ECE 5371 ENGINEERING ANALYSIS

Assignment 3

Rishikesh
R11643260
Texas Tech University
03/28/2024

Contents

1 Problem 1

We are given a Markov process with the following transition diagram and need to find the stationary distribution (π_1, π_2, π_3).

The stationary distribution is found by solving the linear system given by $\pi P = \pi$ where P is the transition matrix, and $\sum_{i=1}^{3} \pi_i = 1$. From the diagram, the transition matrix P can be constructed as:

$$P = \begin{bmatrix} 1/2 & 1/4 & 1/4 \\ 1/3 & 0 & 2/3 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

The equations for the stationary distribution are:

$$\begin{aligned} \pi_1 &= \frac{1}{2}\pi_1 + \frac{1}{4}\pi_2 + \frac{1}{4}\pi_3, \\ \pi_2 &= \frac{1}{3}\pi_1 + 0 + \frac{2}{3}\pi_3, \\ \pi_3 &= \frac{1}{2}\pi_1 + \frac{1}{2}\pi_2 + 0, \\ 1 &= \pi_1 + \pi_2 + \pi_3. \end{aligned}$$

Upon solving the system, we find that the stationary distribution is:

$$\begin{aligned} \pi_1 &= 0.4571, \\ \pi_2 &= 0.2571, \\ \pi_3 &= 0.2857. \end{aligned}$$

```python
from sympy import symbols, Eq, solve, Matrix

pi1, pi2, pi3 = symbols('pi1 pi2 pi3')

# transition matrix
P = Matrix([
    [1/2, 1/4, 1/4],
    [1/3, 0, 2/3],
    [1/2, 1/2, 0]
])
print(P)
eqs = [Eq(pi1, P[0,0]*pi1 + P[1,0]*pi2 + P[2,0]*pi3),
       Eq(pi2, P[0,1]*pi1 + P[1,1]*pi2 + P[2,1]*pi3),
       Eq(pi3, P[0,2]*pi1 + P[1,2]*pi2 + P[2,2]*pi3),
       Eq(pi1 + pi2 + pi3, 1)]

solution = solve(eqs, (pi1, pi2, pi3))
print(solution)
sum(solution.values())
```

Listing 1: Python Code for solving the equations

2 Problem 2

A rental car agency has three locations with the following transition matrix for car pick-ups and returns:

$$P = \begin{bmatrix} 0.4 & 0.4 & 0.2 \\ 0.3 & 0.4 & 0.3 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}$$

Here, P_{ij} represents the probability of a car being picked up at location i and returned to location j. To find the steady-state distribution $\vec{\pi} = (\pi_1, \pi_2, \pi_3)$, we solve the system of linear equations given by $\vec{\pi}P = \vec{\pi}$ subject to the condition $\pi_1 + \pi_2 + \pi_3 = 1$.

The system of equations can be set up as:

$$\begin{aligned} \pi_1 &= 0.4\pi_1 + 0.3\pi_2 + 0.2\pi_3, \\ \pi_2 &= 0.4\pi_1 + 0.4\pi_2 + 0.2\pi_3, \\ \pi_3 &= 0.2\pi_1 + 0.3\pi_2 + 0.6\pi_3, \\ 1 &= \pi_1 + \pi_2 + \pi_3. \end{aligned}$$

Upon solving the system, we find that the stationary distribution is:

$$\begin{aligned} \pi_1 &= 0.2903, \\ \pi_2 &= 0.3226, \\ \pi_3 &= 0.3871. \end{aligned}$$

```python
from sympy import symbols, Eq, solve, Matrix

pi1, pi2, pi3 = symbols('pi1 pi2 pi3')

# transition matrix
P = Matrix([
    [0.4, 0.4, 0.2],
    [0.3, 0.4, 0.3],
    [0.2, 0.2, 0.6]
])
print(P)
eqs = [Eq(pi1, P[0,0]*pi1 + P[1,0]*pi2 + P[2,0]*pi3),
       Eq(pi2, P[0,1]*pi1 + P[1,1]*pi2 + P[2,1]*pi3),
       Eq(pi3, P[0,2]*pi1 + P[1,2]*pi2 + P[2,2]*pi3),
       Eq(pi1 + pi2 + pi3, 1)]

solution = solve(eqs, (pi1, pi2, pi3))
print(solution)
sum(solution.values())
```

Listing 2: Python Code for solving the equations

3 Problem 3

The failure probability is given by an exponential distribution. Thus, if t is the time to failure, the associated cumulative distribution function $F_T(t)$ is given as:

$$F_T(t) = 1 - \exp(-2t).$$

To generate a string of 500 times to failure (e.g., $t_1, t_2, \ldots, t_{500}$) for this random process based on a Monte Carlo scheme involving the generation of random numbers r_i, we use the inverse transform method. Since the random number r_i is uniformly distributed, we set:

$$F_T(t) = r_i.$$

The equation $F_T(t) = 1 - \exp(-2t)$ can be written as:

$$1 - \exp(-2t) = r_i.$$

This implies:

$$\exp(-2t) = 1 - r_i.$$

Taking the natural logarithm on both sides, we get:

$$-2t = \ln(1 - r_i).$$

Hence, t becomes:

$$t = -\frac{1}{2}\ln(1 - r_i).$$

This formula allows us to link the ith failure time t_i to the ith random number r_i.

4 Problem 4

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

particles = 10_000
particles_locations = np.zeros(particles)
time_steps = [50, 500]

for time in range(1, max(time_steps) + 1 ):
    steps = np.random.choice([-0.1, 0.1], size = particles)
    particles_locations += steps

    if time in time_steps:
        plt.figure(figsize = (10,8))
        plt.hist(particles_locations, bins = 100, density = True, alpha = 0.5, label = f'{time} time steps')
        plt.xlabel('position')
        plt.ylabel('Number of particles')
        plt.title('Random walk of particles')
        plt.legend()
        plt.show()
```

Listing 3: Python code Monte Carlo Random Walk for 50 and 500 time steps

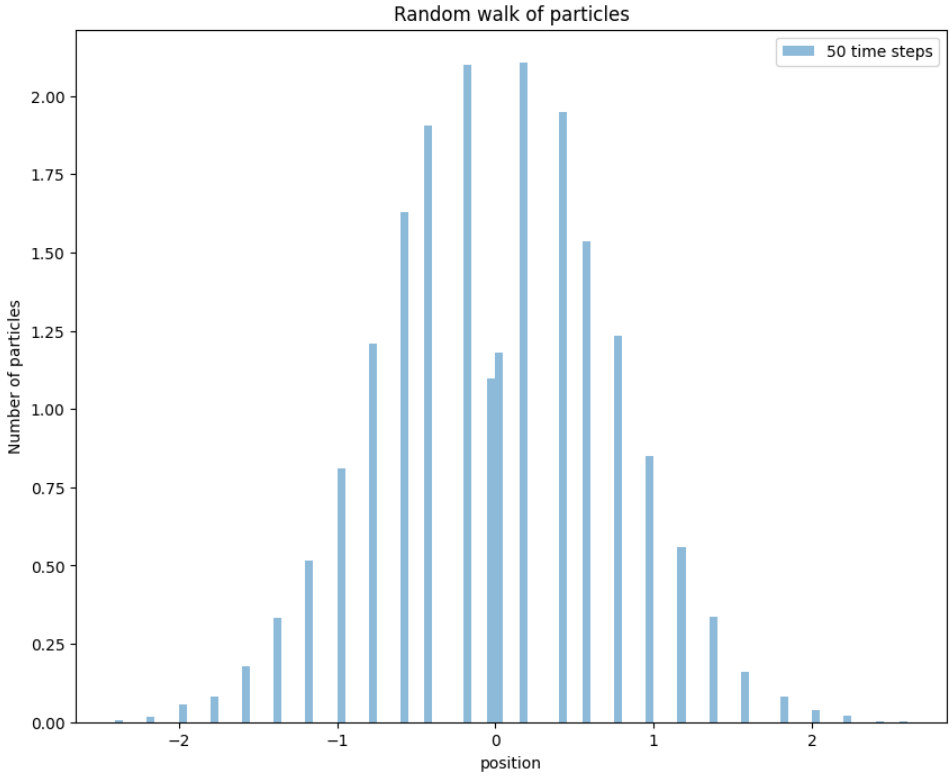

Figure 1: Histogram of the particle distribution for 50 time steps

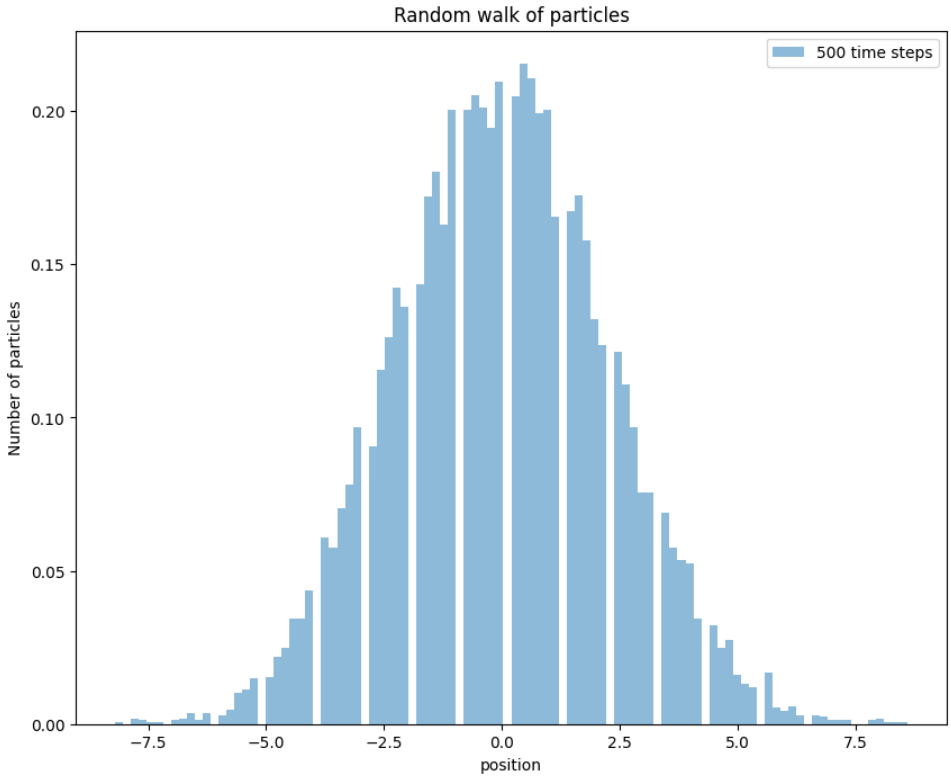

Figure 2: Histogram of the particle distribution for 500 time steps