Applied stochastic processes assignment 2

Giovanni Bianco

March 11, 2025

1 Theory

Consider a Markov chain (X_t) with state space $\chi = \{1, 2, \ldots, n\}$ and transition matrix $P = (p_{ij})_{i,j\in\chi}$ where $p_{ij} = P(X_{t+1} = j \mid X_t = i)$ represents the probability of transitioning from state i to state j.

1.1 Invariant Distribution

A probability distribution $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ is called an **invariant distribution** if it satisfies:

$$\pi P = \pi, \tag{1}$$

which, in component form, is given by:

$$\pi_j = \sum_{i\in\chi} \pi_i p_{ij}, \quad \forall j \in \chi. \tag{2}$$

This means that if the Markov chain is in the distribution π at time t, then it remains in this distribution indefinitely.

1.2 Limiting Distribution

If the Markov chain has a limiting distribution, it is given by:

$$\lim_{t\to\infty} P^t = \Pi, \tag{3}$$

where Π is a matrix whose rows represent the steady-state behavior of the system. Explicitly, the probability of being in state j in the long run is:

$$\lim_{t\to\infty} P^t_{ij} = \pi_j, \quad \forall i, j \in \chi. \tag{4}$$

1.3 Asymptotic Distribution

The **asymptotic distribution** is defined as the long-run proportion of time the Markov chain spends in each state:

$$f_i = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbf{I}_{\{X_t=i\}}, \tag{5}$$

which describes the empirical frequency of visits to state i over time.

1.4 Convergence Theorem

Theorem (Convergence Theorem): In a finite, irreducible and aperiodic MC, the (unique) stationary distribution is also limiting distribution (meaning in the long run the distribution will converge to the stationary one and then remain like that) and asymptotoc distribution.

2 Computation

1)

$$P = \begin{bmatrix} 0 & 0.3333 & 0.3333 & 0.3333 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Analysis of Convergence Theorem for the Given Matrix

To satisfy the Convergence Theorem for Markov chains, P must be:
1. **Irreducible**: Every state must be reachable from any other state.
2. **Aperiodic**: The chain should not have cycles that repeat at fixed intervals.
3. **Finite state space**: The number of states should be finite.

1. Irreducibility

There exist a path from each state to each state: it is irreducible

2. Aperiodicity

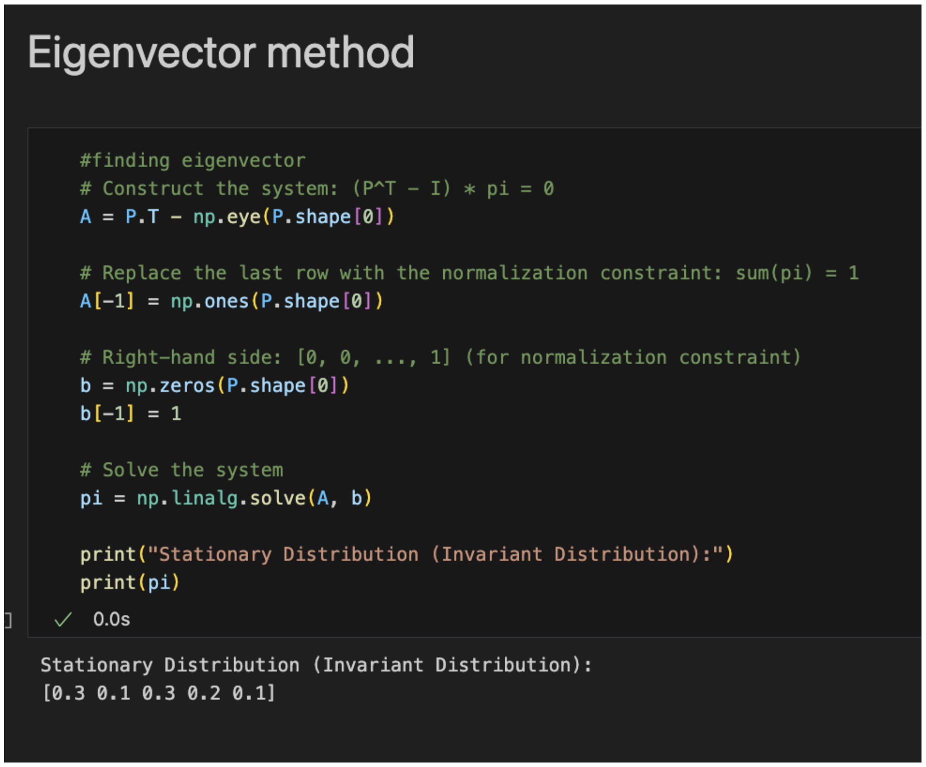
Since there is not a speficic path that starting from a point leads back to that point in a fixed amount of step, the distribution is aperiodic.

3. Finite State Space

The state space is $\{1, 2, 3, 4, 5\}$, which is clearly **finite**.
Therefore it satisfies convergence theorem.

3 Simulations

I simulated using all the 3 methods, in the end the probability vector was (approximately) the same for all of the methods, so i ranked the states only once after the last method(the ranking is the same for all the methods given the vector is the same)

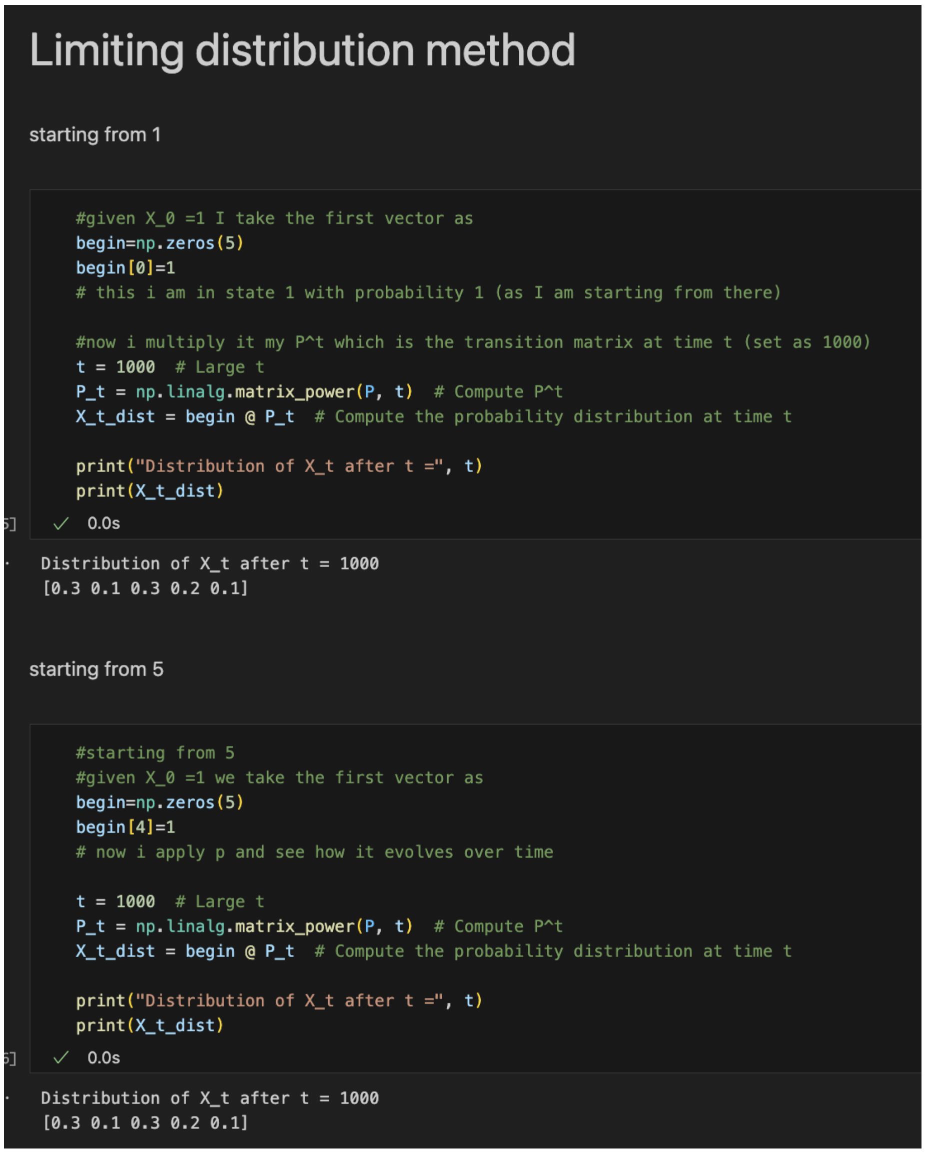


Eigenvector method

```
#finding eigenvector
# Construct the system: (P^T - I) * pi = 0
A = P.T - np.eye(P.shape[0])

# Replace the last row with the normalization constraint: sum(pi) = 1
A[-1] = np.ones(P.shape[0])

# Right-hand side: [0, 0, ..., 1] (for normalization constraint)
b = np.zeros(P.shape[0])
b[-1] = 1

# Solve the system
pi = np.linalg.solve(A, b)

print("Stationary Distribution (Invariant Distribution):")
print(pi)
```

✓ 0.0s

```
Stationary Distribution (Invariant Distribution):
[0.3 0.1 0.3 0.2 0.1]
```



Limiting distribution method

starting from 1

```
#given X_0 =1 I take the first vector as
begin=np.zeros(5)
begin[0]=1
# this i am in state 1 with probability 1 (as I am starting from there)

#now i multiply it my P^t which is the transition matrix at time t (set as 1000)
t = 1000  # Large t
P_t = np.linalg.matrix_power(P, t)  # Compute P^t
X_t_dist = begin @ P_t  # Compute the probability distribution at time t

print("Distribution of X_t after t =", t)
print(X_t_dist)
```

✓ 0.0s

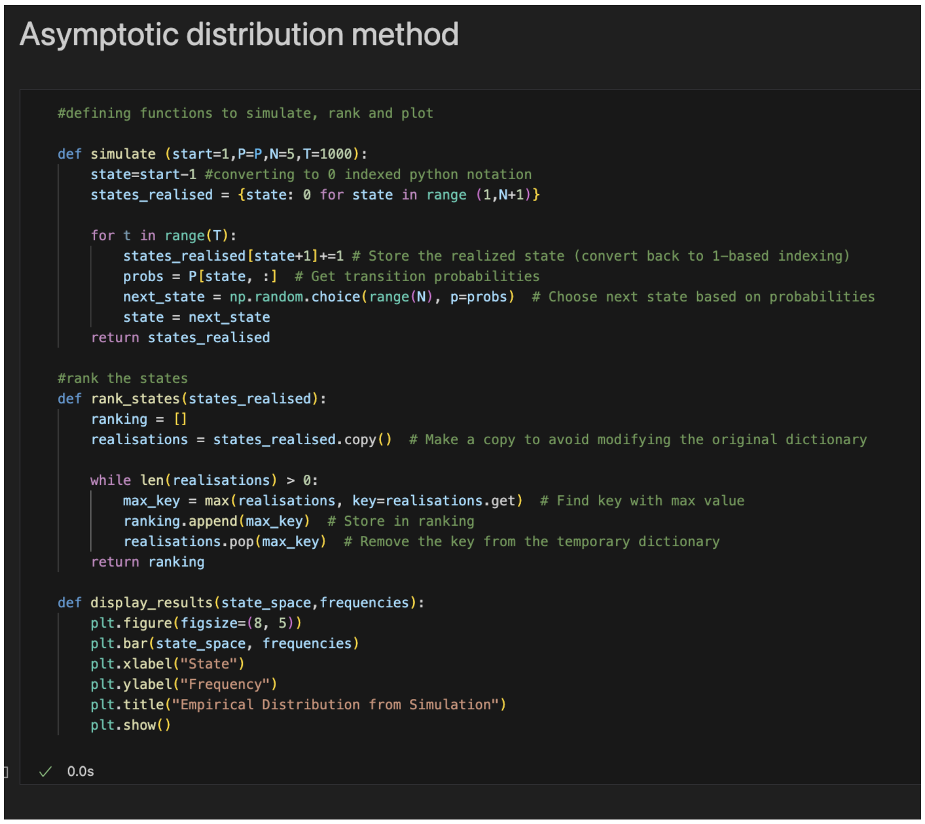
```
Distribution of X_t after t = 1000
[0.3 0.1 0.3 0.2 0.1]
```

starting from 5

```
#starting from 5
#given X_0 =1 we take the first vector as
begin=np.zeros(5)
begin[4]=1
# now i apply p and see how it evolves over time

t = 1000  # Large t
P_t = np.linalg.matrix_power(P, t)  # Compute P^t
X_t_dist = begin @ P_t  # Compute the probability distribution at time t

print("Distribution of X_t after t =", t)
print(X_t_dist)
```

✓ 0.0s

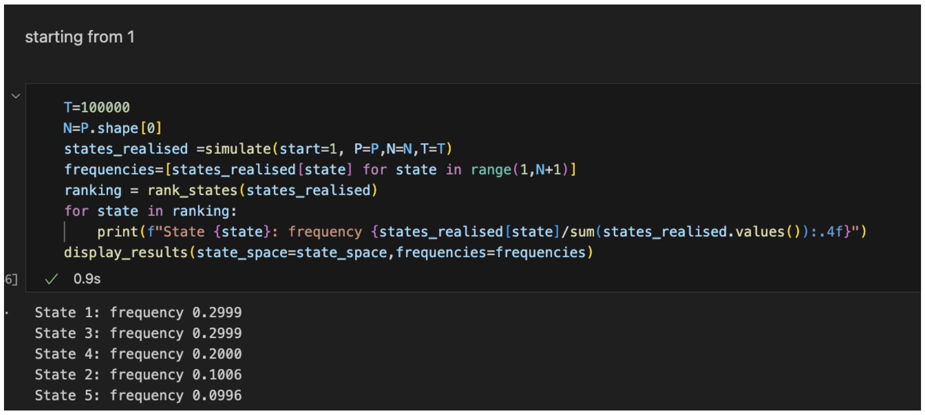
```
Distribution of X_t after t = 1000
[0.3 0.1 0.3 0.2 0.1]
```



Asymptotic distribution method

```python
#defining functions to simulate, rank and plot

def simulate (start=1,P=P,N=5,T=1000):
    state=start-1 #converting to 0 indexed python notation
    states_realised = {state: 0 for state in range (1,N+1)}

    for t in range(T):
        states_realised[state+1]+=1 # Store the realized state (convert back to 1-based indexing)
        probs = P[state, :]  # Get transition probabilities
        next_state = np.random.choice(range(N), p=probs)  # Choose next state based on probabilities
        state = next_state
    return states_realised

#rank the states
def rank_states(states_realised):
    ranking = []
    realisations = states_realised.copy()  # Make a copy to avoid modifying the original dictionary

    while len(realisations) > 0:
        max_key = max(realisations, key=realisations.get)  # Find key with max value
        ranking.append(max_key)  # Store in ranking
        realisations.pop(max_key)  # Remove the key from the temporary dictionary
    return ranking

def display_results(state_space,frequencies):
    plt.figure(figsize=(8, 5))
    plt.bar(state_space, frequencies)
    plt.xlabel("State")
    plt.ylabel("Frequency")
    plt.title("Empirical Distribution from Simulation")
    plt.show()
```

✓ 0.0s

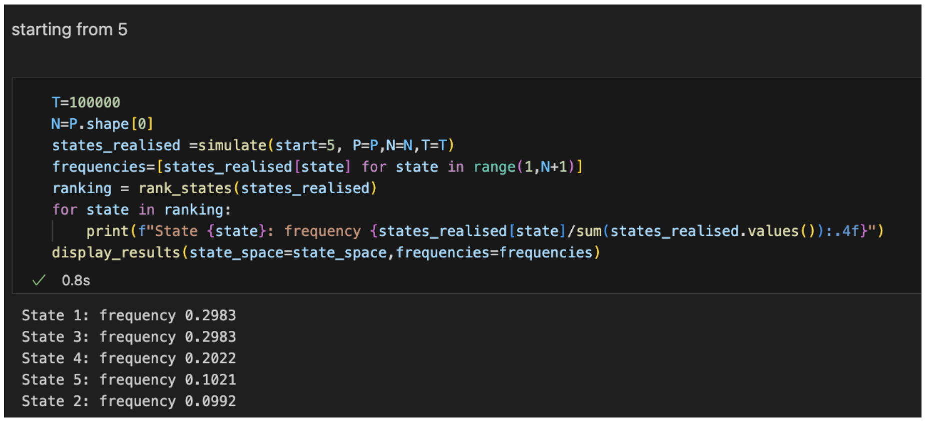


starting from 1

```python
T=100000
N=P.shape[0]
states_realised =simulate(start=1, P=P,N=N,T=T)
frequencies=[states_realised[state] for state in range(1,N+1)]
ranking = rank_states(states_realised)
for state in ranking:
    print(f"State {state}: frequency {states_realised[state]/sum(states_realised.values()):.4f}")
display_results(state_space=state_space,frequencies=frequencies)
```

✓ 0.9s

```
State 1: frequency 0.2999
State 3: frequency 0.2999
State 4: frequency 0.2000
State 2: frequency 0.1006
State 5: frequency 0.0996
```


starting from 5

```
T=100000
N=P.shape[0]
states_realised =simulate(start=5, P=P,N=N,T=T)
frequencies=[states_realised[state] for state in range(1,N+1)]
ranking = rank_states(states_realised)
for state in ranking:
    print(f"State {state}: frequency {states_realised[state]/sum(states_realised.values()):.4f}")
display_results(state_space=state_space,frequencies=frequencies)
```

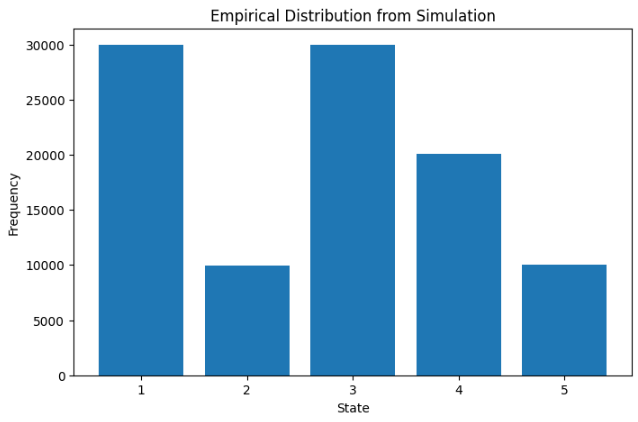
✓ 0.8s

```
State 1: frequency 0.2983
State 3: frequency 0.2983
State 4: frequency 0.2022
State 5: frequency 0.1021
State 2: frequency 0.0992
```

The results do not change if we start from 1 or 5.

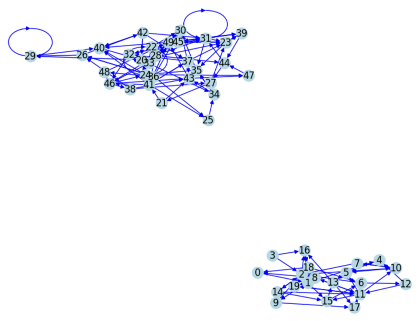
Ranking

- State 1: frequency 0.2996
- State 3: frequency 0.2996
- State 4: frequency 0.2009
- State 5: frequency 0.1003
- State 2: frequency 0.0997

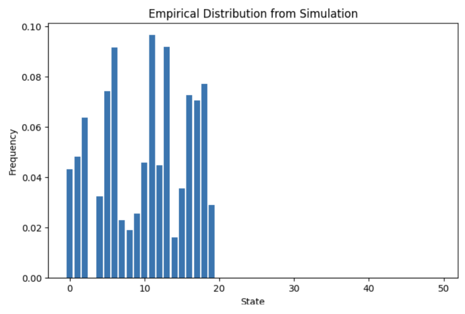


4 Part 3

Loading the dataset from the csv, I normalized the rows so that they represent probabilities and not only connections (each node has uniform probability to go to any node it points to).

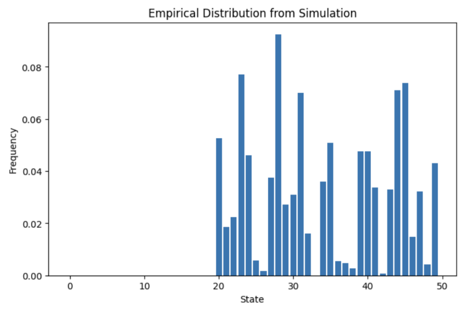


Plotting the graph it is possible to notice that it is divided into 2 connected components, meaning the transition matrix is reducible and therefore the convergence theroem does not apply. This means that starting from a node in a component we can never reach nodes from the other component. Therefore, running simulations it is possible to notice how the asymptotic probability is dependent to the starting node. In fact, starting from node 1, we obtain the following sequences and ranking



- State 12: frequency 0.0966
- State 14: frequency 0.0919
- State 7: frequency 0.0916
-
- ...
- State 49: frequency 0.0000
- State 50: frequency 0.0000

While, if we start from node 50, the results are the following

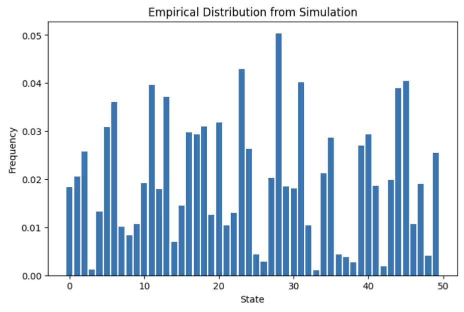


- State 29: frequency 0.0925
- State 24: frequency 0.0770
- State 46: frequency 0.0739
- ...
- State 20: frequency 0.0000
- State 34: frequency 0.0000

4.1 Modified page rank

Thanks to the modified PageRank algorithm, the problem is resolved. Since, with probability 1 - α, we choose a random node at any given time t, setting $\alpha = 0.95$ implies that, on average, we restart at a random node approximately 500 times over the 100,000 steps of the simulation.

This effectively simulates starting from all possible nodes, as the total number of nodes is only 50, ensuring coverage of all connected components. In fact, regardless of whether we start from node 1 or node 50, the frequencies and rankings remain unchanged. They are:



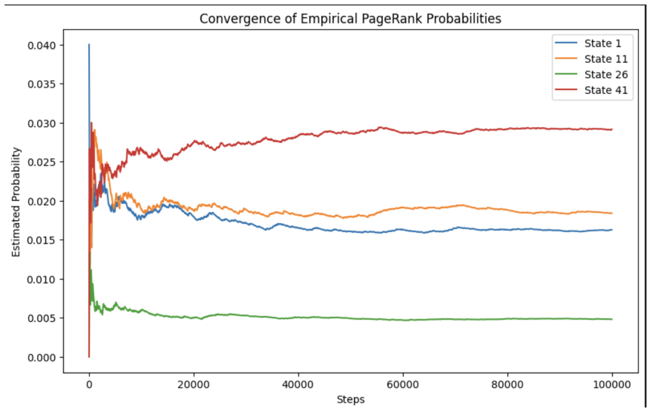
Ranking (check notebook for more results)

- State 29: frequency 0.0503

- State 24: frequency 0.0430
- State 46: frequency 0.0405
- State 32: frequency 0.0402
- State 12: frequency 0.0396
- ...
- State 39: frequency 0.0027
- State 43: frequency 0.0019
- State 4: frequency 0.0012
- State 34: frequency 0.0010

4.2 Showing convergence of asymptotic probabilities

If during the simulation we keep track of how the frequencies for some nodes vary over time, it is possible to node how they converge to steady values (both starting from 1 and 50).


Convergence of Empirical PageRank Probabilities
State 1
State 11
State 26
State 41
Estimated Probability
Steps