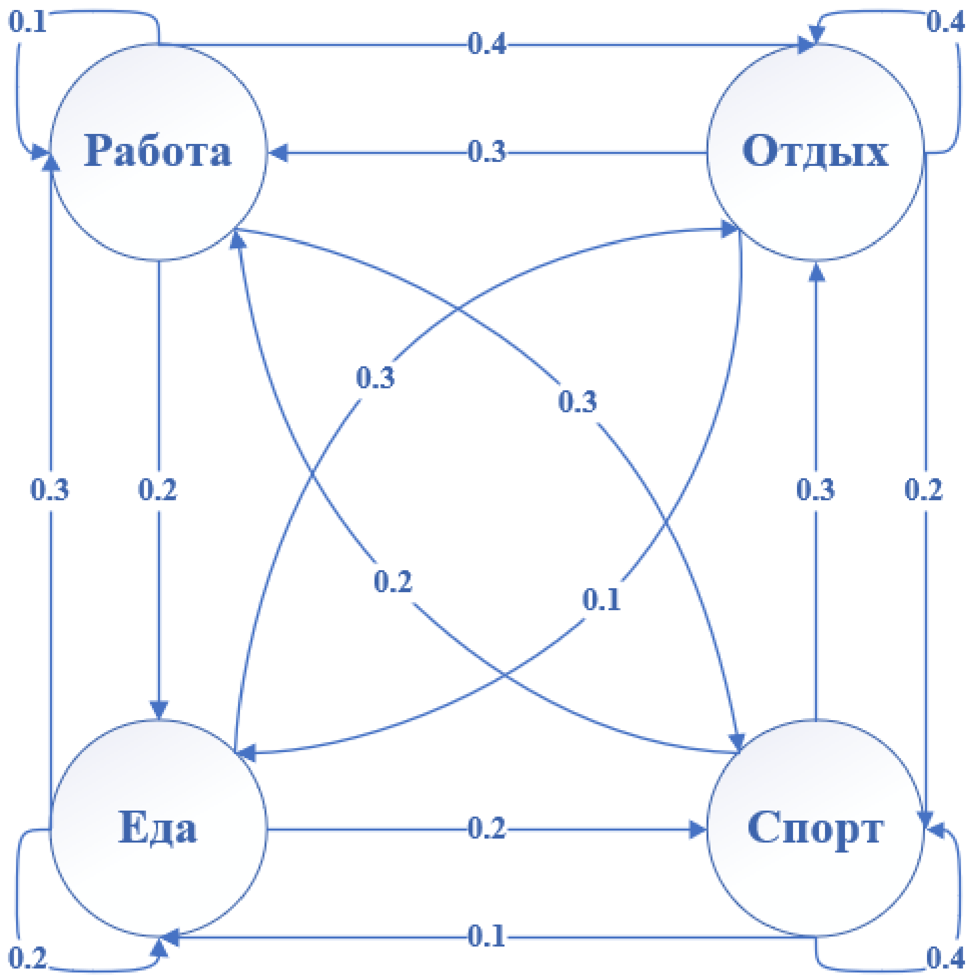
Марковские цепи

1. Придумать эргодическую марковскую цепь, содержащую не менее 4-х состояний

Для моделирования была выбрана марковская цепь с четырьмя состояниями: "Работа", "Отдых", "Спорт", "Еда". Переходы между этими состояниями заданы матрицей переходных вероятностей.

2. Диаграмма переходов и матрица переходных вероятностей

Диаграмма переходов (граф) изображена на следующем рисунке:



Матрица переходных вероятностей имеет следующий вид:

$$P = \begin{pmatrix} 0.1 & 0.4 & 0.3 & 0.2 \\ 0.3 & 0.4 & 0.2 & 0.1 \\ 0.2 & 0.3 & 0.4 & 0.1 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix}$$

3. Реализация марковой цепи и построение графиков изменения компонентов финальных векторов.

Моделирование Марковской цепи пошагово с различными начальными векторами вероятностей состояний

Были проведены симуляции для разных начальных состояний:

1. Начальное состояние: "Отдых"
2. Начальное состояние: "Работа"
3. Начальное состояние: "Спорт"
4. Начальное состояние: "Еда"

Из диаграммы состояний видно, что все состояния имеют ненулевые вероятности перехода в другие состояния. Это предполагает, что марковская цепь может быть неприводимой, так как существует вероятность перейти из любого состояния в любое другое. Также в диаграмме не видны явные циклы с периодом больше единицы, что делает возможным апериодичность цепи. Однако, для точного утверждения о неприводимости и апериодичности требуется дополнительный анализ, которым мы еще займемся. Если цепь неприводима и апериодична, и если каждое состояние возвратно с конечным средним временем возврата, то представленная марковская цепь является эргодической.

Код реализации:

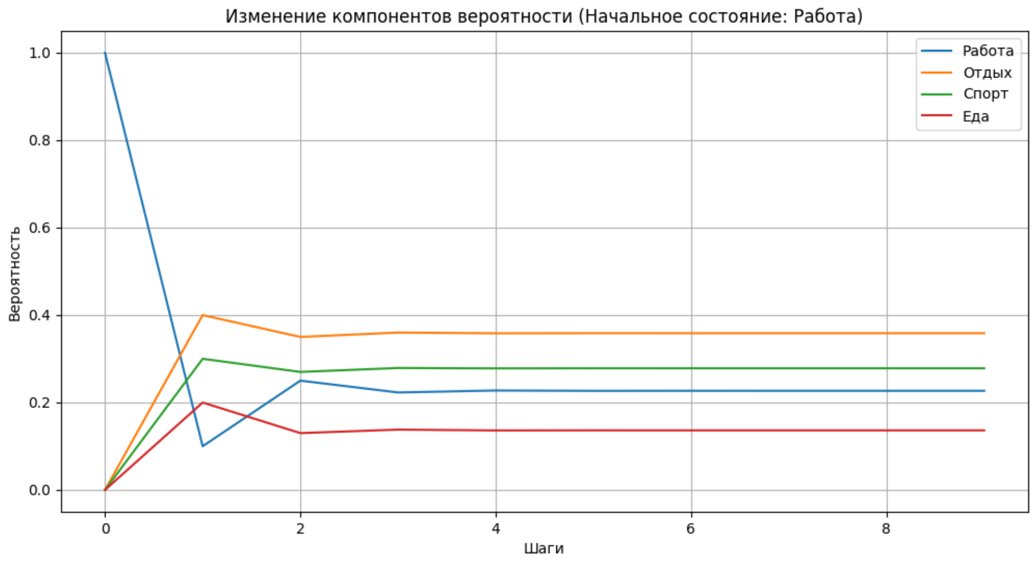
```
import numpy as np
import matplotlib.pyplot as plt

# Пространство состояний
states = ["Работа", "Отдых", "Спорт", "Еда"]
# Матрица вероятностей (матрица переходов) из main.py
trans_matrix = np.array([
    [0.1, 0.4, 0.3, 0.2],   # Работа -> Работа, Отдых, Спорт, Еда
    [0.3, 0.4, 0.2, 0.1],   # Отдых -> Работа, Отдых, Спорт, Еда
    [0.2, 0.3, 0.4, 0.1],   # Спорт -> Работа, Отдых, Спорт, Еда
    [0.3, 0.3, 0.2, 0.2]    # Еда -> Работа, Отдых, Спорт, Еда
])

# Начальные вектора вероятностей для каждого состояния
init_prob_vectors = [
    np.array([1.0, 0.0, 0.0, 0.0]),   # Начало в "Работа"
    np.array([0.0, 1.0, 0.0, 0.0]),   # Начало в "Отдых"
    np.array([0.0, 0.0, 1.0, 0.0]),   # Начало в "Спорт"
    np.array([0.0, 0.0, 0.0, 1.0]),   # Начало в "Еда"
```

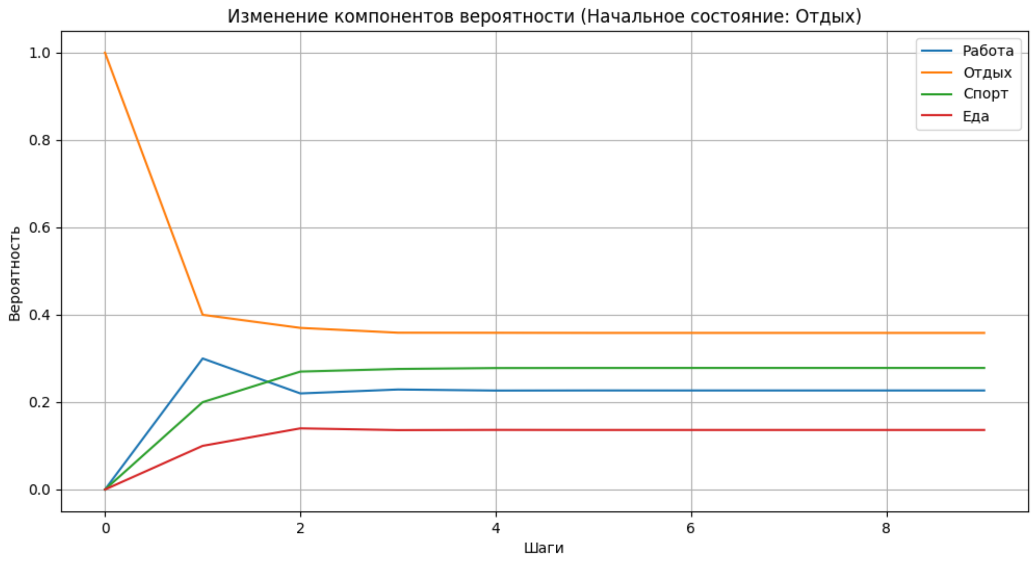
```python
    np.array([0.25, 0.25, 0.25, 0.25])  # Равномерное начальное распределение
]

def markov_step(state_prob, trans_matrix):
    return state_prob @ trans_matrix

def simulate_chain(init_prob, trans_matrix, steps, epsilon=1e-6):
    current_prob = init_prob
    history = [current_prob]
    for _ in range(steps):
        next_prob = markov_step(current_prob, trans_matrix)
        history.append(next_prob)
        if np.linalg.norm(next_prob - current_prob) < epsilon:
            break
        current_prob = next_prob
    return history


# Моделирование марковской цепи
steps = 50
chains = [simulate_chain(vec, trans_matrix, steps) for vec in
init_prob_vectors]

# Вывод результатов моделирования
for i, chain in enumerate(chains):
    init_state = states[i] if i < len(states) else 'Равномерное
распределение'
    print(f"Начальное состояние: {init_state}")
    print(f"Возможные состояния:\n{' -> '.join(states)}")
    print("Результаты моделирования:")

    for step, vec in enumerate(chain):
        print(f"Шаг {step}: {', '.join([f'{p:.4f}' for p in vec])}")

    final_state = states[np.argmax(chain[-1])]
    print(f"Конечное состояние через {len(chain) - 1} шагов: {final_state}")
    print(f"Последовательность вероятностей на последнем шаге: {chain[-1]}")
    print("-" * 50)
```

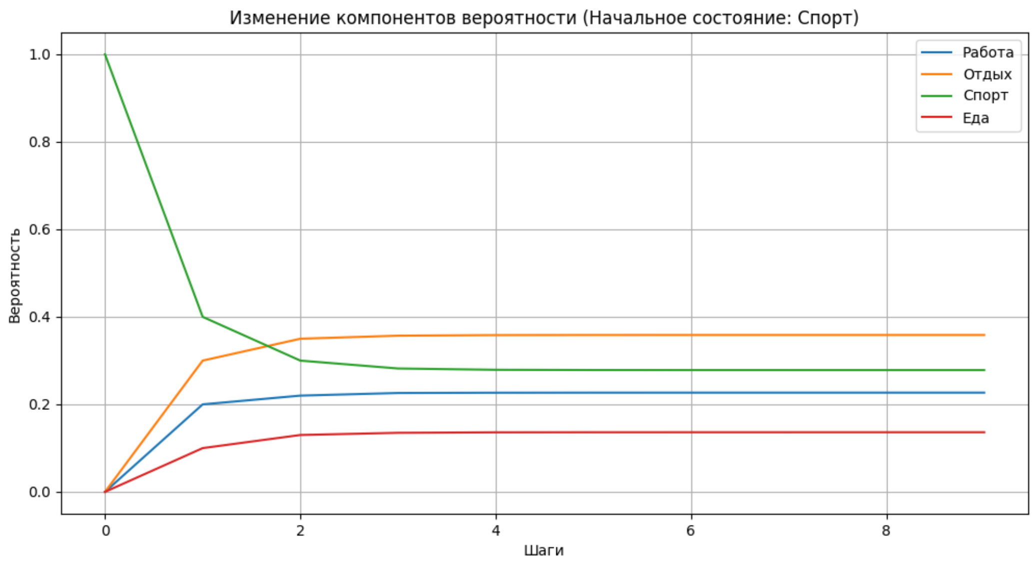
```
Начальное состояние: Работа
Возможные состояния:
Работа -> Отдых -> Спорт -> Еда
Результаты моделирования:
Шаг 0: 1.0000, 0.0000, 0.0000, 0.0000
Шаг 1: 0.1000, 0.4000, 0.3000, 0.2000
Шаг 2: 0.2500, 0.3500, 0.2700, 0.1300
Шаг 3: 0.2230, 0.3600, 0.2790, 0.1380
Шаг 4: 0.2275, 0.3583, 0.2781, 0.1361
Шаг 5: 0.2267, 0.3586, 0.2784, 0.1364
Шаг 6: 0.2268, 0.3585, 0.2783, 0.1363
Шаг 7: 0.2268, 0.3585, 0.2784, 0.1363
Шаг 8: 0.2268, 0.3585, 0.2784, 0.1363
Шаг 9: 0.2268, 0.3585, 0.2784, 0.1363
Конечное состояние через 9 шагов: Отдых
Последовательность вероятностей на последнем шаге: [0.22680402 0.35853383 0.27835053 0.13631161]
```

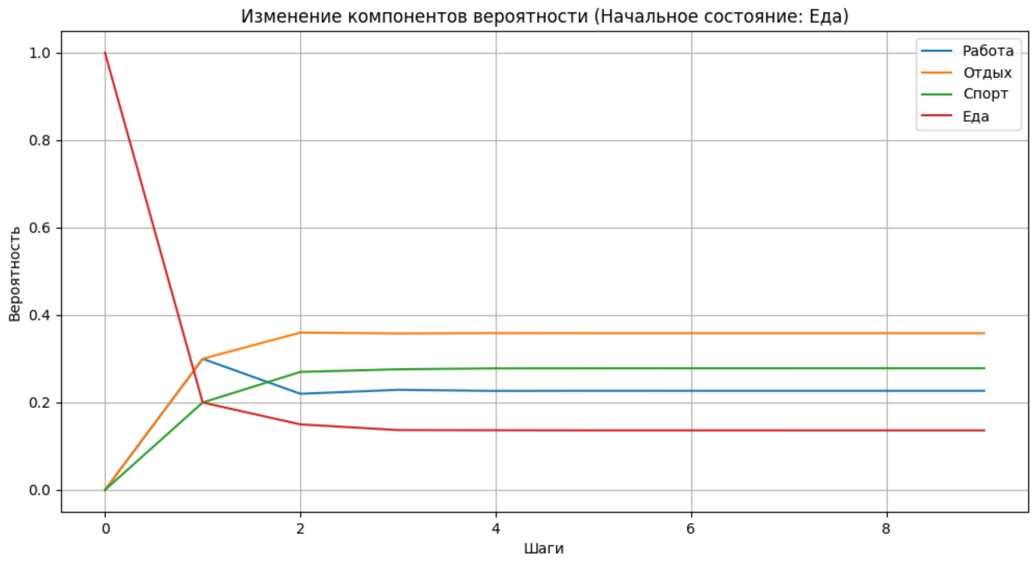
```
Начальное состояние: Отдых
Возможные состояния:
Работа -> Отдых -> Спорт -> Еда
Результаты моделирования:
Шаг 0: 0.0000, 1.0000, 0.0000, 0.0000
Шаг 1: 0.3000, 0.4000, 0.2000, 0.1000
Шаг 2: 0.2200, 0.3700, 0.2700, 0.1400
Шаг 3: 0.2290, 0.3590, 0.2760, 0.1360
Шаг 4: 0.2266, 0.3588, 0.2781, 0.1365
Шаг 5: 0.2269, 0.3585, 0.2783, 0.1363
Шаг 6: 0.2268, 0.3585, 0.2783, 0.1363
Шаг 7: 0.2268, 0.3585, 0.2783, 0.1363
Шаг 8: 0.2268, 0.3585, 0.2784, 0.1363
Шаг 9: 0.2268, 0.3585, 0.2784, 0.1363
Конечное состояние через 9 шагов: Отдых
Последовательность вероятностей на последнем шаге: [0.22680418 0.35853379 0.27835045 0.13631157]
```

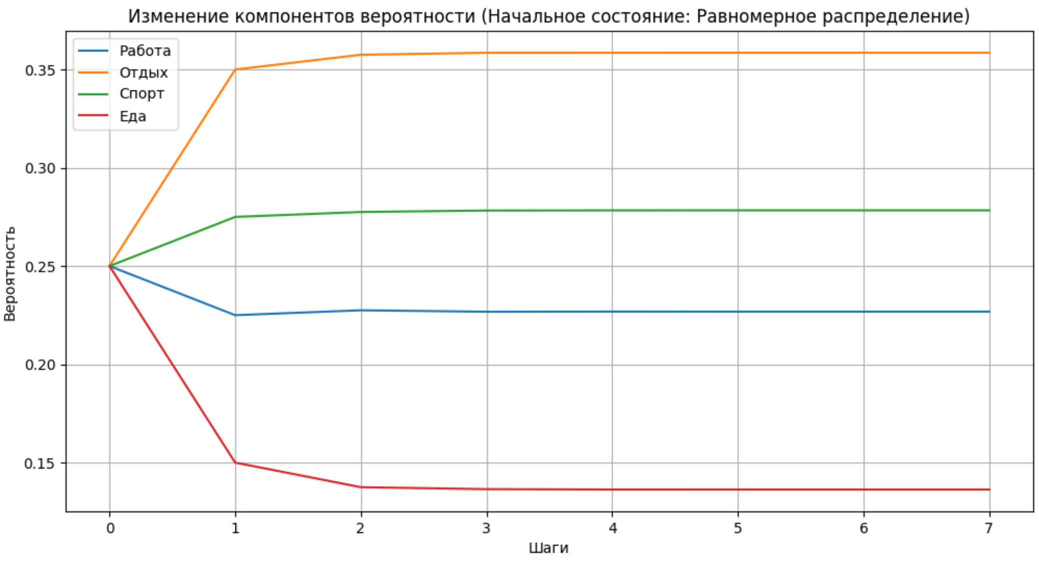
```
def plot_chains(chains, states, init_states):
    for i, chain in enumerate(chains):
        chain = np.array(chain)
        plt.figure(figsize=(12, 6))
        for j, state in enumerate(states):
            plt.plot(range(len(chain)), chain[:, j], label=f'{state}')
        plt.title(f'Изменение компонентов вероятности (Начальное состояние: 
{init_states[i]})')
        plt.xlabel('Шаги')
        plt.ylabel('Вероятность')
        plt.legend()
        plt.grid(True)
        plt.show()

def plot_rmse(chains, init_states):
    for i, chain in enumerate(chains):
        rmse = []
        for j in range(1, len(chain)):
            rmse.append(np.sqrt(np.mean((chain[j] - chain[j-1])**2)))
        plt.figure(figsize=(12, 6))
        plt.plot(range(1, len(chain)), rmse, label=f'Начальное состояние: 
{init_states[i]}')
        plt.title(f'Изменение среднеквадратического отклонения (Начальное 
состояние: {init_states[i]})')
        plt.xlabel('Шаги')
        plt.ylabel('RMSE')
        plt.legend()
        plt.grid(True)
        plt.show()


# Пример использования функций для визуализации:
init_states = ["Работа", "Отдых", "Спорт", "Еда", "Равномерное 
распределение"]

plot_chains(chains, states, init_states)
plot_rmse(chains, init_states)
```

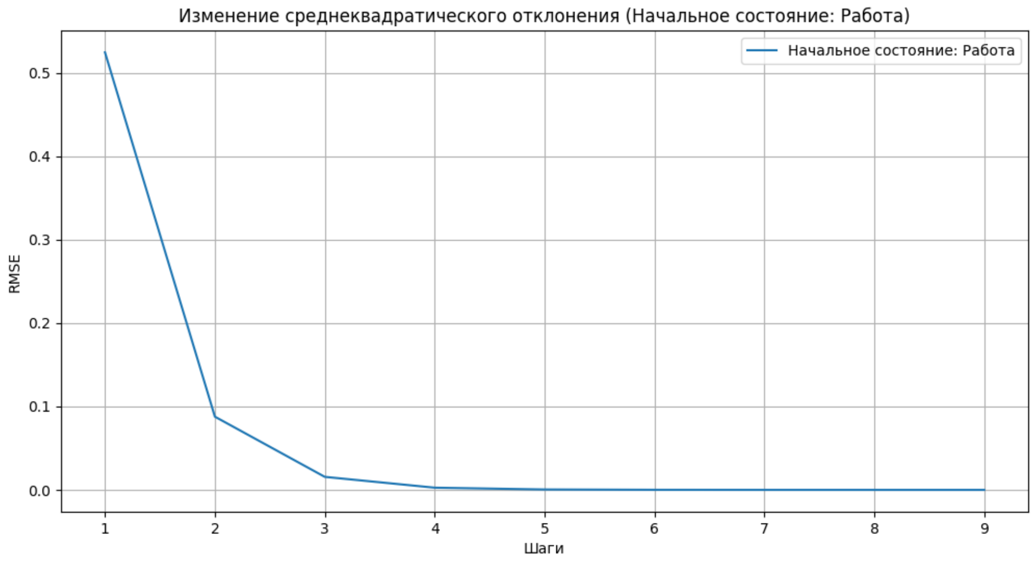

Изменение компонентов вероятности (Начальное состояние: Работа)
Работа
Отдых
Спорт
Еда
Вероятность
Шаги
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8

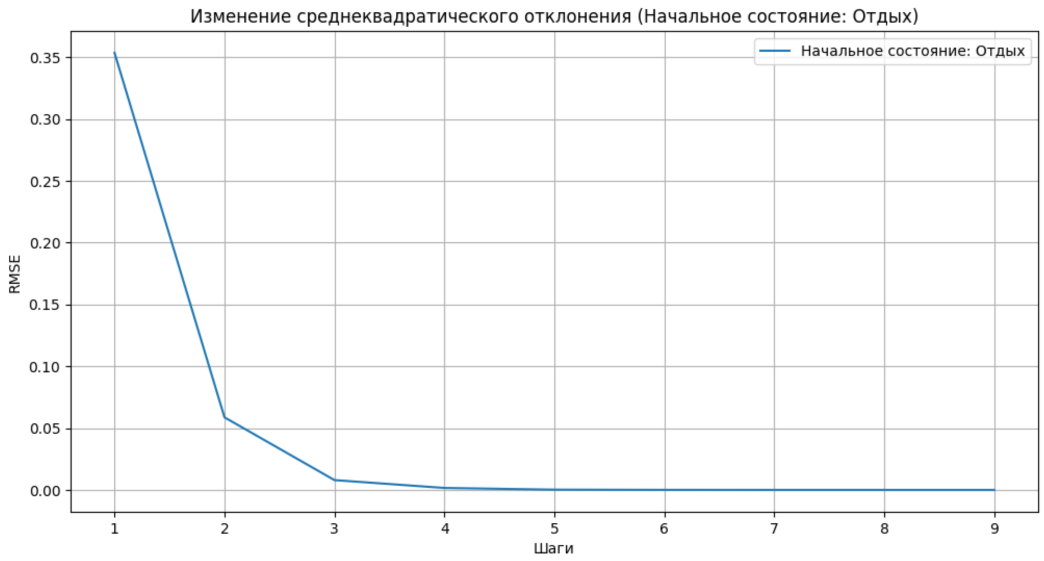

Изменение компонентов вероятности (Начальное состояние: Отдых)
Работа
Отдых
Спорт
Еда
Вероятность
Шаги
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8

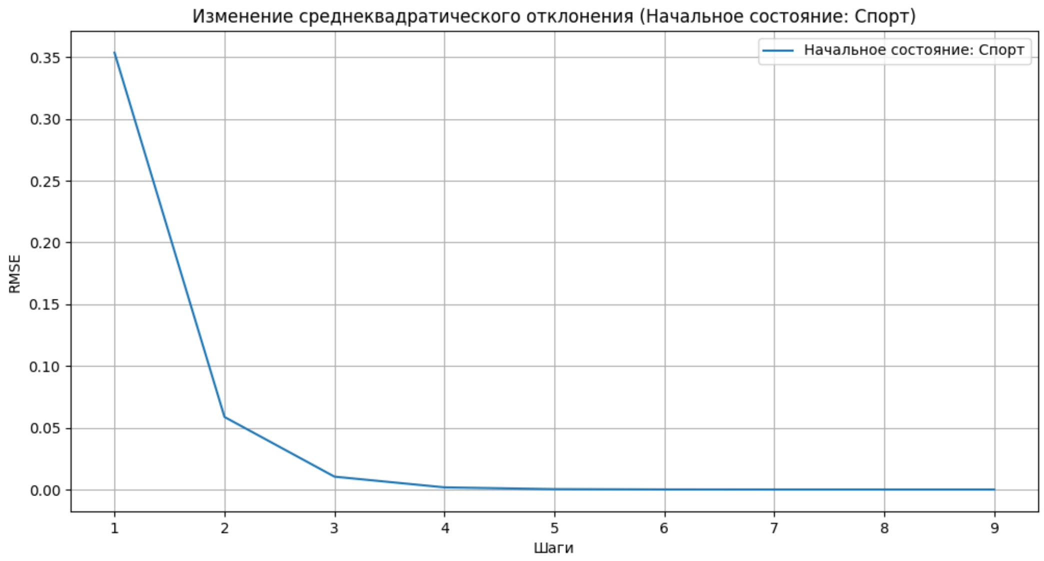

Изменение компонентов вероятности (Начальное состояние: Спорт)
Работа
Отдых
Спорт
Еда
Вероятность
Шаги
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8

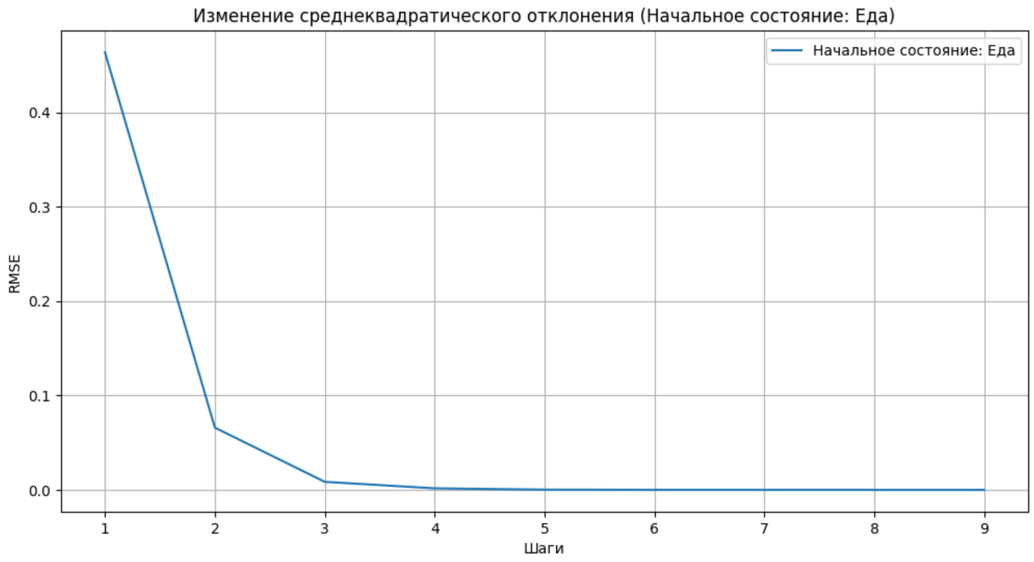

Изменение компонентов вероятности (Начальное состояние: Еда)
Работа
Отдых
Спорт
Еда
Вероятность
Шаги
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8

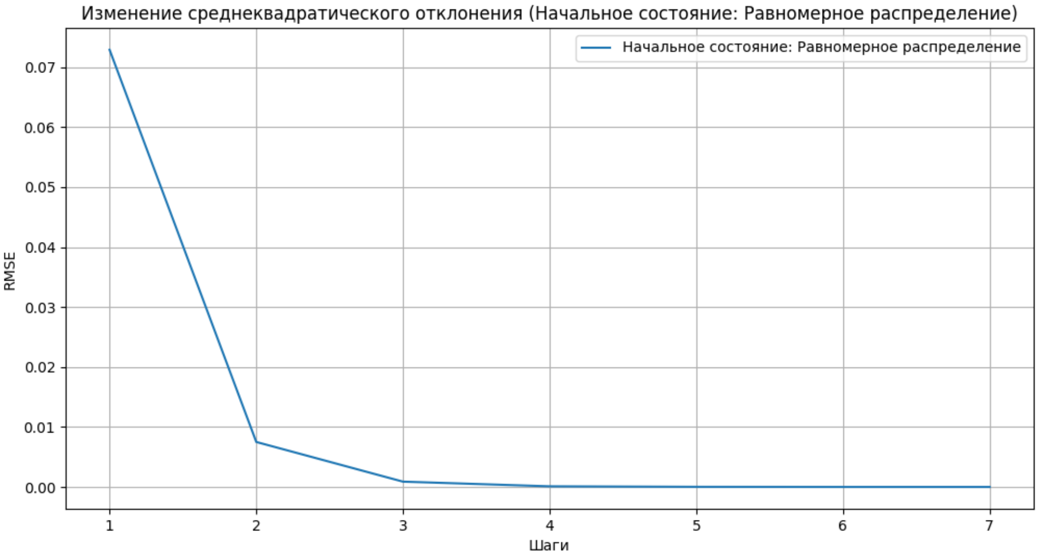

Изменение компонентов вероятности (Начальное состояние: Равномерное распределение)
Работа
Отдых
Спорт
Еда
Вероятность
Шаги
0.35
0.30
0.25
0.20
0.15
0
1
2
3
4
5
6
7


Изменение среднеквадратического отклонения (Начальное состояние: Работа)
Начальное состояние: Работа
RMSE
Шаги
0.5
0.4
0.3
0.2
0.1
0.0
1
2
3
4
5
6
7
8
9


Изменение среднеквадратического отклонения (Начальное состояние: Отдых)
Начальное состояние: Отдых
RMSE
Шаги
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
1
2
3
4
5
6
7
8
9


Изменение среднеквадратического отклонения (Начальное состояние: Спорт)
Начальное состояние: Спорт
RMSE
Шаги
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
1
2
3
4
5
6
7
8
9


Изменение среднеквадратического отклонения (Начальное состояние: Еда)
Начальное состояние: Еда
RMSE
Шаги
0.0
0.1
0.2
0.3
0.4
1
2
3
4
5
6
7
8
9



Поиск стационарного распределения аналитически:

```
#Аналитическое решение
def find_stationary_distribution(transition_matrix):
    n = len(transition_matrix)
    #  (P^T - I) * pi = 0
    A = np.transpose(transition_matrix) - np.eye(n)
    A = np.vstack([A, np.ones(n)])
    b = np.zeros(n + 1)
    b[-1] = 1
    # lsm
    pi, residuals, rank, s = np.linalg.lstsq(A, b, rcond=None)
    return pi


stationary_distribution = find_stationary_distribution(trans_matrix)
print("Стационарное распределение:", stationary_distribution)
sum = 0
for i in stationary_distribution:
    sum += i
print("sum = ", sum)
```

```
--------------------------------------------------
Стационарное распределение: [0.22680412 0.35853379 0.27835052 0.13631157]
sum =  1.0
```

Сравнение вектора из пункта 3 и вектор, рассчитанный аналитически:

```python
#Сравнение финальные распределения цепей со стационарным
def compare_distributions(chains, stationary_dist, init_states):
    """Сравнивает финальные распределения цепей со стационарным
распределением."""
    for i, chain in enumerate(chains):
        final_dist = chain[-1]
        print(f"Начальное состояние: {init_states[i]}")
        print(f"Финальное распределение: {final_dist}")
        print(f"Стационарное распределение: {stationary_dist}")
        print(f"Абсолютная разница: {np.abs(final_dist -
stationary_dist)}\n")

# Пример использования:
compare_distributions(chains, stationary_distribution, init_states)
```

```
Начальное состояние: Работа
Финальное распределение: [0.22680402 0.35853383 0.27835053 0.13631161]
Стационарное распределение: [0.22680412 0.35853379 0.27835052 0.13631157]
Абсолютная разница: [1.02711340e-07 4.24765177e-08 1.75360825e-08 4.26987401e-08]

Начальное состояние: Отдых
Финальное распределение: [0.22680418 0.35853379 0.27835045 0.13631157]
Стационарное распределение: [0.22680412 0.35853379 0.27835052 0.13631157]
Абсолютная разница: [5.92886598e-08 2.47651771e-09 6.34639176e-08 1.69874009e-09]

Начальное состояние: Спорт
Финальное распределение: [0.2268041  0.35853375 0.27835061 0.13631153]
Стационарное распределение: [0.22680412 0.35853379 0.27835052 0.13631157]
Абсолютная разница: [2.17113401e-08 3.85234822e-08 9.85360825e-08 3.83012599e-08]

Начальное состояние: Еда
Финальное распределение: [0.22680418 0.35853379 0.27835045 0.13631157]
Стационарное распределение: [0.22680412 0.35853379 0.27835052 0.13631157]
Абсолютная разница: [5.92886599e-08 1.47651780e-09 6.34639175e-08 2.69874012e-09]

Начальное состояние: Равномерное распределение
Финальное распределение: [0.22680408 0.35853385 0.27835043 0.13631165]
Стационарное распределение: [0.22680412 0.35853379 0.27835052 0.13631157]
Абсолютная разница: [4.87113401e-08 5.84765178e-08 9.04639175e-08 8.06987401e-08]
```

Выводы

Полученные результаты моделирования подтверждают аналитически найденное стационарное распределение. Сравнение данных показывает, что финальные векторы вероятностей, полученные в результате моделирования, хорошо согласуются с аналитическим решением.