключение:

В ходе лабораторной работы мы успешно освоили основы моделирования дискретных цепей Маркова в среде MATLAB. Мы научились создавать модели на основе матрицы переходов, визуализировать их структуру, симулировать траектории состояний и оценивать параметры модели на основе экспериментальных данных.

Дополнительные задания:

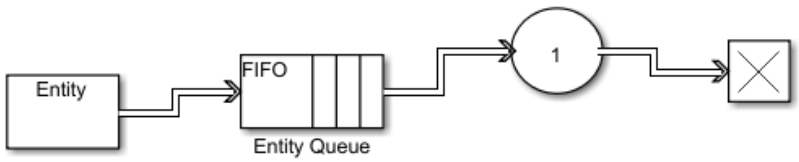
1-

```
import numpy as np


# Определение матрицы переходов
P = np.array([
    [0.2, 0.7, 0.1],  # Состояние 1
    [0.1, 0.6, 0.3],  # Состояние 2
```

```
    [0.5, 0.3, 0.2]   # Состояние 3
])

# Начальное состояние
initial_state = 0  # Индекс начального состояния (от 0)

# Количество шагов симуляции
num_steps = 10

# Симуляция цепи Маркова
current_state = initial_state
states = [current_state]  # Список для хранения посещенных состояний

for _ in range(num_steps):
    # Выбор следующего состояния на основе матрицы переходов
    next_state = np.random.choice(len(P), p=P[current_state])

    # Обновление текущего состояния и добавление в список
    current_state = next_state
    states.append(current_state)

# Вывод результата
print("Посещенные состояния:", states)
```

Вывод: Посещенные состояния: [0, 1, 2, 0, 1, 1, 1, 1, 0, 0, 1]

2-



Блоки:

1. **Entity Generator (Сущность):**
    - *Функция:* Создает заявки (транзакции, клиенты и т.д.), которые поступают в СМО.
    - *Настройка:*
        - **Interarrival time** (Время между прибытиями): задает, как часто появляются заявки (экспоненциальное, детерминированное распределение и т.д.)
2. **FIFO Entity Queue (Очередь FIFO):**
    - *Функция:* Хранит заявки, ожидающие обслуживания, в порядке их поступления (First-In, First-Out).
    - *Настройка:*

        - **Queue capacity** (Емкость очереди): максимальное число заявок, которые могут одновременно находиться в очереди (бесконечная или ограниченная).

3. **Entity Server (Сервер):**
    - *Функция:* Обрабатывает (обслуживает) заявки, поступающие из очереди.
    - *Настройка:*
        - **Service time** (Время обслуживания): задает, сколько времени занимает обработка одной заявки (экспоненциальное, нормальное распределение и т.д.)
4. **Entity Terminator (Терминатор):**
    - *Функция:* Удаляет заявки из модели после их обслуживания.

Приложение:

Sibsutis/3st Year/5th semester/TMO/3lab at main · kibatora/Sibsutis (github.com)