Homework 3.

Question 1.

(a) Find $P(W_2 \geq c)$

$$W_2 = \max(0, V_1 - T_2)$$

$$\begin{aligned} P(W_2 \geq c) &= 1 - P(W_2 < c) \\ &= 1 - P(\max(0, V_1 - T_2) < c) \\ &= 1 - P(V_1 - T_2 < c) \\ &= 1 - P(V_1 < T_2 + c) \end{aligned}$$

$$P(V_1 < T_2 + c) = \int_0^\infty \int_0^{t_2+c} \mu e^{-\mu v_1} \lambda e^{-\lambda t_2} \, dv_1 \, dt_2$$

$$\begin{aligned} \int_0^{t_2+c} \mu e^{-\mu v_1} \, dv_1 &= \left[-e^{-\mu v_1}\right]_0^{t_2+c} \\ &= -e^{-\mu(t_2+c)} + 1 \\ &= 1 - e^{-\mu t_2} e^{-\mu c} \end{aligned}$$

$$\begin{aligned} P(V_1 < T_2 + c) &= \int_0^\infty \lambda e^{-\lambda t_2} \left(1 - e^{-\mu t_2} e^{-\mu c}\right) dt_2 \\ &= \int_0^\infty \lambda e^{-\lambda t_2} \, dt_2 - e^{-\mu c} \int_0^\infty \lambda e^{-(\lambda+\mu) t_2} \, dt_2 \end{aligned}$$

$$\int_0^\infty \lambda e^{-\lambda t_2} \, dt_2 = \left[-e^{-\lambda t_2}\right]_0^\infty = 1$$

$$\int_0^\infty \lambda e^{-(\lambda+\mu) t_2} \, dt_2 = \left[\frac{-\lambda}{\lambda+\mu} e^{-(\lambda+\mu) t_2}\right]_0^\infty = \frac{\lambda}{\lambda+\mu}$$

$$P(V_1 < T_2 + c) = 1 - e^{-\mu c} \cdot \frac{\lambda}{\lambda+\mu} = 1 - \frac{\lambda}{\lambda+\mu} e^{-\mu c}$$

$$\begin{aligned} P(W_2 \geq c) &= 1 - \left(1 - \frac{\lambda}{\lambda+\mu} e^{-\mu c}\right) \\ &= \frac{\lambda}{\lambda+\mu} e^{-\mu c} \end{aligned}$$

(b) Find $P(W_3 \geq c)$

$$W_3 = \max(0, D_2 - A_3)$$

$$\begin{aligned} D_2 &= V_2 + \max(A_2, D_1) \\ &= V_2 + \max(A_1 + T_2, A_1 + V_1) \\ &= V_2 + A_1 + \max(T_2, V_1) \end{aligned}$$

$$A_3 = A_1 + T_2 + T_3$$

$$\begin{aligned} W_3 &= \max(0, V_2 + A_1 + \max(T_2, V_1) - A_1 - T_2 - T_3) \\ &= \max(0, V_2 + \max(T_2, V_1) - T_2 - T_3) \end{aligned}$$

$$\begin{aligned} P(W_3 \geq c) &= P(V_2 + \max(T_2, V_1) - T_2 - T_3 \geq c) \\ &= \int_0^\infty \int_0^\infty \int_0^\infty P(V_2 + \max(t_2, v_1) - t_2 - T_3 \geq c) \mu e^{-\mu v_1} \lambda e^{-\lambda t_2} \mu e^{-\mu v_2} \, dv_1 \, dt_2 \, dv \\ &= \int_0^\infty \int_0^\infty \int_0^\infty P(V_2 \geq c + t_2 + T_3 - \max(t_2, v_1)) \mu e^{-\mu v_1} \lambda e^{-\lambda t_2} \mu e^{-\mu v_2} \, dv_1 \, dt_2 \, dv \\ &= \int_0^\infty \int_0^{t_2} \int_{c+t_2-t_2}^\infty \mu e^{-\mu v_2} \mu e^{-\mu v_1} \lambda e^{-\lambda t_2} \, dv_2 \, dv_1 \, dt_2 \\ &\quad + \int_0^\infty \int_{t_2}^\infty \int_{c+t_2-v_1}^\infty \mu e^{-\mu v_2} \mu e^{-\mu v_1} \lambda e^{-\lambda t_2} \, dv_2 \, dv_1 \, dt_2 \end{aligned}$$

$$\begin{aligned} P(W_3 \geq c) &= \int_0^\infty \int_0^{t_2} \int_c^\infty \mu \lambda \mu e^{-\mu v_2} e^{-\mu v_1} e^{-\lambda t_2} \, dv_2 \, dv_1 \, dt_2 \\ &\quad + \int_0^\infty \int_{t_2}^\infty \int_{c+t_2-v_1}^\infty \mu \lambda \mu e^{-\mu v_2} e^{-\mu v_1} e^{-\lambda t_2} \, dv_2 \, dv_1 \, dt_2 \end{aligned}$$

In [ ]:

```python
import numpy as np

def WaitingTimes(n, lam, mu):
    """
    Simulate waiting times for n customers in a single-server queue.

    Parameters:
    n: number of customers
    lam: arrival rate (lambda)
    mu: service rate (mu)

    Returns:
    Array of waiting times for each customer
    """
    # Generate interarrival times and service times
    T = np.random.exponential(1/lam, n)  # Interarrival times
    # Creates array of n random samples from Exp(λ)
    # np.random.exponential takes SCALE parameter = 1/rate
    # So for rate λ, we pass 1/λ

    V = np.random.exponential(1/mu, n)   # Service times
    # Creates array of n random samples from Exp(μ)
    # For rate μ, we pass 1/μ

    # Initialize arrays
    A = np.zeros(n)  # Arrival times
    # Creates array of n zeros to store arrival time of each customer

    D = np.zeros(n)  # Departure times
    # Creates array of n zeros to store departure time of each customer

    W = np.zeros(n)  # Waiting times
    # Creates array of n zeros to store waiting time of each customer

    # First customer
    A[0] = T[0]
    # Customer 1 arrives at time T[0] (first interarrival time from time 0)

    D[0] = A[0] + V[0]
    # Customer 1 departs at arrival time + service time
    # No waiting since queue starts empty

    W[0] = 0
    # Customer 1 has zero waiting time (given in problem)

    # Remaining customers
    for i in range(1, n):
        # Loop through customers 2 through n (indices 1 through n-1)

        A[i] = A[i-1] + T[i]
        # Customer i arrives T[i] time units after customer i-1
        # This implements: A_i = A_{i-1} + T_i

        D[i] = V[i] + max(A[i], D[i-1])
        # Customer i departs at: service time + max(arrival, previous departure)
        # If A[i] > D[i-1]: server idle, depart at A[i] + V[i]
        # If A[i] <= D[i-1]: server busy, depart at D[i-1] + V[i]
        # This implements: D_i = V_i + max(A_i, D_{i-1})
```

```python
        W[i] = max(D[i-1] - A[i], 0)
        # Waiting time = time until previous customer finishes - arrival time
        # If D[i-1] > A[i]: must wait D[i-1] - A[i]
        # If D[i-1] <= A[i]: server idle, no wait (0)
        # This implements: W_i = max(D_{i-1} - A_i, 0)

    return W
    # Return the array of all waiting times

# Test with n=10, lambda=1, mu=1
np.random.seed(42)
# Set random seed for reproducibility
# Same seed = same "random" numbers every time

waiting_times = WaitingTimes(10, 1, 1)
# Call function to simulate 10 customers with λ=1, μ=1

print("Waiting times for n=10, λ=1, μ=1:")
print(waiting_times)
# Display the array of 10 waiting times

print(f"\nMean waiting time: {np.mean(waiting_times):.4f}")
# Compute and display average waiting time
# np.mean() computes average of array
# :.4f formats to 4 decimal places
```

```
Waiting times for n=10, λ=1, μ=1:
[0.         0.         2.18681178 3.06029877 3.12936153 3.16044422
 3.30321688 1.65473974 1.47958542 0.81387242]

Mean waiting time: 1.8788
```

Question 1d.

$$
\begin{aligned}
E[W_2] &= \int_0^\infty P(W_2 \geq c)\, dc \quad \text{(survival function formula)} \\
&= \int_0^\infty \frac{\lambda}{\lambda+\mu} e^{-\mu c}\, dc \quad \text{(from part (a))} \\
&= \frac{\lambda}{\lambda+\mu} \int_0^\infty e^{-\mu c}\, dc \\
&= \frac{\lambda}{\lambda+\mu} \left[\frac{-1}{\mu} e^{-\mu c} \right]_0^\infty \\
&= \frac{\lambda}{\lambda+\mu} \left(0 - \frac{-1}{\mu} \right) \\
&= \frac{\lambda}{\lambda+\mu} \cdot \frac{1}{\mu} \\
&= \frac{\lambda}{\mu(\lambda+\mu)}
\end{aligned}
$$

$$
\begin{aligned}
&\text{With } \lambda = 1, \mu = 1: \\
&E[W_2] = \frac{1}{1 \cdot (1+1)} = \frac{1}{2} = 0.5
\end{aligned}
$$

In [ ]:

```
# ==============================================================================
# Estimating E[W_2] and P(W_2 > 1) using Monte Carlo
# ==============================================================================

# Set parameters
lam = 1  # Arrival rate λ
mu = 1   # Service rate μ
n_simulations = 1000000  # Number of Monte Carlo simulations

# Initialize list to store W_2 samples
W2_samples = []

# Run Monte Carlo simulations
for _ in range(n_simulations):
    # Generate waiting times for first 3 customers
    # We need 3 customers to get W_0, W_1, W_2
    W = WaitingTimes(2, lam, mu)

    # Extract W_2 (the waiting time of the 2nd customer)
    # Remember: W[0] = W_1, W[1] = W_2, W[2] = W_3
    W2_samples.append(W[1])

# Convert list to numpy array for easier computation
W2_samples = np.array(W2_samples)

# Estimate E[W_2] using sample mean
```

```python
# By Law of Large Numbers: sample mean → true mean as n → ∞
E_W2_mc = np.mean(W2_samples)

# Estimate P(W_2 > 1) using indicator function
# np.mean(W2_samples > 1) computes the fraction of samples where W_2 > 1
# This is equivalent to: (number of times W_2 > 1) / (total simulations)
P_W2_gt_1_mc = np.mean(W2_samples > 1)

# Compute theoretical values from part (a)
# E[W_2] = λ / (μ(λ + μ))
E_W2_theory = lam / (mu * (lam + mu))

# P(W_2 > 1) = (λ/(λ+μ)) * e^(-μ*c) with c = 1
P_W2_gt_1_theory = (lam / (lam + mu)) * np.exp(-mu * 1)

# Display results
print("="*60)
print("Estimating W_2 with λ=1, μ=1")
print("="*60)

print(f"\nE[W_2]:")
print(f"  Monte Carlo estimate: {E_W2_mc:.4f}")
print(f"  Theoretical value:    {E_W2_theory:.4f}")
print(f"  Absolute error:       {abs(E_W2_mc - E_W2_theory):.4f}")

print(f"\nP(W_2 > 1):")
print(f"  Monte Carlo estimate: {P_W2_gt_1_mc:.4f}")
print(f"  Theoretical value:    {P_W2_gt_1_theory:.4f}")
print(f"  Absolute error:       {abs(P_W2_gt_1_mc - P_W2_gt_1_theory):.4f}")
```

```
============================================================
Estimating W_2 with λ=1, μ=1
============================================================

E[W_2]:
  Monte Carlo estimate: 0.4998
  Theoretical value:    0.5000
  Absolute error:       0.0002

P(W_2 > 1):
  Monte Carlo estimate: 0.1839
  Theoretical value:    0.1839
  Absolute error:       0.0001
```

In [ ]:

```python
# ============================================================================
# Estimating E[W_100] using Monte Carlo
# ============================================================================

# Set parameters (same as before)
lam = 1  # Arrival rate λ
mu = 1   # Service rate μ
n_simulations = 100000  # Number of Monte Carlo simulations

# Initialize list to store W_100 samples
W100_samples = []

# Run Monte Carlo simulations
for _ in range(n_simulations):
    # Generate waiting times for first 101 customers
    # We need 101 customers to get W_0, W_1, ..., W_99, W_100
```

```python
    W = WaitingTimes(100, lam, mu)

    # Extract W_100 (the waiting time of the 100th customer)
    # Remember: W[0] = W_1, W[1] = W_2, ..., W[99] = W_100
    W100_samples.append(W[99])

# Convert list to numpy array
W100_samples = np.array(W100_samples)

# Estimate E[W_100] using sample mean
# By Law of Large Numbers: sample mean → true mean as n → ∞
E_W100_mc = np.mean(W100_samples)

# Note: There is no simple closed-form theoretical value for E[W_100]
# For λ = μ (our case), the system is at the boundary of stability
# The queue is critically loaded (utilization ρ = λ/μ = 1)
# In steady-state (as i → ∞), waiting times would grow unbounded
# But for finite i = 100, we can estimate via simulation

# Display results
print("\n" + "="*60)
print("Estimating W_100 with λ=1, μ=1")
print("="*60)

print(f"\nE[W_100]:")
print(f"  Monte Carlo estimate: {E_W100_mc:.4f}")
print(f"  (No simple closed-form theoretical value available)")

# Optional: Compute some additional statistics
print(f"\nAdditional statistics for W_100:")
print(f"  Standard deviation:   {np.std(W100_samples):.4f}")
print(f"  Median:               {np.median(W100_samples):.4f}")
print(f"  95th percentile:      {np.percentile(W100_samples, 95):.4f}")
```

```
============================================================
Estimating W_100 with λ=1, μ=1
============================================================

E[W_100]:
  Monte Carlo estimate: 10.2527
  (No simple closed-form theoretical value available)

Additional statistics for W_100:
  Standard deviation:   8.4487
  Median:               8.4726
  95th percentile:      26.5015
```

Question 2.

Solution to 2a.

Let $X \sim \mathcal{N}(0, D)$ where D is diagonal with $d_1, d_2, \ldots, d_p$ on the diagonal.

Properties of D

$$\det(D) = d_1 \cdot d_2 \cdot \ldots \cdot d_p = \prod_{i=1}^{p} d_i$$

$$D^{-1} = \operatorname{diag}\left(\frac{1}{d_1}, \frac{1}{d_2}, \ldots, \frac{1}{d_p}\right)$$

PDF of X

$$f(x) = \frac{1}{(2\pi)^{p/2}|D|^{1/2}} \exp\left(-\frac{1}{2}x^T D^{-1} x\right)$$

$$= \frac{1}{(2\pi)^{p/2}(d_1 \cdot d_2 \cdot \ldots \cdot d_p)^{1/2}} \exp\left(-\frac{1}{2}x^T D^{-1} x\right)$$

Expand the quadratic form

$$x^T D^{-1} x = \frac{x_1^2}{d_1} + \frac{x_2^2}{d_2} + \cdots + \frac{x_p^2}{d_p}$$

Factor the PDF

$$f(x) = \frac{1}{(2\pi d_1)^{1/2}} \exp\left(-\frac{x_1^2}{2d_1}\right) \cdot \frac{1}{(2\pi d_2)^{1/2}} \exp\left(-\frac{x_2^2}{2d_2}\right) \cdot \ldots \cdot \frac{1}{(2\pi d_p)^{1/2}} \exp\left(-\frac{x_p^2}{2d_p}\right)$$

$$= \prod_{i=1}^{p} \frac{1}{\sqrt{2\pi d_i}} \exp\left(-\frac{x_i^2}{2d_i}\right)$$

This shows that $X_1, X_2, \ldots, X_p$ are independent, with $X_i \sim \mathcal{N}(0, d_i)$.

Solution to 2b.

Given: $Y \sim \mathcal{N}(0, D)$ where D is diagonal with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$.

Goal: Show that $QY \sim \mathcal{N}(0, \Sigma)$ where $\Sigma = QDQ^T$.

Let $X = Q^T Y$, which means $Y = QX$ (since $Q^T = Q^{-1}$ for orthonormal Q).

We want to find the distribution of X.

1. Change of Variables

For the transformation $x = Q^T y$ (equivalently, $y = Qx$), we use:

$$f_X(x) = \left|\det\left(\frac{dy}{dx}\right)\right| f_Y(y)$$

The Jacobian is:

$$\frac{dy}{dx} = Q$$

$$\det\left(\frac{dy}{dx}\right) = \det(Q)$$

2. Determinant of Orthonormal Matrix

Since Q is orthonormal: $QQ^T = I$

Taking determinants:

$$\det(QQ^T) = \det(I)$$

$$\det(Q)\det(Q^T) = 1$$

$$[\det(Q)]^2 = 1$$

$$\det(Q) = \pm 1$$

Therefore: $|\det(Q)| = 1$

3. Apply the Transformation

$$f_X(x) = 1 \cdot f_Y(Qx)$$

$$f_X(x) = \frac{1}{(2\pi)^{p/2}(\lambda_1 \cdot \lambda_2 \cdot \ldots \cdot \lambda_p)^{1/2}} \exp\left(-\frac{1}{2}(Qx)^T D^{-1}(Qx)\right)$$

$$= \frac{1}{(2\pi)^{p/2}|D|^{1/2}} \exp\left(-\frac{1}{2}x^T Q^T D^{-1} Qx\right)$$

4.Simplify using $\Sigma = QDQ^T$

From the spectral decomposition:

$$\Sigma = QDQ^T$$

$$\Sigma^{-1} = QD^{-1}Q^T$$

(since $(QDQ^T)^{-1} = (Q^T)^{-1}D^{-1}Q^{-1} = QD^{-1}Q^T$)

Also:

$$\det(\Sigma) = \det(Q)\det(D)\det(Q^T) = \det(Q)^2\det(D) = 1 \cdot \det(D) = \lambda_1 \cdot \lambda_2 \cdot \ldots \cdot \lambda_p$$

5. Final Result

$$f_X(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}x^T \Sigma^{-1} x\right)$$

This is the PDF of $\mathcal{N}(0, \Sigma)$.

6. Conclusion

Since $X = Q^T Y$ and $X \sim \mathcal{N}(0, \Sigma)$, we have:

$$Y = QX \sim \mathcal{N}(0, \Sigma)$$

Therefore: $\boxed{QY \sim \mathcal{N}(0, \Sigma)}$, which means $X \sim QY$ where $X \sim \mathcal{N}(0, \Sigma)$.

Solution to 2c.

Given: $X \sim \mathcal{N}(0, \Sigma)$ and $w \in \mathbb{R}^p$.

Goal: Show that $w \cdot X \sim \mathcal{N}(0, w^T \Sigma w)$.

1: Use the spectral decomposition

Let $\Sigma = QDQ^T$ where Q is orthonormal and $D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$.

Choose $U = Q^T$ (which is orthonormal since Q is).

Choosing $U = Q^T$ to make Y have independent entries

Let $\Sigma = QDQ^T$ be the spectral decomposition and set $U = Q^T$.

Let $Y = UX = Q^T X$. The covariance matrix of Y is:

$$\text{Cov}(Y) = \text{Cov}(Q^T X) = Q^T \cdot \text{Cov}(X) \cdot (Q^T)^T = Q^T \Sigma Q$$

$$= Q^T (QDQ^T) Q = (Q^T Q) D (Q^T Q) = I \cdot D \cdot I = D$$

Since D is diagonal, $Y_1, Y_2, \ldots, Y_p$ are independent with $Y_i \sim \mathcal{N}(0, \lambda_i)$.

2: Show that $w \cdot X = (Uw) \cdot (UX)$

$$w \cdot X = w^T X$$

Since $U^T U = I$:

$$w^T X = w^T (U^T U) X = (Uw)^T (UX)$$

Therefore:

$$w \cdot X = (Uw) \cdot (UX)$$

4: Express $w \cdot X$ as a sum of independent normals

Let $c = Uw = Q^T w = (c_1, c_2, \ldots, c_p)^T$. Then:

$$w \cdot X = c \cdot Y = c^T Y = c_1 Y_1 + c_2 Y_2 + \cdots + c_p Y_p$$

Since the components are independent:

$$Y_i \sim \mathcal{N}(0, \lambda_i)$$

$$c_i Y_i \sim \mathcal{N}(0, c_i^2 \lambda_i)$$

$$\sum_{i=1}^{p} c_i Y_i \sim \mathcal{N}\left(0, \sum_{i=1}^{p} c_i^2 \lambda_i\right) \quad \text{(as independent)}$$

6: Compute the variance

$$\sum_{i=1}^{p} c_i^2 \lambda_i = c^T D c = (Q^T w)^T D (Q^T w)$$

b

$$= w^T Q D Q^T w = w^T \Sigma w$$

$$\boxed{w \cdot X \sim \mathcal{N}(0, w^T \Sigma w)}$$

This shows that any linear projection of a multivariate normal is a univariate normal.