Project
on
Random Variables and Queue Theory

Course Title: Modeling & Simulation

Course No: CSE 562

Submitted to: Md. Shahriar Karim (MSK1)

Submitted by: Sumaiya Tarannum Noor

Section: 01

ID: 2425410650

Submission date: 23rd August, 2025

Random Variables and Queue Theory

Sumaiya Tarannum Noor

August 23, 2025

1 Answer to the Question No 1

This problem is about Monte Carlo simulation for approximating probabilities and expectations of random variables.
Some of the important features for this are described below:

1. **Random Variable (RV):**
   A random variable is a mapping from the sample space (outcomes of an experiment) to real numbers. In this code, we generate random numbers (using `numpy.random`) to simulate outcomes of RVs.

2. **Monte Carlo Method:**
   Instead of solving integrals analytically, we simulate many samples from the distribution and use their averages to approximate:

   - Probabilities
   - Expected values (means)
   - Variances

   The Law of Large Numbers guarantees that as the number of samples increases, the simulated averages converge to the true values.

3. **Distributions in the Code:**

   - Uniform Distribution (`np.random.rand`) → values between 0 and 1
   - Exponential Distribution (`np.random.exponential`) → models waiting times
   - Normal Distribution (`np.random.normal`) → bell-shaped curve

4. **Validation using CF/MGF (mentioned in Q2):**

   - Characteristic Function (CF) or Moment Generating Function (MGF) can give mean and variance analytically
   - Simulation then verifies those theoretical results

1.1 Algorithm

1.1.1 Exponential Random Variable

Let $X \sim \text{Exponential}(\lambda)$ with pdf

$$f_X(x) = \lambda e^{-\lambda x}, \quad x \geq 0.$$

Mean:

$$\mu = E[X] = \int_0^\infty x\lambda e^{-\lambda x}\, dx = \frac{1}{\lambda}.$$

Variance:

$$\sigma^2 = E[X^2] - (E[X])^2 = \frac{2}{\lambda^2} - \left(\frac{1}{\lambda}\right)^2 = \frac{1}{\lambda^2}.$$

Verification using MGF: The moment generating function (MGF) is

$$M_X(t) = \frac{\lambda}{\lambda - t}, \quad t < \lambda.$$

Then

$$M_X'(0) = \frac{1}{\lambda}, \quad M_X''(0) = \frac{2}{\lambda^2}.$$

Thus $\mu = 1/\lambda$ and $\sigma^2 = 1/\lambda^2$.

—

1.1.2 Uniform Random Variable

Let $X \sim U(a, b)$ with pdf

$$f_X(x) = \frac{1}{b-a}, \quad a \le x \le b.$$

Mean:

$$\mu = \frac{a+b}{2}.$$

Variance:

$$\sigma^2 = \frac{(b-a)^2}{12}.$$

Verification using MGF: The MGF is

$$M_X(t) = \frac{e^{tb} - e^{ta}}{t(b-a)}.$$

Differentiating at $t = 0$ gives $\mu = \frac{a+b}{2}$ and $\sigma^2 = \frac{(b-a)^2}{12}$.

—

1.1.3 Gaussian (Normal) Random Variable

Let $X \sim \mathcal{N}(\mu, \sigma^2)$ with pdf

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

Mean and Variance (Analytical): By definition,

$$E[X] = \mu, \quad \mathrm{Var}(X) = \sigma^2.$$

Verification using CF: The characteristic function is

$$\varphi_X(t) = \exp\left(i\mu t - \frac{\sigma^2 t^2}{2}\right).$$

Differentiating,

$$\varphi_X'(0) = i\mu, \quad \varphi_X''(0) = -\sigma^2 - \mu^2.$$

Thus,

$$E[X] = \mu, \quad \mathrm{Var}(X) = \sigma^2.$$

Algorithm 1 Properties of Exponential Random Variable

Input: Rate parameter $\lambda > 0$ Define random variable $X \sim \text{Exponential}(\lambda)$ with pdf:

$$f_X(x) = \lambda e^{-\lambda x}, \quad x \geq 0$$

Compute mean:

$$\mu = E[X] = \int_0^\infty x\lambda e^{-\lambda x}\, dx = \frac{1}{\lambda}$$

Compute variance:

$$\sigma^2 = E[X^2] - (E[X])^2 = \frac{1}{\lambda^2}$$

Derive MGF:

$$M_X(t) = \frac{\lambda}{\lambda - t}, \quad t < \lambda$$

Differentiate:

$$M_X'(0) = \frac{1}{\lambda}, \quad M_X''(0) = \frac{2}{\lambda^2}$$

Output: $\mu = 1/\lambda$, $\sigma^2 = 1/\lambda^2$ verified via MGF

1.1.4 Result

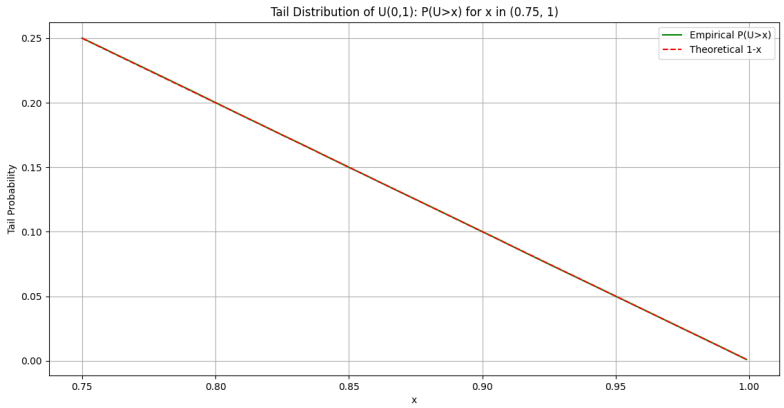

Figure 1: The plot for $P(U > x)$, where $x \in (0.75, 1)$.

—

2 Answer to the Question No. 2

2.1 Theory

In probability and statistics, random variables (RVs) are used to model uncertain outcomes. To analyze their behavior, we often study two key measures: the **mean** (or expectation) and the **variance**. These describe the central tendency and the spread of the distribution, respectively.

2.1.1 Mean and Variance

Let X be a random variable with probability density function (PDF) $f_X(x)$. The mean (expected value) of X is defined as

$$\mu = E[X] = \int_{-\infty}^{\infty} x f_X(x)\, dx.$$

This value represents the long-run average outcome of the random variable.

The variance measures the average squared deviation of X from its mean:

$$\sigma^2 = V[X] = \int_{-\infty}^{\infty} (x-\mu)^2 f_X(x)\,dx.$$

A larger variance indicates that the outcomes are more spread out around the mean.

2.1.2 Moment Generating Function (MGF) and Characteristic Function (CF)

Instead of directly computing integrals, we often use the **moment generating function (MGF)** and the **characteristic function (CF)**.

The MGF of X is defined as

$$M_X(t) = E[e^{tX}] = \int_{-\infty}^{\infty} e^{tx} f_X(x)\,dx,$$

whenever this integral converges.

The MGF is powerful because it encodes all the moments of X. In particular:

$$\mu = M_X'(0), \quad \sigma^2 = M_X''(0) - \left(M_X'(0)\right)^2.$$

That is, the mean is the first derivative of the MGF evaluated at $t = 0$, and the variance is obtained from the second derivative.

Similarly, the characteristic function (CF) is defined as

$$\varphi_X(t) = E[e^{itX}] = \int_{-\infty}^{\infty} e^{itx} f_X(x)\,dx.$$

While the CF always exists (even when the MGF does not), the MGF is often more convenient for finding moments.

2.1.3 Effect of Parameters

Each distribution has parameters that control its shape. For example:

- In the **Normal distribution** $\mathcal{N}(\mu, \sigma^2)$, the parameter μ shifts the distribution horizontally (changing its center), while σ^2 controls the spread.
- In the **Exponential distribution** with rate λ, the mean is $\frac{1}{\lambda}$ and the variance is $\frac{1}{\lambda^2}$. Increasing λ makes the distribution decay faster, concentrating values near zero.
- In the **Uniform distribution** on $[a, b]$, the mean is the midpoint $\frac{a+b}{2}$, while the variance depends on the square of the interval length: $\frac{(b-a)^2}{12}$.

Thus, by computing the mean and variance using MGFs or CFs, and then varying the parameters, we can study how the distribution shifts and spreads.

2.2 Algorithm

The proposed algorithm is designed to estimate the probability distribution of a random variable by combining analytical derivations with numerical verification. It consists of the following key phases:

Step 1: Define the Random Variable Let X be the random variable of interest, with a known or assumed probability density function (pdf) $f_X(x)$. The first step is to clearly state its domain and distributional properties.

Step 2: Transformation of Variables For a transformation $Y = g(X)$, derive the distribution of Y using the change-of-variable technique:

$$f_Y(y) = f_X\big(g^{-1}(y)\big) \cdot \left|\frac{d}{dy} g^{-1}(y)\right|$$

Step 3: Compute Statistical Moments The mean and variance are computed using their definitions:

$$\mu_Y = E[Y] = \int_{-\infty}^{\infty} y\, f_Y(y)\, dy$$

$$\sigma_Y^2 = E[(Y - \mu_Y)^2] = \int_{-\infty}^{\infty} (y - \mu_Y)^2\, f_Y(y)\, dy$$

Step 4: Verification Using MGFs/CFs The moment generating function (MGF) of Y is defined as

$$M_Y(t) = E[e^{tY}]$$

and its derivatives at $t = 0$ provide the moments. Similarly, the characteristic function (CF) can be used:

$$\phi_Y(t) = E[e^{itY}]$$

Step 5: Numerical Validation (Python Simulation) Monte Carlo simulation is employed to verify the analytical results. The algorithm samples N instances of X, applies the transformation $g(\cdot)$ to generate Y, and then empirically estimates the mean, variance, and probability distribution.

Algorithm 2 Distribution Estimation via Analytical and Simulation Methods

1: **Input:** Probability density function (pdf) $f_X(x)$, transformation $g(x)$ (if any), number of samples N
2: Define random variable X with domain $\mathcal{D}_X$
3: If a transformation exists, compute new variable $Y = g(X)$
4: Derive pdf $f_Y(y)$ analytically using the change-of-variable method
5: Compute theoretical mean $\mu_Y = E[Y]$ and variance $\sigma_Y^2 = V[Y]$
6: Obtain moment generating function (MGF) $M_Y(t)$ and/or characteristic function (CF) $\phi_Y(t)$ for validation
7: **Simulation Procedure:**
8: Generate N independent samples $x_i \sim f_X(x)$
9: Apply transformation $y_i = g(x_i)$
10: Estimate empirical mean:

$$\hat{\mu}_Y = \frac{1}{N} \sum_{i=1}^{N} y_i$$

11: Estimate empirical variance:

$$\hat{\sigma}_Y^2 = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{\mu}_Y)^2$$

12: Construct histogram of simulated values $\{y_i\}$ and overlay with theoretical pdf $f_Y(y)$
13: **Output:** Analytical μ_Y, σ_Y^2, empirical $\hat{\mu}_Y, \hat{\sigma}_Y^2$, and validation plots

2.3 Results

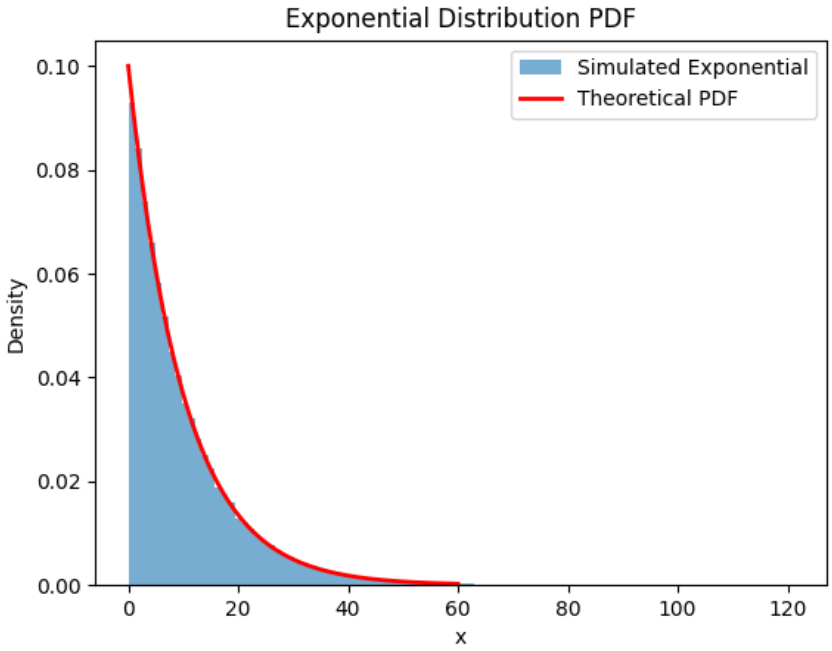

Figure 2: Exponential distribution histogram vs theoretical PDF.

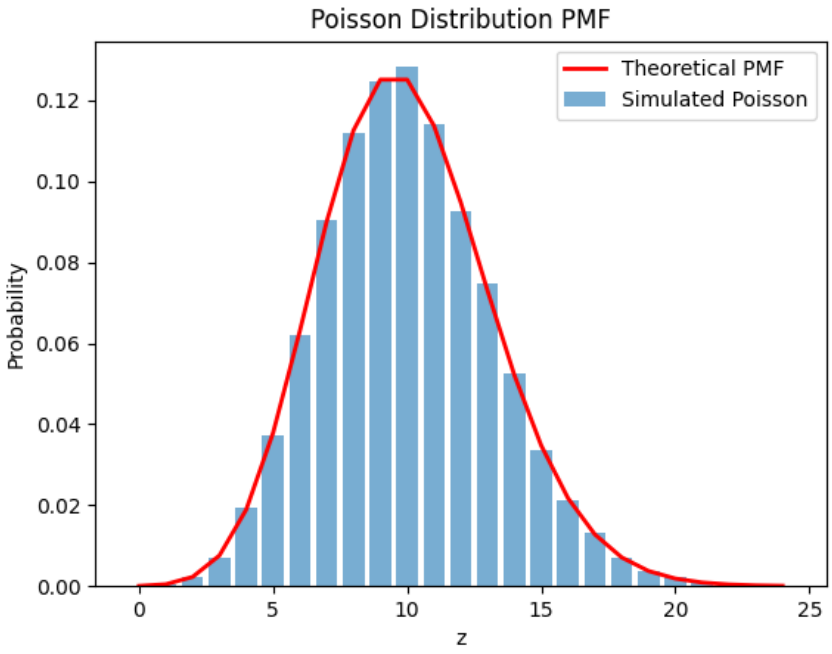

Figure 3: Poisson distribution histogram vs theoretical PMF.

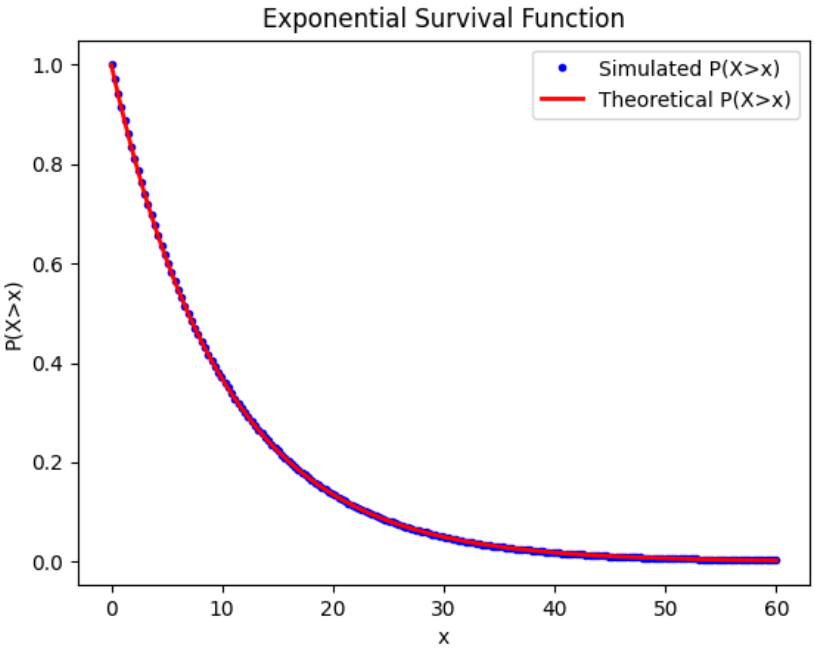

Figure 4: Exponential survival function $P(X > x)$: simulated vs theoretical.

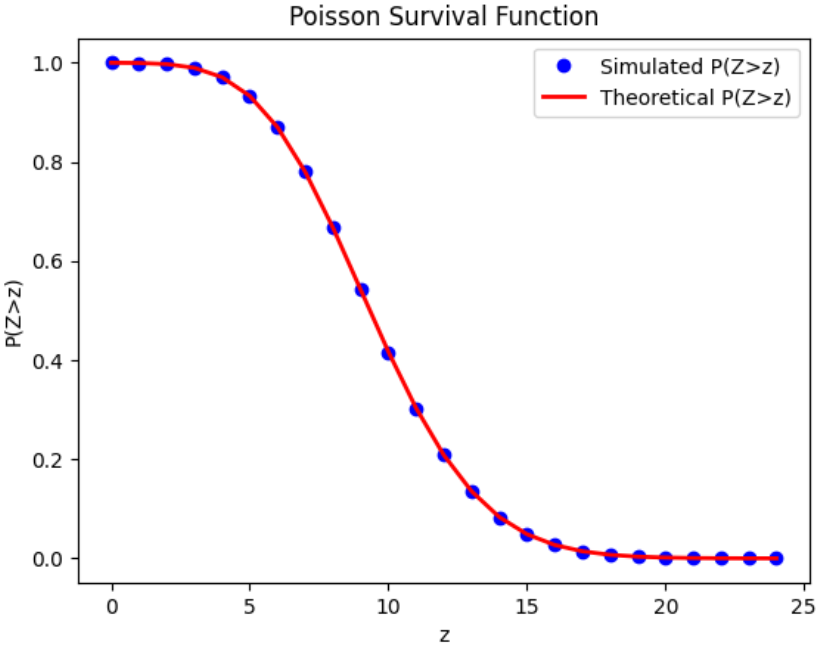

Figure 5: Poisson survival function $P(Z > z)$: simulated vs theoretical.

3 Answer to the Question No. 3

3.1 Theory: M/M/1 Queue

The M/M/1 queue is a classical queueing model where arrivals follow a **Poisson process** with rate λ, and service times are **exponentially distributed** with rate μ. There is a single server and the system can hold an unlimited number of customers.

3.1.1 Key Concepts

- **Arrival rate** λ: average number of customers arriving per unit time.
- **Service rate** μ: average number of customers the server can handle per unit time.
- **Traffic intensity** $\rho = \frac{\lambda}{\mu}$: measures system utilization.
- **Stationary probability** P_n: probability of having n customers in the system at steady state.

For an M/M/1 queue, the stationary probabilities are:

$$P_n = (1-\rho)\rho^n, \quad n = 0, 1, 2, \ldots$$

The mean number of customers in the system is:

$$L = \frac{\rho}{1-\rho}$$

The probability of the system being empty:

$$P_0 = 1 - \rho$$

3.2 Algorithm: M/M/1 Queue Simulation

The M/M/1 queue simulation can be implemented in the following steps:

Step 1: Initialize Parameters Set arrival rate λ, service rate μ, number of arrivals N, and traffic intensity $\rho = \lambda/\mu$.

Step 2: Generate Exponential Random Variables Generate interarrival times and service times using:

$$t_{\text{interarrival}} \sim \text{Exponential}(\lambda), \quad t_{\text{service}} \sim \text{Exponential}(\mu)$$

Step 3: Compute Arrival and Departure Times

$$\text{arrival}_i = \sum_{j=1}^{i} t_{\text{interarrival},j}, \quad \text{start\_service}_i = \max(\text{arrival}_i, \text{departure}_{i-1}), \quad \text{departure}_i = \text{start\_service}_i + t_{\text{service},i}$$

Step 4: Determine Customers in System At each time point t, compute the number of customers in the system:

$$n(t) = \sum_i \mathbf{1}_{\{\text{arrival}_i \le t < \text{departure}_i\}}$$

Step 5: Compute Stationary Probabilities Compute empirical stationary probabilities P_n as the fraction of time the system has exactly n customers:

$$P_n = \text{mean}\big(n(t) = n\big)$$

Step 6: Analytical Comparison Compare simulated P_n with analytical solution:

$$P_n = (1-\rho)\rho^n$$

Step 7: Traffic Intensity Variation Repeat the computation for different traffic intensities $\rho = 0.9, 0.5, 0.25$ and compare the probability distributions.

Algorithm 3 M/M/1 Queue Simulation and Analytical Validation

Input: Arrival rate λ, service rate μ, number of arrivals N, traffic intensities $\rho = \lambda/\mu$
Initialize empty arrays for interarrival times, service times, arrival times, departure times
Generate interarrival times $t_i \sim \text{Exponential}(\lambda)$ for $i = 1, \ldots, N$
Generate service times $s_i \sim \text{Exponential}(\mu)$ for $i = 1, \ldots, N$
Compute arrival times:

$$A_i = \sum_{j=1}^{i} t_j$$

For each customer i:
    Set service start time:

$$S_i = \max(A_i, D_{i-1})$$

    Compute departure time:

$$D_i = S_i + s_i$$

At each observation time t, compute number of customers in system:

$$n(t) = \sum_{i=1}^{N} \mathbf{1}_{\{A_i \le t < D_i\}}$$

Estimate stationary probabilities empirically:

$$\hat{P}_n = \text{mean}(n(t) = n)$$

Compute analytical stationary probabilities:

$$P_n = (1-\rho)\rho^n, \quad n = 0, 1, 2, \ldots$$

Repeat the procedure for different traffic intensities $\rho = 0.9, 0.5, 0.25$
Plot simulated $\hat{P}_n$ vs. analytical P_n and compare results
Output: Empirical distribution $\hat{P}_n$, analytical distribution P_n, and validation plots

3.3 Results

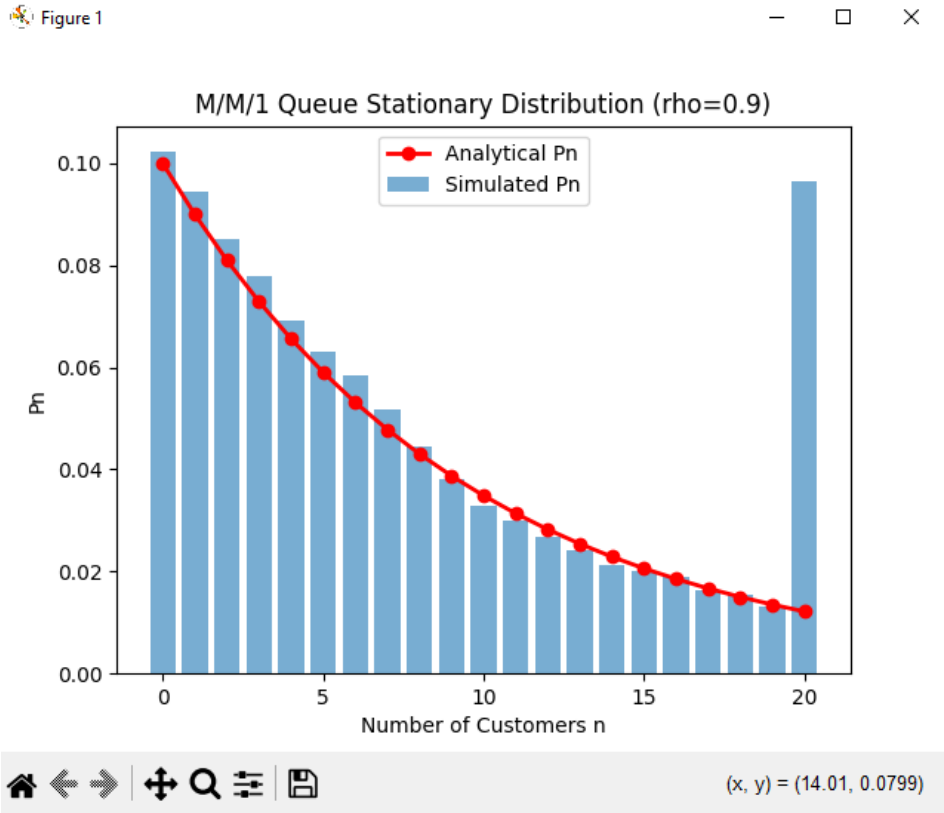

Figure 6: M/M/1 Queue stationary probability distribution at $\rho = 0.9$: simulated vs analytical.

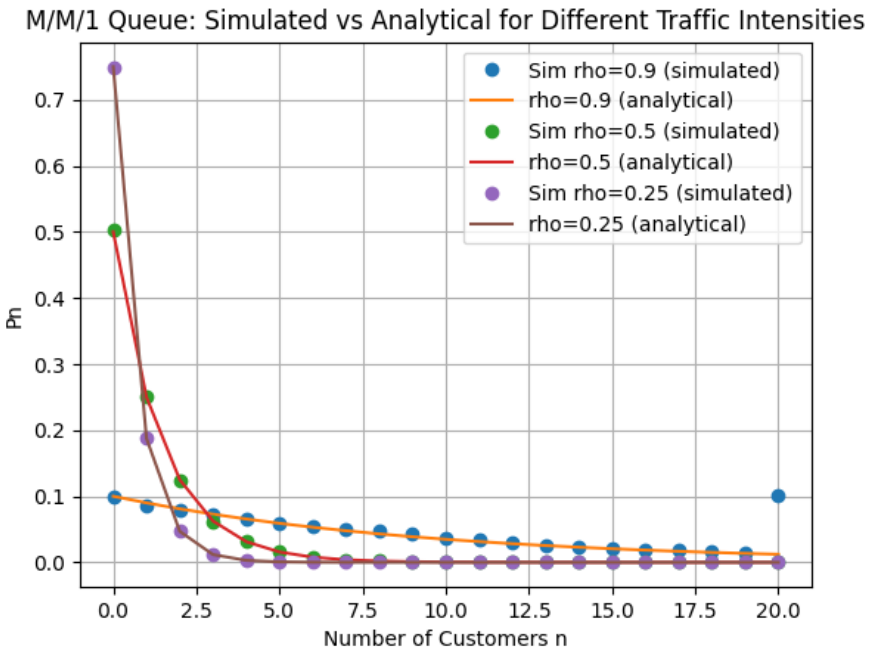

Figure 7: M/M/1 Queue stationary probability P_n for different traffic intensities ρ: simulated vs analytical.

4 Codes

4.1 Code for Question No. 1

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

np.random.seed(42)

# Exponential
lam = 2
exp_data = np.random.exponential(1/lam, 100000)
print("Exponential Mean:", np.mean(exp_data))
print("Exponential Variance:", np.var(exp_data))
plt.hist(exp_data, bins=100, density=True, alpha=0.6)
plt.title("Exponential Distribution")
plt.show()

# Uniform
a, b = 0, 5
uni_data = np.random.uniform(a, b, 100000)
print("Uniform Mean:", np.mean(uni_data))
print("Uniform Variance:", np.var(uni_data))
plt.hist(uni_data, bins=100, density=True, alpha=0.6)
plt.title("Uniform Distribution")
plt.show()

# Normal
mu, sigma = 0, 1
norm_data = np.random.normal(mu, sigma, 100000)
```

```
print("Normal Mean:", np.mean(norm_data))
print("Normal Variance:", np.var(norm_data))
plt.hist(norm_data, bins=100, density=True, alpha=0.6)
plt.title("Normal Distribution")
plt.show()
```

4.2 Code for Question No. 2

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import expon, poisson

np.random.seed(42)

def exponential_rv(lam, size=10000):
    U = np.random.rand(size)
    return -np.log(U) / lam

lam = 1 / 10  # E[X] = 10
X = exponential_rv(lam, size=100000)

def poisson_rv(mu, size=10000):
    samples = []
    for _ in range(size):
        L = np.exp(-mu)
        k = 0
        p = 1
        while p > L:
            k += 1
            p *= np.random.rand()
        samples.append(k-1)
    return np.array(samples)

mu = 10  # E[Z] = 10
Z = poisson_rv(mu, size=50000)

#Exponential histogram vs theoretical PDF
x_vals = np.linspace(0, 60, 200)
plt.figure()
plt.hist(X, bins=100, density=True, alpha=0.6, label="Simulated Exponential")
plt.plot(x_vals, expon.pdf(x_vals, scale=1/lam), 'r-', lw=2, label="Theoretical PDF")
plt.title("Exponential Distribution PDF")
plt.xlabel("x"); plt.ylabel("Density")
plt.legend()

# Poisson histogram vs theoretical PMF
z_vals = np.arange(0, 25)
counts, bins = np.histogram(Z, bins=np.arange(-0.5, 25.5, 1), density=True)

plt.figure()
plt.bar(z_vals, counts, alpha=0.6, label="Simulated Poisson")
plt.plot(z_vals, poisson.pmf(z_vals, mu), 'r-', lw=2, label="Theoretical PMF")
plt.title("Poisson Distribution PMF")
plt.xlabel("z"); plt.ylabel("Probability")
plt.legend()

# Exponential: P(X > x)
```

```
plt.figure()
plt.plot(x_vals, np.mean(X[:,None] > x_vals, axis=0), 'b.', label="Simulated P(X>x)")
plt.plot(x_vals, np.exp(-lam*x_vals), 'r-', lw=2, label="Theoretical P(X>x)")
plt.title("Exponential Survival Function")
plt.xlabel("x"); plt.ylabel("P(X>x)")
plt.legend()

# Poisson: P(Z > z)
plt.figure()
plt.plot(z_vals, [np.mean(Z > z) for z in z_vals], 'bo', label="Simulated P(Z>z)")
plt.plot(z_vals, 1-poisson.cdf(z_vals, mu), 'r-', lw=2, label="Theoretical P(Z>z)")
plt.title("Poisson Survival Function")
plt.xlabel("z"); plt.ylabel("P(Z>z)")
plt.legend()

plt.show()
```

4.3 Code for Question No. 3

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

np.random.seed(42)

def exponential_rv(lam, size=10000):
    U = np.random.rand(size)
    return -np.log(U) / lam

def simulate_mm1(lambda_arrival, mu_service, N=100000, max_customers=20):
    # Generate interarrival and service times
    inter_arrival_times = exponential_rv(lambda_arrival, N)
    service_times = exponential_rv(mu_service, N)

    arrival_times = np.cumsum(inter_arrival_times)
    start_service_times = np.zeros(N)
    departure_times = np.zeros(N)

    for i in range(N):
        if i == 0:
            start_service_times[i] = arrival_times[i]
        else:
            start_service_times[i] = max(arrival_times[i], departure_times[i-1])
        departure_times[i] = start_service_times[i] + service_times[i]

    # ---- Event-driven state tracking ----
    events = np.sort(np.concatenate([arrival_times, departure_times]))
    customers = 0
    last_t = 0
    time_in_state = np.zeros(max_customers+1)

    for t in events:
        # accumulate time spent in current state
        dt = t - last_t
        if customers <= max_customers:
            time_in_state[customers] += dt
        else:
```

```
            time_in_state[max_customers] += dt  # lump overflow into max_customers
        last_t = t

        # update state
        if t in arrival_times:
            customers += 1
        else:
            customers -= 1

    # normalize to get probabilities
    Pn_simulated = time_in_state / np.sum(time_in_state)
    return Pn_simulated

# Parameters
mu_service = 10
max_customers = 20
N = 50000   # reduce a bit for faster simulation

# ---- Simulation for rho = 0.9 ----
lambda_arrival = 9
Pn_simulated = simulate_mm1(lambda_arrival, mu_service, N, max_customers)
rho = lambda_arrival / mu_service
Pn_analytical = rho**np.arange(max_customers+1) * (1-rho)

plt.figure()
plt.bar(range(max_customers+1), Pn_simulated, alpha=0.6, label='Simulated Pn')
plt.plot(range(max_customers+1), Pn_analytical, 'r-o', lw=2, label='Analytical Pn')
plt.xlabel('Number of Customers n')
plt.ylabel('Pn')
plt.title('M/M/1 Queue Stationary Distribution (rho=0.9)')
plt.legend()
plt.show()

# ---- Compare for different rho ----
plt.figure()
for rho_val in [0.9, 0.5, 0.25]:
    lambda_val = rho_val * mu_service
    Pn_sim = simulate_mm1(lambda_val, mu_service, N, max_customers)
    Pn_theory = rho_val**np.arange(max_customers+1) * (1-rho_val)

    plt.plot(range(max_customers+1), Pn_sim, 'o', label=f'Sim rho={rho_val} (simulated)')
    plt.plot(range(max_customers+1), Pn_theory, '-', label=f'rho={rho_val} (analytical)')

plt.xlabel('Number of Customers n')
plt.ylabel('Pn')
plt.title('M/M/1 Queue: Simulated vs Analytical for Different Traffic Intensities')
plt.legend()
plt.grid(True)
plt.show()
```

References

[1] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed., McGraw-Hill, 2002.

[2] S. Chan, *Introduction to Probability for Data Science*, Springer, 2020.

[3] S. M. Ross, *Introduction to Probability Models*, 11th ed., Academic Press, 2014.

[4] H. C. Tijms, *Understanding Probability: Chance Rules in Everyday Life*, 2nd ed., Cambridge University Press, 2007.

[5] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*, 5th ed., Wiley, 2018.

[6] L. Kleinrock, *Queueing Systems, Volume I: Theory*, Wiley, 1975.

[7] J. VanderPlas, *Python Data Science Handbook: Essential Tools for Working with Data*, O'Reilly Media, 2016.