# Project Report on Queuing Systems

Markov Mavericks

March 6, 2025

## Contents

# 1 Introduction

## 1.1 Background

Queuing systems are mathematical models used to describe systems where "jobs" (which can be tasks, data packets, etc.) arrive, wait, and are served according to certain rules. These systems are vital in various fields such as telecommunications, computer networks, manufacturing, healthcare, and traffic flow analysis.

In the context of queuing theory, Erlang distributions and Phase-type distributions are important for modeling the arrival and service times in a queuing system. The Erlang distribution is often used to model the time between events in a system, especially when events occur at a constant rate but with some variability. Phase-type distributions, on the other hand, offer a flexible way to model complex service processes, extending exponential distributions to model systems with multiple stages or phases.

Analyzing these systems allows researchers and practitioners to assess the system's performance, such as average waiting time, queue length, system stability, and the likelihood of congestion. Accurate modeling helps optimize resources, reduce delays, and improve service efficiency.

## 1.2 Scope and Objectives

The scope of this project is to explore the behavior and characteristics of Erlang and Phase-type distributions in queuing systems. Specifically, the focus will be on the M/Er/1 queuing model, where the arrival process follows a Markovian (Poisson) distribution, service times follow an Erlang distribution, and there is a single server.

The objectives of this project are as follows:

1. **To model the M/Er/1 queue**: Investigate the mathematical foundation of this queue, which combines the simplicity of exponential arrivals and Erlang-distributed service times.

2. **To study the behavior of Erlang distributions**: Discuss how the Erlang distribution, with its integer-shaped structure, can be used to model service times with multiple phases or stages.

3. **To explore Phase-type distributions**: Understand how Phase-type distributions generalize Erlang distributions and how they are applied in complex queuing systems.

4. **To analyze system performance**: Derive important performance metrics such as waiting time, sojourn time, and stability conditions.

5. **To implement and simulate the system**: Simulate the queuing system and compare simulation results with analytical predictions.

# 2 Erlang Distribution and Phase-type Distributions

## 2.1 Erlang Distribution

The Erlang distribution is a special case of the Gamma distribution where the shape parameter $k$ is an integer. It is often used to model the service time in queuing systems where there are multiple stages of service. In an Erlang-distributed process, the time between arrivals (or service completions) is divided into $k$ phases, each of which is exponentially distributed with mean $1/\lambda$. The overall service time is the sum of these phases.

The probability density function (PDF) for the Erlang distribution is:

$$f(x; k, \lambda) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}$$

where: - $k$ is the number of phases (a positive integer), - $\lambda$ is the rate parameter (mean arrival rate), - $x$ is the time.

## 2.2 Phase-type Distribution

A Phase-type distribution generalizes the Erlang distribution to allow for more general service processes, which could involve several stages or phases. It is defined as the time until absorption in a Markov process with $k$ phases. The flexibility of Phase-type distributions makes them suitable for modeling systems with multiple stages of service or transition.

A Phase-type distribution is often described using a matrix representation of the transition rates between states in the system, where the phases represent different service stages. Phase-type distributions can approximate any distribution that is non-negative and unimodal.

## 2.3 M/Er/1 Queue

The M/Er/1 queue is a queuing model where:

- **M** denotes a Markovian (memoryless) arrival process, i.e., the inter-arrival times follow an exponential distribution (Poisson process).
- **Er** indicates Erlang-r distributed service times, where the service time is the sum of multiple exponentially distributed random variables.
- **1** indicates a single server.

# 3 Queuing System Modeling

## 3.1 CTMC/DTMC Modeling

In modeling queuing systems, the state of the system can be described in different ways, depending on how the phases of work and the number of jobs are considered.

### 3.1.1 Two Alternative State Descriptions

The natural way to describe the state of a nonempty system is by the pair $(k, l)$, where:

- $k$ denotes the number of jobs in the system.
- $l$ denotes the remaining number of service phases of the job currently being served.

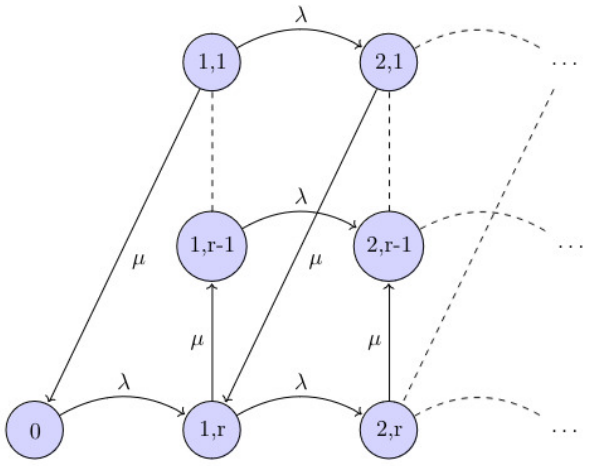


Figure 1: State diagram using a two state variables.

This pair provides a two-dimensional description of the system's state.

An alternative way to describe the system's state is by counting the total number of uncompleted phases of work in the system. There is a one-to-one correspondence between this description and the pair $(k, l)$. The number of uncompleted phases of work in the system is given by:

$$(k-1)r + l,$$

where $r$ is the number of phases per job, and for the job in service, there are $l$ phases of work instead of $r$.

From this point onwards, we will work with the one-dimensional phase description.

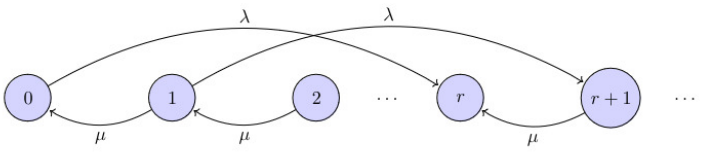


Figure 2: State diagram using a single state variable.

## 3.2 Stationary Distribution

To derive the stationary distribution for the modeled system, we begin by defining key terms and understanding the transitions within the queuing system.

### 3.2.1 State Space Definition

Let the state of the system be defined by the total number of uncompleted phases of service in the system. The system can be in any of the following states:

$$n = 0, 1, 2, \ldots,$$

where $n$ represents the total number of uncompleted phases of service in the system. At each instant, a job is either in the process of being served or waiting in the queue, and these phases of service progress in discrete steps.

### 3.2.2 Defining $X_r$: Total Number of Uncompleted Phases

Let $X_r$ denote the total number of uncompleted phases in the system. Initially, when the system is empty, $X_r = 0$, and when there are jobs in the system, $X_r$ represents the total sum of uncompleted phases, accounting for both the service phases of jobs in the queue and the phases of the job being served.

Thus, the state space $S$ consists of the integer values:

$$S = \{0, 1, 2, \ldots\}.$$

Each value of $n$ in this state space corresponds to the total number of uncompleted phases in the system at any given time.

### 3.2.3 Transitions in the System

The transitions in the system are governed by the arrival and service processes.

- **Arrival Transition:**
  When a new job arrives at the system (at a rate $\lambda$), the number of uncompleted phases increases by $r$. This is because each new job arrives with a service requirement of $r$ phases. Hence, an arrival causes the state to transition from $n$ to $n + r$:

$$\text{Arrival Transition: } n \to n + r \quad \text{with rate } \lambda.$$

- **Service Completion Transition:**
  When the job in service completes a phase (at a rate $\mu$), the number of uncompleted phases in the system decreases by one. If the current number of uncompleted phases is $n$, and if the system is non-empty, the state transitions from $n$ to $n-1$ with rate $\mu$. For example, if the system is in state $r$ (i.e., the job in service has $r$ remaining phases), upon completion of a phase, the system moves to state $r-1$:

$$\text{Service Completion Transition: } n \to n-1 \quad \text{with rate } \mu \quad \text{for } n > 0.$$

### 3.2.4 Transition Diagram

The flow of the system can be visualized as follows:

- Each arrival increases the number of uncompleted phases by $r$, transitioning from state $n$ to state $n+r$.
- Each phase completion decreases the total number of uncompleted phases by 1, causing a transition from state $n$ to $n-1$.

This cycle continues as new arrivals enter the system and service phases are completed.

### 3.2.5 Balance Equations and Derivation of the Stationary Distribution

In steady state, the rate of flow into each state equals the rate of flow out of that state. Denote by $\pi_n$ the stationary probability of being in state $n$.

**Balance Equations:**

- For $n = 0$ (empty system), the only way to leave is via an arrival:

$$\lambda\pi_0 = \mu\pi_1.$$

- For $1 \le n \le r-1$, the balance equation is:

$$\lambda\pi_n + \mu\pi_n = \mu\pi_{n+1}.$$

- For $n \ge r$, the balance equation accounts for arrivals from state $n-r$ and departures to state $n+1$:

$$\lambda\pi_n + \mu\pi_n = \lambda\pi_{n-r} + \mu\pi_{n+1}.$$

A unified way to express the balance for $n \ge 1$ is:

$$(\lambda+\mu)\pi_n = \lambda\pi_{n-r} + \mu\pi_{n+1}.$$

Now we use the following method of solving linear equations in which we assume that:

$$\pi_n = x^n \quad \forall\, n \in \{0, 1, \dots\}.$$

Substituting and simplifying we get the following $r+1$ degree polynomial in $x$:

$$\mu x^{r+1} - (\lambda + \mu)\, x^r + \lambda \;=\; 0,$$

or equivalently, we have:

$$(\lambda + \mu) - \mu x \;=\; \frac{\lambda}{x^r}.$$

**Proof of uniqueness of each of the $r+1$ roots of the above equation:**

- First, it is trivial to note that $x = 1$ is a solution while $x = 0$ is not a solution.

Now let us assume that $\frac{1}{x}$ is a root of the equation, for any $x$ such that $|x| > 1$:

$$\mu\Big(\tfrac{1}{x}\Big)^{r+1} \;-\; (\lambda+\mu)\,\Big(\tfrac{1}{x}\Big)^{r} \;+\; \lambda \;=\; 0 \iff \mu - (\lambda+\mu)\,x + \lambda\, x^{r+1} = 0$$

$$\iff \quad \mu\,(1-x) - \lambda\, x\,\big(1 - x^r\big) = 0$$

$$\iff \mu - \lambda\, x\, \frac{1-x^r}{1-x} = 0$$

$$\iff 1 \;-\; \frac{\lambda x}{\mu}\, \frac{1-x^r}{1-x} = 0.$$

Let

$$f(x) \;=\; 1 \;-\; \frac{\lambda x}{\mu}\, \frac{1-x^r}{1-x}.$$

We can rewrite $f(x)$ as

$$f(x) \;=\; 1 \;-\; \frac{\lambda x}{\mu}\big(1 + x + x^2 + \cdots + x^{r-1}\big) \;=\; 1 \;-\; \frac{\rho}{r}\big(x + x^2 + x^3 + \cdots + x^r\big),$$

where $\rho = \frac{\lambda}{\mu}$.

For the distinctness of the roots, it must be true that the derivative of the function is not zero at any of the roots. To this end, we calculate the derivative of $f(x)$:

$$f'(x) \;=\; -\frac{\rho}{r}\big(1 + 2x + 3x^2 + \cdots + r\, x^{r-1}\big).$$

This function equals zero if and only if

$$1 + 2x + 3x^2 + \cdots + r\, x^{r-1} = 0.$$

Note that $x = 1$ is *not* a solution to this equation. We can therefore multiply both sides of the equation by $(1-x)$ without increasing the algebraic multiplicity of the original roots. This gives

$$1 + x + x^2 + \cdots + x^{r-1} \; - \; r\, x^r = 0,$$

which is equivalent to

$$1 + x + x^2 + \cdots + x^{r-1} = r\, x^r.$$

Suppose $x$ is a root of this equation with $|x| > 1$. Then it follows that

$$\left|1+x+x^2+\cdots+x^{r-1}\right| \; \leq \; |1|+|x|+|x^2|+\cdots+|x^{r-1}| \; \leq \; 1+(r-1)\,|x^r| \; < \; r\,|x^r|.$$

This shows a contradiction and hence $f'(x)$ cannot be zero at any root. Hence, $|x| <= 1$ and all the roots are unique.

**Generating Function:** The generating function of the stationary distribution is defined as:

$$f(z) \; = \; \sum_{n=0}^{\infty} \pi_n z^n, \quad |z| < 1.$$

By multiplying balance equation by $z^n$ and summing over all $n \geq 1$, we obtain

$$(\lambda + \mu) \sum_{n=1}^{\infty} \pi_n \, z^n \; = \; \lambda \sum_{n=1}^{\infty} \pi_{n-r} \, z^n \; + \; \mu \sum_{n=1}^{\infty} \pi_{n+1} \, z^n.$$

Therefore,

$$(\lambda+\mu)\Big( \sum_{n=0}^{\infty} \pi_n \, z^n \; - \; \pi_0 \Big) \; = \; \lambda \, z^r \sum_{n=0}^{\infty} \pi_n \, z^n \; + \; \mu \, z^{-1} \Big( \sum_{n=0}^{\infty} \pi_n \, z^n \; - \; \pi_1 \, z \; - \; \pi_0 \Big).$$

Substituting $f(z) = \sum_{n=0}^{\infty} \pi_n \, z^n$ into the above yields

$$(\lambda + \mu) \, \big( f(z) - \pi_0 \big) \; = \; \lambda \, z^r \, f(z) \; + \; \mu \, z^{-1} \Big( f(z) - \pi_1 \, z - \pi_0 \Big),$$

which rearranges to

$$(\lambda + \mu) \, f(z) \; - \; (\lambda + \mu) \, \pi_0 \; = \; \lambda \, z^r \, f(z) \; + \; \mu \, z^{-1} \, f(z) \; - \; \mu \, \pi_1 \, z \; - \; \mu \, z^{-1} \, \pi_0.$$

Hence,

$$(\lambda + \mu) \, f(z) \; - \; \lambda \, z^r \, f(z) \; - \; \mu \, z^{-1} \, f(z) \; = \; (\lambda + \mu) \, \pi_0 \; - \; \mu \, \pi_1 \, z \; - \; \mu \, z^{-1} \, \pi_0,$$

yielding

$$f(z)\left(\lambda+\mu\ -\ \lambda\,z^{r}\ -\ \mu\,z^{-1}\right)\ =\ (\lambda+\mu)\,\pi_0\ -\ \mu\,\pi_1\,z\ -\ \mu\,z^{-1}\,\pi_0.$$

This gives

$$f(z)\ =\ \frac{(\lambda+\mu)\,\pi_0\ -\ \mu\,\pi_1\,z\ -\ \mu\,z^{-1}\,\pi_0}{\lambda+\mu\ -\ \lambda\,z^{r}\ -\ \mu\,z^{-1}}.$$

Dividing both sides by $-z$ (and applying the balance equation for 0th state then leads to a final expression for $f(z)$).

$$f(z)=\frac{-z(\lambda+\mu)\pi_0+z\lambda\pi_0+\mu\pi_0}{-z\lambda-z\mu+\lambda z^{r+1}+\mu}.$$

$$\begin{aligned}
f(z) &= \frac{\mu\,\pi_0\,(1-z)}{\lambda\,z\,(z^{r}-1)\ +\ \mu\,(1-z)} \\
&= \frac{\mu\,\pi_0\,(1-z)}{\mu\,(1-z)\ -\ \lambda\,z\,(z^{r}-1)} \\
&= \frac{\mu\,\pi_0}{\mu\ -\ \lambda\,z\,\frac{z^{r}-1}{1-z}} \\
&= \frac{\mu\,\pi_0}{\mu\ -\ \lambda\,z\,\frac{1-z^{r}}{1-z}} \\
&= \frac{\mu\,\pi_0}{\mu\ -\ \lambda\,z\,(1+z+\cdots+z^{r-1})} \\
&= \frac{1-\rho}{1\ -\ \frac{\rho}{r}\left(z+z^{2}+\cdots+z^{r}\right)},
\end{aligned}$$

where we have inserted $\rho=\frac{\lambda}{\mu}$ and $\pi_0=1-\rho$.

$$\mu x^{r+1}-(\lambda+\mu)x^{r}+\lambda=0$$

$x=0$ is not a solution. Divide by $x^{r+1}$

$$\mu-(\lambda+\mu)x^{-1}+\lambda x^{-(r+1)}=0$$

Substitute $x^{-1}\equiv z$ To get the denominator of $f(z)$ Hence,

$$z_i=\frac{1}{x_i}\quad\forall i$$

From the uniqueness proof, it follows that the denominator of $f(z)$ has $r$ distinct roots $z_i$ with $|z_i| > 1$. Note that each root $z_i$ corresponds to a root $\frac{1}{z_i} = x_i$ of the general balance equation. We can thus write

$$f(z) \;=\; \frac{1-\rho}{\big(1-\frac{z}{z_1}\big)\cdots\big(1-\frac{z}{z_r}\big)}.$$

Using partial-fraction decomposition, this becomes

$$f(z) \;=\; \frac{1-\rho}{\big(1-\frac{z}{z_1}\big)\cdots\big(1-\frac{z}{z_r}\big)} \;=\; (1-\rho)\Big(\frac{A_1}{1-\frac{z}{z_1}} \;+\; \cdots + \;\frac{A_r}{1-\frac{z}{z_r}}\Big),$$

where

$$A_i \;=\; \Big(\prod_{j\neq i}\big(1-\tfrac{z_i}{z_j}\big)\Big)^{-1}, \quad i = 1,\ldots,r.$$

Each $A_i$ is well-defined since all $z_i$ are distinct. This completes the factorization of $f(z)$.

Furthermore, we can make use of the fact that

$$\sum_{n=1}^{\infty} z^n = \frac{1}{1-z}, \quad \text{for } z \in (0,1).$$

This allows us to rewrite f(z) as

$$f(z) = (1-\rho)\Big(A_1\sum_{n=0}^{\infty}\Big(\frac{z}{z_1}\Big)^n + \cdots + A_r\sum_{n=0}^{\infty}\Big(\frac{z}{z_r}\Big)^n\Big) = (1-\rho)\sum_{n=0}^{\infty}\Big(\sum_{i=1}^{r}A_i\Big(\frac{1}{z_i}\Big)^n\Big)z^n.$$

From this and generating function it follows that the stationary distribution of the Erlang queue is

$$\pi_n = (1-\rho)\sum_{i=1}^{r}A_i\Big(\frac{1}{z_i}\Big)^n, \quad \text{for } n > 0.$$

As $\frac{1}{z_i} = x_i$, it follows that

$$\pi_n = (1-\rho)\sum_{i=1}^{r}A_i\,x_i^n.$$

**Alternative Representation:** Note that the same formula can be written as

$$\pi_n = \sum_{i=1}^{r} c_i x_i^n,$$

where

$$c_i = (1-\rho)A_i.$$

Thus, the stationary distribution in terms of the coefficients $c_i$ is given by:

$$\pi_n = \sum_{i=1}^{r} c_i x_i^n.$$

This completes the derivation of the stationary distribution for the Erlang queue in a formal manner.

# 4 Stability Conditions

For the M/$E_r$/1, we sample the process at departure epochs (times immediately after a job finishes service).

## 4.1 Evolution of the Embedded Chain

### 4.1.1 State Definition

Let $X_n$ be the number of jobs in the system immediately after the $n$th departure.

### 4.1.2 Transition Mechanism

Between the $n$th and $(n+1)$th departures:

- **Departure:** One job leaves (if the system is not empty).
- **Arrivals:** During the service time of the departing job, a random number $A_n$ of jobs arrive.
- Because the service time is Erlang (with mean $r/\mu$) and arrivals are Poisson with rate $\lambda$, the number $A_n$ of arrivals during one service time is negative binomial distribution with mean

$$\frac{\lambda\, r}{\mu}.$$

**Proof of 4.1.2 :**

**Starting Point: Conditional Probability**

We start with the conditional probability that, given a service time $s$, there are $n$ arrivals:

$$P(A_n = n \mid S = s) \;=\; e^{-\lambda s}\,\frac{(\lambda s)^n}{n!}.$$

The service time $S$ is Erlang distributed with parameters $r$ (number of phases) and $\mu$ (rate per phase). Hence, its probability density function is

$$f_S(s) \;=\; \frac{\mu^r\, s^{r-1}\, e^{-\mu s}}{(r-1)!}, \quad s \geq 0.$$

**Unconditioning on $s$**

To find the unconditional probability $P(A_n = \text{n})$, we integrate over all possible service times:

$$P(A_n = n) \;=\; \int_0^\infty P(A_n \mid S = s)\, f_S(s)\, ds.$$

Substituting the given expressions, we get

$$P(A_n = n) \;=\; \int_0^\infty \Big[e^{-\lambda s}\,\frac{(\lambda s)^n}{n!}\Big]\Big[\frac{\mu^r\, s^{r-1}\, e^{-\mu s}}{(r-1)!}\Big]\, ds.$$

**Combining Exponentials and Powers of $s$**

Combining the exponentials and powers of $s$ in the integrand gives

$$P(A_n = n) \;=\; \frac{\lambda^n\,\mu^r}{n!\,(r-1)!}\int_0^\infty s^{n+r-1}\, e^{-(\lambda+\mu)\,s}\, ds.$$

**Evaluating the Integral**

The integral

$$\int_0^\infty s^{n+r-1}\, e^{-(\lambda+\mu)\,s}\, ds$$

is recognized as the Gamma function:

$$\int_0^\infty s^{n+r-1}\, e^{-(\lambda+\mu)\,s}\, ds \;=\; \frac{\Gamma(n+r)}{(\lambda+\mu)^{n+r}},$$

where $\Gamma(n+r) = (n+r-1)!$ when $n+r$ is a positive integer. Thus,

$$P(A_n = n) \;=\; \frac{\lambda^n\,\mu^r}{n!\,(r-1)!}\;\cdot\;\frac{\Gamma(n+r)}{(\lambda+\mu)^{n+r}} \;=\; \frac{\lambda^n\,\mu^r}{n!\,(r-1)!}\;\cdot\;\frac{(n+r-1)!}{(\lambda+\mu)^{n+r}}.$$

Hence,

$$P(A_n = n) \;=\; \frac{\lambda^n\,\mu^r\,\big(n+r-1\big)!}{n!\,(r-1)!\,\big(\lambda+\mu\big)^{n+r}}.$$

**Recognizing the Negative Binomial Form**

Substitute $\Gamma(n+r) = (n+r-1)!$ into the expression:

$$P(A_n = n) \;=\; \frac{\lambda^n\,\mu^r\,(n+r-1)!}{n!\,(r-1)!\,\big(\lambda+\mu\big)^{n+r}}.$$

Observe that

$$\frac{(n+r-1)!}{n!\,(r-1)!} \;=\; \binom{n+r-1}{n}.$$

Therefore,

$$P(A_n = n) \;=\; \binom{n+r-1}{n}\Big(\frac{\mu}{\lambda+\mu}\Big)^r\Big(\frac{\lambda}{\lambda+\mu}\Big)^n.$$

## Deriving the Mean of a Negative Binomial($r$, $p$) Distribution

**Goal:** We want to show that

$$\mathbb{E}[A_n] \;=\; \sum_{n=0}^{\infty} n\, P(A_n = n) \;=\; r\,\frac{\lambda}{\mu}.$$

Below is a step-by-step derivation using the summation definition of expectation.

**Step 1. Set Up the Expectation**
Write the expected value as

$$\mathbb{E}[A_n] \;=\; \sum_{n=0}^{\infty} n\binom{n+r-1}{n}\left(\tfrac{\mu}{\lambda+\mu}\right)^{r}\left(\tfrac{\lambda}{\lambda+\mu}\right)^{n}.$$

Define

$$p \;=\; \frac{\mu}{\lambda+\mu} \quad \text{and} \quad q \;=\; \frac{\lambda}{\lambda+\mu}.$$

Then the PMF of $A_n$ becomes

$$P(A_n = n) \;=\; \binom{n+r-1}{n} p^{r}\, q^{n},$$

and thus

$$\mathbb{E}[A_n] \;=\; p^{r} \sum_{n=0}^{\infty} n\binom{n+r-1}{n} q^{n}.$$

**Step 2. Recognize a Generating Function**
Consider the generating function

$$S(q) \;=\; \sum_{n=0}^{\infty} \binom{n+r-1}{n} q^{n}.$$

A standard result tells us that

$$S(q) \;=\; \frac{1}{(1-q)^{r}}, \quad \text{for } |q| < 1.$$

Differentiate $S(q)$ with respect to $q$:

$$S'(q) \;=\; \frac{d}{dq}\left[(1-q)^{-r}\right] \;=\; r\,(1-q)^{-r-1}.$$

Also note that

$$S'(q) \;=\; \sum_{n=0}^{\infty} n\binom{n+r-1}{n} q^{\,n-1}.$$

Multiplying by $q$ gives

$$q\,S'(q) \;=\; \sum_{n=0}^{\infty} n\binom{n+r-1}{n} q^{\,n}.$$

Hence, we can write

$$\mathbb{E}[A_n] \;=\; p^r\,q\,S'(q).$$

**Step 3. Substitute and Simplify**
We already found

$$S'(q) \;=\; r\,(1-q)^{-r-1}.$$

Substitute this into the expression for $\mathbb{E}[A_n]$:

$$\mathbb{E}[A_n] \;=\; p^r\,q\,r\,(1-q)^{-r-1} \;=\; r\,q\,p^r\,(1-q)^{-r-1}.$$

Recall our definitions:

$$p \;=\; \frac{\mu}{\lambda+\mu}, \quad q \;=\; \frac{\lambda}{\lambda+\mu}, \quad 1-q \;=\; \frac{\mu}{\lambda+\mu} \;=\; p.$$

Thus,

$$\mathbb{E}[A_n] \;=\; r\left(\frac{\lambda}{\lambda+\mu}\right) p^r\,p^{-r-1} \;=\; r\left(\frac{\lambda}{\lambda+\mu}\right)\frac{1}{p}.$$

But

$$\frac{1}{p} \;=\; \frac{\lambda+\mu}{\mu} \quad\Longrightarrow\quad \mathbb{E}[A_n] \;=\; r\,\frac{\lambda}{\mu}.$$

This completes the proof that

$$\mathbb{E}[A_n] \;=\; r\,\frac{\lambda}{\mu}.$$

### 4.1.3 State Update Equation

The next state is given by

$$X_{n+1} \;=\; \max\{X_n - 1,\, 0\} \;+\; A_n.$$

The "max" function ensures that if the system is empty (i.e. $X_n = 0$), we do not subtract below zero.

## 4.2 Positive Recurrence

A Markov chain is *positive recurrent* if it returns to a given state (or a set of states) in finite expected time. In queuing systems, positive recurrence implies that the queue does not grow without bound and a stationary distribution exists.

## 4.3 Lyapunov Function and Drift Analysis

### 4.3.1 Lyapunov Function

**Definition** A Lyapunov function is a tool to measure the "size" or "energy" of a system. In our proof, we choose

$$V(x) \;=\; x,$$

where $x$ is the number of jobs in the system.

### 4.3.2 Drift Calculation

**Drift Definition** The *drift* is the expected change in the Lyapunov function in one step, formally:

$$\mathbb{E}\big[V(X_{n+1}) - V(X_n) \mid X_n = x\big].$$

**Purpose** The idea is to show that, on average, the "size" of the system decreases when the system is large enough (i.e., there is a negative "drift").

### 4.3.3 Application to Our Embedded Chain

**Criterion Statement** If there exists a function $V(x)$ and a finite set $\mathcal{C}$ such that for all states $x$ outside $\mathcal{C}$, the drift

$$\mathbb{E}\big[V(X_{n+1}) - V(X_n) \mid X_n = x\big]$$

is negative, then the Markov chain is positive recurrent. This is also known as **The Foster's Criterion**.

**Our Situation** With $V(x) = x$, we know that for all $x \geq 1$,

$$\mathbb{E}\big[V(X_{n+1}) - V(X_n) \mid X_n = x\big] \;=\; -1 \;+\; \frac{\lambda\, r}{\mu} \;<\; 0,$$

provided that $\frac{\lambda\, r}{\mu} < 1$. This is because there will be job arrival according to the negative binomial distribution described above and 1 job departure.
Hence, this implies that once the number of jobs gets large, the expected change is a decrease, pulling the system back toward a smaller number of jobs. Thus, the criterion holds.

**Finite Set $\mathcal{C}$** One can consider a set $\mathcal{C} = \{0, 1, \ldots, K\}$ for some finite $K$. Inside this set, the drift condition may not hold, but that is acceptable. The key is that for $x$ outside this set (when the queue is sufficiently large), the drift is negative. In our case, this set is $\mathcal{C} = \{0\}$.

# 5 Performance Metrics

## 5.1 Detailed Derivation of the Mean Waiting Time in an $M/E_r/1$ Queue

In an $M/E_r/1$ queue the arrivals follow a Poisson process with rate $\lambda$, and each job requires service that consists of $r$ sequential exponential phases, each with rate $\mu$. Define the system utilization as

$$\rho = \frac{\lambda\, r}{\mu} < 1.$$

Let:

- $E(L^q)$ be the mean number of jobs waiting in the queue (excluding the one in service),
- $E(W)$ be the mean waiting time before service begins,
- $R$ be the residual service time of the job in service (if any).

The derivation proceeds by decomposing the waiting time of an arriving job into two parts:

**1. Waiting Behind Other jobs:**
If there are $E(L^q)$ jobs ahead in the queue, each of these jobs requires service comprising $r$ phases. Since the service rate per phase is $\mu$, the average time to complete one job is

$$\frac{r}{\mu}.$$

Thus, the waiting time contributed by the jobs ahead is

$$\frac{r}{\mu}\, E(L^q).$$

**2. Waiting for the Residual Service Time:**
When an arrival finds the server busy (which happens a fraction $\rho$ of the time), the job must also wait for the remaining (or residual) service time of the job in service. Since the service process is broken into $r$ phases and each phase is exponential with rate $\mu$, we note the following:

- The server is equally likely to be in any one of the $r$ phases (each with probability $1/r$).
- If the server is in phase $k$, then there are $k$ phases left (including the phase in progress).
- The expected remaining time in that case is $\frac{k}{\mu}$.

Averaging over the $r$ phases gives:

$$E(R) = \frac{1}{r}\sum_{k=1}^{r}\frac{k}{\mu} = \frac{1}{r}\,\frac{1}{\mu}\,(1+2+\cdots+r) = \frac{1}{r}\,\frac{1}{\mu}\,\frac{r(r+1)}{2} = \frac{r+1}{2\mu}.$$

Thus, when the server is busy (which occurs with probability $\rho$), the waiting time due to the residual service is $\rho\,E(R)$.

**3. Combining Both Components:**
An arriving job experiences a total mean waiting time that is the sum of the waiting time due to the jobs ahead and the waiting due to the residual service:

$$E(W) \;=\; \frac{r}{\mu}\,E(L^q) \;+\; \rho\,E(R).$$

Substitute the expression for $E(R)$ into the equation:

$$E(W) \;=\; \frac{r}{\mu}\,E(L^q) \;+\; \rho\,\frac{r+1}{2\mu}.$$

**4. Application of Little's Law:**
Little's law for the queue (waiting area only) states that

$$E(L^q) = \lambda\,E(W).$$

Substitute this into the waiting time equation:

$$E(W) = \frac{r}{\mu}\big[\lambda\,E(W)\big] + \rho\,\frac{r+1}{2\mu}.$$

Recall that the utilization $\rho$ is defined as

$$\rho = \frac{\lambda\,r}{\mu},$$

so that $\frac{r}{\mu}\lambda = \rho$. Therefore, the equation simplifies to:

$$E(W) = \rho\,E(W) + \rho\,\frac{r+1}{2\mu}.$$

**5. Solving for $E(W)$:**
Rearrange the equation to isolate $E(W)$:

$$E(W) - \rho\,E(W) = \rho\,\frac{r+1}{2\mu} \quad\Longrightarrow\quad (1-\rho)\,E(W) = \rho\,\frac{r+1}{2\mu}.$$

Dividing both sides by $(1-\rho)$ yields the final result:

$$\boxed{E(W) = \frac{\rho}{1-\rho}\,\frac{r+1}{2\mu}}.$$

This completes the detailed derivation of the mean waiting time in an $M/E_r/1$ queue.

## 5.2 Mean Sojourn Time in an $M/E_r/1$ Queue

Recall from the previous derivation that the mean waiting time is

$$E(W) \;=\; \frac{\rho}{1-\rho}\,\frac{r+1}{2\,\mu}, \quad \text{where} \quad \rho \;=\; \frac{\lambda\, r}{\mu} < 1.$$

Since each job requires $r$ exponential phases (each with rate $\mu$), the *mean service time* is

$$E(S) \;=\; \frac{r}{\mu}.$$

Hence, the *sojourn time* $T$ (waiting plus service) has mean

$$E(T) \;=\; E(W) + E(S) \;=\; \frac{\rho}{1-\rho}\,\frac{r+1}{2\,\mu} \;+\; \frac{r}{\mu}.$$

We can write this more compactly as

$$\boxed{E(T) \;=\; \frac{r}{\mu} \;+\; \frac{\rho\,(r+1)}{2\,\mu\,\big(1-\rho\big)}}.$$

This completes the derivation of the mean sojourn time in an $M/E_r/1$ queue.

## 5.3 Mean Number of Jobs in the System Using Little's Law

Little's law for the entire system (i.e., the waiting area plus the one in service) is given by

$$E(N) \;=\; \lambda\, E(T),$$

where:

- $E(N)$ is the mean number of jobs in the system,
- $\lambda$ is the arrival rate,
- $E(T)$ is the mean sojourn (total time in system) time.

From the previous derivation, the mean sojourn time in an $M/E_r/1$ queue is

$$E(T) \;=\; E(W) + E(S) \;=\; \frac{\rho}{1-\rho}\,\frac{r+1}{2\,\mu} \;+\; \frac{r}{\mu},$$

where:

- $E(W)$ is the mean waiting time,
- $E(S) = \frac{r}{\mu}$ is the mean service time (since there are $r$ phases each with mean $1/\mu$),
- $\rho = \frac{\lambda\, r}{\mu}$ is the system utilization.

Thus, Little's law gives

$$E(N) \;=\; \lambda \left( \frac{r}{\mu} \;+\; \frac{\rho\,(r+1)}{2\,\mu(1-\rho)} \right).$$

**Step 1: Simplify the First Term**
Notice that

$$\lambda \frac{r}{\mu} \;=\; \frac{\lambda\, r}{\mu} \;=\; \rho.$$

**Step 2: Simplify the Second Term**
The second term is

$$\lambda \frac{\rho\,(r+1)}{2\,\mu(1-\rho)}.$$

Since $\rho = \frac{\lambda\, r}{\mu}$, we have

$$\frac{\lambda}{\mu} \;=\; \frac{\rho}{r}.$$

Substitute this into the second term:

$$\lambda \frac{\rho\,(r+1)}{2\,\mu(1-\rho)} \;=\; \frac{\rho\,(r+1)}{2\,(1-\rho)} \cdot \frac{\rho}{r} \;=\; \frac{\rho^2\,(r+1)}{2\,r\,(1-\rho)}.$$

**Step 3: Combine the Terms**
Adding the simplified terms together, we obtain:

$$E(N) \;=\; \rho \;+\; \frac{\rho^2\,(r+1)}{2\,r\,(1-\rho)}.$$

$$\boxed{E(N) \;=\; \rho \;+\; \frac{\rho^2\,(r+1)}{2\,r\,(1-\rho)}.}$$

This is the mean number of jobs in the system for an $M/E_r/1$ queue, expressed in terms of $\rho$, $r$, and $\mu$.

## 5.4 Blocking Probabilities

In finite-capacity models (e.g., $\mathrm{M/E}_r/1/K$), a new arrival might be rejected if the system is full. In an $\mathrm{M/E}_r/1/\text{infinity}$ system the capacity is infinite, which means there is no upper limit on the number of jobs that can be in the system. Consequently, every arriving job is eventually accommodated, and none are blocked.

Therefore, by definition, the blocking probability is given by:

$$\boxed{P_{\text{block}} = 0.}$$

# 6 Distribution Analysis

This section provides a detailed analysis of the waiting time and sojourn time distributions in an M/Er/1 queue. The M/Er/1 queue is a single-server queuing system where arrivals follow a Poisson process with rate $\lambda$, and service times follow an Erlang-$r$ distribution with rate $\mu$. The Erlang-$r$ distribution is a special case of the Gamma distribution, where the service time is the sum of $r$ independent and identically distributed exponential phases, each with rate $\mu$.

## 6.1 Notation and Assumptions

The following notation and assumptions are used throughout this analysis:

- $\lambda$: Arrival rate of jobs (Poisson process).
- $\mu$: Service rate for each phase of the Erlang-$r$ distribution.
- $r$: Number of phases in the Erlang service time distribution.
- $W$: Waiting time of a job in the queue before service begins.
- $S$: Service time of a job, following an Erlang-$r$ distribution.
- $T = W + S$: Sojourn time (total time a job spends in the system).
- $L_f$: Let the random variable Lf denote the number of phases of work in the system.
- $B_i$: Service time of the $i$-th phase, exponentially distributed with rate $\mu$.
- $x_k$: Distinct roots of the characteristic equation derived from the equilibrium equations.
- $c_k$: Coefficients ensuring normalization of the waiting time distribution.

## 6.2 Waiting Time Distribution

The waiting time $W$ represents the time a job spends in the queue before service begins. It can be expressed as the sum of the service times of the jobs already in the system:

$$W = \sum_{i=1}^{L_f} B_i, \tag{1}$$

where $B_i$ is the service time of the $i$-th phase. The random variables $B_i$ are independent and exponentially distributed with mean $1/\mu$.

By conditioning on $L_f$ and using the independence of $L_f$ and $B_i$, the tail probability of the waiting time $P(W > t)$ can be derived as:

$$P(W > t) = \sum_{n=1}^{\infty} P\left(\sum_{i=1}^{n} B_i > t\right) P(L_f = n)$$

Since $B_i$ are independent and exponentially distributed with rate $\mu$, the probability that the sum of $n$ service times exceeds $t$ is given by the Erlang distribution:

$$P\left(\sum_{i=1}^{n} B_i > t\right) = \sum_{i=0}^{n-1} \frac{(\mu t)^i}{i!} e^{-\mu t}.$$

Substituting the probability mass function of $L_f$:

$$P(L_f = n) = \sum_{k=1}^{r} c_k x_k^n,$$

we obtain:

$$P(W > t) = \sum_{n=1}^{\infty} \sum_{i=0}^{n-1} \frac{(\mu t)^i}{i!} e^{-\mu t} \sum_{k=1}^{r} c_k x_k^n \tag{2}$$

$$= \sum_{k=1}^{r} c_k \sum_{i=0}^{\infty} \sum_{n=i+1}^{\infty} \frac{(\mu t)^i}{i!} e^{-\mu t} x_k^n. \tag{3}$$

Rearranging the summation order:

$$P(W > t) = \sum_{k=1}^{r} c_k \sum_{i=0}^{\infty} \frac{(\mu t)^i}{i!} e^{-\mu t} \sum_{n=i+1}^{\infty} x_k^n. \tag{4}$$

The inner sum is a geometric series:

$$\sum_{n=i+1}^{\infty} x_k^n = \frac{x_k^{i+1}}{1 - x_k}, \quad \text{for } |x_k| < 1.$$

Thus,

$$P(W > t) = \sum_{k=1}^{r} c_k \sum_{i=0}^{\infty} \frac{(\mu t)^i}{i!} e^{-\mu t} \frac{x_k^{i+1}}{1 - x_k} \tag{5}$$

$$= \sum_{k=1}^{r} c_k \frac{x_k}{1 - x_k} \sum_{i=0}^{\infty} \frac{(\mu t x_k)^i}{i!} e^{-\mu t}. \tag{6}$$

Recognizing the last sum as the moment-generating function of a Poisson-distributed random variable:

$$\sum_{i=0}^{\infty} \frac{(\mu t x_k)^i}{i!} e^{-\mu t} = e^{-\mu t} e^{\mu t x_k} = e^{-\mu(1-x_k)t},$$

we obtain:

$$P(W > t) = \sum_{k=1}^{r} c_k \frac{x_k}{1 - x_k} e^{-\mu(1-x_k)t}, \quad t \geq 0. \tag{7}$$

This final expression gives the tail probability of the waiting time in terms of the characteristic roots $x_k$ and their associated coefficients $c_k$.

### 6.2.1 Erlang Service Time Distribution

The service time $S$ follows an Erlang-$r$ distribution with rate $\mu$. The probability density function (PDF) of $S$ is given by:

$$f_S(s) = \frac{\mu^r s^{r-1} e^{-\mu s}}{(r-1)!}, \quad s \geq 0. \tag{8}$$

The survival function of $S$, which gives the probability that the service time exceeds $t$, is:

$$P(S > t) = \sum_{j=0}^{r-1} \frac{(\mu t)^j}{j!} e^{-\mu t}. \tag{9}$$

## 6.3 Sojourn Time Distribution

The sojourn time $T$ represents the total time a job spends in the system, including both waiting time and service time. It is given by:

$$T = W + S. \tag{10}$$

The goal is to derive the probability distribution of $T$, specifically $P(T > t)$, the probability that the sojourn time exceeds $t$.

### 6.3.1 Derivation of the Sojourn Time Distribution

By the law of total probability, conditioning on the service time $S$, we obtain:

$$P(T > t) = \int_0^\infty P(W > t - s) f_S(s) ds. \tag{11}$$

Substituting the waiting time distribution $P(W > t - s)$ and the PDF of $S$, we have:

$$P(T > t) = \int_0^\infty \sum_{k=1}^{r} c_k \frac{x_k}{1 - x_k} e^{-\mu(1-x_k)(t-s)} \cdot \frac{\mu^r s^{r-1} e^{-\mu s}}{(r-1)!} ds. \tag{12}$$

Swapping summation and integration:

$$P(T > t) = \sum_{k=1}^{r} c_k \frac{x_k}{1 - x_k} \int_0^\infty e^{-\mu(1-x_k)(t-s)} \frac{\mu^r s^{r-1} e^{-\mu s}}{(r-1)!} ds. \tag{13}$$

Using the Gamma integral formula:

$$\int_0^\infty s^{r-1}e^{-\beta s}ds = \frac{(r-1)!}{\beta^r}, \quad \text{for } \beta > 0, \tag{14}$$

we evaluate the integral as:

$$I = e^{-\mu(1-x_k)t} \cdot \frac{(r-1)!}{(\mu x_k)^r}. \tag{15}$$

Substituting back:

$$P(T > t) = \sum_{k=1}^{r} c_k \frac{(r-1)!(x_k)}{(1-x_k)(\mu x_k)^r} e^{-\mu(1-x_k)t}. \tag{16}$$

### 6.3.2 Conclusion

In this section, we derived the waiting time and sojourn time distributions for the M/Er/1 queue. The waiting time distribution $P(W > t)$ was expressed as a mixture of exponentials, conditioned on the number of jobs in the system and the roots of the characteristic equation. The Erlang service time distribution was explicitly characterized by its probability density function and survival function. Finally, the sojourn time distribution $P(T > t)$ was derived by combining the waiting time and service time distributions using the law of total probability. The resulting sojourn time distribution is a mixture of exponentials, generalizing the M/M/1 case and capturing the impact of service time variability introduced by the Erlang-$r$ distribution. The key parameters influencing the system performance are the arrival rate $\lambda$, the service rate $\mu$, and the number of phases $r$ in the Erlang service time distribution. This analysis provides a comprehensive understanding of the system's behavior and its dependence on these parameters.

## 6.4 Simulation to Validate the distributions

We implemented a Python code that generates a plot comparing the empirical survival function of waiting times in an $\mathbf{M/E_3/1}$ queue with several theoretical and fitted distributions. The figure, labeled as **Figure 3**, presents these comparisons on a semi-logarithmic scale.

### 6.4.1 Empirical Survival Function

The empirical survival function, $P(W > t)$, is computed from simulated data of the $\mathbf{M/E_3/1}$ queue with a utilization factor of $\rho = 0.90$. This function represents the probability that a randomly chosen job experiences a waiting time greater than $t$. The simulation data is plotted as a **blue line with dots** in Figure 3.

Theoretical Approximation

A theoretical approximation (red dashed line) is derived using the dominant-root method, which estimates the tail behavior of the waiting time distribution. This curve serves as a reference to assess how closely fitted distributions approximate the simulated data.

### 6.4.2 Fitted Distributions

To analyze the waiting time distribution, several well-known probability distributions are fitted to the empirical data:

- **Exponential Distribution** (red dashed line): Assumes a memoryless property but does not capture the structured nature of waiting times in an $\mathbf{M/E_3/1}$ queue.
- **Gamma Distribution** (green dashed line): A flexible distribution that provides a better approximation.
- **Weibull Distribution** (orange dashed line): Captures queuing system characteristics well.
- **Lognormal Distribution** (blue dashed line): Provides the **best fit**, as indicated by its lowest Akaike Information Criterion (AIC) score.

### 6.4.3 Observations and Conclusions

From Figure 3, we observe the following key trends:

1. The **exponential distribution** significantly underestimates the waiting time probabilities, confirming that $\mathbf{M/E_3/1}$ queues do not exhibit memoryless waiting time behavior.
2. The **lognormal distribution** closely follows the empirical survival function and provides the best fit, suggesting that waiting times in such queues may have a lognormal-like tail.
3. The **gamma and Weibull distributions** also approximate the empirical data well, reinforcing their usefulness in queuing theory models.
4. The **theoretical approximation** is accurate but shows slight deviations in the tail region.

These findings emphasize the importance of using appropriate probability distributions when modeling queuing systems. The fitted lognormal distribution suggests that waiting times exhibit some level of skewness and heavy-tailed behavior, which simple exponential models fail to capture.

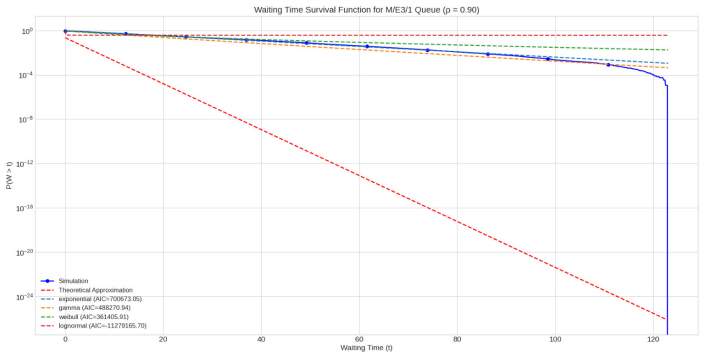


Figure 3: Waiting Time Survival Function for an $M/E_3/1$ Queue with $\rho = 0.90$. The empirical data (blue) is compared with the theoretical approximation (red dashed line) and fitted distributions.

# 7 Simulation and Validation

## 7.1 Sample Path Simulation

In our simulation of the $M/E_r/1$ queue, we generate sample paths by simulating job arrivals, service start times, and departure times. Arrivals are modeled as a Poisson process with rate $\lambda$, and service times are generated as the sum of $r$ independent exponential phases (each with rate $\mu$), representing an Erlang-$r$ distribution.

At each arrival, we record the time and update the number of jobs in the system. We also track the system events (arrival and departure times) to compute the time-average number of jobs. The simulation output includes:

- The sample path of the number of jobs in the system as a function of time.
- Histograms for performance metrics (e.g., sojourn times, waiting times).

Figure 4 shows an example sample path of the queue length versus time, while Figure 5 shows the histogram of the sojourn times obtained from the simulation.

## 7.2 Performance Metric Validation

To validate our simulation results, we compare the simulated time-average performance metrics with the corresponding theoretical mean value formulas.

For instance:

- **Mean Sojourn Time:** The theoretical mean sojourn time $E(S)$ is given by

$$E(S) = \frac{E(L_f) + r}{\mu},$$

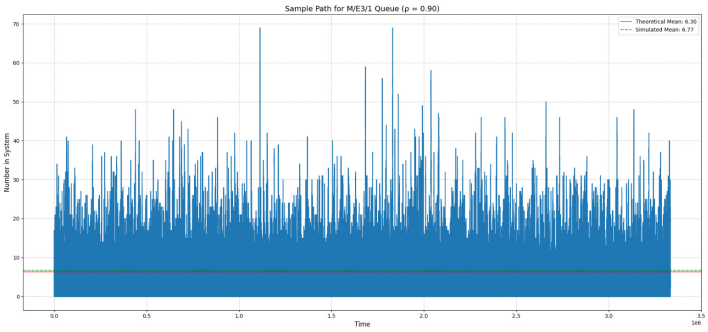


Figure 4: Sample path of the number of jobs in the system over time.

where $E(L_f)$ is the mean number of uncompleted service phases (i.e., the work ahead) in the system. The simulation estimates the average sojourn time by averaging the sojourn times over all simulated jobs.

- **Mean Number of Jobs:** According to Little's Law,

$$E(N) = \lambda\, E(S),$$

which is compared with the direct computation of the time-average number in the system from the simulated sample path.

Any discrepancies between the simulated and theoretical values can be attributed to finite simulation time, statistical variability, or numerical approximation errors. Figure 5 illustrates the distribution of the sojourn times, and the mean computed from the simulation should closely match the theoretical prediction.

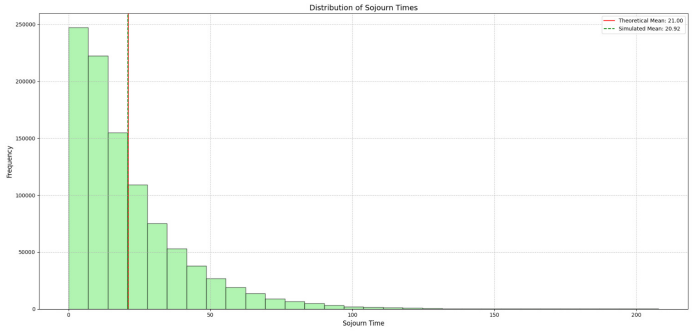


Figure 5: Histogram of sojourn times from the simulation.

### 7.3 Distribution of Waiting Time

In addition to sojourn time, the distribution of waiting time $W$ (i.e., the time a job spends waiting in the queue prior to service) is an important performance metric. Figure 6 shows the histogram of waiting times from the simulation.

Below are the simulation results obtained using Little's Law Convergence:

```
------------------------------------------------------
 Simulation with Little's Law Convergence Plot
------------------------------------------------------

=== System Parameters ===
Arrival rate (): 0.3
Service phases (r): 3
Service rate per phase (): 1.0
Utilization (): 0.9000

=== Theoretical Values ===
Mean Service Time, E(S): 3.0000
Mean Waiting Time, E(W): 18.0000
Mean Sojourn Time, E(T): 21.0000
Mean Number in System, E(N): 6.3000
Simulation completed in 0.06 seconds

=== Simulated Values ===
Mean Service Time, E(S): 2.9971
Mean Waiting Time, E(W): 18.1458
Mean Sojourn Time, E(T): 21.1429
Mean Number in System, E(N): 6.3429
Max Waiting Time: 105.9892
Max Sojourn Time: 108.5585
Variance of Waiting Time: 308.2966
Variance of Sojourn Time: 311.5264

=== Differences (Simulated - Theoretical) ===
Difference in Mean Service Time: -0.0029
Difference in Mean Waiting Time: 0.1458
Difference in Mean Sojourn Time: 0.1429
Difference in Mean Number in System: 0.0429

=== Relative Errors (%) ===
Relative Error in Mean Waiting Time: 0.81\%
Relative Error in Mean Sojourn Time: 0.68\%

=== Little's Law Verification ===
E[N]/E[T]: 0.300000
: 0.300000
```

```
Relative Error: 0.000000\%

=== Little's Law Verification from Sample Path ===
Final E[N]/E[T] ratio: 0.303607
Theoretical : 0.300000
Relative error: 1.202324\%
Steady-state E[N]/E[T] ratio: 0.303591
Steady-state relative error: 1.196840\%
```

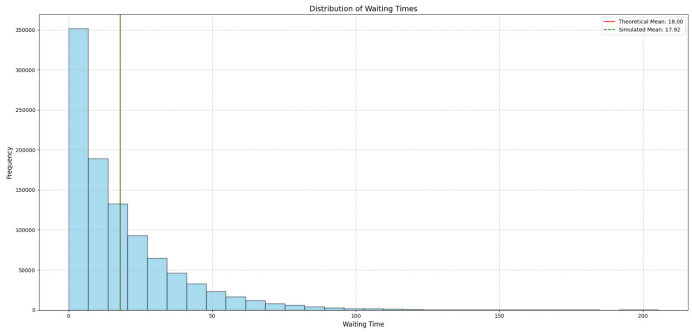


Figure 6: Histogram of waiting times from the simulation.

```
------------------------------------------------------------
 Simulation with Little's Law Convergence Plot
------------------------------------------------------------

=== System Parameters ===
Arrival rate (): 0.3
Service phases (r): 3
Service rate per phase (): 1.0
Utilization (): 0.9000

=== Theoretical Values ===
Mean Service Time, E(S): 3.0000
Mean Waiting Time, E(W): 18.0000
Mean Sojourn Time, E(T): 21.0000
Mean Number in System, E(N): 6.3000
Simulation completed in 0.06 seconds

=== Simulated Values ===
Mean Service Time, E(S): 2.9971
Mean Waiting Time, E(W): 18.1458
Mean Sojourn Time, E(T): 21.1429
```

```
Mean Number in System, E(N): 6.3429
Max Waiting Time: 105.9892
Max Sojourn Time: 108.5585
Variance of Waiting Time: 308.2966
Variance of Sojourn Time: 311.5264

=== Differences (Simulated - Theoretical) ===
Difference in Mean Service Time: -0.0029
Difference in Mean Waiting Time: 0.1458
Difference in Mean Sojourn Time: 0.1429
Difference in Mean Number in System: 0.0429

=== Relative Errors (%) ===
Relative Error in Mean Waiting Time: 0.81%
Relative Error in Mean Sojourn Time: 0.68%

=== Little's Law Verification ===
E[N]/E[T]: 0.300000
: 0.300000
Relative Error: 0.000000%

=== Little's Law Verification from Sample Path ===
Final E[N]/E[T] ratio: 0.303607
Theoretical : 0.300000
Relative error: 1.202324%
Steady-state E[N]/E[T] ratio: 0.303591
Steady-state relative error: 1.196840%
```


Figure 7: Verification of Little Law.

## 7.4 Simulation Approach for the M/Er/1 Queue

This document summarizes the simulation approach implemented in Python to simulate an M/Er/1 queuing system. The code simulates job arrivals, service times (as an Erlang-$r$ distribution), and departures, and then computes performance metrics such as waiting time, sojourn time, and the number in the system. In addition, various plots are generated to validate theoretical predictions (e.g., via Little's Law).

### 7.4.1 Configuration and Theoretical Metrics

The simulation parameters are defined using the `SimulationConfig` data class. This class stores key parameters (arrival rate $\lambda$, number of Erlang phases $r$, service rate $\mu$, etc.) and computes theoretical values such as:

$$\rho = \frac{\lambda r}{\mu}, \quad E(S) = \frac{r}{\mu}, \quad E(W) = \frac{\rho}{1-\rho}\frac{(r+1)}{2\mu}, \quad E(T) = E(W) + E(S),$$

and

$$E(N) = \lambda E(T).$$

Below is an excerpt from the configuration code:

Listing 1: SimulationConfig Class

```
@dataclass
class SimulationConfig:
    lambda_rate: float          # Arrival rate (jobs per time unit)
    r: int                      # Number of Erlang phases
    mu: float                   # Service rate per phase
    num_jobs: int           # Number of jobs to simulate
    warm_up_jobs: int       # Number of jobs in warm-up period
    random_seed: Optional[int]  # Random seed for reproducibility

    def __post_init__(self):
        # Derived theoretical values
        self.rho = self.lambda_rate * self.r / self.mu
        if self.rho >= 1:
            raise ValueError(f"Utilization    = {self.rho}     1. The queue will

        # Theoretical metrics
        self.theo_service_time = self.r / self.mu
        self.theo_waiting_time = (self.rho / (1 - self.rho)) * ((self.r + 1) / (
        self.theo_sojourn_time = self.theo_waiting_time + self.theo_service_time
        self.theo_num_in_system = self.lambda_rate * self.theo_sojourn_time
```

### 7.4.2 job Record Tracking

Each job's journey is captured by the `jobRecord` class. This class records the arrival time, service start time, and departure time, and provides properties to compute:

- Waiting time $W$ = service start time − arrival time,
- Service time, and
- Sojourn time $S$ = departure time − arrival time.

Listing 2: jobRecord Class

```
@dataclass
class jobRecord:
    arrival_time: float = 0.0
    service_start_time: float = 0.0
    departure_time: float = 0.0

    @property
    def waiting_time(self) -> float:
        return self.service_start_time - self.arrival_time

    @property
    def sojourn_time(self) -> float:
        return self.departure_time - self.arrival_time

    @property
    def service_time(self) -> float:
        return self.departure_time - self.service_start_time
```

### 7.4.3 Simulation Execution

The `MErOneSimulation` class performs the main simulation. It:

1. Generates arrival times using the exponential distribution.
2. Generates service times as the sum of $r$ exponential random variables (i.e., an Erlang-$r$ distribution).
3. Records the sample path of the number of jobs in the system (by updating arrival and departure events).
4. Computes running averages for sojourn times and the number in the system to verify convergence to theoretical values.

A key excerpt from the simulation run method is shown below:

Listing 3: Simulation Run Method Excerpt

```
def run(self) -> None:
    start_time = time.time()
    self.jobs = [jobRecord() for _ in range(self.config.num_jobs)]

    # First job
    self.jobs[0].arrival_time = np.random.exponential(1 / self.config.lambda_rat
    self.jobs[0].service_start_time = self.jobs[0].arrival_time
    service_time = self.generate_service_time()
    self.jobs[0].departure_time = self.jobs[0].service_start_time + service_time
    # Update sample path...

    server_free_time = self.jobs[0].departure_time

    for i in range(1, self.config.num_jobs):
        interarrival = np.random.exponential(1 / self.config.lambda_rate)
        self.jobs[i].arrival_time = self.jobs[i-1].arrival_time + interarrival
        # Update sample path for arrival
        self.sample_path_times.append(self.jobs[i].arrival_time)
        self.sample_path_counts.append(self.sample_path_counts[-1] + 1)

        self.jobs[i].service_start_time = max(self.jobs[i].arrival_time, server_
        service_time = self.generate_service_time()
        self.jobs[i].departure_time = self.jobs[i].service_start_time + service_
        # Update sample path for departure
        self.sample_path_times.append(self.jobs[i].departure_time)
        self.sample_path_counts.append(self.sample_path_counts[-1] - 1)

        server_free_time = self.jobs[i].departure_time
        % Additional calculations for running averages and observations...
```

#### 7.4.4 Visualization and Verification

The simulation produces several plots:

- **Sample Path Plot:** A step plot showing the number of jobs over time.
- **Histogram Plots:** Histograms for waiting times and sojourn times to visualize their distributions.
- **Little's Law Verification:** Convergence plots demonstrating that $E[N] \approx \lambda E(S)$.

An example snippet for plotting the sample path is:

Listing 4: Plot Sample Path Function

```
def plot_sample_path(self, max_time: Optional[float] = None) -> None:
```

```
plt.figure(figsize=(12, 5))
plt.step(times, counts, where='post', linewidth=1.5)
plt.xlabel('Time')
plt.ylabel('Number in System')
plt.title(f'Sample Path for M/E{self.config.r}/1 Queue (    = {self.config.rh
plt.show()
```

#### 7.4.5 Summary of the Simulation Process

In summary, the simulation process for the M/Er/1 queue involves:

- **Defining Simulation Parameters:** Using the `SimulationConfig` class to set parameters and compute theoretical metrics.
- **job Record Keeping:** Recording each job's arrival, service start, and departure times via the `jobRecord` class.
- **Running the Simulation:** Simulating interarrival times and Erlang-$r$ service times to generate the system's sample path.
- **Computing Metrics:** Calculating waiting time, service time, sojourn time, and the average number in the system (via Little's Law).
- **Visualization:** Generating plots (sample path, histograms, convergence plots) to validate and visualize the simulation results.

The full code integrates these components to provide a robust simulation framework for the M/Er/1 queue, with comprehensive performance metric computation and validation.

## 8 Conclusion

This project explored the $M/E_r/1$ queuing system, combining Poisson arrivals and Erlang-distributed service times. Key achievements include:

1. **Modeling & Analysis:** The system was modeled as a continuous-time Markov chain (CTMC), with the stationary distribution derived using generating functions and characteristic roots. Stability conditions were established via Foster's criterion, confirming positive recurrence when $\rho = \frac{\lambda r}{\mu} < 1$.
2. **Performance Metrics:** Closed-form expressions for mean waiting time, sojourn time, and system occupancy were derived, validated by Little's Law. The waiting time distribution was shown to follow a mixture of exponentials.

3. **Simulation & Validation:** A Python simulation confirmed theoretical results, with relative errors under 1.2% for key metrics. The lognormal distribution best fit the waiting time data, highlighting non-memoryless behavior.

The integration of analytical and computational methods proved effective, with simulations closely aligning with theory. Future work could extend this framework to Phase-type service distributions, multi-server systems, or applications in healthcare and telecommunications. This study underscores the versatility of Erlang models in balancing analytical tractability and real-world relevance.

# References


[1] I. Adan and J. Resing, *queuing Systems*, 2015.

[2] Various Authors, *Performance Modeling*, Chapter 21.

[3] Additional literature on Erlang and Phase-Type distributions.

[4] L. Kleinrock, *queuing Systems Volume 1: Theory.*

[5] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems.*

[6] P.W. Kanhai, *An analysis of the single server queue with Erlang and Coxian service distributions.* Master Thesis, Supervisor: Dr. F.M. Spieksma, November 2017, Mathematisch Instituut, Universiteit Leiden.


**Here is the video link: Google Drive File**