Winter 2019

An Examination of Computational Methods Related to G/M/c Queueing

Thomas Michaud

An Examination of Computational Methods Related to G/M/c Queueing

A thesis submitted in partial fulfillment of the requirements
for the degree Master of Science in Statistics
University of Southern Maine

By Thomas R. Michaud

December 20, 2019

We hereby recommend that the thesis of Thomas Michaud entitled

An Examination of Computational Models Related to G/M/c Queueing

be accepted in partial fulfillment of the requirements for the degree

Master of Science in Statistics.

Muhammad El-Taha

Abou El-Makarim Aboueissa

Weston Viles

Accepted

Dean, College of Science, Technology, and Health.

Acknowledgements

The author would like to thank the members of the Committee both for their participation and for their contribution to my development as a student. Dr. Abou's courses developed the programming ability demonstrated here. Dr. Viles' course and conversations deepened my understanding of current statistical thought. Dr. El-Taha's courses and guidance as my advisor have deepened both my theoretical understanding and my awareness of the potential applications.

A special thanks to Peggy Moore for her mentoring from my first real Statistics course through to teaching this semester, and to my wife Heather for her forbearance during this time.

Abstract

The following examination of computational methods related to queues with general arrivals (*i.i.d.* but of unknown distribution), multiple identical servers with *i.i.d.* exponential service times, and ordinary first come, first served service (hereafter referred to as G/M/c to use existing naming conventions) seeks to investigate the current models and provide new results based on a draft convolution method proposed by El-Taha[3]. The new model will demonstrate the use of distributions with coefficients of variance ranging from zero to near infinity to provide flexibility in simulating a range of potential arrival distributions, and we include detailed results and software for both small-scale and large-scale models.

Contents

List of Tables

List of Figures

Chapter 1

Overview of Queueing and the G/M/c System

1 Introduction

We begin with an overview of queueing theory and the general properties and conditions necessary to evaluate systems with general arrivals and exponential service times. We then proceed to review the existing literature on this topic, and define the transition states present in the system.

2 Overview of Queueing Theory

The most general description of queueing theory may be that found in Cooper[2], who describes it as "the mathematical analysis of systems subject to demands whose occurrences and lengths can, in general, be specified only probabilistically." We can add to this the situation found in most queueing systems, that of having a finite limit for the number of demands that can be satisfied concurrently, which typically delays the beginning of any demand entering the system after that limit is reached. We may also include a limit on the total number of demands allowed in the system (pending or in-process). And for any system with a finite limit on concurrently satisfied demands, we must consider the ordering of demands within the system, which can affect the amount of delay for a particular demand.

Thus in a typical queueing model we have:

1. A random variable defining the length of the interval between demands entering the system. We consider this the distribution of interarrival times.

2. A random variable defining the duration of a demand. We consider this the distribution of service times.

3. A (usually finite) number of demands allowed to occur simultaneously in the system. We consider this the number of "servers" in the system.

4. A method of ordering demands within the system. We call this the queue discipline, the simplest form of which is "first come, first served", i.e.: demands are ordered within the system in the order of their arrival.

5. In some cases, a finite limit on the number of concurrent demands in the system (pending or being served). The capacity available beyond the number of servers shall be referred to as the "buffer" or "waiting space" of the system.

We limit our examination to those systems capable of long-run stability, that is to say, systems in which the probability distribution of the number of demands in the system reaches a finite limit as the overall interval approaches infinity. In general, it should be apparent that, as long as the rate at which demands are allowed to enter the system is less than the rate at which the system can serve demands, we have such a system. Otherwise, unless the system has finite capacity and rejects arrivals whenever this limit is reached (thereby restricting the *effective* arrival rate to be less than the service rate), the number of demands in the system will grow without bound over the long run, with the probability distribution of the number of demands in the system becoming zero for any specific value.

Of great convenience in the analysis of queueing systems in the presence of the memoryless property of the exponential distribution, when used for arrivals and/or service. This allows us to evaluate the future state of the system based only on the present state, without dependence on the past history of the system. In practice, the simplest queueing systems have both exponential arrival times and exponential service times, allowing them to be modeled as birth-death processes.

Our focus in this paper is queueing systems with the following characteristics:

1. A general distribution (G) of interarrival times, which we shall demonstrate with deterministic, Erlang, exponential, and hyperexponential distributions.

2. An exponential distribution (M) for service times. This provides the Markovian property of memorylessness for the service time distribution.

3. A finite number of servers, which we represent with the constant c.

4. A queue discipline of "first come, first served", i.e.: demands are ordered within the system in the order of their arrival.

5. Either a finite or infinite buffer (we shall examine both).

Thus our label for this type of system, with general arrivals, exponential service, and a finite number of servers, is G/M/c.

3 Review of Existing Literature

Multiple approaches exist for finding the performance values of a G/M/c system, however, the most straightforward appears to be the defining of an embedded Markov chain by conditioning the transition probabilities on the instants immediately preceding an arrival, as seen in Kleinrock [8] for example. Thus the elapsed time between transitions is from the distribution of interarrival times, and given that the only events occuring between arrivals are service completions, the state transitions during an interarrival time are determined only by the service distribution. If our service distribution is exponential (as ours is), then we have the memoryless property such that the future state of the system is dependent only on the present state, and not the earlier states.

Takacs [10], Kleinrock[8], and Gross and Harris [5] proceed to define a random variable for the number of demands in the system immediately prior to an arrival, which forms the states of the embedded Markov Chain. They then define a random variable for the number of demands serviced (completed) during the interval between two sequential such instants, and from this comes the conclusion that the number of demands in the system immediately prior to the next arrival must be equal to the number in the system immediately prior to the previous arrival, plus the arrival that occurred, minus the number of demands serviced during the interarrival time.

Thus for a given transition probability p_{ij} wherein i defines the number in the system immediately prior to an arrival and j which defines the number in the system immediately prior to the next arrival, it is clear via the relationship above that the probability of the transition from state i to state j is equal to the probability that $i + 1 - j$ demands are served during an interarrival time [Kleinrock]. It is this premise that allows us to formulate the matrix of transition probabilities. All that remains is to prove that the limiting distribution exists when the total service rate is greater than the arrival rate, and this proof is provided by Takacs [10]. Takacs[10] then provides a detailed but lengthy and complex method for obtaining the limiting distribution of probabilities for the system states. Kleinrock[8] instead proceeds to evaluate the conditional distribution of queue size, demonstrating that, given a queue exists, the distribution of the conditional queue length is geometric.

4 Transition Probabilities Defined

The core of these methods is the formulation of the matrix of state transition probabilities. However, given the multi-server system with individual service rates of μ for each server, the one-step transition matrix has different structures in the four regions given below:

(i) Region 1, where $j \leq i+1 \leq c$, is the region where the system is not fully utilized (at least one server is available to the next arrival). Here, some servers may complete service while others do not, thus the service rate may vary from $c\mu$ down to $j\mu$.

(ii) Region 2, where $c \leq j \leq i + 1$, is the region where both the current arrival and the next arrival find a busy system, that is, a system where all servers are engaged. Thus we have a system with overall service $c\mu$. Note that the special case $i = c - 1, j = c$ is equivalent for Region 1 and Region 2.

(iii) Region 3, where $j \leq c - 1 < i$, is the region where a busy system empties the entire waiting queue and has at least one server available for the next arrival. This is the most complex region, both theoretically and computationally, because a portion of the interarrival time is needed to empty the queue with rate $c\mu$, after which the service rate varies from $c\mu$ down to $j\mu$.

(iv) Region 4, where $j > i + 1$, is the region where one or more arrivals occurs between arrivals, which is a contradiction. Therefore all transitions in this region have probability zero.

A visual representation of this matrix for a system with five servers is shown in Figure 1.1.

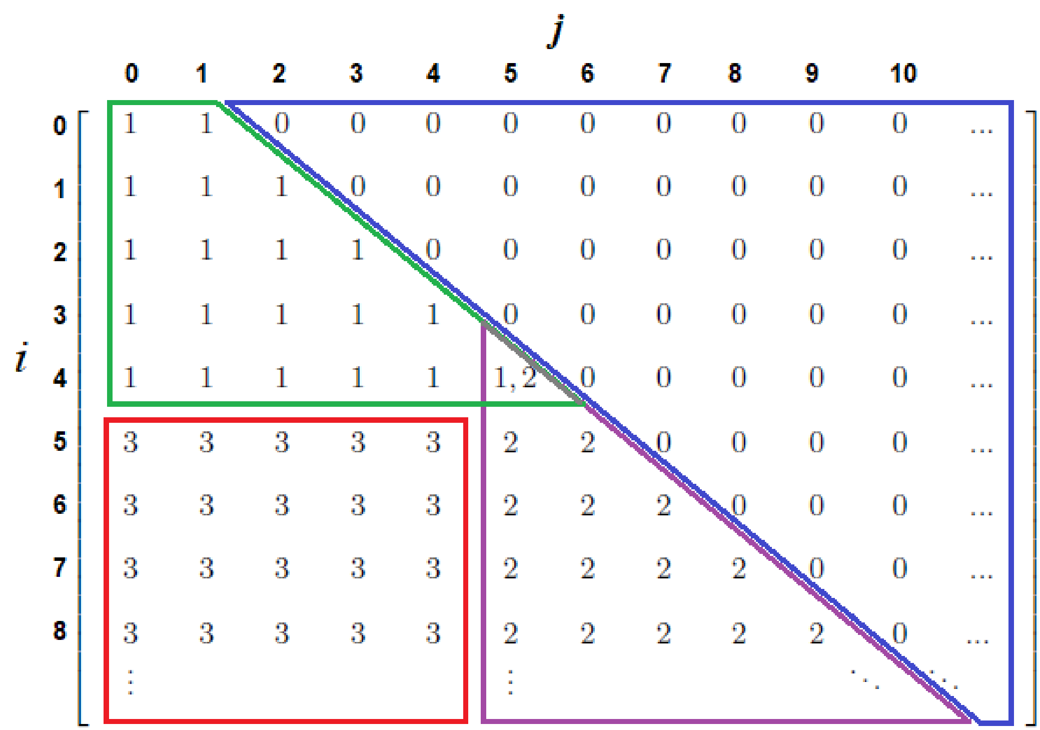

Figure 1.1: Transition matrix for G/M/c system with $c = 5$

As noted above, we require an ergodic system to avoid the trivial case where $p_{ij} = 0$ for all i, j. This case occurs when the average arrival rate exceeds the total service rate, thus we need only require that the overall ratio of arrivals to services be less than one.

Chapter 2

Transition Probabilities for $G/M/c$ Queues

1 Introduction

We begin our solutions with the transition matrix, noting that in all regions other than Region 4, the service is conditioned on the cumulative interarrival distribution, which we label $A(t)$. Once defined, these regions can be evaluated with respect to this distribution. We then evaluate with respect to the following distributions: deterministic, Erlang, exponential, and hyper-exponential. These four distributions allow us to evaluate coefficients of variation of zero, between zero and one exclusive, one, and greater than one. Lastly, we examine the convolution method mentioned in Chapter 1 for use in Region 3.

2 Direct Integration

Here we directly evaluate the transition probabilities in the four regions with respect to the arrival distribution.

2.1 Region 1

For Region 1 of the transition matrix (where the arrival does not wait to be served), we must have $i-j+1$ service completions between arrivals. Given that the probability for an individual server to complete within time t is the cumulative distribution function of the exponential service distribution $1-e^{-\mu t}$, and the probability of not completing is therefore $1-(1-e^{-\mu t}) = e^{-\mu t}$, we have a binomial distribution of completions with $i+1$ trials and $i-j+1$ successes (completions), dependent on the arrival distribution. Thus the probability in this region is

$$
\begin{aligned}
p(i,j) &= \int_0^\infty \binom{i+1}{i-j+1} e^{-(\mu t)j}(1-e^{-\mu t})^{i-j+1} dA(t) \\
&= \binom{i+1}{i-j+1} \int_0^\infty e^{-j\mu t}(1-e^{-\mu t})^{i-j+1} dA(t)
\end{aligned}
$$

Note: Given $a \in \mathbb{R}, n \in \mathbb{N}$, we have the binomial expansion

$$
(1-a)^n = \sum_{k=0}^{n} \binom{n}{k} (-1)^k a^k
$$

therefore:

$$
(1-e^{-\mu t})^{i-j+1} = \sum_{k=0}^{i-j+1} \binom{i-j+1}{k} (-1)^k e^{-k\mu t}
$$

thus we continue:

$$p(i,j) = \binom{i+1}{i-j+1} \int_0^\infty e^{-j\mu t} \left[\sum_{k=0}^{i-j+1} \binom{i-j+1}{k} (-1)^k e^{-k\mu t} \right] dA(t)$$

$$= \binom{i+1}{i-j+1} \int_0^\infty \left[\sum_{k=0}^{i-j+1} \binom{i-j+1}{k} (-1)^k e^{-j\mu t} e^{-k\mu t} \right] dA(t)$$

$$= \binom{i+1}{i-j+1} \sum_{k=0}^{i-j+1} \binom{i-j+1}{k} (-1)^k \int_0^\infty e^{-j\mu t - k\mu t} dA(t)$$

$$= \binom{i+1}{i-j+1} \sum_{k=0}^{i-j+1} \binom{i-j+1}{k} (-1)^k \int_0^\infty e^{-(j+k)\mu t} dA(t)$$

Here we use the Laplace-Stieltjes transform $\int_0^\infty e^{-st} dA(t) = A^*(s)$, where $s = (j+k)\mu$, thus

$$p(i,j) = \binom{i+1}{i-j+1} \sum_{k=0}^{i-j+1} \binom{i-j+1}{k} (-1)^k A^*((j+k)\mu)$$

Expanding the combinations provides some limited simplification:

$$p(i,j) = \frac{(i+1)!}{(i+1-(i-j+1))!(i-j+1)!} \sum_{k=0}^{i-j+1} \frac{(i-j+1)!}{(i-j+1-k)!k!} (-1)^k A^*((j+k)\mu)$$

$$= \frac{(i+1)!}{j!} \sum_{k=0}^{i-j+1} \frac{(-1)^k A^*((j+k)\mu)}{(i-j-k+1)!k!} \tag{2.1}$$

2.2 Region 2

Region 2 is the area where a busy system remains busy (a queue exists prior to the previous arrival and is not eliminated during the interarrival time), we have Poisson service completions at a rate of $c\mu$ (again dependent on the arrival distribution), thus we have

$$p(i,j) = \int_0^\infty \frac{e^{-c\mu t}(c\mu t)^{i-j+1}}{(i-j+1)!} dA(t) \quad = \quad \frac{(c\mu)^{i-j+1}}{(i-j+1)!} \int_0^\infty t^{i-j+1} e^{-c\mu t} dA(t)$$

Again we utilize the Laplace-Stieltjes transform

$$\int_0^\infty e^{-st} dA(t) = A^*(s)$$

and further define

$$A_n^*(s) = (-1)^n \frac{d^n A^*(s)}{ds^n} = \int_0^\infty t^n e^{-st} dA(t)$$

where $\frac{d^n A^*(s)}{ds^n}$ is the n^{th} derivative of $A^*(s)$. (See Appendix A for proof.)

In this case, $s = c\mu$, leading to

$$p(i,j) = \frac{(c\mu)^{i-j+1}}{(i-j+1)!} A_{i-j+1}^*(c\mu) \tag{2.2}$$

2.3 Region 3

Region 3 is the most complicated because we have a busy system that transitions to having at least one server idle (a queue exists immediately prior to an arrival but is emptied before the next arrival). Thus we begin the interarrival time with Poisson service completions (as in Region 2) until the queue is empty, and then end the interarrival time with the binomial distribution shown for Region 1 as the service rate varies down to $j\mu$. Thus we have the binomial from Region 1 for the $c-j$ demands completed after the queue is emptied, conditioned on the emptying of the queue.

Following Gross and Harris[5], we define ν as the time required to empty the queue of the $i-c+1$ demands awaiting service, with all c servers working. If we name the CDF of ν as $H(\nu)$, and our binomial has $c-j$ successes (completions) out of c trials with probability of success $1-e^{-\mu(t-\nu)}$ then we have

$$p(i,j) = \int_0^\infty \int_0^t \binom{c}{c-j} (1-e^{-\mu(t-\nu)})^{c-j} \; e^{-\mu(t-\nu)j} \, dH(\nu) \, dA(t)$$

We know that, as with Region 3, completions for a busy system are Poisson distributed with rate $c\mu$. To be more specific, the distribution of completions in this region is the sum of the individual completion distributions, each having rate μ. Thus to determine the distribution of the time ν required to complete all $i-c+1$ demands in the queue, we have the sum of the individual exponential service times, which is Erlang distributed with shape $i-c+1$ and rate $c\mu$.

Therefore we can replace $dH(\nu)$ with $h(\nu)d\nu$, where

$$h(\nu) = \frac{(c\mu)^{i-c+1}\nu^{i-c}e^{-c\mu\nu}}{(i-c)!}$$

Thus we have

$$p(i,j) = \int_0^\infty\int_0^t \binom{c}{c-j} e^{-\mu(t-\nu)j}(1-e^{-\mu(t-\nu)})^{c-j}\frac{(c\mu)^{i-c+1}\nu^{i-c}e^{-c\mu\nu}}{(i-c)!}\,d\nu\,dA(t)$$

which we integrate as follows:

$$\begin{aligned}
p(i,j) &= \binom{c}{c-j}\frac{(c\mu)^{i-c+1}}{(i-c)!}\int_0^\infty\int_0^t e^{-\mu(t-\nu)j}(1-e^{-\mu(t-\nu)})^{c-j}\nu^{i-c}e^{-c\mu\nu}\,d\nu\,dA(t) \\
&= \binom{c}{c-j}\frac{(c\mu)^{i-c+1}}{(i-c)!}\int_0^\infty\int_0^t \nu^{i-c}e^{-c\mu\nu}e^{-\mu(t-\nu)j}(1-e^{-\mu(t-\nu)})^{c-j}\,d\nu\,dA(t) \\
&= \binom{c}{c-j}\frac{(c\mu)^{i-c+1}}{(i-c)!}\int_0^\infty\int_0^t \nu^{i-c}e^{-t\mu j+\nu\mu j-\nu\mu c}(1-e^{-\mu(t-\nu)})^{c-j}\,d\nu\,dA(t) \\
&= \binom{c}{c-j}\frac{(c\mu)^{i-c+1}}{(i-c)!}\int_0^\infty\int_0^t \nu^{i-c}e^{-t\mu j}e^{\nu\mu j-\nu\mu c}(1-e^{-\mu(t-\nu)})^{c-j}\,d\nu\,dA(t) \\
&= \binom{c}{c-j}\frac{(c\mu)^{i-c+1}}{(i-c)!}\int_0^\infty\int_0^t \nu^{i-c}e^{-t\mu j}e^{-\nu\mu(c-j)}(1-e^{-\mu(t-\nu)})^{c-j}\,d\nu\,dA(t) \\
&= \binom{c}{c-j}\frac{(c\mu)^{i-c+1}}{(i-c)!}\int_0^\infty e^{-t\mu j}\int_0^t \nu^{i-c}e^{-\nu\mu(c-j)}(1-e^{-\mu(t-\nu)})^{c-j}\,d\nu\,dA(t)
\end{aligned}$$

Note:

$$(1-a)^b = \sum_{i=0}^{b}\binom{b}{i}(-1)^i a^i$$

therefore:

$$(1-e^{-\mu(t-\nu)})^{c-j} = \sum_{m=0}^{c-j} \binom{c-j}{m} (-1)^m e^{-m\mu(t-\nu)}$$

to yield:

$$p(i,j) = \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \int_0^\infty e^{-t\mu j} \int_0^t \nu^{i-c} e^{-\nu\mu(c-j)} \sum_{m=0}^{c-j} \binom{c-j}{m} (-1)^m e^{-m\mu(t-\nu)}\, d\nu\, dA(t)$$

We separate the final term of the binomial expansion and continue the integration:

$$\begin{aligned}
p(i,j) &= \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \int_0^\infty e^{-t\mu j} \int_0^t \nu^{i-c} e^{-\nu\mu(c-j)} \sum_{m=0}^{c-j-1} \binom{c-j}{m} (-1)^m e^{-m\mu(t-\nu)}\, d\nu\, dA(t) \\
&\quad + \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \int_0^\infty e^{-t\mu j} \int_0^t \nu^{i-c} e^{-\nu\mu(c-j)} (-1)^{c-j} e^{-(c-j)\mu(t-\nu)}\, d\nu\, dA(t) \\
&= \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \sum_{m=0}^{c-j-1} \binom{c-j}{m} (-1)^m \int_0^\infty e^{-t\mu j} \int_0^t \nu^{i-c} e^{-\nu\mu c} e^{\nu\mu j} e^{-m\mu t} e^{m\mu\nu}\, d\nu\, dA(t) \\
&\quad + \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} (-1)^{c-j} \int_0^\infty e^{-t\mu j} \int_0^t \nu^{i-c} e^{-\nu\mu c} e^{\nu\mu j} e^{-(c-j)\mu t} e^{(c-j)\mu\nu}\, d\nu\, dA(t) \\
&= \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \sum_{m=0}^{c-j-1} \binom{c-j}{m} (-1)^m \int_0^\infty e^{-j\mu t} e^{-m\mu t} \int_0^t \nu^{i-c} e^{-\nu\mu c} e^{\nu\mu j} e^{m\mu\nu}\, d\nu\, dA(t) \\
&\quad + \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} (-1)^{c-j} \int_0^\infty e^{-j\mu t} e^{-(c-j)\mu t} \int_0^t \nu^{i-c} e^{-\nu\mu c} e^{\nu\mu j} e^{(c-j)\mu\nu}\, d\nu\, dA(t) \\
&= \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \sum_{m=0}^{c-j-1} \binom{c-j}{m} (-1)^m \int_0^\infty e^{-(j+m)\mu t} \int_0^t \nu^{i-c} e^{-\nu\mu(c-j+m)}\, d\nu\, dA(t) \\
&\quad + \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} (-1)^{c-j} \int_0^\infty e^{-(j+(c-j))\mu t} \int_0^t \nu^{i-c} e^{-\nu\mu(c-j-(c-j))}\, d\nu\, dA(t)
\end{aligned}$$

$$= \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \sum_{m=0}^{c-j-1} \binom{c-j}{m} (-1)^m \int_0^\infty e^{-(j+m)\mu t} \int_0^t \nu^{i-c} e^{-\nu\mu(c-j+m)}\, d\nu\, dA(t)$$

$$+ \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} (-1)^{c-j} \int_0^\infty e^{-c\mu t} \int_0^t \nu^{i-c}\, d\nu\, dA(t)$$

$$= \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \sum_{m=0}^{c-j-1} \binom{c-j}{m} (-1)^m \int_0^\infty e^{-(j+m)\mu t} \int_0^t \nu^{i-c} e^{-\nu\mu(c-j+m)}\, d\nu\, dA(t)$$

$$+ \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} (-1)^{c-j} \int_0^\infty e^{-c\mu t} \left(\left. \frac{\nu^{i-c+1}}{i-c+1} \right|_0^t \right) dA(t)$$

$$= \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \sum_{m=0}^{c-j-1} \binom{c-j}{m} (-1)^m \int_0^\infty e^{-(j+m)\mu t} \int_0^t \nu^{i-c} e^{-\nu\mu(c-j+m)}\, d\nu\, dA(t)$$

$$+ \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} (-1)^{c-j} \int_0^\infty e^{-c\mu t} \left(\frac{t^{i-c+1}}{i-c+1} \right) dA(t)$$

$$= \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \sum_{m=0}^{c-j-1} \binom{c-j}{m} (-1)^m \int_0^\infty e^{-(j+m)\mu t} \int_0^t \nu^{i-c} e^{-\nu\mu(c-j+m)}\, d\nu\, dA(t)$$

$$+ \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \cdot \frac{(-1)^{c-j}}{(i-c+1)} \int_0^\infty t^{i-c+1} e^{-c\mu t}\, dA(t)$$

Note that $\int_0^t \nu^{i-c} e^{-\mu(c-j-m)\nu} d\nu$ is resolved through repeated integration by parts, so that

$$\int_0^t \nu^{i-c} e^{-\mu(c-j-m)\nu} d\nu = \frac{(i-c)!}{(\mu(c-j-m))^{i-c+1}} - \sum_{n=0}^{i-c} \frac{(i-c)!(\mu(c-j-m))^n t^n e^{-\mu(c-j-m)t}}{n!(\mu(c-j-m))^{i-c+1}}$$

$$= \frac{(i-c)!}{(\mu(c-j-m))^{i-c+1}} \left[1 - \sum_{n=0}^{i-c} \frac{(\mu(c-j-m))^n t^n e^{-\mu(c-j-m)t}}{n!} \right]$$

Thus we continue

$$p(i,j) = \binom{c}{c-j}\frac{(c\mu)^{i-c+1}}{(i-c)!}\sum_{m=0}^{c-j-1}\binom{c-j}{m}(-1)^m$$

$$\times \int_0^\infty e^{-(j+m)\mu t}\frac{(i-c)!}{(\mu(c-j-m))^{i-c+1}}\left[1-\sum_{n=0}^{i-c}\frac{(\mu(c-j-m))^n t^n e^{-\mu(c-j-m)t}}{n!}\right]dA(t)$$

$$+ \binom{c}{c-j}\frac{(c\mu)^{i-c+1}(-1)^{c-j}}{(i-c+1)!}\int_0^\infty t^{i-c+1}e^{-c\mu t}\,dA(t)$$

$$= \binom{c}{c-j}\frac{(c\mu)^{i-c+1}}{(i-c)!}\sum_{m=0}^{c-j-1}\binom{c-j}{m}(-1)^m\frac{(i-c)!}{(\mu(c-j-m))^{i-c+1}}$$

$$\times \int_0^\infty e^{-(j+m)\mu t}\left[1-\sum_{n=0}^{i-c}\frac{(\mu(c-j-m))^n t^n e^{-\mu(c-j-m)t}}{n!}\right]dA(t)$$

$$+ \binom{c}{c-j}\frac{(c\mu)^{i-c+1}(-1)^{c-j}}{(i-c+1)!}\int_0^\infty t^{i-c+1}e^{-c\mu t}\,dA(t)$$

$$= \binom{c}{c-j}\frac{(c\mu)^{i-c+1}}{(i-c)!}\sum_{m=0}^{c-j-1}\binom{c-j}{m}(-1)^m\frac{(i-c)!}{(\mu(c-j-m))^{i-c+1}}$$

$$\times \int_0^\infty \left[e^{-(j+m)\mu t}-\sum_{n=0}^{i-c}\frac{(\mu(c-j-m))^n t^n e^{-(j+m)\mu t}e^{-\mu(c-j-m)t}}{n!}\right]dA(t)$$

$$+ \binom{c}{c-j}\frac{(c\mu)^{i-c+1}(-1)^{c-j}}{(i-c+1)!}\int_0^\infty t^{i-c+1}e^{-c\mu t}\,dA(t)$$

$$= \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \sum_{m=0}^{c-j-1} \binom{c-j}{m} (-1)^m \frac{(i-c)!}{(\mu(c-j-m))^{i-c+1}}$$

$$\times \left[\int_0^\infty e^{-(j+m)\mu t} dA(t) - \int_0^\infty \sum_{n=0}^{i-c} \frac{(\mu(c-j-m))^n t^n e^{-(j+m)\mu t} e^{-\mu(c-j-m)t}}{n!} dA(t) \right]$$

$$+ \binom{c}{c-j} \frac{(c\mu)^{i-c+1}(-1)^{c-j}}{(i-c+1)!} \int_0^\infty t^{i-c+1} e^{-c\mu t}\, dA(t)$$

$$= \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \sum_{m=0}^{c-j-1} \binom{c-j}{m} (-1)^m \frac{(i-c)!}{(\mu(c-j-m))^{i-c+1}}$$

$$\times \left[\int_0^\infty e^{-(j+m)\mu t} dA(t) - \sum_{n=0}^{i-c} \frac{(\mu(c-j-m))^n}{n!} \int_0^\infty t^n e^{-(j+m+c-j-m)\mu t} dA(t) \right]$$

$$+ \binom{c}{c-j} \frac{(c\mu)^{i-c+1}(-1)^{c-j}}{(i-c+1)!} \int_0^\infty t^{i-c+1} e^{-c\mu t}\, dA(t)$$

$$= \binom{c}{c-j} \frac{(c\mu)^{i-c+1}}{(i-c)!} \sum_{m=0}^{c-j-1} \binom{c-j}{m} (-1)^m \frac{(i-c)!}{(\mu(c-j-m))^{i-c+1}}$$

$$\times \left[\int_0^\infty e^{-(j+m)\mu t} dA(t) - \sum_{n=0}^{i-c} \frac{(\mu(c-j-m))^n}{n!} \int_0^\infty t^n e^{-c\mu t} dA(t) \right]$$

$$+ \binom{c}{c-j} \frac{(c\mu)^{i-c+1}(-1)^{c-j}}{(i-c+1)!} \int_0^\infty t^{i-c+1} e^{-c\mu t}\, dA(t)$$

As with Regions 1 and 2, we again use the Laplace transforms $A^*(s)$ and $A_n^*(s)$

$$A^*(s) = \int_0^\infty e^{-st} dA(t) \qquad\qquad A_n^*(s) = \int_0^\infty t^n e^{-st} dA(t)$$

thus we have

$$p(i,j) = \binom{c}{c-j}\frac{(c\mu)^{i-c+1}}{(i-c)!}\sum_{m=0}^{c-j-1}\binom{c-j}{m}(-1)^m\frac{(i-c)!}{(\mu(c-j-m))^{i-c+1}}$$

$$\times\left[\int_0^\infty e^{-(j+m)\mu t}dA(t) - \sum_{n=0}^{i-c}\frac{(\mu(c-j-m))^n}{n!}\int_0^\infty t^n e^{-c\mu t}dA(t)\right]$$

$$+\binom{c}{c-j}\frac{(c\mu)^{i-c+1}(-1)^{c-j}}{(i-c+1)!}\int_0^\infty t^{i-c+1}e^{-c\mu t}\,dA(t)$$

$$= \binom{c}{c-j}\frac{(c\mu)^{i-c+1}}{(i-c)!}\sum_{m=0}^{c-j-1}\binom{c-j}{m}(-1)^m\frac{(i-c)!}{(\mu(c-j-m))^{i-c+1}}$$

$$\times\left[A^*((j+m)\mu) - \sum_{n=0}^{i-c}\frac{(\mu(c-j-m))^n A_n^*(c\mu)}{n!}\right]$$

$$+\binom{c}{c-j}\frac{(c\mu)^{i-c+1}(-1)^{c-j}A_{i-c+1}^*(c\mu)}{(i-c+1)!}$$

$$= \binom{c}{c-j}c^{i-c+1}\sum_{m=0}^{c-j-1}\binom{c-j}{m}\frac{(-1)^m}{(c-j-m)^{i-c+1}}$$

$$\times\left[A^*((j+m)\mu) - \sum_{n=0}^{i-c}\frac{(\mu(c-j-m))^n A_n^*(c\mu)}{n!}\right]$$

$$+\binom{c}{c-j}\frac{(c\mu)^{i-c+1}(-1)^{c-j}A_{i-c+1}^*(c\mu)}{(i-c+1)!}$$

3 Convolution

An alternative approach proposed by El-Taha[3] shows that while the inter-event times must be independent, it is possible to relax the assumption of identical distribution. From this, El-Taha proceeds to obtain one-step transition probabilities for Region 3 in the form of

$$\begin{aligned} p(i,j) \;=\; & \sum_{k=1}^{c-j-1} \frac{C_{k,c-j}(c-k)}{j} \left(\frac{c}{k}\right)^{i-c+2} \left[A^*((c-k)\mu) - \sum_{r=0}^{i-c+1} \frac{(k\mu)^r A_r^*(c\mu)}{r!} \right] \\ & + \; C_{c-j,c-j} \left(\frac{c}{c-j}\right)^{i-c+2} \left[A^*(j\mu) - \sum_{r=0}^{i-c+1} \frac{((c-j)\mu)^r A_r^*(c\mu)}{r!} \right] \end{aligned} \tag{2.3}$$

where

$$C_{k,c-j} = \prod_{m=1}^{k-1} \frac{c-m}{k-m} \times \prod_{m=k+1}^{c-j} \frac{c-m}{k-m}$$

and $\prod$ and $\sum$ over empty sets are 1 and 0 respectively.

We will use this convolution method for the computation of Region 3.

Chapter 3

Small-scale Finite Buffer Examples

1 Introduction

In this section we present small-scale, finite-buffer examples for a system with $c = 3$ servers and a total capacity of $K = 6$. Laplace-Stieltjes transforms of the various distributions are provided in Appendix A.

2 General Arrivals

First, we note (as does El-Taha[3]) that, given i as the number in the system immediately prior to an arrival, then the probabilities when $i = K - 1$ must be the same as the probabilities when $i = K$, because in the first case, the system becomes full, and in the second case, the system is already full and the new arrival is lost. Given the memoryless property of the individual service distributions, the probabilities of transitioning out of a full system are unaffected by any additional arrivals while the system is full, and thus the transition probabilities for $i - 1$ and i must be equal.

Thus for the four regions described, we have the following:

(i) For Region 1, where $i \le c-1$ and $j \le i+1$, (i.e.: $i = 0, 1, 2$ and $j = 0, 1, ..., i+1$), we use

$$p(i,j) = \frac{(i+1)!}{j!} \sum_{k=0}^{i-j+1} \frac{(-1)^k A^*((j+k)\mu)}{(i-j-k+1)!k!}$$

(ii) For Region 2, where $i = c, c+1, ..., N, \quad j = c, c+1, ..., i+1, \quad i+1 \le N$ (i.e.: $i = 3, 4, 5$ and $j = 3, 4, 5$), we use

$$p(i,j) = \frac{(c\mu)^{i-j+1}}{(i-j+1)!} A^*_{i-j+1}(c\mu)$$

$$= \frac{(3\mu)^{i-j+1} A^*_{i-j+1}(3\mu)}{(i-j+1)!}$$

(iii) For Region 3, where $1 \le j \le c-1 < i$ (i.e.: $i = 3, 4, 5$ and $j = 1, 2$), we use the convolution result from El-Taha [3]

$$p(i,j) = \sum_{k=1}^{2-j} \frac{C_{k,3-j}(3-k)}{j} \left(\frac{3}{k}\right)^{i-1} \left[A^*((3-k)\mu) - \sum_{r=0}^{i-2} \frac{(k\mu)^r A^*_r(3\mu)}{r!}\right]$$

$$+ C_{3-j,3-j} \left(\frac{3}{3-j}\right)^{i-1} \left[A^*(j\mu) - \sum_{r=0}^{i-2} \frac{((3-j)\mu)^r A^*_r(3\mu)}{r!}\right]$$

where

$$C_{k,3-j} = \prod_{m=1}^{k-1} \frac{3-m}{k-m} \cdot \prod_{m=k+1}^{3-j} \frac{3-m}{k-m} \quad \text{and} \quad C_{3-j,3-j} = \prod_{m=1}^{2-j} \frac{3-m}{3-j-m} \quad \text{or} \quad C_{v,v} = \prod_{m=1}^{v-1} \frac{3-m}{v-m}$$

thus we have $C_{1,1} = 1$, $C_{1,2} = -1$, $C_{2,2} = 2$

And for $i \geq c$ and $j = 0$, we simply use

$$p(i,j) = 1 - \sum_{n=1}^{N} p_{i,n}$$

(iv) Lastly for Region 4, where $j > i+1$ we have $p(i,j) = 0$. Thus

$$p_{0,2}, p_{0,3}, p_{0,4}, p_{0,5}, p_{0,6}, p_{1,3}, p_{1,4}, p_{1,5}, p_{1,6}, p_{2,4}, p_{2,5}, p_{2,6}, p_{3,5}, p_{3,6}, \text{and } p_{4,6}$$

all equal 0.

Thus our transition matrix is as follows:

$$P_{i,j} = \begin{bmatrix} 1 - A^*(\mu) & A^*(\mu) & 0 & 0 & 0 & 0 & 0 \\ 1 - 2A^*(\mu) + A^*(2\mu) & 2A^*(\mu) - 2A^*(2\mu) & A^*(2\mu) & 0 & 0 & 0 & 0 \\ 1 - 3A^*(\mu) + 3A^*(2\mu) - A^*(3\mu) & 3A^*(\mu) - 6A^*(2\mu) + 3A^*(3\mu) & 3A^*(2\mu) - 3A^*(3\mu) & A^*(3\mu) & 0 & 0 & 0 \\ 1 - \sum_{n=1}^{4} p_{3,n} & p_{3,1} & p_{3,2} & 3\mu A_1^*(3\mu) & A^*(3\mu) & 0 & 0 \\ 1 - \sum_{n=1}^{5} p_{4,n} & p_{4,1} & p_{4,2} & \frac{9}{2}\mu^2 A_2^*(3\mu) & 3\mu A_1^*(3\mu) & A^*(3\mu) & 0 \\ 1 - \sum_{n=1}^{6} p_{5,n} & p_{5,1} & p_{5,2} & \frac{9}{2}\mu^3 A_3^*(3\mu) & \frac{9}{2}\mu^2 A_2^*(3\mu) & 3\mu A_1^*(3\mu) & A^*(3\mu) \\ 1 - \sum_{n=1}^{6} p_{6,n} & p_{6,1} & p_{6,2} & \frac{9}{2}\mu^3 A_3^*(3\mu) & \frac{9}{2}\mu^2 A_2^*(3\mu) & 3\mu A_1^*(3\mu) & A^*(3\mu) \end{bmatrix}$$

where

$$p_{3,1} = \frac{9}{2}A^*(\mu) - 18A^*(2\mu) + \frac{27}{2}A^*(3\mu) + 9\mu A_1^*(3\mu)$$

$$p_{3,2} = 9A^*(2\mu) - 9A^*(3\mu) - 9\mu A_1^*(3\mu)$$

$$p_{4,1} = \frac{27}{4}A^*(\mu) - 54A^*(2\mu) + \frac{189}{4}A^*(3\mu) + \frac{81}{2}\mu A_1^*(3\mu) + \frac{27}{2}\mu^2 A_2^*(3\mu)$$

$$p_{4,2} = 27A^*(2\mu) - 27A^*(3\mu) - 27\mu A_1^*(3\mu) - \frac{27}{2}\mu^2 A_2^*(3\mu)$$

$$p_{5,1} = \frac{81}{8}A^*(\mu) - 162A^*(2\mu) + \frac{1215}{8}A^*(3\mu) + \frac{567}{4}\mu A_1^*(3\mu) + \frac{243}{4}\mu^2 A_2^*(3\mu) + \frac{27}{2}\mu^3 A_3^*(3\mu)$$

$$p_{5,2} = 81A^*(2\mu) - 81A^*(3\mu) - 81\mu A_1^*(3\mu) - \frac{81}{2}\mu^2 A_2^*(3\mu) - \frac{27}{2}\mu^3 A_3^*(3\mu)$$

$$p_{6,1} = p_{5,1}$$

$$p_{6,2} = p_{5,2}$$

3 Deterministic Arrivals

For deterministic arrivals with rate λ and $a(t) = 1/\lambda$ w.p. 1, we have

$$A_n^*(s) = \lambda^{-n} e^{-s/\lambda}$$

and

$$A_n^*(c\mu) = \lambda^{-n} e^{-c\mu/\lambda}$$

Letting $\lambda^{-1} = a$ we have

$$A_n^*(s) = a^n e^{-sa}$$

thus

$$A^*(\mu(j+m)) = e^{-\mu(j+m)a}$$

and

$$A_n^*(c\mu) = a^n e^{-c\mu a}$$

This gives the following:

$$A^*(0) = 1$$

$$A^*(\mu) = e^{-a\mu} \qquad A_1^*(3\mu) = ae^{-3a\mu}$$

$$A^*(2\mu) = e^{-2a\mu} \qquad A_2^*(3\mu) = a^2 e^{-3a\mu}$$

$$A^*(3\mu) = e^{-3a\mu} \qquad A_3^*(3\mu) = a^3 e^{-3a\mu}$$

Thus our transition matrix is as follows:

$$P_{i,j} = \begin{bmatrix} 1-e^{-a\mu} & e^{-a\mu} & 0 & 0 & 0 & 0 & 0 \\ 1-2e^{-a\mu}+e^{-2a\mu} & 2e^{-a\mu}-2e^{-2a\mu} & e^{-2a\mu} & 0 & 0 & 0 & 0 \\ 1-3e^{-a\mu}+3e^{-2a\mu}-e^{-3a\mu} & 3e^{-a\mu}-6e^{-2a\mu}+3e^{-3a\mu} & 3e^{-2a\mu}-3e^{-3a\mu} & e^{-3a\mu} & 0 & 0 & 0 \\ 1-\sum_{n=1}^{4} p_{3,n} & p_{3,1} & p_{3,2} & 3a\mu e^{-3a\mu} & e^{-3a\mu} & 0 & 0 \\ 1-\sum_{n=1}^{5} p_{4,n} & p_{4,1} & p_{4,2} & \frac{9}{2}a^2\mu^2 e^{-3a\mu} & 3a\mu e^{-3a\mu} & e^{-3a\mu} & 0 \\ 1-\sum_{n=1}^{6} p_{5,n} & p_{5,1} & p_{5,2} & \frac{9}{2}a^3\mu^3 e^{-3a\mu} & \frac{9}{2}a^2\mu^2 e^{-3a\mu} & 3a\mu e^{-3a\mu} & e^{-3a\mu} \\ 1-\sum_{n=1}^{6} p_{6,n} & p_{6,1} & p_{6,2} & \frac{9}{2}a^3\mu^3 e^{-3a\mu} & \frac{9}{2}a^2\mu^2 e^{-3a\mu} & 3a\mu e^{-3a\mu} & e^{-3a\mu} \end{bmatrix}$$

where

$$p_{3,1} = \frac{9}{2}e^{-a\mu} - 18e^{-2a\mu} + \frac{27}{2}e^{-3a\mu} + 9\mu a e^{-3a\mu}$$

$$p_{3,2} = 9e^{-2a\mu} - 9e^{-3a\mu} - 9\mu a e^{-3a\mu}$$

$$p_{4,1} = \frac{27}{4}e^{-a\mu} - 54e^{-2a\mu} + \frac{189}{4}e^{-3a\mu} + \frac{81}{2}\mu a e^{-3a\mu} + \frac{27}{2}\mu^2 a^2 e^{-3a\mu}$$

$$p_{4,2} = 27e^{-2a\mu} - 27e^{-3a\mu} - 27\mu a e^{-3a\mu} - \frac{27}{2}\mu^2 a^2 e^{-3a\mu}$$

$$p_{5,1} = \frac{81}{8}e^{-a\mu} - 162e^{-2a\mu} + \frac{1215}{8}e^{-3a\mu} + \frac{567}{4}\mu a e^{-3a\mu} + \frac{243}{4}\mu^2 a^2 e^{-3a\mu} + \frac{27}{2}\mu^3 a^3 e^{-3a\mu}$$

$$p_{5,2} = 81e^{-2a\mu} - 81e^{-3a\mu} - 81\mu a e^{-3a\mu} - \frac{81}{2}\mu^2 a^2 e^{-3a\mu} - \frac{27}{2}\mu^3 a^3 e^{-3a\mu}$$

$$p_{6,1} = p_{5,1}$$

$$p_{6,2} = p_{5,2}$$

4 Exponential arrivals

For exponential arrivals with $a(t) = \lambda e^{-\lambda t}, \lambda \geq 0, t \geq 0$ we have :

$$A_n^*(s) = n! \frac{\lambda}{(s+\lambda)^{n+1}}$$

thus

$$A^*(\mu(j+m)) = \frac{\lambda}{\mu(j+m)+\lambda}$$

This gives the following:

$$\begin{aligned} A^*(0) &= 1 \\ A^*(\mu) &= \frac{\lambda}{\mu+\lambda} & A_1^*(3\mu) &= \frac{\lambda}{(3\mu+\lambda)^2} \\ A^*(2\mu) &= \frac{\lambda}{2\mu+\lambda} & A_2^*(3\mu) &= \frac{2\lambda}{(3\mu+\lambda)^3} \\ A^*(3\mu) &= \frac{\lambda}{3\mu+\lambda} & A_3^*(3\mu) &= \frac{6\lambda}{(3\mu+\lambda)^4} \end{aligned}$$

Thus our transition matrix is as follows:

$$P_{i,j} = \begin{bmatrix} 1-\frac{\lambda}{\mu+\lambda} & \frac{\lambda}{\mu+\lambda} & 0 & 0 & 0 & 0 & 0 \\ 1-\frac{2\lambda}{\mu+\lambda}+\frac{\lambda}{2\mu+\lambda} & \frac{2\lambda}{\mu+\lambda}-\frac{2\lambda}{2\mu+\lambda} & \frac{\lambda}{2\mu+\lambda} & 0 & 0 & 0 & 0 \\ 1-\frac{3\lambda}{\mu+\lambda}+\frac{3\lambda}{2\mu+\lambda}-\frac{\lambda}{3\mu+\lambda} & \frac{3\lambda}{\mu+\lambda}-\frac{6\lambda}{2\mu+\lambda}+\frac{3\lambda}{3\mu+\lambda} & \frac{3\lambda}{2\mu+\lambda}-\frac{3\lambda}{3\mu+\lambda} & \frac{\lambda}{3\mu+\lambda} & 0 & 0 & 0 \\ 1-\sum_{n=1}^4 p_{3,n} & p_{3,1} & p_{3,2} & \frac{3\mu\lambda}{(3\mu+\lambda)^2} & \frac{\lambda}{3\mu+\lambda} & 0 & 0 \\ 1-\sum_{n=1}^5 p_{4,n} & p_{4,1} & p_{4,2} & \frac{9\mu^2\lambda}{(3\mu+\lambda)^3} & \frac{3\mu\lambda}{(3\mu+\lambda)^2} & \frac{\lambda}{3\mu+\lambda} & 0 \\ 1-\sum_{n=1}^6 p_{5,n} & p_{5,1} & p_{5,2} & \frac{27\mu^3\lambda}{(3\mu+\lambda)^4} & \frac{9\mu^2\lambda}{(3\mu+\lambda)^3} & \frac{3\mu\lambda}{(3\mu+\lambda)^2} & \frac{\lambda}{3\mu+\lambda} \\ 1-\sum_{n=1}^6 p_{6,n} & p_{6,1} & p_{6,2} & \frac{27\mu^3\lambda}{(3\mu+\lambda)^4} & \frac{9\mu^2\lambda}{(3\mu+\lambda)^3} & \frac{3\mu\lambda}{(3\mu+\lambda)^2} & \frac{\lambda}{3\mu+\lambda} \end{bmatrix}$$

where

$$p_{3,1} = \frac{9\lambda}{2(\mu+\lambda)} - \frac{18\lambda}{2\mu+\lambda} + \frac{27\lambda}{2(3\mu+\lambda)} + \frac{9\mu\lambda}{(3\mu+\lambda)^2}$$

$$p_{3,2} = \frac{9\lambda}{2\mu+\lambda} - \frac{9\lambda}{3\mu+\lambda} - \frac{9\mu\lambda}{(3\mu+\lambda)^2}$$

$$p_{4,1} = \frac{27\lambda}{4(\mu+\lambda)} - \frac{54\lambda}{2\mu+\lambda} + \frac{189\lambda}{4(3\mu+\lambda)} + \frac{81\mu\lambda}{2(3\mu+\lambda)^2} + \frac{27\mu^2\lambda}{(3\mu+\lambda)^3}$$

$$p_{4,2} = \frac{27\lambda}{2\mu+\lambda} - \frac{27\lambda}{3\mu+\lambda} - \frac{27\mu\lambda}{(3\mu+\lambda)^2} - \frac{27\mu^2\lambda}{(3\mu+\lambda)^3}$$

$$p_{5,1} = \frac{81\lambda}{8(\mu+\lambda)} - \frac{162\lambda}{2\mu+\lambda} + \frac{1215\lambda}{8(3\mu+\lambda)} + \frac{567\mu\lambda}{4(3\mu+\lambda)^2} + \frac{243\mu^2\lambda}{2(3\mu+\lambda)^3} + \frac{81\mu^3\lambda}{(3\mu+\lambda)^4}$$

$$p_{5,2} = \frac{81\lambda}{2\mu+\lambda} - \frac{81\lambda}{3\mu+\lambda} - \frac{81\mu\lambda}{(3\mu+\lambda)^2} - \frac{81\mu^2\lambda}{(3\mu+\lambda)^3} - \frac{81\mu^3\lambda}{(3\mu+\lambda)^4}$$

$$p_{6,1} = p_{5,1}$$

$$p_{6,2} = p_{5,2}$$

5 Erlang Arrivals

For an Erlang distribution of arrivals, with $a(t) = \frac{(k\lambda)^k t^{k-1}}{(k-1)!} e^{-k\lambda t}, \lambda \geq 0, t \geq 0$ we have

:

$$A_n^*(s) = n! \binom{k+n-1}{n} \frac{(k\lambda)^k}{(s+k\lambda)^{n+k}}$$

Thus for $k = 2$ we have:

$$A_n^*(s) = n!(n+1) \frac{4\lambda^2}{(s+2\lambda)^{n+2}}$$

This gives the following:

$$\begin{aligned} A^*(0) &= 1 \\ A^*(\mu) &= \frac{4\lambda^2}{(\mu+2\lambda)^2} & A_1^*(3\mu) &= \frac{8\lambda^2}{(3\mu+2\lambda)^3} \\ A^*(2\mu) &= \frac{4\lambda^2}{(2\mu+2\lambda)^2} = \frac{\lambda^2}{(\mu+\lambda)^2} & A_2^*(3\mu) &= \frac{24\lambda^2}{(3\mu+2\lambda)^4} \\ A^*(3\mu) &= \frac{4\lambda^2}{(3\mu+2\lambda)^2} & A_3^*(3\mu) &= \frac{96\lambda^2}{(3\mu+2\lambda)^5} \end{aligned}$$

$$P_{i,j} = \begin{bmatrix} 1 - \frac{4\lambda^2}{(\mu+2\lambda)^2} & \frac{4\lambda^2}{(\mu+2\lambda)^2} & 0 & 0 & 0 & 0 & 0 \\ 1 - \frac{8\lambda^2}{(\mu+2\lambda)^2} + \frac{\lambda^2}{(\mu+\lambda)^2} & \frac{8\lambda^2}{(\mu+2\lambda)^2} - \frac{2\lambda^2}{(\mu+\lambda)^2} & \frac{\lambda^2}{(\mu+\lambda)^2} & 0 & 0 & 0 & 0 \\ 1 - \frac{12\lambda^2}{(\mu+2\lambda)^2} + \frac{3\lambda^2}{(\mu+\lambda)^2} - \frac{4\lambda^2}{(3\mu+2\lambda)^2} & \frac{12\lambda^2}{(\mu+2\lambda)^2} - \frac{6\lambda^2}{(\mu+\lambda)^2} + \frac{12\lambda^2}{(3\mu+2\lambda)^2} & \frac{3\lambda^2}{(\mu+\lambda)^2} - \frac{12\lambda^2}{(3\mu+2\lambda)^2} & \frac{4\lambda^2}{(3\mu+2\lambda)^2} & 0 & 0 & 0 \\ 1 - \sum_{n=1}^{4} p_{3,n} & p_{3,1} & p_{3,2} & \frac{24\mu\lambda^2}{(3\mu+2\lambda)^3} & \frac{4\lambda^2}{(3\mu+2\lambda)^2} & 0 & 0 \\ 1 - \sum_{n=1}^{5} p_{4,n} & p_{4,1} & p_{4,2} & \frac{108\mu^2\lambda^2}{(3\mu+2\lambda)^4} & \frac{24\mu\lambda^2}{(3\mu+2\lambda)^3} & \frac{4\lambda^2}{(3\mu+2\lambda)^2} & 0 \\ 1 - \sum_{n=1}^{6} p_{5,n} & p_{5,1} & p_{5,2} & \frac{432\mu^3\lambda^2}{(3\mu+2\lambda)^5} & \frac{108\mu^2\lambda^2}{(3\mu+2\lambda)^4} & \frac{24\mu\lambda^2}{(3\mu+2\lambda)^3} & \frac{4\lambda^2}{(3\mu+2\lambda)^2} \\ 1 - \sum_{n=1}^{6} p_{6,n} & p_{6,1} & p_{6,2} & \frac{432\mu^3\lambda^2}{(3\mu+2\lambda)^5} & \frac{108\mu^2\lambda^2}{(3\mu+2\lambda)^4} & \frac{24\mu\lambda^2}{(3\mu+2\lambda)^3} & \frac{4\lambda^2}{(3\mu+2\lambda)^2} \end{bmatrix}$$

where

$$p_{3,1} = \frac{18\lambda^2}{(\mu+2\lambda)^2} - \frac{18\lambda^2}{(\mu+\lambda)^2} + \frac{54\lambda^2}{(3\mu+2\lambda)^2} + \frac{72\mu\lambda^2}{(3\mu+2\lambda)^3}$$

$$p_{3,2} = \frac{36\lambda^2}{(2\mu+2\lambda)^2} - \frac{36\lambda^2}{(3\mu+2\lambda)^2} - \frac{72\mu\lambda^2}{(3\mu+2\lambda)^3}$$

$$p_{4,1} = \frac{27\lambda^2}{(\mu+2\lambda)^2} - \frac{54\lambda^2}{(\mu+\lambda)^2} + \frac{189\lambda^2}{(3\mu+2\lambda)^2} + \frac{324\mu\lambda^2}{(3\mu+2\lambda)^3} + \frac{324\mu^2\lambda^2}{(3\mu+2\lambda)^4}$$

$$p_{4,2} = \frac{108\lambda^2}{(2\mu+2\lambda)^2} - \frac{108\lambda^2}{(3\mu+2\lambda)^2} - \frac{216\mu\lambda^2}{(3\mu+2\lambda)^3} - \frac{324\mu^2\lambda^2}{(3\mu+2\lambda)^4}$$

$$p_{5,1} = \frac{81\lambda^2}{2(\mu+2\lambda)^2} - \frac{162\lambda^2}{(\mu+\lambda)^2} + \frac{1215\lambda^2}{2(3\mu+2\lambda)^2} + \frac{1134\mu\lambda^2}{(3\mu+2\lambda)^3} + \frac{1458\mu^2\lambda^2}{(3\mu+2\lambda)^4} + \frac{1296\mu^3\lambda^2}{(3\mu+2\lambda)^5}$$

$$p_{5,2} = \frac{324\lambda^2}{(2\mu+2\lambda)^2} - \frac{324\lambda^2}{(3\mu+2\lambda)^2} - \frac{648\mu\lambda^2}{(3\mu+2\lambda)^3} - \frac{972\mu^2\lambda^2}{(3\mu+2\lambda)^4} - \frac{1296\mu^3\lambda^2}{(3\mu+2\lambda)^5}$$

$$p_{6,1} = p_{5,1}$$

$$p_{6,2} = p_{5,2}$$

6 Hyperexponential Arrivals

For a hyperexponential mixture of exponential distributions with

$$a(t) = \sum_{i=1}^{k} p_i \lambda_i e^{-\lambda_i t}, \quad 0 \leq p_i \leq 1, \ \sum_{i=0}^{k} p_i = 1, \ \lambda_i \geq 0, \ t \geq 0$$

where λ_i are the means of the exponential distributions, and events are members of distribution i with probability p_i, we have :

$$A_n^*(s) \;\; = n! \sum_{i=1}^{k} \frac{p_i \lambda_i}{(s+\lambda_i)^{n+1}}$$

where p_i are the relative weights, and λ_i are the individual means of the exponential distributions composing the mixture.

Thus, for a mixture of two exponentials we have:

$$A_n^*(s) = n! \left[\frac{p\lambda_1}{(s+\lambda_1)^{n+1}} + \frac{(1-p)\lambda_2}{(s+\lambda_2)^{n+1}} \right]$$

This gives the following:

$$A^*(0) \quad = 1$$

$$A^*(\mu) \quad = \frac{p\lambda_1}{\mu+\lambda_1} + \frac{(1-p)\lambda_2}{\mu+\lambda_2} \qquad A_1^*(3\mu) \quad = \frac{p\lambda_1}{(3\mu+\lambda_1)^2} + \frac{(1-p)\lambda_2}{(3\mu+\lambda_2)^2}$$

$$A^*(2\mu) \quad = \frac{p\lambda_1}{2\mu+\lambda_1} + \frac{(1-p)\lambda_2}{2\mu+\lambda_2} \qquad A_2^*(3\mu) \quad = \frac{2p\lambda_1}{(3\mu+\lambda_1)^3} + \frac{2(1-p)\lambda_2}{(3\mu+\lambda_2)^3}$$

$$A^*(3\mu) \quad = \frac{p\lambda_1}{3\mu+\lambda_1} + \frac{(1-p)\lambda_2}{3\mu+\lambda_2} \qquad A_3^*(3\mu) \quad = \frac{6p\lambda_1}{(3\mu+\lambda_1)^4} + \frac{6(1-p)\lambda_2}{(3\mu+\lambda_2)^4}$$

$$P_{i,j} = \begin{bmatrix} 1-\frac{p\lambda_1}{\mu+\lambda_1}-\frac{(1-p)\lambda_2}{\mu+\lambda_2} & \frac{p\lambda_1}{\mu+\lambda_1}+\frac{(1-p)\lambda_2}{\mu+\lambda_2} & 0 & 0 & 0 & 0 & 0 \\ 1-\frac{2p\lambda_1}{\mu+\lambda_1}-\frac{2(1-p)\lambda_2}{\mu+\lambda_2}+\frac{p\lambda_1}{2\mu+\lambda_1}+\frac{(1-p)\lambda_2}{2\mu+\lambda_2} & \frac{2p\lambda_1}{\mu+\lambda_1}+\frac{2(1-p)\lambda_2}{\mu+\lambda_2}-\frac{2p\lambda_1}{2\mu+\lambda_1}-\frac{2(1-p)\lambda_2}{2\mu+\lambda_2} & \frac{3p\lambda_1}{2\mu+\lambda_1}+\frac{3(1-p)\lambda_2}{2\mu+\lambda_2} & 0 & 0 & 0 & 0 \\ 1-\frac{3p\lambda_1}{\mu+\lambda_1}-\frac{3(1-p)\lambda_2}{\mu+\lambda_2}+\frac{3p\lambda_1}{2\mu+\lambda_1}+\frac{3(1-p)\lambda_2}{2\mu+\lambda_2}-\frac{p\lambda_1}{3\mu+\lambda_1}-\frac{(1-p)\lambda_2}{3\mu+\lambda_2} & \frac{3p\lambda_1}{\mu+\lambda_1}+\frac{3(1-p)\lambda_2}{\mu+\lambda_2}-\frac{6p\lambda_1}{2\mu+\lambda_1}-\frac{6(1-p)\lambda_2}{2\mu+\lambda_2}+\frac{3p\lambda_1}{3\mu+\lambda_1}+\frac{3(1-p)\lambda_2}{3\mu+\lambda_2} & \frac{3p\lambda_1}{2\mu+\lambda_1}+\frac{3(1-p)\lambda_2}{2\mu+\lambda_2}-\frac{3p\lambda_1}{3\mu+\lambda_1}-\frac{3(1-p)\lambda_2}{3\mu+\lambda_2} & \frac{p\lambda_1}{3\mu+\lambda_1}-\frac{(1-p)\lambda_2}{3\mu+\lambda_2} & 0 & 0 & 0 \\ 1-\sum_{n=1}^{4}p_{3,n} & p_{3,1} & p_{3,2} & \frac{3\mu p\lambda_1}{(3\mu+\lambda_1)^2}+\frac{3\mu(1-p)\lambda_2}{(3\mu+\lambda_2)^2} & \frac{p\lambda_1}{3\mu+\lambda_1}+\frac{(1-p)\lambda_2}{3\mu+\lambda_2} & 0 & 0 \\ 1-\sum_{n=1}^{5}p_{4,n} & p_{4,1} & p_{4,2} & \frac{9\mu^2 p\lambda_1}{(3\mu+\lambda_1)^3}+\frac{9\mu^2(1-p)\lambda_2}{(3\mu+\lambda_2)^3} & \frac{3\mu p\lambda_1}{(3\mu+\lambda_1)^2}+\frac{3\mu(1-p)\lambda_2}{(3\mu+\lambda_2)^2} & \frac{p\lambda_1}{3\mu+\lambda_1}+\frac{(1-p)\lambda_2}{3\mu+\lambda_2} & 0 \\ 1-\sum_{n=1}^{6}p_{5,n} & p_{5,1} & p_{5,2} & \frac{27\mu^3 p\lambda_1}{(3\mu+\lambda_1)^4}+\frac{27\mu^3(1-p)\lambda_2}{(3\mu+\lambda_2)^4} & \frac{9\mu^2 p\lambda_1}{(3\mu+\lambda_1)^3}+\frac{9\mu^2(1-p)\lambda_2}{(3\mu+\lambda_2)^3} & \frac{3\mu p\lambda_1}{(3\mu+\lambda_1)^2}+\frac{3\mu(1-p)\lambda_2}{(3\mu+\lambda_2)^2} & \frac{p\lambda_1}{3\mu+\lambda_1}+\frac{(1-p)\lambda_2}{3\mu+\lambda_2} \\ 1-\sum_{n=1}^{6}p_{6,n} & p_{6,1} & p_{6,2} & \frac{27\mu^3 p\lambda_1}{(3\mu+\lambda_1)^4}+\frac{27\mu^3(1-p)\lambda_2}{(3\mu+\lambda_2)^4} & \frac{9\mu^2 p\lambda_1}{(3\mu+\lambda_1)^3}+\frac{9\mu^2(1-p)\lambda_2}{(3\mu+\lambda_2)^3} & \frac{3\mu p\lambda_1}{(3\mu+\lambda_1)^2}+\frac{3\mu(1-p)\lambda_2}{(3\mu+\lambda_2)^2} & \frac{p\lambda_1}{3\mu+\lambda_1}+\frac{(1-p)\lambda_2}{3\mu+\lambda_2} \end{bmatrix}$$

where

$$p_{3,1} = \frac{9p\lambda_1}{2(\mu+\lambda_1)} + \frac{9(1-p)\lambda_2}{2(\mu+\lambda_2)} - \frac{18p\lambda_1}{2\mu+\lambda_1} - \frac{18(1-p)\lambda_2}{2\mu+\lambda_2} + \frac{27p\lambda_1}{2(3\mu+\lambda_1)} + \frac{27(1-p)\lambda_2}{2(3\mu+\lambda_2)} + \frac{9\mu p\lambda_1}{(3\mu+\lambda_1)^2} + \frac{9\mu(1-p)\lambda_2}{(3\mu+\lambda_2)^2}$$

$$p_{3,2} = \frac{9p\lambda_1}{2\mu+\lambda_1} + \frac{9(1-p)\lambda_2}{2\mu+\lambda_2} - \frac{9p\lambda_1}{3\mu+\lambda_1} - \frac{9(1-p)\lambda_2}{3\mu+\lambda_2} - \frac{9\mu p\lambda_1}{(3\mu+\lambda_1)^2} - \frac{9\mu(1-p)\lambda_2}{(3\mu+\lambda_2)^2}$$

$$p_{4,1} = \frac{27p\lambda_1}{4(\mu+\lambda_1)} + \frac{27(1-p)\lambda_2}{4(\mu+\lambda_2)} - \frac{54p\lambda_1}{2\mu+\lambda_1} - \frac{54(1-p)\lambda_1}{2\mu+\lambda_1} + \frac{189p\lambda_1}{4(3\mu+\lambda_1)} + \frac{189(1-p)\lambda_2}{4(3\mu+\lambda_2)} + \frac{81p\mu\lambda_1}{2(3\mu+\lambda_1)^2} + \frac{81(1-p)\mu\lambda_2}{2(3\mu+\lambda_2)^2} + \frac{27p\mu^2\lambda_1}{(3\mu+\lambda_1)^3} + \frac{27(1-p)\mu^2\lambda_2}{(3\mu+\lambda_2)^3}$$

$$p_{4,2} = \frac{27p\lambda_1}{2\mu+\lambda_1} + \frac{27(1-p)\lambda_2}{2\mu+\lambda_2} - \frac{27p\lambda_1}{3\mu+\lambda_1} - \frac{27(1-p)\lambda_2}{3\mu+\lambda_2} - \frac{27p\mu\lambda_1}{(3\mu+\lambda_1)^2} - \frac{27(1-p)\mu\lambda_2}{(3\mu+\lambda_2)^2} - \frac{27p\mu^2\lambda_1}{(3\mu+\lambda_1)^3} - \frac{27(1-p)\mu^2\lambda_2}{(3\mu+\lambda_2)^3}$$

$$p_{5,1} = \frac{81p\lambda_1}{8(\mu+\lambda_1)} + \frac{81(1-p)\lambda_2}{8(\mu+\lambda_2)} - \frac{162p\lambda_1}{2\mu+\lambda_1} - \frac{162(1-p)\lambda_2}{2\mu+\lambda_2} + \frac{1215p\lambda_1}{8(3\mu+\lambda_1)} + \frac{1215(1-p)\lambda_2}{8(3\mu+\lambda_2)} + \frac{567p\mu\lambda_1}{4(3\mu+\lambda_1)^2} + \frac{567(1-p)\mu\lambda_2}{4(3\mu+\lambda_2)^2} + \frac{243p\mu^2\lambda_1}{2(3\mu+\lambda_1)^3} + \frac{243(1-p)\mu^2\lambda_2}{2(3\mu+\lambda_2)^3} + \frac{81p\mu^3\lambda_1}{(3\mu+\lambda_1)^4} + \frac{81(1-p)\mu^3\lambda_2}{(3\mu+\lambda_2)^4}$$

$$p_{5,2} = \frac{81p\lambda_1}{2\mu+\lambda_1} + \frac{81(1-p)\lambda_2}{2\mu+\lambda_2} - \frac{81p\lambda_1}{3\mu+\lambda_1} - \frac{81(1-p)\lambda_2}{3\mu+\lambda_2} - \frac{81p\mu\lambda_1}{(3\mu+\lambda_1)^2} - \frac{81(1-p)\mu\lambda_2}{(3\mu+\lambda_2)^2} - \frac{81p\mu^2\lambda_1}{(3\mu+\lambda_1)^3} - \frac{81(1-p)\mu^2\lambda_2}{(3\mu+\lambda_2)^3} - \frac{81p\mu^3\lambda_1}{(3\mu+\lambda_2)^4} - \frac{81(1-p)\mu^3\lambda_2}{(3\mu+\lambda_2)^4}$$

$$p_{6,1} = p_{5,1}$$

$$p_{6,2} = p_{5,2}$$

7 Numerical Results

Here we provide numerical results for the distributions given above, for a system with $c = 3$ servers and capacity of $K = 6$, and utilization factor $\rho = 5/6$, that is to say with an arrival rate of five and an overall service rate of six.

Table 3.1: Numerical Results: Small-Scale Finite Buffer Model With Exponential Arrivals

p_n	**Exponential - Convolution**	**Exponential - Traditional**
0	0.067958810	0.067958810
1	0.169897026	0.169897026
2	0.212371283	0.212371283
3	0.176976069	0.176976069
4	0.147480057	0.147480057
5	0.122900048	0.122900048
6	0.102416707	0.102416707

Table 3.2: Numerical Results: Small-Scale Finite Buffer Model With Deterministic, Erlang, or Hyperexponential Arrivals

p_n	**Deterministic**	**Erlang**	**Hyper - exponential**
0	0.047234853	0.060394802	0.095667547
1	0.127764093	0.153533580	0.170777140
2	0.254669333	0.228194616	0.182173364
3	0.232414603	0.196990341	0.153721590
4	0.156638062	0.150739248	0.148862427
5	0.107500964	0.117912665	0.131909935
6	0.073778091	0.092234748	0.116887997

Table 3.3: Performance Measures, Small-Scale Finite Buffer Models

Distribution and Method	**L**	**W**
Exponential - Convolution	2.944488506	0.656092538
Exponential - Traditional	2.944488506	0.656092538
Deterministic - Convolution	2.941072182	0.635068584
Erlang - Convolution	2.946822639	0.649247728
Hyperexponential - Convolution	2.952616004	0.668684379

Chapter 4

Large-scale Examples with Infinite Waiting Space

1 Introduction

Here we use the Python programming language and the convolution method from El-Taha [3] to generate probabilities for large-scale examples ($c = 30$) with $\rho = 5/6$. The software code is given in Appendix B. Use of the Decimal package, a fixed-decimal package capable of arbitrarily long mantissas, is notable in addressing the overflow errors associated with large factorials and the underflow issues created by the LST values with large c and n. We address the infinite waiting space model by truncating the transition matrix at $p_{N,N}$ at some value of $N > c$, obtained by defining some $\epsilon > 0$ as the maximum acceptable error (see Step 3 of the methodology).

2 Computational Methodology

Our computational methodology follows that provided in El-Taha[3], wherein the transition matrix is prepared using the methods described in Chapter 2, $|\sigma| < 1$ is determined from $A^*(c\mu(1-\sigma)) = \sigma$, and $\pi(j)$ values are prepared using σ and/or the transition matrix and then normalized to provide steady-state system size probabilities from which we obtain the performance measures.

Given $\epsilon =$ maximum allowable error or $N =$ truncation point, $c =$ number of servers each with mean rate $= \mu$, the type of arrival distribution and its parameters (e.g.: overall mean rate, and number of phases and/or weights if applicable):

1. Compute $\rho = \lambda/c\mu$ where λ is the mean of the arrival distribution, to confirm $\rho < 1$ (i.e.: a long-run solution exists).

2. Root-solve by iterating over the following until $|\sigma_{n+1} - \sigma_n| < \epsilon$

$$\sigma_{n+1} = A^*[c\mu(1-\sigma_n)]$$

3. For a given ϵ, determine N using

$$N = \min\left\{ n \in \mathbb{N} \;\middle|\; n \geq c + \frac{\ln(\epsilon) - 2\ln(1-\sigma)}{\ln(\sigma)} \right\}$$

4. For the specified distribution, compute

$$A^*(s) \text{ for } s = k\mu \text{ where } k = 1, 2, ..., c \text{ and } A_n^*(c\mu) \text{ for } n = 1, 2, ..., N - c + 1$$

Deterministic:

$$A^*(s) = e^{-s\lambda} \qquad\qquad A_n^*(c\mu) = \lambda^n e^{-c\mu\lambda}$$

Exponential:

$$A^*(s) = \frac{\lambda}{s+\lambda} \qquad\qquad A_n^*(c\mu) = \frac{n!\lambda}{(c\mu+\lambda)^{n+1}}$$

Erlang (k-phase):

$$A^*(s) = \frac{(k\lambda)^k}{(s+k\lambda)^k} \qquad\qquad A_n^*(c\mu) = n!\binom{k+n-1}{n}\frac{(k\lambda)^k}{(c\mu+k\lambda)^{k+n}}$$

Erlang (two-phase):

$$A^*(s) = \frac{4\lambda^2}{(s+2\lambda)^2} \qquad\qquad A_n^*(c\mu) = n!(n+1)\frac{4\lambda^2}{(c\mu+2\lambda)^{n+2}}$$

Hyper-exponential:

$$A^*(s) = \sum_{i=1}^{k}\frac{p_i\lambda_i}{s+\lambda_i} \qquad\qquad A_n^*(c\mu) = n!\sum_{i=1}^{k}\frac{p_i\lambda_i}{(c\mu+\lambda_i)^{n+1}}$$

Hyper-exponential (k=2):

$$A^*(s) = \frac{p\lambda_1}{s+\lambda_1} + \frac{(1-p)\lambda_2}{s+\lambda_2} \qquad\qquad A_n^*(c\mu) = n!\left[\frac{p\lambda_1}{(c\mu+\lambda_1)^{n+1}} + \frac{(1-p)\lambda_2}{(c\mu+\lambda_2)^{n+1}}\right]$$

5. Compute $C_{k,c-j}$ for $k = 1, 2, ..., c-1, \quad (c-j) = 1, 2, ..., c-1$ using

$$C_{k,c-j} = \prod_{m=1}^{k-1} \frac{c-m}{k-m} \cdot \prod_{m=k+1}^{c-j} \frac{c-m}{k-m}$$

where the product over an empty set ($k = 1$ or $c - j \leq k$) is 1.

6. Define $p(i,j) = 0$ for all $i = 0, 1, ..., N-2, \;\; j = i+2, i+3, ..., N$.

7. Compute $p(i,j)$ for $i = 1, 2, ..., c-1, \;\; j = 1, 2, ..., i+1$ using

$$p(i,j) = \binom{i+1}{i-j+1} \sum_{r=0}^{i-j+1} (-1)^r \binom{i-j+1}{r} A^*((j+r)\mu))$$

8. Compute $p(i,j)$ for $i = c, c+1, ..., N, \;\; j = c, c+1, ..., i+1, \quad i+1 \leq N$ using

$$p(i,j) = \frac{(c\mu)^{i-j+1} A^*_{i-j+1}(c\mu)}{(i-j+1)!}$$

9. Compute $p(i,j)$ for $i = c, c+1, ..., N, \;\; j = 1, 2, ..., c-1$ using

$$p(i,j) = \sum_{k=1}^{c-j-1} \frac{C_{k,c-j}(c-k)}{j} \left(\frac{c}{k}\right)^{i-c+2} \left[A^*((c-k)\mu) - \sum_{r=0}^{i-c+1} \frac{(k\mu)^r A^*_r(c\mu)}{r!}\right]$$

$$+ \; C_{c-j,c-j} \left(\frac{c}{c-j}\right)^{i-c+2} \left[A^*(j\mu) - \sum_{r=0}^{i-c+1} \frac{((c-j)\mu)^r A^*_r(c\mu)}{r!}\right]$$

10. Compute $p(i,j)$ for $i = c, c+1, ..., N, \;\; j = 0$ using

$$p(i,j) = 1 - \sum_{n=1}^{N} p_{i,n}$$

11. Define $a(k, j) = 0$ for $k = 0, 1, ..., N, \quad j = k, k + 1, ..., N$

12. Compute $a(k, j)$ for $k = j + 1, j + 2, ..., N, \quad j = 0, 1, ..., c - 1$ using

$$a(k, j) = \sum_{i=0}^{j} p(k, i)$$

13. Compute $\pi'(j) = \sigma^j$ for $j = c, c + 1, ..., N$

14. Compute $\pi'(j)$ for $j = c - 1, c - 2, ..., 0$ recursively using

$$\pi'(j) = \frac{\sum_{k=j+1}^{N} \pi'(k) a(k, j)}{p(j, j + 1)}$$

15. Compute $\pi(j)$ for $j = 0, 1, ..., N$ by normalizing π'_j using

$$\pi(j) = \frac{\pi'(j)}{\phi}$$

where

$$\phi = \sum_{k=0}^{N} \pi'(k) = \sum_{k=0}^{c-1} \pi'(k) + \sum_{k=c}^{N} \sigma^k = \sum_{k=0}^{c-1} \pi'(k) + \frac{\sigma^c(1 - \sigma^{N-c+1})}{(1 - \sigma)}$$

16. Compute p_0 using

$$p_0 = (1 - \rho) + \rho\pi(N) - \rho \sum_{k=0}^{c-2} \frac{c - k - 1}{k + 1} \pi(k)$$

17. Compute p_n for $n = 1, 2, ..., c - 1$ using

$$p_n = \frac{c\rho\pi(n - 1)}{n}$$

18. Compute p_n for $n = c, c+1, ..., N$ using

$$p_n = \rho\pi(n-1)$$

19. Compute performance measures using

$$E[L] = \sum_{i=1}^{N} ip(i)$$

$$E[W] = E[L]/\lambda$$

3 Large-scale Results

3.1 Introduction

Here we present large-scale results using the convolution method for $c = 30, \rho = 5/6$, as generated via the Python software shown in Appendix B. The results for Exponential arrivals are compared with traditional methods for computing $M/M/c$ queues such as are found in Gross & Harris[5] and Kleinrock[8]. This serves to validate the model.

For deterministic, Erlang, and hyperexponential arrivals, our results are compared to the Takacs method as generated by the QTS software provided by Gross & Harris [5].

3.2 Programmatic Methodology

As can be seen from the QTS (Takacs) results for the Erlang, deterministic, and hyperexponential distributions, difficulties with floating-point overflow/underflow exist with this number of servers, and persist as low as $c = 10$. These problems expand with increasing c, limiting usable results from that software.

With the exception of p_0, which compounds the error present in all other values of p_n, our methodology is more numerically stable than Takacs implementations even when using floating-point levels of precision. In addition, our use of the Decimal package to provide a fixed-decimal methodology with longer mantissas provides accuracy for substantially higher values of c while also reducing the error of p_0 below that of floating-point implementations.

Table 4.1: Exponential Arrivals
Comparison with Traditional Methodology

p_n	Exponential - Convolution	Exponential - Traditional	Absolute Difference	Percent Difference
0	0.000000000013	0.000000000013	0.000000000000	0.00000000
1	0.000000000318	0.000000000318	0.000000000000	0.00000000
2	0.000000003980	0.000000003980	0.000000000000	0.00000000
3	0.000000033169	0.000000033169	0.000000000000	0.00000000
4	0.000000207306	0.000000207306	0.000000000000	0.00000000
5	0.000001036530	0.000001036530	0.000000000000	0.00000000
6	0.000004318870	0.000004318870	0.000000000000	0.00000000
7	0.000015424500	0.000015424500	0.000000000000	0.00000000
8	0.000048201700	0.000048201700	0.000000000000	0.00000000
9	0.000133894000	0.000133894000	0.000000000000	0.00000000
10	0.000334734000	0.000334734000	0.000000000000	0.00000000
11	0.000760759000	0.000760759000	0.000000000000	0.00000000
12	0.001584915000	0.001584915000	0.000000000000	0.00000000
13	0.003047914000	0.003047914000	0.000000000000	0.00000000
14	0.005442704000	0.005442704000	0.000000000000	0.00000000
15	0.009071173000	0.009071173000	0.000000000000	0.00000000
16	0.014173708000	0.014173708000	0.000000000000	0.00000000
17	0.020843689000	0.020843689000	0.000000000000	0.00000000
18	0.028949568000	0.028949568000	0.000000000000	0.00000000
19	0.038091536000	0.038091536000	0.000000000000	0.00000000
20	0.047614420000	0.047614420000	0.000000000000	0.00000000
21	0.056683834000	0.056683834000	0.000000000000	0.00000000
22	0.064413447000	0.064413447000	0.000000000000	0.00000000
23	0.070014617000	0.070014617000	0.000000000000	0.00000000
24	0.072931893000	0.072931893000	0.000000000000	0.00000000
25	0.072931893000	0.072931893000	0.000000000000	0.00000000
26	0.070126820000	0.070126820000	0.000000000000	0.00000000
27	0.064932240000	0.064932240000	0.000000000000	0.00000000
28	0.057975215000	0.057975215000	0.000000000000	0.00000000
29	0.049978633000	0.049978633000	0.000000000000	0.00000000
30	0.041648861000	0.041648861000	0.000000000000	0.00000000
31	0.034707384000	0.034707384000	0.000000000000	0.00000000
32	0.028922820000	0.028922820000	0.000000000000	0.00000000
33	0.024102350000	0.024102350000	0.000000000000	0.00000000
34	0.020085292000	0.020085292000	0.000000000000	0.00000000
35	0.016737743000	0.016737743000	0.000000000000	0.00000000

Table 4.1: Exponential Arrivals
Comparison with Traditional Methodology

p_n	Exponential - Convolution	Exponential - Traditional	Absolute Difference	Percent Difference
36	0.013948119000	0.013948119000	0.000000000000	0.00000000
37	0.011623433000	0.011623433000	0.000000000000	0.00000000
38	0.009686194000	0.009686194000	0.000000000000	0.00000000
39	0.008071828000	0.008071828000	0.000000000000	0.00000000
40	0.006726524000	0.006726524000	0.000000000000	0.00000000
41	0.005605436000	0.005605436000	0.000000000000	0.00000000
42	0.004671197000	0.004671197000	0.000000000000	0.00000000
43	0.003892664000	0.003892664000	0.000000000000	0.00000000
44	0.003243887000	0.003243887000	0.000000000000	0.00000000
45	0.002703239000	0.002703239000	0.000000000000	0.00000000
46	0.002252699000	0.002252699000	0.000000000000	0.00000000
47	0.001877249000	0.001877249000	0.000000000000	0.00000000
48	0.001564374000	0.001564374000	0.000000000000	0.00000000
49	0.001303645000	0.001303645000	0.000000000000	0.00000000
50	0.001086371000	0.001086371000	0.000000000000	0.00000000
51	0.000905309000	0.000905309000	0.000000000000	0.00000000
52	0.000754424000	0.000754424000	0.000000000000	0.00000000
53	0.000628687000	0.000628687000	0.000000000000	0.00000000
54	0.000523906000	0.000523906000	0.000000000000	0.00000000
55	0.000436588000	0.000436588000	0.000000000000	0.00000000
56	0.000363823000	0.000363823000	0.000000000000	0.00000000
57	0.000303186000	0.000303186000	0.000000000000	0.00000000
58	0.000252655000	0.000252655000	0.000000000000	0.00000000
59	0.000210546000	0.000210546000	0.000000000000	0.00000000
60	0.000175455000	0.000175455000	0.000000000000	0.00000000
61	0.000146213000	0.000146213000	0.000000000000	0.00000000
62	0.000121844000	0.000121844000	0.000000000000	0.00000000
63	0.000101536000	0.000101536000	0.000000000000	0.00000000
64	0.000084613700	0.000084613700	0.000000000000	0.00000000
65	0.000070511400	0.000070511400	0.000000000000	0.00000000
66	0.000058759500	0.000058759500	0.000000000000	0.00000000
67	0.000048966300	0.000048966300	0.000000000000	0.00000000
68	0.000040805200	0.000040805200	0.000000000000	0.00000000
69	0.000034004400	0.000034004400	0.000000000000	0.00000000
70	0.000028337000	0.000028337000	0.000000000000	0.00000000
71	0.000023614100	0.000023614100	0.000000000000	0.00000000

Table 4.1: Exponential Arrivals
Comparison with Traditional Methodology

p_n	Exponential - Convolution	Exponential - Traditional	Absolute Difference	Percent Difference
72	0.000019678400	0.000019678400	0.000000000000	0.00000000
73	0.000016398700	0.000016398700	0.000000000000	0.00000000
74	0.000013665600	0.000013665600	0.000000000000	0.00000000
75	0.000011388000	0.000011388000	0.000000000000	0.00000000
76	0.000009489990	0.000009489990	0.000000000000	0.00000000
77	0.000007908330	0.000007908330	0.000000000000	0.00000000
78	0.000006590270	0.000006590270	0.000000000000	0.00000000
79	0.000005491890	0.000005491890	0.000000000000	0.00000000
80	0.000004576580	0.000004576580	0.000000000000	0.00000000
81	0.000003813810	0.000003813810	0.000000000000	0.00000000
82	0.000003178180	0.000003178180	0.000000000000	0.00000000
83	0.000002648480	0.000002648480	0.000000000000	0.00000000
84	0.000002207070	0.000002207070	0.000000000000	0.00000000
85	0.000001839220	0.000001839220	0.000000000000	0.00000000
86	0.000001532690	0.000001532690	0.000000000000	0.00000000
87	0.000001277240	0.000001277240	0.000000000000	0.00000000
88	0.000001064370	0.000001064370	0.000000000000	0.00000000
89	0.000000886971	0.000000886971	0.000000000000	0.00000000
90	0.000000739143	0.000000739143	0.000000000000	0.00000000
91	0.000000615952	0.000000615952	0.000000000000	0.00000000
92	0.000000513294	0.000000513294	0.000000000000	0.00000000
93	0.000000427745	0.000000427745	0.000000000000	0.00000000
94	0.000000356454	0.000000356454	0.000000000000	0.00000000
95	0.000000297045	0.000000297045	0.000000000000	0.00000000
96	0.000000247537	0.000000247537	0.000000000000	0.00000000
97	0.000000206281	0.000000206281	0.000000000000	0.00000000
98	0.000000171901	0.000000171901	0.000000000000	0.00000000
99	0.000000143251	0.000000143251	0.000000000000	0.00000000
100	0.000000119376	0.000000119376	0.000000000000	0.00000000
101	0.000000099480	0.000000099480	0.000000000000	0.00000000
102	0.000000082900	0.000000082900	0.000000000000	0.00000000
103	0.000000069083	0.000000069083	0.000000000000	0.00000000
104	0.000000057569	0.000000057569	0.000000000000	0.00000000
105	0.000000047974	0.000000047974	0.000000000000	0.00000000
106	0.000000039979	0.000000039979	0.000000000000	0.00000000
107	0.000000033316	0.000000033316	0.000000000000	0.00000000

Table 4.1: Exponential Arrivals
Comparison with Traditional Methodology

p_n	Exponential - Convolution	Exponential - Traditional	Absolute Difference	Percent Difference
108	0.000000027763	0.000000027763	0.000000000000	0.00000000
109	0.000000023136	0.000000023136	0.000000000000	0.00000000
110	0.000000019280	0.000000019280	0.000000000000	0.00000000
111	0.000000016067	0.000000016067	0.000000000000	0.00000000
112	0.000000013389	0.000000013389	0.000000000000	0.00000000
113	0.000000011157	0.000000011157	0.000000000000	0.00000000
114	0.000000009298	0.000000009298	0.000000000000	0.00000000
115	0.000000007748	0.000000007748	0.000000000000	0.00000000
116	0.000000006457	0.000000006457	0.000000000000	0.00000000
117	0.000000005381	0.000000005381	0.000000000000	0.00000000
118	0.000000004484	0.000000004484	0.000000000000	0.00000000
119	0.000000003737	0.000000003737	0.000000000000	0.00000000
120	0.000000003114	0.000000003114	0.000000000000	0.00000000
121	0.000000002595	0.000000002595	0.000000000000	0.00000000
122	0.000000002162	0.000000002162	0.000000000000	0.00000000
123	0.000000001802	0.000000001802	0.000000000000	0.00000000
124	0.000000001502	0.000000001502	0.000000000000	0.00000000
125	0.000000001251	0.000000001251	0.000000000000	0.00000000
126	0.000000001043	0.000000001043	0.000000000000	0.00000000
127	0.000000000869	0.000000000869	0.000000000000	0.00000000
128	0.000000000724	0.000000000724	0.000000000000	0.00000000
129	0.000000000603	0.000000000603	0.000000000000	0.00000000
130	0.000000000503	0.000000000503	0.000000000000	0.00000000
131	0.000000000419	0.000000000419	0.000000000000	0.00000000
132	0.000000000349	0.000000000349	0.000000000000	0.00000000
133	0.000000000291	0.000000000291	0.000000000000	0.00000000
134	0.000000000243	0.000000000243	0.000000000000	0.00000000
135	0.000000000202	0.000000000202	0.000000000000	0.00000000
136	0.000000000168	0.000000000168	0.000000000000	0.00000000
137	0.000000000140	0.000000000140	0.000000000000	0.00000000
138	0.000000000117	0.000000000117	0.000000000000	0.00000000
139	0.000000000097	0.000000000097	0.000000000000	0.00000000
140	0.000000000081	0.000000000081	0.000000000000	0.00000000
141	0.000000000068	0.000000000068	0.000000000000	0.00000000
142	0.000000000056	0.000000000056	0.000000000000	0.00000000
143	0.000000000047	0.000000000047	0.000000000000	0.00000000

Table 4.1: Exponential Arrivals
Comparison with Traditional Methodology

p_n	Exponential - Convolution	Exponential - Traditional	Absolute Difference	Percent Difference
144	0.000000000039	0.000000000039	0.000000000000	0.00000000
145	0.000000000033	0.000000000033	0.000000000000	0.00000000
146	0.000000000027	0.000000000027	0.000000000000	0.00000000
147	0.000000000023	0.000000000023	0.000000000000	0.00000000
148	0.000000000019	0.000000000019	0.000000000000	0.00000000
149	0.000000000016	0.000000000016	0.000000000000	0.00000000
150	0.000000000013	0.000000000013	0.000000000000	0.00000000
151	0.000000000011	0.000000000011	0.000000000000	0.00000000
152	0.000000000009	0.000000000009	0.000000000000	0.00000000
153	0.000000000008	0.000000000008	0.000000000000	0.00000000
154	0.000000000006	0.000000000006	0.000000000000	0.00000000
155	0.000000000005	0.000000000005	0.000000000000	0.00000000
156	0.000000000004	0.000000000004	0.000000000000	0.00000000
157	0.000000000004	0.000000000004	0.000000000000	0.00000000
158	0.000000000003	0.000000000003	0.000000000000	0.00000000
159	0.000000000003	0.000000000003	0.000000000000	0.00000000
160	0.000000000002	0.000000000002	0.000000000000	0.00000000
161	0.000000000002	0.000000000002	0.000000000000	0.00000000
162	0.000000000001	0.000000000001	0.000000000000	0.00000000
163	0.000000000001	0.000000000001	0.000000000000	0.00000000
164	0.000000000001	0.000000000001	0.000000000000	0.00000000
165	0.000000000001	0.000000000001	0.000000000000	0.00000000
166	0.000000000001	0.000000000001	0.000000000000	0.00000000
167	0.000000000001	0.000000000001	0.000000000000	0.00000000
168	0.000000000000	0.000000000000	0.000000000000	0.00000000
169	0.000000000000	0.000000000000	0.000000000000	0.00000000
170	0.000000000000	0.000000000000	0.000000000000	0.00000000
171	0.000000000000	0.000000000000	0.000000000000	0.00000000
172	0.000000000000	0.000000000000	0.000000000000	0.00000000
173	0.000000000000	0.000000000000	0.000000000000	0.00000000
174	0.000000000000	0.000000000000	0.000000000000	0.00000000
175	0.000000000000	0.000000000000	0.000000000000	0.00000000
176	0.000000000000	0.000000000000	0.000000000000	0.00000000
177	0.000000000000	0.000000000000	0.000000000000	0.00000000
178	0.000000000000	0.000000000000	0.000000000000	0.00000000
179	0.000000000000	0.000000000000	0.000000000000	0.00000000

Table 4.1: Exponential Arrivals
Comparison with Traditional Methodology

p_n	Exponential - Convolution	Exponential - Traditional	Absolute Difference	Percent Difference
180	0.000000000000	0.000000000000	0.000000000000	0.00000000
181	0.000000000000	0.000000000000	0.000000000000	0.00000000
182	0.000000000000	0.000000000000	0.000000000000	0.00000000
183	0.000000000000	0.000000000000	0.000000000000	0.00000000
184	0.000000000000	0.000000000000	0.000000000000	0.00000000
185	0.000000000000	0.000000000000	0.000000000000	0.00000000
186	0.000000000000	0.000000000000	0.000000000000	0.00000000
187	0.000000000000	0.000000000000	0.000000000000	0.00000000
188	0.000000000000	0.000000000000	0.000000000000	0.00000000
189	0.000000000000	0.000000000000	0.000000000000	0.00000000
190	0.000000000000	0.000000000000	0.000000000000	0.00000000
191	0.000000000000	0.000000000000	0.000000000000	0.00000000
192	0.000000000000	0.000000000000	0.000000000000	0.00000000
193	0.000000000000	0.000000000000	0.000000000000	0.00000000
194	0.000000000000	0.000000000000	0.000000000000	0.00000000
195	0.000000000000	0.000000000000	0.000000000000	0.00000000
196	0.000000000000	0.000000000000	0.000000000000	0.00000000
197	0.000000000000	0.000000000000	0.000000000000	0.00000000
198	0.000000000000	0.000000000000	0.000000000000	0.00000000
199	0.000000000000	0.000000000000	0.000000000000	0.00000000
200	0.000000000000	0.000000000000	0.000000000000	0.00000000
201	0.000000000000	0.000000000000	0.000000000000	0.00000000
202	0.000000000000	0.000000000000	0.000000000000	0.00000000
203	0.000000000000	0.000000000000	0.000000000000	0.00000000
204	0.000000000000	0.000000000000	0.000000000000	0.00000000
205	0.000000000000	0.000000000000	0.000000000000	0.00000000
206	0.000000000000	0.000000000000	0.000000000000	0.00000000
207	0.000000000000	0.000000000000	0.000000000000	0.00000000
208	0.000000000000	0.000000000000	0.000000000000	0.00000000
209	0.000000000000	0.000000000000	0.000000000000	0.00000000
210	0.000000000000	0.000000000000	0.000000000000	0.00000000
211	0.000000000000	0.000000000000	0.000000000000	0.00000000
212	0.000000000000	0.000000000000	0.000000000000	0.00000000

Table 4.2: Deterministic and Erlang Arrivals
Comparison with Takacs

p_n	Deterministic - Convolution	Deterministic - Takacs	Erlang - Convolution	Erlang - Takacs
0	0.000000000000	1.594207244000	0.000000000000	−0.143446368000
1	0.000000000000	−15.523542510000	0.000000000003	0.969606785000
2	0.000000000000	67.636154120000	0.000000000054	−2.780039134000
3	0.000000000001	−173.335280700000	0.000000000701	4.359607454000
4	0.000000000011	288.625530500000	0.000000006590	−4.002432846000
5	0.000000000160	−325.018011000000	0.000000048100	2.111086423000
6	0.000000001770	249.059210800000	0.000000285000	−0.562620134000
7	0.000000015700	−126.738921700000	0.000001410000	0.044784353000
8	0.000000114000	39.970060980000	0.000005950000	0.002796984000
9	0.000000688000	−6.572370779000	0.000021800000	0.000428132000
10	0.000003540000	0.284029267000	0.000070600000	0.000330347000
11	0.000015600000	0.016023180000	0.000203263000	0.000145629000
12	0.000060000000	0.002277234000	0.000525700000	0.000641381000
13	0.000201914000	0.000674363000	0.001231254000	0.001164160000
14	0.000599788000	0.000735204000	0.002628879000	0.002670982000
15	0.001583662000	0.001631801000	0.005146435000	0.005124463000
16	0.003738255000	0.003755942000	0.009284190000	0.009294957000
17	0.007929177000	0.007933956000	0.015503024000	0.015501019000
18	0.015181195000	0.015183787000	0.024057155000	0.024057930000
19	0.026342713000	0.026344305000	0.034815226000	0.034815846000
20	0.041578938000	0.041579756000	0.047139080000	0.047139223000
21	0.059892855000	0.059893371000	0.059887964000	0.059888152000
22	0.078970515000	0.078970818000	0.071580022000	0.071580121000
23	0.095571647000	0.095571852000	0.080684009000	0.080684093000
24	0.106427403000	0.106427541000	0.085958828000	0.085958888000
25	0.109304115000	0.109304209000	0.086734062000	0.086734106000
26	0.103752550000	0.103752612000	0.083043883000	0.083043913000
27	0.091199252000	0.091199292000	0.075580435000	0.075580454000
28	0.074371052000	0.074371075000	0.065495940000	0.065495950000
29	0.056365554000	0.056365566000	0.054128663000	0.054128668000
30	0.039811267000	0.039811273000	0.042742240000	0.042742241000
31	0.027322539000	0.027322541000	0.033434235000	0.033434234000
32	0.018751504000	0.018751504000	0.026153240000	0.026153237000
33	0.012869189000	0.012869187000	0.020457832000	0.020457829000
34	0.008832146000	0.008832144000	0.016002717000	0.016002713000
35	0.006061516000	0.006061515000	0.012517795000	0.012517791000

Table 4.2: Deterministic and Erlang Arrivals
Comparison with Takacs

p_n	Deterministic - Convolution	Deterministic - Takacs	Erlang - Convolution	Erlang - Takacs
36	0.004160029000	0.004160027000	0.009791787000	0.009791783000
37	0.002855035000	0.002855034000	0.007659423000	0.007659420000
38	0.001959415000	0.001959414000	0.005991426000	0.005991423000
39	0.001344750000	0.001344749000	0.004686669000	0.004686666000
40	0.000922904000	0.000922903000	0.003666050000	0.003666048000
41	0.000633391000	0.000633390000	0.002867692000	0.002867690000
42	0.000434697000	0.000434697000	0.002243193000	0.002243191000
43	0.000298333000	0.000298333000	0.001754691000	0.001754690000
44	0.000204747000	0.000204746000	0.001372571000	0.001372570000
45	0.000140518000	0.000140518000	0.001073665000	0.001073664000
46	0.000096400000	0.000096400000	0.000839853000	0.000839852000
47	0.000066200000	0.000066200000	0.000656957000	0.000656957000
48	0.000045400000	0.000045400000	0.000513891000	0.000513891000
49	0.000031200000	0.000031200000	0.000401981000	0.000401980000
50	0.000021400000	0.000021400000	0.000314441000	0.000314441000
51	0.000014700000	0.000014700000	0.000245965000	0.000245965000
52	0.000010100000	0.000010100000	0.000192401000	0.000192401000
53	0.000006920000	0.000006920000	0.000150502000	0.000150502000
54	0.000004750000	0.000004750000	0.000117727000	0.000117727000
55	0.000003260000	0.000003260000	0.000092100000	0.000092100000
56	0.000002240000	0.000002240000	0.000072000000	0.000072000000
57	0.000001530000	0.000001530000	0.000056300000	0.000056300000
58	0.000001050000	0.000001050000	0.000044100000	0.000044100000
59	0.000000723000	0.000000723000	0.000034500000	0.000034500000
60	0.000000496000	0.000000496000	0.000027000000	0.000027000000
61	0.000000340000	0.000000340000	0.000021100000	0.000021100000
62	0.000000234000	0.000000234000	0.000016500000	0.000016500000
63	0.000000160000	0.000000160000	0.000012900000	0.000012900000
64	0.000000110000	0.000000110000	0.000010100000	0.000010100000
65	0.000000075500	0.000000075500	0.000007900000	0.000007900000
66	0.000000051800	0.000000051800	0.000006180000	0.000006180000
67	0.000000035600	0.000000035600	0.000004830000	0.000004830000
68	0.000000024400	0.000000024400	0.000003780000	0.000003780000
69	0.000000016800	0.000000016800	0.000002960000	0.000002960000
70	0.000000011500	0.000000011500	0.000002310000	0.000002310000
71	0.000000007890	0.000000007890	0.000001810000	0.000001810000

Table 4.2: Deterministic and Erlang Arrivals
Comparison with Takacs

p_n	Deterministic - Convolution	Deterministic - Takacs	Erlang - Convolution	Erlang - Takacs
72	0.000000005420	0.000000005420	0.000001420000	0.000001420000
73	0.000000003720	0.000000003720	0.000001110000	0.000001110000
74	0.000000002550	0.000000002550	0.000000866000	0.000000866000
75	0.000000001750	0.000000001750	0.000000677000	0.000000677000
76	0.000000001200	0.000000001200	0.000000530000	0.000000530000
77	0.000000000825	0.000000000825	0.000000415000	0.000000415000
78	0.000000000566	0.000000000566	0.000000324000	0.000000324000
79	0.000000000388	0.000000000388	0.000000254000	0.000000254000
80	0.000000000267	0.000000000267	0.000000198000	0.000000198000
81	0.000000000183	0.000000000183	0.000000155000	0.000000155000
82	0.000000000126	0.000000000126	0.000000121000	0.000000121000
83	0.000000000086	0.000000000086	0.000000095000	0.000000095000
84	0.000000000059	0.000000000059	0.000000074300	0.000000074300
85	0.000000000041	0.000000000041	0.000000058100	0.000000058100
86	0.000000000028	0.000000000028	0.000000045500	0.000000045500
87	0.000000000019	0.000000000019	0.000000035600	0.000000035600
88	0.000000000013	0.000000000013	0.000000027800	0.000000027800
89	0.000000000009	0.000000000009	0.000000021800	0.000000021800
90	0.000000000006	0.000000000006	0.000000017000	0.000000017000
91	0.000000000004	0.000000000004	0.000000013300	0.000000013300
92	0.000000000003	0.000000000003	0.000000010400	0.000000010400
93	0.000000000002	0.000000000002	0.000000008150	0.000000008150
94	0.000000000001	0.000000000001	0.000000006370	0.000000006370
95	0.000000000001	0.000000000001	0.000000004980	0.000000004980
96	0.000000000001	0.000000000001	0.000000003900	0.000000003900
97	0.000000000000	0.000000000000	0.000000003050	0.000000003050
98	0.000000000000	0.000000000000	0.000000002390	0.000000002390
99	0.000000000000	0.000000000000	0.000000001870	0.000000001870
100	0.000000000000	0.000000000000	0.000000001460	0.000000001460
101	0.000000000000	0.000000000000	0.000000001140	0.000000001140
102	0.000000000000	0.000000000000	0.000000000893	0.000000000893
103	0.000000000000	0.000000000000	0.000000000699	0.000000000699
104	0.000000000000	0.000000000000	0.000000000546	0.000000000546
105	0.000000000000	0.000000000000	0.000000000427	0.000000000427
106	0.000000000000	0.000000000000	0.000000000334	0.000000000334
107	0.000000000000	0.000000000000	0.000000000262	0.000000000262

Table 4.2: Deterministic and Erlang Arrivals
Comparison with Takacs

p_n	Deterministic - Convolution	Deterministic - Takacs	Erlang - Convolution	Erlang - Takacs
108	0.000000000000	0.000000000000	0.000000000205	0.000000000205
109	0.000000000000	0.000000000000	0.000000000160	0.000000000160
110	0.000000000000	0.000000000000	0.000000000125	0.000000000125
111	0.000000000000	0.000000000000	0.000000000098	0.000000000098
112	0.000000000000	0.000000000000	0.000000000077	0.000000000077
113	0.000000000000	0.000000000000	0.000000000060	0.000000000060
114	0.000000000000	0.000000000000	0.000000000047	0.000000000047
115	0.000000000000	0.000000000000	0.000000000037	0.000000000037
116	0.000000000000	0.000000000000	0.000000000029	0.000000000029
117	0.000000000000	0.000000000000	0.000000000022	0.000000000022
118	0.000000000000	0.000000000000	0.000000000018	0.000000000018
119	0.000000000000	0.000000000000	0.000000000014	0.000000000014
120	0.000000000000	0.000000000000	0.000000000011	0.000000000011
121	0.000000000000	0.000000000000	0.000000000008	0.000000000008
122			0.000000000007	0.000000000007
123			0.000000000005	0.000000000005
124			0.000000000004	0.000000000004
125			0.000000000003	0.000000000003
126			0.000000000002	0.000000000002
127			0.000000000002	0.000000000002
128			0.000000000002	0.000000000002
129			0.000000000001	0.000000000001
130			0.000000000001	0.000000000001
131			0.000000000001	0.000000000001
132			0.000000000001	0.000000000001
133			0.000000000000	0.000000000000
134			0.000000000000	0.000000000000
135			0.000000000000	0.000000000000
136			0.000000000000	0.000000000000
137			0.000000000000	0.000000000000
138			0.000000000000	0.000000000000
139			0.000000000000	0.000000000000
140			0.000000000000	0.000000000000
141			0.000000000000	0.000000000000
142			0.000000000000	0.000000000000
143			0.000000000000	0.000000000000

Table 4.2: Deterministic and Erlang Arrivals
Comparison with Takacs

p_n	Deterministic - Convolution	Deterministic - Takacs	Erlang - Convolution	Erlang - Takacs
144			0.000000000000	0.000000000000
145			0.000000000000	0.000000000000
146			0.000000000000	0.000000000000
147			0.000000000000	0.000000000000
148			0.000000000000	0.000000000000
149			0.000000000000	0.000000000000
150			0.000000000000	0.000000000000
151			0.000000000000	0.000000000000
152			0.000000000000	0.000000000000
153			0.000000000000	0.000000000000
154			0.000000000000	0.000000000000
155			0.000000000000	0.000000000000
156			0.000000000000	0.000000000000
157			0.000000000000	0.000000000000
158			0.000000000000	0.000000000000
159			0.000000000000	0.000000000000
160			0.000000000000	0.000000000000
161			0.000000000000	0.000000000000
162			0.000000000000	0.000000000000
163			0.000000000000	0.000000000000
164			0.000000000000	0.000000000000
165			0.000000000000	0.000000000000
166			0.000000000000	0.000000000000
167			0.000000000000	0.000000000000

Table 4.3: Hyperexponential Arrivals
Comparison with Takacs

p_n	Hyperexponential - Convolution	Hyperexponential - Takacs	p_n	Hyperexponential - Convolution	Hyperexponential - Takacs
0	0.000000036700	0.002134719000	1	0.000000378000	-0.008008966000
2	0.000002100000	0.010040374000	3	0.000008330000	-0.004912430000
4	0.000026400000	0.000543656000	5	0.000071300000	0.000193266000
6	0.000169361000	0.000076900000	7	0.000362914000	0.000599255000
8	0.000713719000	0.000445471000	9	0.001304399000	0.001693183000

Table 4.3: Hyperexponential Arrivals
Comparison with Takacs

p_n	Hyperexponential - Convolution	Hyperexponential - Takacs	p_n	Hyperexponential - Convolution	Hyperexponential - Takacs
10	0.002236414000	0.001900990000	11	0.003623442000	0.003977845000
12	0.005579737000	0.005342266000	13	0.008203995000	0.008388662000
14	0.011560400000	0.011476424000	15	0.015659567000	0.015702208000
16	0.020442809000	0.020426235000	17	0.025773216000	0.025781691000
18	0.031436395000	0.031433744000	19	0.037152301000	0.037153013000
20	0.042597740000	0.042597651000	21	0.047437082000	0.047437196000
22	0.051357039000	0.051357086000	23	0.054100366000	0.054100419000
24	0.055493343000	0.055493386000	25	0.055462841000	0.055462878000
26	0.054040539000	0.054040569000	27	0.051353989000	0.051354013000
28	0.047606260000	0.047606279000	29	0.043047462000	0.043047476000
30	0.037942180000	0.037942190000	31	0.033621314000	0.033621321000
32	0.029792510000	0.029792514000	33	0.026399731000	0.026399733000
34	0.023393323000	0.023393323000	35	0.020729285000	0.020729284000
36	0.018368629000	0.018368627000	37	0.016276804000	0.016276802000
38	0.014423198000	0.014423195000	39	0.012780680000	0.012780677000
40	0.011325213000	0.011325209000	41	0.010035495000	0.010035491000
42	0.008892650000	0.008892646000	43	0.007879953000	0.007879949000
44	0.006982582000	0.006982577000	45	0.006187403000	0.006187399000
46	0.005482780000	0.005482776000	47	0.004858400000	0.004858396000
48	0.004305124000	0.004305120000	49	0.003814855000	0.003814852000
50	0.003380419000	0.003380415000	51	0.002995456000	0.002995453000
52	0.002654332000	0.002654330000	53	0.002352056000	0.002352054000
54	0.002084204000	0.002084201000	55	0.001846854000	0.001846852000
56	0.001636534000	0.001636532000	57	0.001450165000	0.001450163000
58	0.001285020000	0.001285018000	59	0.001138681000	0.001138680000
60	0.001009008000	0.001009006000	61	0.000894102000	0.000894100000
62	0.000792281000	0.000792280000	63	0.000702056000	0.000702055000
64	0.000622106000	0.000622105000	65	0.000551260000	0.000551259000
66	0.000488483000	0.000488482000	67	0.000432854000	0.000432853000
68	0.000383560000	0.000383560000	69	0.000339881000	0.000339880000
70	0.000301175000	0.000301174000	71	0.000266877000	0.000266876000
72	0.000236485000	0.000236484000	73	0.000209554000	0.000209553000
74	0.000185690000	0.000185689000	75	0.000164544000	0.000164543000
76	0.000145805000	0.000145805000	77	0.000129201000	0.000129201000
78	0.000114487000	0.000114487000	79	0.000101450000	0.000101449000
80	0.000089900000	0.000089900000	81	0.000079700000	0.000079700000
82	0.000070600000	0.000070600000	83	0.000062500000	0.000062500000
84	0.000055400000	0.000055400000	85	0.000049100000	0.000049100000

Table 4.3: Hyperexponential Arrivals
Comparison with Takacs

p_n	Hyperexponential - Convolution	Hyperexponential - Takacs	p_n	Hyperexponential - Convolution	Hyperexponential - Takacs
86	0.000043500000	0.000043500000	87	0.000038600000	0.000038600000
88	0.000034200000	0.000034200000	89	0.000030300000	0.000030300000
90	0.000026800000	0.000026800000	91	0.000023800000	0.000023800000
92	0.000021100000	0.000021100000	93	0.000018700000	0.000018700000
94	0.000016500000	0.000016500000	95	0.000014700000	0.000014700000
96	0.000013000000	0.000013000000	97	0.000011500000	0.000011500000
98	0.000010200000	0.000010200000	99	0.000009040000	0.000009040000
100	0.000008010000	0.000008010000	101	0.000007100000	0.000007100000
102	0.000006290000	0.000006290000	103	0.000005570000	0.000005570000
104	0.000004940000	0.000004940000	105	0.000004380000	0.000004380000
106	0.000003880000	0.000003880000	107	0.000003440000	0.000003440000
108	0.000003040000	0.000003040000	109	0.000002700000	0.000002700000
110	0.000002390000	0.000002390000	111	0.000002120000	0.000002120000
112	0.000001880000	0.000001880000	113	0.000001660000	0.000001660000
114	0.000001470000	0.000001470000	115	0.000001310000	0.000001310000
116	0.000001160000	0.000001160000	117	0.000001030000	0.000001030000
118	0.000000909000	0.000000909000	119	0.000000805000	0.000000805000
120	0.000000714000	0.000000714000	121	0.000000632000	0.000000632000
122	0.000000560000	0.000000560000	123	0.000000496000	0.000000496000
124	0.000000440000	0.000000440000	125	0.000000390000	0.000000390000
126	0.000000345000	0.000000345000	127	0.000000306000	0.000000306000
128	0.000000271000	0.000000271000	129	0.000000240000	0.000000240000
130	0.000000213000	0.000000213000	131	0.000000189000	0.000000189000
132	0.000000167000	0.000000167000	133	0.000000148000	0.000000148000
134	0.000000131000	0.000000131000	135	0.000000116000	0.000000116000
136	0.000000103000	0.000000103000	137	0.000000091400	0.000000091400
138	0.000000081000	0.000000081000	139	0.000000071700	0.000000071700
140	0.000000063600	0.000000063600	141	0.000000056300	0.000000056300
142	0.000000049900	0.000000049900	143	0.000000044200	0.000000044200
144	0.000000039200	0.000000039200	145	0.000000034700	0.000000034700
146	0.000000030800	0.000000030800	147	0.000000027300	0.000000027300
148	0.000000024200	0.000000024200	149	0.000000021400	0.000000021400
150	0.000000019000	0.000000019000	151	0.000000016800	0.000000016800
152	0.000000014900	0.000000014900	153	0.000000013200	0.000000013200
154	0.000000011700	0.000000011700	155	0.000000010400	0.000000010400
156	0.000000009190	0.000000009190	157	0.000000008140	0.000000008140
158	0.000000007210	0.000000007210	159	0.000000006390	0.000000006390
160	0.000000005660	0.000000005660	161	0.000000005020	0.000000005020

Table 4.3: Hyperexponential Arrivals
Comparison with Takacs

p_n	Hyperexponential - Convolution	Hyperexponential - Takacs	p_n	Hyperexponential - Convolution	Hyperexponential - Takacs
162	0.000000004450	0.000000004450	163	0.000000003940	0.000000003940
164	0.000000003490	0.000000003490	165	0.000000003090	0.000000003090
166	0.000000002740	0.000000002740	167	0.000000002430	0.000000002430
168	0.000000002150	0.000000002150	169	0.000000001910	0.000000001910
170	0.000000001690	0.000000001690	171	0.000000001500	0.000000001500
172	0.000000001330	0.000000001330	173	0.000000001180	0.000000001180
174	0.000000001040	0.000000001040	175	0.000000000924	0.000000000924
176	0.000000000818	0.000000000818	177	0.000000000725	0.000000000725
178	0.000000000643	0.000000000643	179	0.000000000569	0.000000000569
180	0.000000000505	0.000000000505	181	0.000000000447	0.000000000447
182	0.000000000396	0.000000000396	183	0.000000000351	0.000000000351
184	0.000000000311	0.000000000311	185	0.000000000276	0.000000000276
186	0.000000000244	0.000000000244	187	0.000000000216	0.000000000216
188	0.000000000192	0.000000000192	189	0.000000000170	0.000000000170
190	0.000000000151	0.000000000151	191	0.000000000133	0.000000000133
192	0.000000000118	0.000000000118	193	0.000000000105	0.000000000105
194	0.000000000093	0.000000000093	195	0.000000000082	0.000000000082
196	0.000000000073	0.000000000073	197	0.000000000065	0.000000000065
198	0.000000000057	0.000000000057	199	0.000000000051	0.000000000051
200	0.000000000045	0.000000000045	201	0.000000000040	0.000000000040
202	0.000000000035	0.000000000035	203	0.000000000031	0.000000000031
204	0.000000000028	0.000000000028	205	0.000000000025	0.000000000025
206	0.000000000022	0.000000000022	207	0.000000000019	0.000000000019
208	0.000000000017	0.000000000017	209	0.000000000015	0.000000000015
210	0.000000000013	0.000000000013	211	0.000000000012	0.000000000012
212	0.000000000011	0.000000000011	213	0.000000000009	0.000000000009
214	0.000000000008	0.000000000008	215	0.000000000007	0.000000000007
216	0.000000000007	0.000000000007	217	0.000000000006	0.000000000006
218	0.000000000005	0.000000000005	219	0.000000000005	0.000000000005
220	0.000000000004	0.000000000004	221	0.000000000004	0.000000000004
222	0.000000000003	0.000000000003	223	0.000000000003	0.000000000003
224	0.000000000002	0.000000000002	225	0.000000000002	0.000000000002
226	0.000000000002	0.000000000002	227	0.000000000002	0.000000000002
228	0.000000000002	0.000000000002	229	0.000000000001	0.000000000001
230	0.000000000001	0.000000000001	231	0.000000000001	0.000000000001
232	0.000000000001	0.000000000001	233	0.000000000001	0.000000000001
234	0.000000000001	0.000000000001	235	0.000000000001	0.000000000001
236	0.000000000001	0.000000000001	237	0.000000000001	0.000000000001

Table 4.3: Hyperexponential Arrivals
Comparison with Takacs

p_n	Hyperexponential - Convolution	Hyperexponential - Takacs	p_n	Hyperexponential - Convolution	Hyperexponential - Takacs
238	0.000000000000	0.000000000000	239	0.000000000000	0.000000000000
240	0.000000000000	0.000000000000	241	0.000000000000	0.000000000000
242	0.000000000000	0.000000000000	243	0.000000000000	0.000000000000
244	0.000000000000	0.000000000000	245	0.000000000000	0.000000000000
246	0.000000000000	0.000000000000	247	0.000000000000	0.000000000000
248	0.000000000000	0.000000000000	249	0.000000000000	0.000000000000
250	0.000000000000	0.000000000000	251	0.000000000000	0.000000000000
252	0.000000000000	0.000000000000	253	0.000000000000	0.000000000000
254	0.000000000000	0.000000000000	255	0.000000000000	0.000000000000
256	0.000000000000	0.000000000000	257	0.000000000000	0.000000000000
258	0.000000000000	0.000000000000	259	0.000000000000	0.000000000000
260	0.000000000000	0.000000000000	261	0.000000000000	0.000000000000
262	0.000000000000	0.000000000000	263	0.000000000000	0.000000000000
264	0.000000000000	0.000000000000	265	0.000000000000	0.000000000000
266	0.000000000000	0.000000000000	267	0.000000000000	0.000000000000
268	0.000000000000	0.000000000000	269	0.000000000000	0.000000000000
270	0.000000000000	0.000000000000	271	0.000000000000	0.000000000000
272	0.000000000000	0.000000000000	273	0.000000000000	0.000000000000
274	0.000000000000	0.000000000000	275	0.000000000000	0.000000000000
276	0.000000000000	0.000000000000	277	0.000000000000	0.000000000000
278	0.000000000000	0.000000000000	279	0.000000000000	0.000000000000
280	0.000000000000	0.000000000000	281	0.000000000000	0.000000000000
282	0.000000000000	0.000000000000	283	0.000000000000	0.000000000000
284	0.000000000000	0.000000000000	285	0.000000000000	0.000000000000
286	0.000000000000	0.000000000000	287	0.000000000000	0.000000000000
288	0.000000000000	0.000000000000	289	0.000000000000	0.000000000000
290	0.000000000000	0.000000000000	291	0.000000000000	0.000000000000
292	0.000000000000	0.000000000000	293	0.000000000000	0.000000000000
294	0.000000000000	0.000000000000	295	0.000000000000	0.000000000000
296	0.000000000000	0.000000000000	297	0.000000000000	0.000000000000
298	0.000000000000	0.000000000000			

Table 4.4: Performance Measures

Distribution and Method	**L**	**W**
Deterministic - Convolution	25.2776493	5.0555299
Deterministic - Takacs	25.2776493	5.0555299
Erlang - Convolution	25.7050059	5.1410012
Erlang - Takacs	25.7050055	5.1410011
Exponential - Convolution	26.2494658	5.2498932
Exponential - Traditional	26.2494658	5.2498932
Hyperexponential - Convolution	27.5924940	5.5184988
Hyperexponential - Takacs	27.5924921	5.5184984

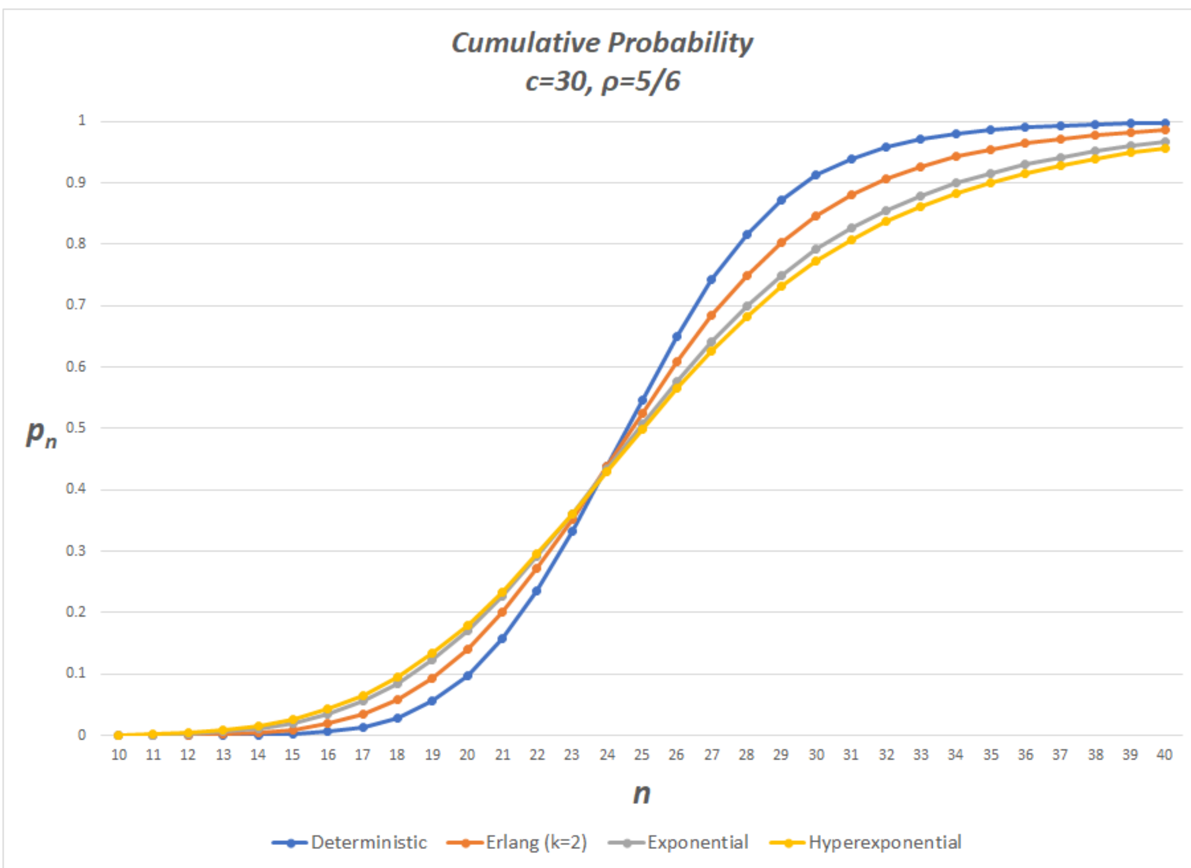

Figure 4.1: Cumulative Probability for $c = 30, \rho = 5/6$

Chapter 5

Future Extensions and Improvements

1 Introduction

The items in this section fall into two broad categories: theoretical improvements or extensions, and programmatical improvements or extentions. This list should not be considered comprehensive.

2 Theoretical Improvements

This model could theoretically be extended to any interarrival distribution with support over the semi-infinite interval $[0, \infty)$, for example the Gamma distribution, simply by formulating its Laplace-Stieltjes transform and the derivatives thereof. Given that the list of such distributions is extensive, this extention of the model would be best suited to distributions of greatest practical interest.

In terms of improving the model, there may be opportunities within particular interarrival distributions to optimize the computational method. One such improvement has already been implemented, namely, that any distributions which contain $n!$ in the A_n^* function (that is, the Erlang, exponential, hyperexponential distributions) have that factor cancelled by the $n!$ in the denominator of the summation, and thus both were removed from computation in the software. On the other hand, the deterministic distribution has no such factor in its A_n^* function and must therefore compute the $n!$ in the denominator of the summation, which creates underflow issues with large n.

Lastly, it remains to be proven that the convolution method is equivalent to direct integration of Region 3. We see many similarities, particularly in the summations that use the Laplace-Stieltjes transform and its derivatives, and numerical equivalence was demonstrated by comparison with models having the exponential distribution for both interarrival and service (which can be computed with much simpler formulas), but no theoretical proof of equivalence has yet been completed.

3 Programmatical Improvements

In terms of the accuracy and precision of the results, the major difficulties are the result of factorials, which in some cases are the denominator of a fraction. This causes underflow and overflow in the summations which can result in inaccurate probabilities (often negative or greater than one) for some of the states. Our primary corrective for this was the use of a fixed-decimal package for our computations. However in the summations of our method, when alternating positive and negative terms exist, underflow issues could be reduced by summing all negative terms separately from the sum of all positive terms before combining. This would reduce the likelyhood of positive and negative terms canceling each other in a manner that causes underflow.

Regarding model extensions, both the Erlang and the hyperexponential distributions could be extended beyond $k = 2$ with relatively little difficulty. This could potentially allow better modeling of real-world applications, where more than two phases may be present.

The addition of simulations for the distributions modeled by the software would be a useful extension for comparing the theoretical (infinite interval) results with shorter-run trials. While some such simulators already exist, they appear to be focused on the performance metrics (average number in system and average time in system) rather than providing the state probabilities.

REFERENCES

[1] Jacob Willem Cohen. *The Single Server Queue, 2nd ed.* Elsevier Science Publishers B.V., Amsterdam, 1992.

[2] R.B. Cooper. *Introduction to Queueing Theory*. North Holland, New York, 2nd edition, 1981.

[3] M. El-Taha. The G/M/c Model Revisited: An Efficient Convolution Method to Compute Transition Probabilities. Preprint, 2019.

[4] W. Feller. *An introduction to probability theory and its applications, Volume 1,* $3^r d$ *edition*. John Wiley and Sons, New York, 1968.

[5] D. Gross, J.F. Shortle, J.M. Thompson, and C. Harris. *Fundamentals of Queueing Theory*. John Wiley, New Jersey, 4th edition, 2008.

[6] Oliver C. Ibe. *Fundamentals of Stochastic Networks*. John Wiley & Sons, Inc., New Jersey, 2011.

[7] D. G. Kendall. Stochastic processes occurring in the theory of queues and their analysis by means of the imbedded Markov chain. *The Annals of Mathematical Statistics*, 24:338–354, 1953.

[8] L. Kleinrock. *Queueing Systems vol. I.* Wiley Intersciences, New York, 1975.

[9] J. Medhi. *Stochastic Models in Queueing Theory*. Academic Press, New York, 2nd edition, 2003.

[10] L. Takacs. *Introduction to the theory of queues.* Oxford University Press, New York, 1962.

Appendices

Appendix A

The Laplace-Stieltjes Transforms

1 Definition

Given a function $A(t)$ with domain $[0, \infty)$, we define the Laplace-Stieltjes transform (LST) $A^*(s)$ as:

$$A^*(s) = \int_0^\infty e^{-st} dA(t)$$

2 The n^{th} Derivative of the Laplace-Stieltjes Transform

Given the LST of the function $A(t)$ defined as

$$\int_0^\infty e^{-st} dA(t)$$

we obtain the first derivative with respect to s as follows:

$$\frac{d}{ds}\int_0^\infty e^{-st}dA(t) = \int_0^\infty \frac{d}{ds}\left[e^{-st}\right]dA(t)$$

$$= \int_0^\infty -te^{-st}dA(t)$$

$$= (-1)\int_0^\infty te^{-st}dA(t)$$

The second derivative is obtained similarly:

$$\frac{d}{ds}\int_0^\infty -te^{-st}dA(t) = \int_0^\infty -t\frac{d}{ds}\left[e^{-st}\right]dA(t)$$

$$= \int_0^\infty -t(-te^{-st})dA(t)$$

$$= \int_0^\infty t^2e^{-st}dA(t)$$

$$= (-1)^2\int_0^\infty t^2e^{-st}dA(t)$$

Thus we see that the n^{th} derivative of the LST of $A(t)$ is therefore:

$$\frac{d^n}{ds^n}\int_0^\infty e^{-st}dA(t) = (-1)^n\int_0^\infty t^ne^{-st}dA(t)$$

So that we may define

$$A_n^*(s) = (-1)^n\frac{d^nA^*(s)}{ds^n} = \int_0^\infty t^ne^{-st}dA(t)$$

3 Laplace-Stieltjes Transforms For Common Arrival Distributions

3.1 Deterministic Distributions

For determinisic distributions with mean λ and $a(t) = \frac{1}{\lambda}$ w.p. 1, we have

$$\begin{aligned}
A^*(s) &= \int_0^\infty e^{-st} dA(t) \\
&= \int_0^\infty e^{-st} \left(\frac{1}{\lambda}\right) dt \\
&= \frac{1}{\lambda} \int_0^\infty e^{-st} dt \\
&= \frac{1}{\lambda} \int_0^\infty e^{-st} dt \\
&= \frac{1}{\lambda} \lambda e^{-s/\lambda} \\
&= e^{-s/\lambda}
\end{aligned}$$

and

$$\begin{aligned}
A_n^*(s) &= (-1)^n \frac{d^n A^*(s)}{ds^n} \\
&= (-1)^n \frac{d^n e^{-s/\lambda}}{ds^n} \\
&= (-1)^n (-1)^n \left(\frac{1}{\lambda}\right)^n e^{-s/\lambda} \\
&= \lambda^{-n} e^{-s/\lambda}
\end{aligned}$$

3.2 Exponential Distributions

For exponential distributions with rate λ and

$$a(t) = \lambda e^{-\lambda t}, \quad \lambda \geq 0, \; t \geq 0$$

we have

$$\begin{aligned}
A^*(s) &= \int_0^\infty e^{-st} dA(t) \\
&= \int_0^\infty e^{-st} \left(\lambda e^{-\lambda t} \right) dt \\
&= \lambda \int_0^\infty e^{-st-\lambda t} dt \\
&= \lambda \int_0^\infty e^{-t(s+\lambda)} dt \\
&= \lambda \left[\frac{-e^{-t(s+\lambda)}}{s+\lambda} \Bigg|_{t=0}^{\infty} \right] \\
&= \lambda \left(\frac{1}{s+\lambda} \right) \\
&= \frac{\lambda}{s+\lambda}
\end{aligned}$$

and

$$
\begin{aligned}
A_n^*(s) &= (-1)^n \frac{d^n A^*(s)}{ds^n} \\
&= (-1)^n \frac{d^n}{ds^n}\left(\frac{\lambda}{s+\lambda}\right) \\
&= (-1)^n \left(n!(-1)^n \frac{\lambda}{(s+\lambda)^{n+1}}\right) \\
&= n! \frac{\lambda}{(s+\lambda)^{n+1}}
\end{aligned}
$$

3.3 Erlang Distributions

For Erlang distributions with k phases, rate $k\lambda$, and

$$
a(t) = \frac{(k\lambda)^k t^{k-1}}{(k-1)!} e^{-k\lambda t}, \quad \lambda \geq 0, \quad t \geq 0
$$

we have

$$
\begin{aligned}
A^*(s) &= \int_0^\infty e^{-st} dA(t) \\
&= \int_0^\infty e^{-st} \left(\frac{(k\lambda)^k t^{k-1}}{(k-1)!} e^{-k\lambda t}\right) dt \\
&= \int_0^\infty \frac{(k\lambda)^k t^{k-1}}{(k-1)!} e^{-t(s+k\lambda)} dt \\
&= \frac{(k\lambda)^k}{(k-1)!} \int_0^\infty t^{k-1} e^{-t(s+k\lambda)} dt
\end{aligned}
$$

which is resolved through repeated integration by parts, so that

$$\int_0^{\infty} t^{k-1} e^{-t(s+k\lambda)} dt$$

$$= \left[-\frac{t^{k-1}e^{-t(s+k\lambda)}}{s+k\lambda}\Bigg|_{t=0}^{\infty} \right] + \int_0^{\infty} \frac{(k-1)}{s+k\lambda} t^{k-2} e^{-t(s+k\lambda)} dt$$

$$= \left[-\frac{t^{k-1}e^{-t(s+k\lambda)}}{s+k\lambda}\Bigg|_{t=0}^{\infty} \right] + \frac{(k-1)}{s+k\lambda}\int_0^{\infty} t^{k-2} e^{-t(s+k\lambda)} dt$$

$$= \left[-\frac{t^{k-1}e^{-t(s+k\lambda)}}{s+k\lambda}\Bigg|_{t=0}^{\infty} \right] + \frac{(k-1)}{s+k\lambda}\left(\left[-\frac{t^{k-2}e^{-t(s+k\lambda)}}{s+k\lambda}\Bigg|_{t=0}^{\infty} \right] + \frac{(k-2)}{s+k\lambda}\int_0^{\infty} t^{k-2} e^{-t(s+k\lambda)} dt \right)$$

$$= \left[-\frac{t^{k-1}e^{-t(s+k\lambda)}}{s+k\lambda} - \frac{(k-1)t^{k-2}e^{-t(s+k\lambda)}}{(s+k\lambda)^2}\Bigg|_{t=0}^{\infty} \right] + \frac{(k-1)(k-2)}{(s+k\lambda)^2}\left(\int_0^{\infty} t^{k-2} e^{-t(s+k\lambda)} dt \right)$$

and so on, until we have

$$= \left[-\frac{t^{k-1}e^{-t(s+k\lambda)}}{s+k\lambda} - \frac{(k-1)t^{k-2}e^{-t(s+k\lambda)}}{(s+k\lambda)^2} - \ldots - \frac{(k-1)!e^{-t(s+k\lambda)}}{(s+k\lambda)^k}\Bigg|_{t=0}^{\infty} \right]$$

$$= \left[-\sum_{n=1}^{k-1} \frac{(k-1)!}{(k-n)!}\frac{t^{k-n}e^{-t(s+k\lambda)}}{(s+k\lambda)^n}\Bigg|_{t=0}^{\infty} \right] - \left[\frac{(k-1)!e^{-t(s+k\lambda)}}{(s+k\lambda)^k}\Bigg|_{t=0}^{\infty} \right]$$

Note that for all terms in the summation above (excluding the term outside the summation), we would need to apply L'Hopital's rule when $t = \infty$, such that each of those terms ultimately becomes $\frac{(k-n)!}{e^{\infty}} = 0$, leaving only

$$-\left[\frac{(k-1)!e^{-t(s+k\lambda)}}{(s+k\lambda)^k}\Bigg|_{t=0}^{\infty} \right] = -\left(0 - \frac{(k-1)!}{(s+k\lambda)^k} \right) = \frac{(k-1)!}{(s+k\lambda)^k}$$

thus we have

$$A^*(s) = \frac{(k\lambda)^k}{(k-1)!}\int_0^\infty t^{k-1}e^{-t(s+k\lambda)}dt$$

$$= \frac{(k\lambda)^k}{(k-1)!}\left(\frac{(k-1)!}{(s+k\lambda)^k}\right)$$

$$= \frac{(k\lambda)^k}{(s+k\lambda)^k}$$

and

$$A_n^*(s) = (-1)^n\frac{d^nA^*(s)}{ds^n}$$

$$= (-1)^n\frac{d^n}{ds^n}\frac{(k\lambda)^k}{(s+k\lambda)^k}$$

$$= (-1)^n(k\lambda)^k\frac{d^n}{ds^n}\frac{1}{(s+k\lambda)^k}$$

$$= (-1)^n(k\lambda)^k\left(\frac{(-1)^n(k+n-1)!}{(k-1)!(s+k\lambda)^{k+n}}\right)$$

$$= \frac{(k-1+n)!(k\lambda)^k}{(k-1)!(s+k\lambda)^{k+n}}$$

$$= \frac{n!(k-1+n)!(k\lambda)^k}{n!(k-1)!(s+k\lambda)^{k+n}}$$

$$= n!\binom{k-1+n}{n}\frac{(k\lambda)^k}{(s+k\lambda)^{k+n}}$$

For $k = 2$, we therefore have

$$A^*(s) = \frac{4\lambda^2}{(s+2\lambda)^2}$$

and

$$\begin{aligned}A_n^*(s) &= n!\binom{2-1+n}{n}\frac{(2\lambda)^2}{(s+2\lambda)^{2+n}} \\ &= n!\binom{n+1}{n}\frac{(2\lambda)^2}{(s+2\lambda)^{2+n}} \\ &= n!(n+1)\frac{(2\lambda)^2}{(s+2\lambda)^{2+n}}\end{aligned}$$

3.4 Hyperexponential Distributions

For a hyperexponential mixture of exponential distributions with

$$a(t) = \sum_{i=1}^{k} p_i \lambda_i e^{-\lambda_i t}, \quad 0 \le p \le 1, \;\; \lambda_i \ge 0, \;\; t \ge 0$$

where λ_i are the means of the exponential distributions composing the mixture and p_i are their relative weights, we have

$$\begin{aligned}A^*(s) &= \int_0^\infty e^{-st} dA(t) \\ &= \int_0^\infty e^{-st}\left(\sum_{i=1}^{k} p_i \lambda_i e^{-\lambda_i t}\right) dt \\ &= \sum_{i=1}^{k} p_i \lambda_i \int_0^\infty e^{-st} e^{-\lambda_i t} dt\end{aligned}$$

$$= \sum_{i=1}^{k} p_i \lambda_i \int_0^\infty e^{-t(s+\lambda_i)} dt$$

$$= \sum_{i=1}^{k} p_i \lambda_i \left[\frac{-e^{-t(s+\lambda_i)}}{s+\lambda_i} \Bigg|_{t=0}^{\infty} \right]$$

$$= \sum_{i=1}^{k} p_i \lambda_i \left(\frac{1}{s+\lambda_i} \right)$$

$$= \sum_{i=1}^{k} \left(\frac{p_i \lambda_i}{s+\lambda_i} \right)$$

and

$$A_n^*(s) = (-1)^n \frac{d^n A^*(s)}{ds^n}$$

$$= (-1)^n \frac{d^n}{ds^n} \sum_{i=1}^{k} \left(\frac{p_i \lambda_i}{s+\lambda_i} \right)$$

$$= (-1)^n \sum_{i=1}^{k} \frac{d^n}{ds^n} \left(\frac{p_i \lambda_i}{s+\lambda_i} \right)$$

$$= (-1)^n \sum_{i=1}^{k} p_i \lambda_i \frac{d^n}{ds^n} \left(\frac{1}{s+\lambda_i} \right)$$

$$= (-1)^n \sum_{i=1}^{k} p_i \lambda_i \left(n!(-1)^n \frac{1}{(s+\lambda_i)^{n+1}} \right)$$

$$= n! \sum_{i=1}^{k} \frac{p_i \lambda_i}{(s+\lambda_i)^{n+1}}$$

Thus, for a mixture of two exponentials with $p_1 = p,\ p_2 = 1 - p$ and rates λ_1, λ_2 we have:

$$A^*(s) = \sum_{i=1}^{k} \left(\frac{p_i \lambda_i}{s + \lambda_i} \right) = \frac{p\lambda_1}{s + \lambda_1} + \frac{(1-p)\lambda_2}{s + \lambda_2}$$

and

$$A_n^*(s) = n! \left(\frac{p\lambda_1}{(s + \lambda_1)^{n+1}} + \frac{(1-p)\lambda_2}{(s + \lambda_2)^{n+1}} \right)$$

Appendix B

Python Code for Numerical Results

```
import math
import string
import scipy.special as prob
import numpy as np
import matplotlib.pyplot as plt
from decimal import *
l_par = str(string.punctuation[7])
r_par = str(string.punctuation[8])
path = "C:/Users/tom18/OneDrive/Documents/2nd Degree/Thesis/"

#### Inputs ####
getcontext().prec = 16       # precision for mantissa
boundary = Decimal('1E-250')  # "near-zero" boundary for terminating summations
epsilon = Decimal('1E-16')  # max error for root-solve and N
servers = int(1)
service_rate = Decimal('6') / servers  # per-server rate = mu
arrival_type = 'Hyp'   # arrival distribution
K = int(0)  #### If non-zero, provide finite-buffer results. Overrides N.
temp_p0 = False  # use complement if False
N_override = int(0)  #### If non-zero, override calculated N with this value

# Arrival parameters vary by distribution.
# For deterministic, param = a = fixed time between arrivals = 1/lambda
# For exponential, param = rate = lambda
```

```
# For Erlang k=2, param = lambda, rate = 2 * lambda   (lambda = rate / 2)
# For hyper exponential, params = [p, lambda_1, lambda_2],
#   rate = 1 / [p * lambda_1 + (1-p) * lambda_2]
lambda_det = Decimal('5.0')
a = 1/lambda_det
lambda_exp = Decimal('5.0')
lambda_Erl = Decimal('5.0')
p = Decimal('0.8')  #for testing Hyperexponential
lambda_1 = Decimal('8.0')
lambda_2 = Decimal('2.0')


#### Validate distribution type
dist = arrival_type
dist_types = ['Det', 'Exp', 'Erl', 'Hyp']
if dist not in dist_types  :
    print("Invalid arrival distribution specified.")
    valid_dist = False
else :
    valid_dist = True

####  Assign inputs to variables
e = epsilon
c = servers
mu = service_rate
precision = abs(epsilon.as_tuple().exponent)

###  Compute overall arrival rate lambda based on distribution
if dist.lower() == str('det') :
    arrival_rate = Decimal('1')/a
    l = arrival_rate
if dist.lower() == str('exp') :
    arrival_rate = lambda_exp
    l = arrival_rate
if dist.lower() == str('erl') :
    arrival_rate = lambda_Erl
    l = lambda_Erl
```

```
if dist.lower() == str('hyp') :
    arrival_rate = 1/(p/lambda_1 + (1-p)/lambda_2)

# Print inputs
print(str(dist) + " arrival rate = " + str(arrival_rate))
print("Per-server rate (mu) = " + str(mu) + ", Number of servers (c) = "
      + str(c))
print("Overall service rate (c*mu) = " + str(c*mu))

#### validate rho < 1
rho = arrival_rate / (c*mu)
if rho >= 1 or rho <= 0 :
    print("System does not have long-run stability.")
    stable = False
else :
    stable = True

####
## execute all steps for stable system (rho<1)
####
while stable and valid_dist:
    print("stable system (rho = " + str(rho) + ") - execute all steps")

    # Root-solve
    old_root = Decimal('0.5')
    err = Decimal('1')
    while (err > e) :
        if dist.lower() == str('det') :
            root = Decimal.exp(Decimal('-1') * (c*mu * (Decimal('1')
                                - old_root) * a))
        elif dist.lower() == str('exp') :
            root = l / ((c*mu * (Decimal('1') - old_root)) + l)
        elif dist.lower() == str('erl') :
            root = ((Decimal('4') * lambda_Erl**Decimal('2')) /
                    ((c*mu * (Decimal('1') - old_root)) +
                    Decimal('2')*lambda_Erl)**Decimal('2'))
        elif dist.lower() == str('hyp') :
```

```
            root = (p*lambda_1 / ((c*mu * (Decimal('1') - old_root))
                    + lambda_1) + (Decimal('1') - p)*lambda_2
                    / ((c*mu * (Decimal('1') - old_root)) + lambda_2))
        else :
            stable = False
            print('Distribution not found.')
            break
        err = Decimal.copy_abs(root - old_root)
        old_root = root
    print("Root = " + str(root))

    # Calculate N based on epsilon and root
    trunc_point = c + (Decimal.ln(e) - Decimal('2') * Decimal.ln(Decimal('1')
                  - root)) / Decimal.ln(root)
    N = int(round(trunc_point,1))
    print('N = ' + str(N))

    #   Forced override of N (if specified)
    if N_override != 0 :
        N = N_override
        print("N overriden.  N = " + str(N))

    #   Finite buffer model (if specified)
    if K != 0 :
        N = K
        print("Finite buffer model.  N = K = " + str(N))

    # Calculate A*_n(s) for each distribution
    if dist.lower() == str('det') :
        A_star = np.empty(c+1, object)
        for k in range(0, c+1) : #define A*(s) for s = mu, 2mu, ..., c mu
            s = Decimal(k)*mu
            A_star[k] = Decimal.exp(Decimal('-1') * s * a)
        A_star_n = np.empty(N-c+2, object)
        for n in range(0, N-c+2) : #define A*_n(c mu) for n = 1,2,...,N-c+1
            A_star_n[n] = (a**n * Decimal.exp(Decimal('-1') * c * mu * a) /
                           Decimal(math.factorial(n)))
```

```
    elif dist.lower() == str('exp') :
        A_star = np.empty(c+1, object)
        for k in range(0, c+1) : #define A*(s) for s = mu, 2mu, ..., c mu
            s = Decimal(k)*mu
            A_star[k] = 1 / (s + 1)
        A_star_n = np.empty(N-c+2, object)
        for n in range(0, N-c+2) : #define A*_n(c mu) for n = 1,2,...,N-c+1
            A_star_n[n] = 1 / (c*mu + 1)**Decimal(n+1)
    elif dist.lower() == str('erl') :
        A_star = np.empty(c+1, object)
        for k in range(0, c+1) : #define A*(s) for s = mu, 2mu, ..., c mu
            s = Decimal(k)*mu
            A_star[k] = ((Decimal('4') * lambda_Erl**Decimal('2') / (s +
                        Decimal('2')*lambda_Erl)**Decimal('2')))
        A_star_n = np.empty(N-c+2, object)
        for n in range(0, N-c+2) : #define A*_n(c mu) for n = 1, 2, ..., N-c+1
            A_star_n[n] = ((n+1) * Decimal('4') *
                        lambda_Erl**Decimal('2') /
                        (c*mu + Decimal('2')*lambda_Erl)**Decimal(n+2))
    elif dist.lower() == str('hyp') :
        A_star = np.empty(c+1, object)
        for k in range(0, c+1) : #define A*(s) for s = mu, 2mu, ..., c mu
            s = Decimal(k)*mu
            A_star[k] = ((p*lambda_1 / (s + lambda_1)) +
                    ((Decimal('1')-p)*lambda_2 / (s + lambda_2)))
        A_star_n = np.empty(N-c+2, object)
        for n in range(0, N-c+2) : #define A*_n(c mu) for n = 1,2,...,N-c+1
            A_star_n[n] = (((p*lambda_1 / (c*mu +
                    lambda_1)**Decimal(n+1)) +
                    ((Decimal('1')-p)*lambda_2 /
                    (c*mu + lambda_2)**Decimal(n+1))))
    print("A* completed")

    # Compute C_ij for i, j = 1,2,...,c-1
    C = np.empty([c,c], object)  #array that will not use 0 row or 0 col
    for i,row in enumerate(C) :
        for j,item in enumerate(row) :
```

```
            C[i,j] = Decimal('1')

    for k,row in enumerate(C) :
        if k == 0 : continue  #skip zero row
        for j,item in enumerate(row) :
            if j == 0 : continue  #skip zero column
            C[k,j] = Decimal('1')  #product over empty set is one.
            for m in range(1,k) :  #does not execute if k==1
                C[k,j] = C[k,j] * (c-m) / (Decimal(k)-m)
            for m in range(k+1,j+1) :  #does not execute if j <= k+1
                C[k,j] = C[k,j] * (c-m) / (Decimal(k)-m)
    print("C_kj completed.")

    # initialize P matrix with zeros
    P = np.empty([N+1,N+1], object)
    for i,row in enumerate(P) :
        for j,item in enumerate(row) :
            P[i,j] = Decimal('0')

    #Compute p_ij's for i = 0, 2, ..., c-1;  j = 0, 1, ..., i+1
    for i in range(0,c) :
        for j in range(0,i+2) :
            if j > N : continue  #avoid violating boundary
            summation = Decimal('0')
            for r in range(0,i-j+2) :
                temp = (Decimal(-1)**Decimal(r) * A_star[j+r]
                            / (Decimal(math.factorial(i-j+1-r))
                            * Decimal(math.factorial(r))))
                if Decimal.copy_abs(temp) < boundary :  #underflow trap
                    continue
                summation += temp
            P[i,j] = (Decimal(math.factorial(i+1)) * summation
                    / Decimal(math.factorial(j)))
            if P[i,j] < Decimal('0') :  # underflow trap
                print("P" + l_par + str(i) + "," + str(j) + r_par
                    + " = " + format(P[i,j], '3.6f'))
                P[i,j] = Decimal('0')
```

```
        if P[i,j] > Decimal('1') :
            print("P" + l_par + str(i) + "," + str(j) + r_par
                  + " = " + format(P[i,j], '3.6f'))
            P[i,j] = Decimal('1')   #prevent propagation of overflow
print("P_ij for 0 <= i <= c-1;  0 <= j <= i+1 completed.")

#Compute p_ij's for i = c, c+1, ..., N;  j = c, c+1, ..., i+1; i+1<=N
for i in range(c,N+1) :
    for j in range(c,i+2) :
        if j > N : continue   #avoid violating boundary
        P[i,j] = ((c*mu)**Decimal(i-j+1) * A_star_n[i-j+1])
        if P[i,j] < Decimal('0') :  #underflow trap
            print("P" + l_par + str(i) + "," + str(j) + r_par
                  + " = " + format(P[i,j], '3.6f'))
            P[i,j] = Decimal('0')
        if P[i,j] > Decimal('1') :
            print("P" + l_par + str(i) + "," + str(j) + r_par
                  + " = " + format(P[i,j], '3.6f'))
            P[i,j] = Decimal('1')   #prevent propagation of overflow
print("P_ij for c <= i <= N;  c <= j <= N completed.")

#Compute p_ij's for i = c, c+1, ..., N;  j = 1, 2, ..., c-1 (Region 3)
for i in range(c,N+1) :
    for j in range(1,c) :
        P[i,j-1] == boundary
        first_term = Decimal('0')
        second_term = Decimal('0')
        summation2 = Decimal('0')
        for r in range(0,i-c+2) :
            summation2 += (((c-j)*mu)**Decimal(r) * A_star_n[r])
        A_star_minus_sum2 = A_star[j] - summation2
        second_term = (C[c-j, c-j] * (c / (c-Decimal(j)))**Decimal(i-c+2)
                       * A_star_minus_sum2)
        if (c-j-1) < 1 :   #first term = sum over empty set = 0
            P[i,j] = second_term
             if P[i,j] < Decimal('0') :  # underflow trap
                 P[i,j] = Decimal('0')
```

```
                if P[i,j] > P[i-1,j] :
                    P[i,j] = Decimal('0')  #prevent propagation of overflow
                continue
            for k in range(1, c-j) :  #compute first term
                summation1 = Decimal('0')
                for r in range(0,i-c+2) : #compute summation over r
                    summation1 += ((Decimal(k)*mu)**Decimal(r) * A_star_n[r])
                A_star_minus_sum1 = A_star[c-k] - summation1
                first_term += ((C[k,c-j] * (c-Decimal(k)) / Decimal(j))
                                * (c/Decimal(k))**Decimal(i-c+2)
                                    * A_star_minus_sum1)
            P[i,j] = first_term + second_term
            if P[i,j] < Decimal('0') :  # eliminate negative probabilities
                P[i,j] = Decimal('0')
            if P[i,j] > P[i-1,j] and P[i,j-1] == Decimal('0') :
                P[i,j] = Decimal('0')   #prevent propagation of overflow
print("P_ij for c <= i <= N;  1 <= j <= c-1 completed.")

#Compute p_ij's for i = c-1, 1, ..., N;  j = 0
for i in range(c-1,N+1) :
    if P[i,1] == Decimal('0') and P[i,2] == Decimal('0') :
        P[i,0] = Decimal('0')
        continue
    summation = Decimal('0')
    for j in range(1,N+1) :
        summation += P[i,j]
    P[i,0] = Decimal('1') - summation
    if P[i,0] < Decimal('0') :  # eliminate negative probabilities
        P[i,0] = Decimal('0')
    if P[i,0] > Decimal('1') :
        P[i,0] = Decimal('1')   #prevent propagation of overflow

# Finite-buffer model: final row = previous row
if K != 0 :
    for j in range(0,N+1) :
        P[N,j] = P[N-1,j]
```

```python
#Initialize array of zeros for a_kj
a = np.empty([N+1,N+1], object)
for k,row in enumerate(P) :
    for j,item in enumerate(row) :
        a[k,j] = Decimal('0')

#Compute a(k,j) for k = j+1, j+2, ..., N;  j = 0, 1, ..., c-1
for j in range(0,c) :
    for k in range(j+1,N+1) :
        for i in range(0,j+1) :
            a[k,j] += P[k,i]

#Compute pi'_j
pi_prime = np.empty(N+1, object)
for j in range(c, N+1) :
    pi_prime[j] = root**Decimal(j)
for j in range(c-1, -1, -1) :
    summation = Decimal('0')
    for k in range(j+1, N+1) :
        summation += pi_prime[k] * a[k,j]
    pi_prime[j] = summation / P[j, j+1]

#Compute pi_j
pi = np.empty(N+1, object)
phi = sum(pi_prime)
for j,item in enumerate(pi) :
    pi[j] = pi_prime[j] / phi

p = np.empty(N+1, object)
#Compute p_n for 1 <= n <= c
for n in range(1, c+1) :
    p[n] = Decimal(c) * rho * pi[n-1] / Decimal(n)

#Compute p_n for c+1 <= n <= N
for n in range(c+1, N+1) :
    p[n] = rho * pi[n-1]
```

```
    if temp_p0 : #Compute p_0 using Lemma 3.3
        summation = Decimal('0')
        for k in range(0,c-1) :
            summation += pi[k] * (c-k-1) / (k+1)
        p[0] = (1 - rho) - (rho * summation) + (rho * pi[N])
    else : #Compute p_0 as 1-sum(p_n)
        summation = Decimal('0')
        for n in range(1, N+1) :
            summation += p[n]
        p[0] = Decimal('1') - summation

    getcontext().prec = 10
    print("p_n = ")
    for n, item in enumerate(p) :
        print("p_" + str(n) + " :  " + str(format(item, '3.32f')))

    # Compute performance measures L, W
    L = Decimal('0')
    for n in range(1, N+1) :
        L += n*p[n]
    W = L / arrival_rate
    if K != 0 :
        W = L / (arrival_rate * (1-p[K]))
    print("L = " + str(format(L, '3.16f')) + ",    W = " + str(format(W, '3.16f'

    stable = False #when finished, end loop
    #### END OF MAIN LOOP ####


print("End of program, exporting results.")

pi_j_short = np.zeros((N+1))
for row_index,item in enumerate(pi) :
    pi_j_short[row_index] = float(format(item, '4.500f'))
np.savetxt(path + "pi_j.csv", pi_j_short, delimiter=",")

pi_prime_short = np.zeros((N+1))
```

```
for row_index,item in enumerate(pi_prime) :
    pi_prime_short[row_index] = float(format(item, '4.500f'))
np.savetxt(path + "pi_prime.csv", pi_prime_short, delimiter=",")

A_star_n_short = np.zeros((N-c+2))
for row_index,item in enumerate(A_star_n) :
    A_star_n_short[row_index] = float(format(item, '4.500f'))
np.savetxt(path + "A_star_n.csv", A_star_n_short, delimiter=",")

C_short = np.empty([c,c])
for row_index,row in enumerate(C) :
    for col_index,item in enumerate(row) :
#           if col_index == 0 : continue  #skip zero column
#           if item >=1 or item <0 :
        C_short[row_index,col_index] = float(format(item, '4.500f'))
np.savetxt(path + "C.csv", C_short, delimiter=",")

p_n_short = np.zeros(N+1)
for row_index,item in enumerate(p) :
    p_n_short[row_index] = float(format(item, '4.500f'))
    continue
np.savetxt(path + "p_n.csv", p_n_short, delimiter=",")

P_short = np.empty([N+1,N+1])
for row_index,row in enumerate(P) :
    for col_index,item in enumerate(row) :
        P_short[row_index, col_index] = float(format(item, '4.500f'))
np.savetxt(path + "P.csv", P_short, delimiter=",")

Perf = np.empty(2)
Perf[0] = float(format(L, '4.500f'))
Perf[1] = float(format(W, '4.500f'))
np.savetxt(path + "Perf.csv", Perf, delimiter=",")
```