The nonstationary Erlang loss model is a queueing system consisting of a finite number of servers and no waiting room with a nonstationary arrival process or a time-dependent service rate. The Erlang loss model is commonly used to model and evaluate many communication systems. Often, these types of service systems encounter a change in the arrival rate over time while the service rate remains either constant or changes very little over time. In view of this, the focus in this research is the nonstationary Erlang loss queues and network with time-dependent arrival rate and constant service rate. We developed an iterative scheme referred to as the fixed point approximation (FPA) in order to obtain the time-dependent blocking probability and other measures for a single-class nonstationary Erlang loss queue and a nonstationary multi-rate Erlang loss queue. The FPA method was compared against exact numerical results, and two other methods, namely, MOL and PSA, for various nonstationary Erlang loss queues with sinusoidal arrival rates. Although we used sinusoidal functions to model the time-dependent arrival rate, the solution can be obtained for any arrival rate function. Experimental results demonstrate that the FPA algorithm provides an exact solution for nonstationary Erlang loss queue. The FPA algorithm was also applied to the case of multi-rate nonstationary Erlang loss queues and the results obtained were compared with simulation. We generalized the FPA algorithm for networks of nonstationary Erlang loss queues with Markovian branching, and compared its accuracy to simulation. Finally, FPA was used to analyze networks of nonstationary Erlang loss queues with population constraints. Numerical results showed that FPA provides a good approximation.
NONSTATIONARY ERLANG LOSS QUEUES AND NETWORKS

by

KHALID ABDULAZIZ ALNOWIBET

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial satisfaction of the requirements for the Degree of Doctor of Philosophy

OPERATIONS RESEARCH

Raleigh 2004

Approved By

Dr. Yannis Viniotis
Dr. Thomas Reiland

Dr. William Stewart
Dr. Harry Perros
Chair of Advisory Committee
In the name of Allah, Most Gracious, Most Merciful • Praise be to Allah, the Cherisher and Sustainer of the worlds • Most Gracious, Most Merciful • Master of the Day of Judgment. • Thee do we worship, and Thine aid we seek • Show us the straight way • The way of those on whom Thou hast bestowed Thy Grace, those whose (portion) is not wrath, and who go not astray.
Biography

KHALID ABDULAZIZ ALNOWIBET was born in Riyadh, Kingdom of Saudi Arabia in 1971. He received his B.Sc. in 1992 from the department of Statistics and Operations Research at King Saud University in Riyadh, Kingdom of Saudi Arabia. After graduation, he worked as a planner for the Ministry of Planning for one year before he got a teaching assistant position at the department of Statistics and Operations Research at King Saud University. In 1995, King Saud University granted him a scholarship to pursue graduate studies in Operation Research abroad. He received his M.S. in 1998 from George Washington University, DC, and in 1999, he enrolled in North Carolina State University for the Ph.D. degree in Operations Research.
Acknowledgments

First of all, I would like thank Allah, my lord, for His blessings and mercy upon me for having this great opportunity. I pray to Allah to make me among His grateful slaves and to hold my heart steadfast on the straight path till my last breath.

I owe a debt of gratitude to my advisor, Professor Harry Perros, for his continuous support and patience. I also thank him for his valuable advice and expert insight in this research as well as the composition of this dissertation. I am also thankful to Professors Thomas Reiland, William Stewart, and Yannis Viniotis, for serving as advisors on my committee and for their constructive input. Lastly, I would like to express my sincere thanks to my father, my mother and my wife for their encouragement, patience and prayers.
# Table of Contents

List of Tables viii

List of Figures ix

1 Introduction 1
   1.1 Research Motivation ........................................ 1
   1.2 Organization of the Dissertation .......................... 3

2 Literature Survey 4
   2.1 The Simple Stationary Approximation (SSA) ............ 4
   2.2 The Stationary Peakness Approximation (PK) .......... 8
   2.3 The Average Stationary Approximation (ASA) .......... 11
   2.4 The Closure Approximation for Nonstationary Queues 12
   2.5 The Pointwise Stationary Approximation (PSA) ........ 15
   2.6 The Pointwise Stationary Fluid Flow Approximation (PSFFA) 20
   2.7 The Modified Offered Load Approximation (MOL) ....... 23

3 Nonstationary Erlang Loss Queues 29
   3.1 The Mathematical Model ................................... 29
   3.2 The Fixed Point Approximation (FPA) .................... 32
   3.3 The Erlang Loss Model with Sinusoidal Arrival Rate .... 37
      3.3.1 Accuracy of FPA as $\Delta t \to \infty$ .................. 39
      3.3.2 Accuracy of FPA with Respect to the Number of Servers 39
      3.3.3 Accuracy of FPA with Respect to Average Arrival Rate 45
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2 A Constrained Network of Nonstationary Erlang Loss Queues</td>
<td>108</td>
</tr>
<tr>
<td>5.3 The FPA for Constrained Networks of Nonstationary Erlang Loss Queues</td>
<td>110</td>
</tr>
<tr>
<td>5.4 Applications and Numerical Examples.</td>
<td>116</td>
</tr>
<tr>
<td>5.4.1 A Traffic-Groomed Tandem Nonstationary Optical Network</td>
<td>117</td>
</tr>
<tr>
<td>5.4.1.1 A Numerical Example</td>
<td>118</td>
</tr>
<tr>
<td>5.4.2 A Low Earth Satellite Network on a Single Orbit</td>
<td>122</td>
</tr>
<tr>
<td>5.4.2.1 A Numerical Example</td>
<td>124</td>
</tr>
<tr>
<td>6 Conclusion and Future Research</td>
<td>130</td>
</tr>
<tr>
<td>6.1 Future Research</td>
<td>131</td>
</tr>
<tr>
<td>References</td>
<td>132</td>
</tr>
<tr>
<td>Appendices</td>
<td>136</td>
</tr>
<tr>
<td>A Plots for Numerical examples in Chapter 3</td>
<td>137</td>
</tr>
<tr>
<td>B Examples of Other Time-Dependent Arrival Rate Functions</td>
<td>146</td>
</tr>
<tr>
<td>C Plots for Numerical examples in Chapter 4</td>
<td>154</td>
</tr>
<tr>
<td>D Plots for Numerical examples in Chapter 5</td>
<td>157</td>
</tr>
<tr>
<td>E MATLAB and Simulation Codes</td>
<td>162</td>
</tr>
</tbody>
</table>
# List of Tables

3.1 Average absolute error of FPA as $\Delta t \to 0$ ........................................... 39

3.2 Error of FPA compared with the error of MOL and PSA as the number of servers changes ................................................................. 40

3.3 Error of FPA compared with the error of MOL and PSA as the average arrival rate changes ................................................................. 45

3.4 Error of FPA compared with the error of MOL and PSA as the amplitude of arrival rate changes ................................................................. 46

3.5 Mean and standard deviation of the error and relative error of each method ................................................................. 46

4.1 Error of FPA for five-queue Erlang loss network with the diverging topology 78

4.2 Error of FPA for five-queue loss network with the diverging topology 82

4.3 Error of FPA for five-queue loss network with the series topology 85

4.4 Error of FPA for Six-queue Erlang loss network with feedback. 89

4.5 Error in blocking probabilities using FPA for six-queue multi-rate loss network in Figure 4.25 104

5.1. Simulated and approximate average blocking probability and relative and average absolute error for each class in the optical network in Figure 5.4 122

5.2. Simulate and approximate average blocking probability and relative and average absolute error for each class in the satellite network in Figure 5.7 128
# List of Figures

2.1 Types of periodic arrival rate functions ........................................... 5
2.2 Periodic Averaging of the arrival rate functions ................................. 6
2.3 Average arrival rate over subintervals with equal lengths ...................... 11
2.4 PSA approximation for average number of customers in system1 and system 2 18
2.5 Exact and PSA approximation of average number of customers in the system for an Erlang loss system with $\lambda(t) = 20 + 15 \sin(2t)$, $\mu = 2$ and $s = 15$ ......................................................... 19
2.6 Exact and PSA and MOL approximation of average customers in the system for an Erlang loss system with $\lambda(t) = 20 + 15 \sin(2t)$, $\mu = 2$ and $s = 15$ ......................................................... 27
3.1 Rate diagram for $M(t)/M/s/0$ ............................................................. 29
3.2 Effective arrival and departure rates in a loss queue ................................. 33
3.3 Results of Erlang loss system with $\lambda(t) = 10 + 5 \sin(2t)$, $\mu = 1$, and $s = 5$ ................................. 41
3.4 Results of Erlang loss system with $\lambda(t) = 10 + 5 \sin(2t)$, $\mu = 1$, and $s = 10$ ......................................................... 42
3.5 Results of Erlang loss system with $\lambda(t) = 10 + 5 \sin(2t)$, $\mu = 1$, and $s = 15$ ......................................................... 43
3.6 Results of Erlang loss system with $\lambda(t) = 10 + 5 \sin(2t)$, $\mu = 1$, and $s = 20$ ......................................................... 44
3.7 Collecting and analyzing observations of time-dependent simulation ........ 62
3.8 Results of FPA and simulation for two-class Erlang loss system with parameters $\lambda_1(t) = 100 + 70 \sin(t)$, $\lambda_2(t) = 80 + 50 \sin(2t)$, $\mu_1 = 2$, $\mu_2 = 3$ and $s = 100$ ......................................................... 64
4.1 Arrival and departure rates of a queue in an infinite-server queueing network 69
4.2 Arrival and departure rates of a queue in an Erlang loss queueing network 71
4.3 A Five-queue Erlang loss network with the diverging topology . . . . . . . 77
4.4 Simulation and FPA results of queue 1. . . . . . . . . . . . . . . . . . . . . . . 79
4.5 Simulation and FPA results of queue 2. . . . . . . . . . . . . . . . . . . . . . . 79
4.6 Simulation and FPA results of queue 3. . . . . . . . . . . . . . . . . . . . . . . 79
4.7 Simulation and FPA results of queue 4. . . . . . . . . . . . . . . . . . . . . . . 80
4.8 Simulation and FPA results of queue 5. . . . . . . . . . . . . . . . . . . . . . . 80
4.9 A five-queue loss network with the converging topology . . . . . . . . . . 81
4.10 Simulation and FPA results of queue 1. . . . . . . . . . . . . . . . . . . . . . . 83
4.11 Five-queue loss network with the series topology . . . . . . . . . . . . . . 84
4.12 Simulation and FPA results of queue 1. . . . . . . . . . . . . . . . . . . . . . . 86
4.13 Simulation and FPA results of queue 2 . . . . . . . . . . . . . . . . . . . . . . . 86
4.14 Simulation and FPA results of queue 3. . . . . . . . . . . . . . . . . . . . . . . 86
4.15 Simulation and FPA results of queue 4. . . . . . . . . . . . . . . . . . . . . . . 87
4.16 Simulation and FPA results of queue 5. . . . . . . . . . . . . . . . . . . . . . . 87
4.17 A Six-queue loss network with the feedback topology . . . . . . . . . . . . . 88
4.18 Simulation and FPA results of queue 1. . . . . . . . . . . . . . . . . . . . . . . 90
4.19 Simulation and FPA results of queue 2. . . . . . . . . . . . . . . . . . . . . . . 90
4.20 Simulation and FPA results of queue 3. . . . . . . . . . . . . . . . . . . . . . . 90
4.21 Simulation and FPA results of queue 4. . . . . . . . . . . . . . . . . . . . . . . 91
4.22 Simulation and FPA results of queue 5. . . . . . . . . . . . . . . . . . . . . . . 91
4.23 Simulation and FPA results of queue 6. . . . . . . . . . . . . . . . . . . . . . . 91
4.24 Arrival and departure rates of a queue in a multi-rate Erlang loss network . . 98
4.25  A six-queue Erlang loss network with three classes of arrivals . . . . . . 103

4.26  Simulation and FPA results blocking for class-3 arrivals . . . . . . . . 105

5.1  Simple telephone network . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 107

5.2  Independent multi-rate Erlang loss queues of the network in Figure 5.2 . . . 112

5.3  A six-node tandem optical network with all source-destination arrivals . . . 118

5.4  Blocking probability using FPA and the 95% confidence interval of simulation
    for the classes (1,2), (2,3), (1,3) and (2,4) . . . . . . . . . . . . . . . . . . . 120

5.5  Blocking probability using FPA and the 95% confidence interval of simulation
    for the classes (1,4), (2,5), (1,5) and (1,6) . . . . . . . . . . . . . . . . . . . 121

5.6  (a) Link occupancy over a single footprint. (b) Link occupancy over two
    footprints . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123

5.7  A constellation of five LEO satellites . . . . . . . . . . . . . . . . . . . . . 125

5.8  Blocking probability using FPA and the 95% confidence interval of simulation
    for the classes (1,1), (1,2), (1,3), (1,4) and (1,5) . . . . . . . . . . . . . . . 127
Chapter 1

Introduction

1.1 Research Motivation

The Erlang loss model is a queueing system consisting of $s$ servers and no waiting room. A customer is lost if it arrives at a time when all servers are busy. The Erlang loss model is commonly used to model service systems, such as a telephone switch, a tower for a cellular telephony and a satellite that provides telephone and data services. It has also been recently used to model a wavelength division multiplexing (WDM) optical networks.

The Erlang loss model has been extensively studied in the stationary case (i.e. assuming that arrival rate and service rate are time invariant). The nonstationary Erlang loss model, where the arrival rate and/or service rate are time-dependent is also of a great interest. Specifically, in most communication systems the arrival rate varies over time. As an example, consider a tower used for cellular telephony. The actual population that uses the tower is not constant, since mobile users continuously enter and leave the
tower’s coverage area. In view of this, it makes sense to model this system with an Erlang loss model assuming that the arrival rate $\lambda(t)$ is some arbitrary function of time $t$.

The nonstationary Erlang loss model also has some applications in the context of queueing networks. A typical example of this problem is the global system for mobile (GSM) network with fixed channel allocation. The area covered by this system is divided into cells. Usually, the cells are represented as hexagons. Each cell has a limited number of available circuits. When a person uses his mobile phone within a cell, he may finish his call without leaving the cell and, hence, depart the network. Alternatively, he may traverse a number of cells. The sequence of cells taken depends on the route of the mobile caller. From a previous call data and road network, the movement from one cell to another can be translated into probability distribution. These probabilities represent the branching probabilities in the network.

Theoretically, what makes the stationary Erlang loss model more attractive than the nonstationary model is that it has a closed form expression for the steady-state solution. In addition, it possesses the insensitivity property to the service processes. In contrast, the nonstationary Erlang loss model does not have a closed-form solution due to the variability of rates over time. In order to obtain the steady state distribution, one must solve the forward differential equations as functions of time. As a result, the steady-state distribution is a function of time.

Most of the real life queueing applications have nonstationary arrival rate functions that are periodic in nature. On the other hand, the service rate in most applications is constant over time or nearly constant where its variability over time can be neglected. Therefore, in this thesis we concentrate on the cases where the arrival rate is time-dependent and the service rate is time-independent.

Throughout this thesis, a modified Kendall’s notation is used to express the dependency of rates on time. The time-dependent Erlang loss model is denoted as $M(t)/M/s/0$ where the last term represents the waiting space. In some literature, the notation $M(t)/M/s/s$ is also used where the last term represents the maximum total number of customers in the system. The first term $M(t)$ denotes that the arrival process is
Poisson with time-dependent rate function. A similar notation could be used for nonstationary service process.

1.2 Organization of the Dissertation

The remainder of this dissertation is organized into five chapters. In Chapter 2, we review in detail the most important approximation methods found in the literature, and describe some of the advantages and disadvantages of each method. In Chapter 3, we introduce the mathematical model of the nonstationary Erlang loss queue and discuss the difficulties that arise when attempting to solve the forward equations. We also introduce our method, FPA, to obtain the time-dependent blocking probability and provide some numerical examples and comparisons against the MOL and PSA methods. In this chapter also, we applied FPA to the multi-class and multi-rate nonstationary Erlang loss queue. In Chapter 4, we generalize the idea of FPA to solve nonstationary Erlang loss networks with Markovian branching. We also describe the simulation methodology used to measure the accuracy of FPA. Finally, we extend our method to networks of multi-rate nonstationary Erlang loss queues. In Chapter 5, we used the FPA method to approximately analyze the time-dependent blocking probability of a network of nonstationary Erlang loss queues with population constraints. Finally, in Chapter 6, we give our conclusion and describe future research work.
Chapter 2

Literature Survey

In this chapter we review the literature for the analysis of nonstationary Erlang loss queues and networks.

2.1 The Simple Stationary Approximation (SSA)

This method uses the average arrival rate of the nonstationary model to obtain the steady-state results. The average arrival rate for a cycle of length $T$ is

$$\bar{\lambda} = \frac{1}{T} \int_{0}^{T} \lambda(t) \, dt. \quad (2.1)$$

Let $Q(t)$ be number of customers in the system at time $t$. Then, the steady-state distribution can be obtained using the expression:

$$P\{Q(t) = n\} = \frac{\rho^n / n!}{\sum_{i=0}^{s} \rho^i / i!}, \quad n = 0, 1, 2, \ldots, s, \quad \text{where} \quad \rho = \frac{\bar{\lambda}}{\mu}$$
and the blocking probability, $BP(t)$, at time $t$ is the Erlang loss formula with parameter $(s, \rho)$, as follows:

$$BP(t) = P\{Q(t) = s\} = \frac{\rho^s/s!}{\sum_{i=0}^{s} \rho^i/i!}, \text{ for all } t.$$

This method provides a reasonable approximation for the nonstationary system with a weakly varying arrival rate. An arrival rate is considered weakly varying over time if the arrival rate function remains within $\pm 10\%$ interval from the average arrival rate for all $t$ [13]. Figure 2.1 shows two examples of weakly changing arrival rates and one example of strongly varying.

![Figure 2.1: Types of periodic arrival rate functions](image)

The SSA method is simple and can be applied to a wide range of queueing systems that have been already studied extensively. However, this method noticeably underestimates the average performance measures of a nonstationary system with a strongly varying arrival rate. Green et al. [13] have numerically investigated the level of nonstationarity at which this method provides reasonable accuracy. A sinusoidal arrival rate is used in their paper as a typical representation for nonstationary arrival process.
The authors studied the effect of nonstationarity with respect to amplitude, frequency of events and the size of the system (i.e. number of servers.) They numerically showed that this method is applicable to relatively small systems (e.g. one or two servers) with small relative amplitude (e.g. less than 10%), and short cycle length (equivalently infrequent events). In addition, the SSA method provides a time-independent solution for nonstationary systems which does not reflect the effect of the variability in the arrival rate over time on the measure of performance.

Whitt [38] pointed out that this approximation method has a long history in the literature that has been addressed in a fairly sensible way for strongly varying arrival rate functions. The common practice, especially in communication, was to average the arrival rate over an appropriate busy hours and non-busy hours instead of a daily average as shown in Figure 2.2.

The SSA can be extended to cover the time-dependent networks of Erlang loss queues with Markovian branching as described by Abdalla and Boucherie [1]. This type of queueing network consists of \( N \) independent Erlang loss queues each has a time-dependent Poisson stream of external arrivals at rate \( \lambda_i(t) \) and \( s_i \) servers, \( i = 1,2,\ldots,N \).

![Figure 2.2: Periodic Averaging of the arrival rate functions](image-url)
Upon service completion at queue $i$, customer will move to node $j$ for the next service with probability $q_{ij}$. Probability of a customer depart the system from queue $i$ after service is $q_{ii}$. Any external or internal arrival to node $i$ finds all servers busy will be lost.

As proposed in the single loss queue, the arrival rate of each queue in the network is averaged over time using equation (2.1) for all queues in the network. Then the probability that the system is in state $n$ where $n \in S = \{n \in \mathbb{N}^N: 0 \leq n_i \leq s_i, i = 1,2,\ldots,N\}$ is:

$$P(n; t) = \frac{\prod_{j=1}^{N} \rho_j^{n_j} / \prod_{n \in S} \prod_{j=1}^{N} \frac{\rho_j^{n_j}}{n_j!}}{\sum_{n \in S} \prod_{j=1}^{N} \frac{\rho_j^{n_j}}{n_j!}}.$$  

The offered load of queue $j$, $\rho_j$, satisfies the solution of the following traffic equations:

$$0 = \bar{\lambda}_j + \sum_{i=1}^{N} \mu_i q_{ij} \rho_i - \mu_j \rho_j, \quad j = 1,2,\ldots,N.$$  

It is worth mentioning that this method is an exact solution for loss networks with Markovian branching and stationary arrivals if and only the rates of queue $j$, $j=1, 2,\ldots,N$, in the network satisfy the following conditions:

$$\lambda_j = q_{jj} \rho_j \quad (2.2)$$

$$q_{ij} \rho_i = q_{ji} \rho_j; \quad j = 1,2,\ldots,N \quad (2.3)$$

In general, the SSA underestimates the average performance measure of nonstationary systems even when (2.2) and (2.3) conditions are satisfied by the average arrival rates. Therefore, it is expected that the SSA will behave poorly when applied on highly nonstationary loss systems. In addition, the SSA provides the average performance measure of the nonstationary system as oppose to a time-dependent measure.
2.2 The Stationary Peakedness Approximation (PK)

As discussed in SSA method, the simple stationary approximation does not consider the nonstationarity factor of the system. However, it is possible to introduce an extra stochastic characteristic into the stationary process approximation to capture the effect of variability over time in the arrival rate function. This can be done by using a non-Poisson stationary point process to approximate the non-homogeneous Poisson arrival process. Massey and Whitt [26] presented two approaches to do that and used the heavy traffic peakedness on the resulting point process to approximate the blocking probability of the nonstationary Erlang loss model. The peakedness is defined as the ratio of the variance to the mean of the steady-state number of customers in an infinite-server model with the same service time distribution and arrival process.

To explain how this method works, consider a periodic Poisson arrival process with period $T$. This method starts by approximating the nonstationary arrival process by dividing the cycle $T$ into $n$ subintervals. It is assumed that the arrival rate at each subinterval is approximately constant. The arrival rate in any one subinterval is:

$$
\lambda_k = \int_{(k-1)T/n}^{kT/n} \lambda(u) \, du , \quad 1 \leq k < n.
$$

Then, the mean number of arrivals is:

$$
\bar{\lambda}_n = \frac{1}{n} \sum_{k=1}^{n} \lambda_k = \frac{\bar{\lambda} T}{n} , \quad \text{where} \quad \bar{\lambda} = \frac{1}{T} \int_0^T \lambda(u) \, du ,
$$

and its variance is:

$$
\sigma_n^2 = \bar{\lambda}_n + \frac{1}{n} \sum_{k=1}^{n} (\lambda_k - \bar{\lambda}_n)^2 .
$$
Based on the above analysis, the overall arrival process $M(t)$ in the interval $(0, T]$ is approximated by the stationary point process $[N(t) : t \geq 0]$ with mean and variance:

$$n \bar{\lambda}_n = \bar{\lambda}T \quad \text{and} \quad n \sigma_n^2 = \bar{\lambda}T + \sum_{k=1}^{n} (\lambda_k - \bar{\lambda}_n)^2.$$ 

One may notice that the variance depends heavily on $n$. For example, for $n = 1$, $n \sigma_n^2 = \bar{\lambda}T$; while $n \sigma_n^2 \to \bar{\lambda}T$ as $n \to \infty$. Therefore, $n$ should be an intermediate point to capture the variability in arrival process. Next, the peakedness $c^2$ for number of customers in the infinite server system, $Q(t)$, is calculated:

$$c^2 = \frac{\text{Var}[N(T)]}{E[N(T)]} = 1 + \frac{1}{\bar{\lambda}T} \sum_{k=0}^{n} (\lambda_k - \bar{\lambda}_n)^2.$$ 

Assuming that the arrival rate over each subinterval is constant, $c^2$ could be approximated as follows:

$$c^2 \approx 1 + \frac{1}{\bar{\lambda}T} \left( \int_{0}^{T} (\lambda(u) - \bar{\lambda})^2 \, du \right) \approx 1 + \frac{1}{n \bar{\lambda}} \int_{0}^{T} (\lambda(u) - \bar{\lambda})^2 \, du.$$ 

It is always possible to rescale the problem to make the unit time equal to the mean service time. This means that $\mu = 1$. In this case, a good choice for $n$ is to be equal to $T$. Thus,

$$c^2 \approx 1 + \frac{1}{\bar{\lambda}T} \int_{0}^{T} (\lambda(u) - \bar{\lambda})^2 \, du.$$ 

Then, $c^2$ is used to compute the heavy traffic peakedness of the nonstationary process. The heavy-traffic peakedness for an infinite-server system with exponential service
distribution \( (\mu = 1) \) is:

\[
z = 1 + \frac{c^2 - 1}{2} = 1 + \frac{1}{2\lambda T} \int_0^T (\lambda(u) - \bar{\lambda})^2 \, du.
\]

Finally, the approximate blocking probability for the M(t)/M/s/0 queue with time unit equals to the mean service time (i.e., \( \mu = 1 \)) is given by the Erlang loss formula with updated number of servers \( s/z \) (\( s/z \) is integer) and updated offered load \( \bar{\lambda}/z \) as follows:

\[
BP(t) = P\{Q(t) = s\} = \frac{\left(\frac{\lambda}{z}\right)^{s/z} / (s/z)!}{\sum_{i=0}^{s} \left(\frac{\lambda}{z}\right)^i / i!},
\]

for all \( t \), where \( s/z \) is integer and the average number of customers:

\[
E[Q(t)] = (1 - BP(t)) \frac{\bar{\lambda}}{z}
\]

This method is a stationary approximation of the original system. In other words, the PK method finds non-Poisson stationary parameters that better approximate the time-dependent arrival process. Therefore, the resulting approximation with the new parameters is a time reversible process. Although this approximation does not provide a solution for the system as a function time, it provides a better approximation than the SSA method for the average measure of performance of nonstationary Erlang loss models.
2.3 The Average Stationary Approximation (ASA)

This method was introduced by Whitt [38] for loss queues with periodic arrival rates. This approximation starts by dividing the arrival rate cycle $T$ into sub-intervals each of length $\tau$ where $\tau$ is proportional (or equal) to the mean service time. The arrival rate $\lambda(t)$ over subinterval $[t-\tau, t]$ is taken to be equal to the average arrival rate during $[t-\tau, t]$ as follows:

$$\bar{\lambda}_k(t) = \frac{1}{\alpha \mu^{-1}} \int_{t_{k-\alpha \mu^{-1}}}^{t_k} \lambda(u) \, du , \quad t \in [t_{k-\alpha \mu^{-1}}, t_k], \quad \alpha \mu^{-1} = \tau$$

Figure 2.3 shows an example of an arrival rate function after approximation. The arrival process at each sub-interval should be nearly independent of the system state several mean service times later, if the instantaneous offered load is sufficiently below 1.

![Figure 2.3: Average arrival rate over subintervals with equal lengths](image)

Finally, the stationary results are used as a function of $t$ and $\bar{\lambda}_k(t)$ to approximate the performance measures. Namely, the blocking probability, $BP(t,k)$, during subinterval $k$
is approximated as follows:

\[ BP(t, k) = \frac{\rho_i^k}{s!} \sum_{i=0}^{\infty} \frac{\rho_i^k}{i!}, \quad \rho_i^k = \frac{\bar{\lambda}_i(t)}{\mu}, \quad t \in [t_k - \alpha \mu^{-1}, t_k], \]

and the average number of customers, \( E[Q(t,k)] \), during sub-interval \( k \):

\[ E[Q(t,k)] = (1 - BP(t,k)) \frac{\bar{\lambda}_i(t)}{\mu}, \quad \text{for} \quad t \in [t_k - \alpha \mu^{-1}, t_k]. \]

Obviously, the performance measures are going to be step functions due to the discretization of the arrival process. This method is simple to apply and it produces an insight into the behavior of the performance measures over time. In addition, this method provides an exact solution for the M(\( t \))/D/\( \infty \) queue when \( \alpha = 1 \). This method depends mainly on the choice of the subinterval length (\( \tau \)) which is strongly related to the choice of \( \alpha \). If \( \alpha \) is chosen to be small when it should not be, the approximation will pick up more variability from the arrival process than needed. In contrast, if \( \alpha \) is chosen to be large then this method will approach the stationary approximation which kills the variability of the performance measures over time.

2.4 The Closure Approximation for Nonstationary Queues

In general, the solution of the forward equations is needed to obtain any performance measure of interest for a queueing system. The large number and complexity of differential equations give rise to approximation techniques that reduce the number of equations necessary in order to reduce the computation burden.
This method reduces the number of differential equation of the queueing system by considering the differential equations of the mean and the variance of the number of customers in the system. In many systems, the equations for the mean and the variance involve more variables than number of equations. Consequently, an additional number of equations are required to obtain a system of equations that leads to a unique solution.

Consider an M(t)/M/1 queue with arrival rate \( \lambda(t) \) and service rate \( \mu \). The probability \( P_n(t) \) of having \( n \) in the system at time \( t \) is given by the following set of differential equations:

\[
\begin{align*}
\frac{d}{dt} P_0(t) &= -\lambda(t)P_0(t) + \mu P_1(t), \\
\frac{d}{dt} P_n(t) &= -(\lambda(t) + \mu)P_n(t) + \lambda(t)P_{n-1}(t) + \mu P_{n+1}(t), \quad n > 0
\end{align*}
\] (2.4)

Multiplying equations (2.4) by \( n \) and summing over all \( n \) gives:

\[
\frac{d}{dt} E[n] = \sum_{n=0}^{\infty} n \frac{d}{dt} P_n(t) = \lambda(t) - \mu(1 - P_0(t))
\] (2.5)

Multiplying equations (2.4) by \( n^2 \) and summing over all \( n \) gives:

\[
\frac{d}{dt} E[n^2] = \sum_{n=0}^{\infty} n^2 \frac{d}{dt} P_n(t) = \lambda(t) - \mu(1 - P_0(t)) + 2 E[n]\lambda(t) - \mu
\]

So, the variance

\[
\frac{d}{dt} Var[n] = \frac{d}{dt} E[n^2] - \frac{d}{dt} E[n]^2 = \lambda(t) + \mu P_0(t)(2 E[n] + 1)
\] (2.6)

Equations (2.5) and (2.6) provide a system of two differential equations in three unknowns \( (Var[n], E[n] \text{ and } P_0(t)) \). To obtain a unique solution using equations (2.5) and (2.6), an additional equation of \( Var[n], E[n] \text{ and } P_0(t) \) is required to bound the solution of equations (2.5) and (2.6).
Rothkopf and Oren [33] consider the negative binomial distribution to provide a closure function for the M(t)/M(t)/s system. The negative binomial with probability of success $q$ and parameters $n$ and $r$ has the form:

$$p_n(q,r) = \binom{r+n-1}{n} q^n (1-q)^r, \ n = 0, 1, 2, \ldots$$

with mean $r(1-q)q^{-1}$ and variance is $r(1-q)q^{-2}$.

The negative binomial reduces to the geometric distribution if its parameters ($q$ and $r$) are chosen such that:

$$Var[n] = E[n] (1 + E[n]). \quad (2.7)$$

The number of customers in an M/M/1 system has a geometric distribution. Therefore, the parameters $q$ and $r$ can be chosen as functions of the mean and variance of the system so that the resulting negative binomial satisfies property (2.7). The new negative binomial distribution will have the following parameters:

$$q(t) = \frac{E[n]}{Var[n]} \quad \text{and} \quad r(t) = \frac{E[n]^2}{Var[n] - E[n]},$$

where $Var[n]$ and $E[n]$ are functions of time. Finally, the closure function $P_0(t)$ of equations (2.3) and (2.4) is obtained by setting $n$ equals to zero in the negative binomial with the parameters $q(t)$ and $r(t)$:

$$P_0(t) = p_0(q(t), r(t)) = q(t)^{r(0)}.$$

Similarly, the mean and variance of M(t)/M/s system are:

$$\frac{d}{dt} E[n] = \lambda(t) - \mu s + \mu \sum_{n=0}^{s-1} (s - n) P_n(t) \quad (2.8)$$
and

\[
\frac{d}{dt} \text{Var}[n] = \lambda(t) + \mu s - \mu \sum_{n=0}^{s-1} (2E[n] + 1 - 2n)(s - n) P_n(t). \tag{2.9}
\]

The closure functions \((P_n(t)\) for \(n = 0, 1, 2, \ldots, s-1\)) of equations (2.8) and (2.9) are obtained by evaluating the negative binomial distribution described earlier at \(n = 0, 1, 2, \ldots, s-1\).

This method provides an exact solution for the stationary \(M/M/1\) queue and a very good approximation for the \(M(t)/M/1\) queue due to the fact that the stationary system has a geometric steady-state solution. However, for the \(M(t)/M/s\) queue the error of the approximation increases very quickly as the number of servers increases. Rothkopf and Oren [33] provided an error correction term to improve the accuracy of the approximation for the \(M(t)/M/s\) queue.

### 2.5 The Pointwise Stationary Approximation (PSA)

This method is based on the idea that the nonstationary Erlang loss model approximately behaves like a stationary model at each instance of time. Thus, the steady-state results of the stationary Erlang loss model can be used to approximate the nonstationary Erlang loss model at each point on time. This method was first introduced by Grassman [8] in 1983 as a way of constructing an upper bound on the expected number of customers in the queue. Green et al. [13] showed numerically that PSA gives an upper bound on the expected number of customers in the system and probability of delay, if the maximum traffic intensity is strictly less than one. In addition, Green and Kolesar [9] used PSA to approximate the steady-state average performance measures of the periodic \(M(t)/M/s\) queue.
Consider a stationary Erlang loss queue with arrival rate $\lambda$ and service rate $\mu$. Let $Q(t)$ be number of customers in the system at time $t$. Then, the probability $P_n$ that there are $n$ customers in the system is:

$$P_n = \lim_{t \to \infty} P\{Q(t) = n\} = \frac{\rho^n/n!}{\sum_{i=0}^{\infty} \rho^i/i!}, \quad \rho = \frac{\lambda}{\mu}, \quad n = 0, 1, 2, \ldots, s.$$ 

The probability of blocking $BP$ is:

$$BP = \lim_{t \to \infty} P\{Q(t) = s\} = \frac{\rho^s/s!}{\sum_{i=0}^{\infty} \rho^i/i!},$$

and the average number of customers in the system (i.e. average number of busy servers) is:

$$\lim_{t \to \infty} E[Q(t)] = E[Q] = (1 - BP)\rho.$$

In the PSA method, the time dependent-steady state distribution of the nonstationary Erlang loss system, given that the arrival rate is $\lambda(t)$ and service rate is $\mu$, is calculated as follows:

$$P_n(t) = \frac{\rho(t)^n/n!}{\sum_{i=0}^{\infty} \rho(t)^i/i!}, \quad \rho(t) = \frac{\lambda(t)}{\mu} \quad \text{and} \quad n = 0, 1, 2, \ldots, s.$$ 

The time-dependent steady-state blocking probability is

$$BP(t) = P_s(t) = \frac{\rho(t)^s/s!}{\sum_{i=0}^{\infty} \rho(t)^i/i!},$$
and the time-dependent steady-state average number of customers in the system is:

\[ E[Q(t)] = (1 - BP(t)) \rho(t). \]

The PSA method can be easily generalized to most of the queueing systems. To guarantee stability, the formulae for most of the stationary systems are valid when \( \rho < 1 \). Similarly, one has to make sure that \( \rho(t) < 1 \) for all \( t \) when applying the PSA. Thus, PSA is valid for nonstationary systems where \( \max \{ \rho(t) : t > 0 \} < 1 \) whenever \( \rho < 1 \) is required for stability in stationary systems. An important factor that affects the accuracy of the PSA is the arrival rate function. The PSA method will provide a good approximation as the arrival rate increases. For example, consider the following two nonstationary Erlang loss systems:

- System 1: \( \lambda(t) = 5 + 2.5 \sin(t) \), \( \mu = 0.5 \), \( s = 10 \),
- System 2: \( \lambda(t) = 20 + 10 \sin(t) \), \( \mu = 2 \), \( s = 10 \),

According to PSA, the time-dependent offered load for both systems is

\[ \rho(t) = \frac{\lambda(t)}{\mu} = 10 + 5 \sin(t). \]

Although both systems have the same offered load, PSA will provide better approximation for system 2. System 2 needs shorter time to reach the steady state since it has higher arrival and service rates. On the other hand, system 1 needs longer time to achieve the steady state due to the low rates. This means that system 2 will behave more like a stationary system within reasonably small interval of time than system 1. In Figure 2.4, we plot the exact time-dependent average number \( E[Q(t)] \) for system 1, system 2 and the PSA values as a function of time \( t \). As can be seen, PSA provides better approximation for system 2 than for system 1.
The PSA method uses the instantaneous offered load to estimate the dependency of the performance measures of the nonstationary system on time. As shown in Figure 2.4, PSA overestimates the peak of some performance measures. In addition, the peak of PSA lags the peak of exact performance measures. However, these two problems become negligible as the arrival and service rates increase, as shown in Figure 2.4. In fact, Whitt [38] mathematically proved that PSA for the M(t)/M(t)/s queues is asymptotically correct as the rates increase.

In practice, peak period measures are of great concern to researchers and designers. Green and Kolesar [12] proposed a technique based on PSA to compute the average performance measures during peak period for periodic systems, which they called Simple Peak Hour Approximation (SPHA). SPHA starts by obtaining the measure of interest (say $X(t)$) using the PSA method. Next, the peak time $t^*$ at which the $X(t)$ achieves its maximum is determined. The average of $X(t)$ over the interval $[a, b]$ where $t^*$ is the center of the interval is the SPHA for $X(t)$.

In practice, many applications do not experience high arrival or service rates. In that case, the lag between the actual peak time and PSA peak time becomes large. In this
case, PSA will not provide good estimates for the peak period. Figure 2.5 shows the affect of using the instantaneous arrival rate to estimate the performance measures. In Figure 2.5, the peak time of PSA is exactly the same for $\lambda(t)$ but it lags the peak time for the actual measure. To avoid that, Green and Kolesar [11] provided an approximation for the lag in PSA using the time-dependent offered load of the $M(t)/M/\infty$ model in order to improve the accuracy of PSA during the peak period.

![Figure 2.5: Exact and PSA approximation of average customers in the system for an Erlang loss system with $\lambda(t)=20+15 \sin(2t)$, $\mu = 2$ and $s = 15$](image)

SSA and PSA can be seen as two extreme cases of averaging the arrival rate. The SSA averages the arrival rate $\lambda(t)$ over a long period of time (equal to the cycle length $T$) which is then used as the mean arrival rate. On the other hand, the PSA uses the average arrival rate over an infinitesimally small interval that results using the instantaneous arrival rate $\lambda(t)$.

Massey and Whitt [24] and Abdalla and Boucherie [1] introduced PSA for the nonstationary loss network. The PSA method for loss networks uses the same system of algebraic traffic equations used in the SSA method with arrival rates taken to be
functions of time instead of averages. As a result, the offered load functions, $\rho_j(t)$ for all $j$, obtained by PSA are simply the offered load functions obtained by SSA with each $\lambda_j$ is substituted by $\lambda_j(t)$. In other words, the time-dependent offered loads $\rho_j(t)$ are obtained by solving the following system of traffic equations in time $t$:

$$0 = \lambda_j(t) + \sum_{i=1}^{N} \mu_i q_{ij} \rho_i(t) - \mu_j \rho_j(t); \quad j = 1,2,...,N$$

(2.10)

Then, the probability, $P(n; t)$, of having $n$ in the system, for $n \in S = \{ n \in \mathbb{N}^N : 0 \leq n_i \leq s_i ; i = 1,2,...,N \}$, is

$$P(n; t) = \prod_{i=1}^{N} \frac{\rho_i(t)^{n_i}}{n_i!} / \sum_{n \in S} \prod_{i=1}^{N} \frac{\rho_i(t)^{n_i}}{n_i!}$$

(2.11)

### 2.6 The Pointwise Stationary Fluid Flow Approximation (PSFFA)

This method combines fluid flow differential equations and the PSA method into a single nonlinear differential equation which is solved numerically. This approximation was presented by Wang et al [36].

Consider a single server queue $M(t)/M/1$ with a nonstationary arrival process with rate $\lambda(t)$ and a service rate $\mu$. Let $x(t)$ be the state variable representing the average number of customers in the system at time $t$. From the flow conservation principle, the rate of change in $x(t)$ at time $t$ is equal to the difference between the average rate in the system at time $t$, $f_{in}(t)$, and the average rate out of the system at time $t$, $f_{out}(t)$. Therefore, we have:

$$\frac{d}{dt} x(t) = f_{in}(t) - f_{out}(t).$$

(2.12)
The average rate into the M(t)/M/1 queue $f_{in}(t)$ is basically $\lambda(t)$ since the system has infinite capacity. The departure rate $f_{out}(t)$ is proportional to the utilization of the system $\rho(t)$ at time $t$. Therefore, the fluid flow equation (2.12) becomes:

$$\frac{d}{dt} x(t) = \lambda(t) - \mu \rho(t). \quad (2.13)$$

An approximation for the utilization $\rho(t)$ is required to complete equation (2.13). The PSA method uses the results of the stationary M/M/1 system to approximate the performance measures for the nonstationary M/M/1 system. According to PSA, the average number in the system for M(t)/M/1 system is:

$$x(t) = \frac{\rho(t)}{1 - \rho(t)} = G(\rho(t)), \rho(t) < 1. \quad (2.14)$$

Equation (1.14) could be inverted to be a function of $x(t)$ to obtain $\rho(t)$ as follows:

$$\rho(t) = \frac{x(t)}{1 + x(t)} = G^{-1}(x(t)).$$

Then, the utilization function is substituted in the fluid flow model (2.13) to obtain a nonlinear differential equation (2.15) which can be solved numerically with a given initial conditions.

$$\frac{d}{dt} x(t) = \lambda(t) - \mu G^{-1}(x(t)) = \lambda(t) - \mu \frac{x(t)}{1 + x(t)}. \quad (2.15)$$

This method does not require a closed-form expression for $G^{-1}(x(t))$. A numerical graph showing the relation between $\rho(t)$ and $x(t)$ is sufficient to apply the PSFFA method in the absence of closed-form expression for $G^{-1}(x(t))$. PSFFA formulas of the average number of customers in the system for M(t)/D/1, M(t)/E_k/1, M(t)/G/1, GI(t)/M/1 and GI(t)/G/1 systems were obtained explicitly by Wang et al [36].
The authors also considered a single server queue $M(t)/M/1$ with finite capacity $k$. The average rate into the system $f_{in}(t)$, in this case, depends on the probability that the system is full $P_i(t)$. On the other hand, the departure rate $f_{out}(t)$ depends on the probability that the system is idle $P_0(t)$. Therefore, the fluid flow model for this system becomes:

$$\frac{d}{dt} x(t) = \lambda(t) (1 - P_k(t)) - \mu (1 - P_0(t)).$$

(2.16)

Assume that the following functional relationships can be determined for the stationary $M/M/1/k$ system:

$$x = G(\rho), P_0 = G_1(\rho), P_k = G_2(\rho).$$

Then, using PSA, the corresponding functional relationships for the nonstationary $M(t)/M/1/k$ system are:

$$x(t) = G(\rho(t)), P_0(t) = G_1(\rho(t)), P_k(t) = G_2(\rho(t)).$$

Assuming that $G_1(\rho(t))$ is numerically invertible, $\rho(t) = G^{-1}(x(t))$. Thus, $P_0(t)$ and $P_k(t)$ can re-written as functions of $x(t)$ as follows:

$$\rho(t) = G^{-1}(x(t)), P_0(t) = G_1(G^{-1}(x(t))), P_k(t) = G_2(G^{-1}(x(t))).$$

The above transformation provides the following single fluid flow model (differential equation) for the finite capacity system which can be solved numerically:

$$\frac{d}{dt} x(t) = \lambda(t) [1 - G_2(G^{-1}(x(t)))] - \mu [1 - G_1(G^{-1}(x(t)))].$$

Tipper and Sundarershan [34] used this method to model computer networks under nonstationary conditions. The computer network they consider in their paper consists of a number of nodes each node is a single server queue with finite buffer.
Arrivals to any node in the network are packets of fixed length. Their main focus was obtaining the time-dependent average number of packets at each node.

### 2.7 The Modified Offered Load Approximation (MOL)

Let us first consider the stationary Erlang loss queue and the stationary infinite server queue. Let $Q_\infty(t)$ be number of customers in the infinite server queue at time $t$. The probability $P_n$ that there are $n$ customers in an M/M/$\infty$ with an arrival rate $\lambda$ and a service rate $\mu$ can be obtained by solving the steady-state equations. We have:

$$P_n = \lim_{t \to \infty} P\{Q_\infty(t) = n\} = \frac{\rho^n}{n!} e^{-\rho}, \text{ where } \rho = \frac{\lambda}{\mu} \text{ and } n = 0, 1, 2, \ldots.$$

Likewise, let $Q(t)$ be number of customers in the Erlang loss queue with $s$ servers at time $t$. The probability $P_n$ that there are $n$ customers in an M/M/$s/0$ with an arrival rate $\lambda$ and a service rate $\mu$, is:

$$P_n = \lim_{t \to \infty} P\{Q(t) = n\} = \frac{\rho^n/n!}{\sum_{i=0}^{s} \rho^i/i!},$$

where $\rho = \frac{\lambda}{\mu}$ and $n = 0, 1, 2, \ldots, s$.

Another way to obtain the stationary distribution of the M/M/$s/0$ queue is to use the fact that the M/M/$s/0$ queue is a truncated process of an M/M/$\infty$ queue which is a reversible Markov process. Since the arrival rates are time invariant we dropped the time variable from the random variables. Then, we have:

$$P\{Q = n\} = P\{Q_\infty = n \mid Q_\infty < s\} = \frac{e^{-\rho} (\rho^n/n!)}{\sum_{i=0}^{s} \rho^i/i!} = \frac{\rho^n/n!}{\sum_{i=0}^{s} \rho^i/i!}.$$
Palm [30] and Khinchine [21] independently showed in 1943 and 1955 that the steady-state probability distribution for the \( M(t)/G/\infty \) queue is still Poisson as in the stationary \( M/M/\infty \) queue. The only difference between the two solutions is the offered load \( \rho(t) \). Eick et al [6] provide a comprehensive study for the properties of \( M(t)/G/\infty \) queue. In addition, they introduced the following theorem to express the average of busy servers in terms of a new random variable, \( S_e \), called the \textit{stationary-excess} or \textit{equilibrium-residual-lifetime} of a stochastic process which has the following cdf:

\[
P(S_e \leq t) = \frac{1}{E[S]} \int_0^t (1 - G(u)) \, du
\]

where \( S \) is a generic service-time random variable with cumulative \( G \).

**Theorem**

For each \( t \), the number of customers \( Q_\infty(t) \) in an \( M(t)/G/\infty \) has a Poisson distribution with mean

\[
E[Q_\infty(t)] = E\left[ \int_{S}^{t} \lambda(u) \, du \right] = E[\lambda(t - S_e)] E[S].
\]

The departure process is a Poisson process with time dependent rate \( \delta(t) = E[\lambda(t - S)] \).

For each \( t \), \( Q_\infty(t) \) is independent of the departure process in the interval \( (-\infty, t] \).

This nice expression explicitly shows that in time-dependent \( M(t)/G/\infty \) queue, the average number of customers depends on the service distribution and not only on its mean as in the case of \( M/G/\infty \). In other words, the insensitivity property that the \( M/G/\infty \) possesses is lost in the case of nonstationary arrivals and/or service rates.

In the \( M(t)/M/\infty \) queue, the rate of change in the average number of customers at time \( t \) is equal to the difference between the arrival rate and the departure rate due to the Markovian property. Then,

\[
\frac{d}{dt} E[Q_\infty(t)] = \lambda(t) - \mu E[Q_\infty(t)].
\]
Recall that there is always an idle server for each arriving customer to the M(t)/M/∞ queue. This means that no customers are lost and all customers in the system at time t are being served. Therefore, the average number of customers at time t is equal to the average number of customers in the system at time t which equals to the offered load ρ(t). We have the following differential equation for ρ(t):

\[
\frac{d}{dt} \rho(t) = \lambda(t) - \mu \rho(t)
\] (2.17)

Analogous to the stationary queues, one can approximate the M(t)/M/s/0 by truncating the M(t)/M/∞ queue. This method is called the modified offered load method (MOL). The MOL approximation was first developed by Jagerman [16] in 1975. The probability P_n(t) that there are n customers in the system using MOL is:

\[
P_n(t) \approx P\{Q_∞(t) = n \mid Q_∞(t) < s\} = \frac{\rho(t)^n / n!}{\sum_{j=0}^{s} \rho(t)^j / j!}, \quad n = 0, 1, 2, \ldots, s.
\]

The truncated M/M/∞ queue provides an exact solution to the M/M/s/0 queue due to the reversibility property. In the case of nonstationary arrival process, the reversibility property is lost and hence the truncated M(t)/M/∞ will not provide an exact solution to the M(t)/M/s/0. Massy and Whitt [23] developed analytical bounds on the error between the MOL approximation and the exact solution of the M(t)/M/s/0 system.

**Theorem**

If λ(t) is differentiable and bounded on [0,∞), its derivative λ′(t) is also bounded on [0, ∞), μ = 1 and E[Q(0)] = E[Q_∞(0)] = λ(0), then:

\[
\sup_{t \geq 0} \left| E[Q(t)] - \rho(t)(1 - B(s, \rho(t))) \right| \leq \left| \lambda(t) \right|_∞ \left| \lambda'(t) \right|_∞ B(s - 1, \left| \lambda(t) \right|_∞),
\]

where \(|\lambda(t)|_∞ = \sup_{t \geq 0} |\lambda(t)|\) and ρ′(t) = λ(t) − μ ρ(t).
The $\text{M}(t)/\text{M}/s/0$ behaves like $\text{M}(t)/\text{M}/\infty$ as the blocking probability gets smaller. In view of this, the MOL method provides a good approximation for the $\text{M}(t)/\text{M}/s/0$ as long as the system has a small blocking probability. Experiments show that the actual blocking probability of the $\text{M}(t)/\text{M}/s/0$ queue should be less than 0.1 in order for the MOL to provide a good approximation. As expected, the MOL underestimates the blocking probability of loss systems with high load (i.e. when the exact blocking probability is high).

Comparing MOL with PSA and SSA, the MOL method is the intermediate approximation between SSA and PSA. The MOL method averages the arrival rate over an interval that depends on the service time mean and distribution. This can explicitly be seen from the average the MOL for the $\text{M}(t)/\text{G}/\infty$ system. Unlike the SSA that averages $\lambda(t)$ over very long interval or the PSA that averages $\lambda(t)$ over an infinitesimally small interval.

The MOL method will provide a good estimation for the peak time congestion for queues with small blocking probabilities. This is due to the fact that the MOL method is sensitive to the service process through its mean and distribution. The PSA method depends on the service distribution only through its mean. As a result, PSA appears to lag the actual performance measures values of the system. Figure 2.6 shows the exact peak time and PSA and MOL peaks for the same $\text{M}(t)/\text{M}/s/0$ queue plotted in Figure 2.5 in section 2.5. Obviously the difference in peak time using MOL is very small compared to PAS.

This method appeared in literature as a way to approximate the loss networks in different contexts. Grier et al [14] proposed using the MOL method to approximate the blocking probability of an Erlang loss model where blocked customers have a probability to reenter the system for retry. In [14], they modeled the Erlang loss model with retrials as a two queue model. In this model, blocked customers, who decide to retry, go to an infinite server queue where they got delayed for some time and, then, they go back to the Erlang loss queue to get the service, if possible. As a generalization for Grier et al [14] work, Whitt [40] described the use of MOL in a decomposition
algorithm for a general time-dependent loss networks. In addition, Massey and Whitt [24] extensively studied the time-dependent infinite-server networks and presented the MOL approximation for the time dependent Erlang loss networks described in section 2.1.

The MOL method also has been studied in the context of communication networks. For example, Jennings and Massey [17] used MOL to approximate the performance of the time-dependent circuit-switched networks. Another example, Abdalla and Boucherie [1] applied MOL on a mobile communication model with time-varying rates and redialing. In fact, Abdalla and Boucherie [1] established an exact expression for the error in the MOL approximation as well as bounds on the error.

The main idea of the MOL approximation for Erlang loss networks is to solve the traffic equations of the infinite server network to obtain time-dependent offered loads and hence apply them in the stationary results. Therefore, the MOL method for Erlang loss networks uses the system of traffic equations (2.10) described in the PSA method.
with the exception that \( \frac{d}{dt} \rho_i(t), i=1,2,...,N \) is not taken to be zero in the right hand side of the equations. As a result, the system of offered load functions (2.10) is a system of ordinary differential equations of \( \rho_i(t) \) for all \( i \). In other words, the time dependent offered loads \( \rho_j(t) \) are obtained by solving the following system of differential equations in time \( t \):

\[
\frac{d}{dt} \rho_j(t) = \lambda_j(t) + \sum_{i=1}^{N} \mu_i q_{ij} \rho_i(t) - \mu_j \rho_j(t); \quad j = 1,2,...,N
\]

Then, for \( n \in S = \{ n \in \mathbb{N}^N; 0 \leq n_i \leq s_i; i = 1,2,...,N \} \)

\[
P(n; t) = \prod_{i=1}^{N} \frac{\rho_i(t)^{n_i}}{n_i!} / \sum_{n \in S} \prod_{i=1}^{N} \frac{\rho_i(t)^{n_i}}{n_i!}.
\]

It is worth mentioning that the \( M(t)/M/s/0 \) behaves like \( M(t)/M/\infty \) as the blocking probability gets smaller. Then, it is expected that the MOL method will provide a good approximation for the nodes in the system that has a small blocking probability.
Chapter 3

Nonstationary Erlang Loss Queues

3.1 The Mathematical Model

Consider a stream of arrivals that arrive to an Erlang loss model (M(t)/M/s/0) according to a Poisson process with time-dependent rate $\lambda(t)$. Each arrival requests a single service that requires an exponential amount of time with mean $\mu^{-1}$. The service requested by any arriving customer is performed by a single server. The system has $s$ identical servers. Any arriving customer finds all $s$ servers in the system busy will be lost. In other words, the waiting space in the system is zero. The rates at which the state of the system changes are depicted in Figure 3.1.

Figure 3.1: Rate diagram for M(t)/M/s/0
Since the arrival and the service processes are Poisson, the probability of \( n (n = 0, 1, \ldots, s) \) customers in the system at time \( t \), \( P_n(t) \), is represented by the following set of forward differential equations:

\[
\begin{align*}
\frac{d}{dt} P_0(t) &= \mu P_1(t) - \lambda(t) P_0(t), \\
\frac{d}{dt} P_n(t) &= \lambda(t) P_{n-1}(t) + (n+1) \mu P_{n+1}(t) - (\lambda(t) + n \mu) P_n(t), \quad 0 < n < s, \\
\frac{d}{dt} P_s(t) &= \lambda(t) P_{s-1}(t) - s \mu P_s(t),
\end{align*}
\]

(3.1)

where

\[
P_0(t) + P_1(t) + P_2(t) + \ldots + P_s(t) = 1, \quad t \geq 0
\]

and

\[
0 \leq P_n(t) \leq 1 \text{ for } t \geq 0 \text{ and } n = 0, 1, 2, \ldots, s
\]

with initial conditions: \( P_0(0) = 1 \) and \( P_n(0) = 0 \); \( n = 1, 2, 3, \ldots, s \).

A special case of arrival rate is when \( \lambda(t) = \lambda \) for all \( t \). As a result, \( \frac{d}{dt}[P_n(t)] \rightarrow 0 \) for all \( n \) as \( t \rightarrow \infty \). The above system of differential equations reduces to a set of linear equations that provides the following familiar solution:

\[
P_n(t) = \left( \frac{\lambda}{\mu} \right)^n / n! \sum_{i=0}^{s} \binom{n}{i} \left( \frac{\lambda}{\mu} \right)^i / i!, \quad n = 0, 1, 2, \ldots, s
\]

Note that the expression for \( P_n(t) \) is independent of time, for sufficiently large \( t \), due to the stationary arrival process. In the case of nonstationary arrival rate, \( \frac{d}{dt}[P_n(t)] \) will converge to some value which is not necessarily zero at all time. In fact, \( \frac{d}{dt}[P_n(t)] \) will converge to some function of time depending on the structure of \( \lambda(t) \). Therefore,
there is no closed-form solution to obtain the time-dependent stationary probability distribution for the model for any \( \lambda(t) \).

The solution of the above forward differential equations is extremely complex even for fairly small systems with very special rate functions \( \lambda(t) \). Even with stationary arrival rates, it is not easy to obtain a closed-form solution for the probability distribution of number of customers in the system during its transient period (warm-up period). To demonstrate the complexity of analyzing a nonstationary Erlang loss model, we consider the simplest case where there is only one server in the system. Let \( \lambda(t) \) and \( \mu \) be the arrival and service rates, respectively. Then, the forward equations are:

\[
\frac{d}{dt} P_0(t) = \mu P_1(t) - \lambda(t) P_0(t),
\]

and

\[
\frac{d}{dt} P_1(t) = \lambda(t) P_0(t) - \mu P_1(t),
\]

where

\[
P_0(0) = 1 \text{ and } P_0(t) + P_1(t) = 1, \quad t \geq 0
\]

This system can be reduced to solving a single differential equation:

\[
\frac{d}{dt} P_1(t) + [\lambda(t) + \mu] P_1(t) = \lambda(t), \quad \text{with } P_1(0) = 0.
\]

Multiplying both sides by \( e^{\int_{\eta}^{\lambda(t)+\mu} d\eta} \)

\[
\frac{d}{dt} P_1(t) e^{\int_{\eta}^{\lambda(t)+\mu} d\eta} + [\lambda(t) + \mu] e^{\int_{\eta}^{\lambda(t)+\mu} d\eta} P_1(t) = \lambda(t) e^{\int_{\eta}^{\lambda(t)+\mu} d\eta}
\]

or

\[
31
\]
Given that \( P_1(0) = 0 \), we finally have the blocking probability, \( BP(t) \):

\[
BP(t) = P_1(t) = \int_0^t \lambda(u) e^{\int_0^u \lambda(\eta) + \mu \, d\eta} \, dt 
\]

One can imagine how complicated the solution is going to get as the number of servers increases. Due to this difficulty in the nonstationary system, approximation methods are employed to obtain the steady-state distribution. Below, we propose a new approximation method for obtaining the blocking probability without the need for solving the forward differential equation.

### 3.2 The Fixed Point Approximation (FPA)

The time-dependent average number of customers in the \( M(t)/M/s/0 \) queue can be expressed as the difference between the effective arrival rate and the departure rate at time \( t \), as shown in Figure 3.2.

Thus, the average number of customers \( E[Q(t)] \) is given by the following differential equation:

\[
\frac{d}{dt} E[Q(t)] = \lambda(t) (1 - BP(t)) - [\mu P[Q(t)=1] + 2\mu P[Q(t)=2] + \ldots + s\mu P[Q(t)=s] ].
\]
or
\[
\frac{d}{dt} E[Q(t)] = \lambda(t) (1 - BP(t)) - \mu E[Q(t)]
\] (3.2)

Obtaining the exact solution of \(E[Q(t)]\) using the differential equation (3.2) is extremely difficult since it involves the evaluation of the time-dependent blocking probability. However, this blocking probability can be approximated iteratively. To illustrate the iterative scheme, consider an \(M(t)/M/s/0\) queue with a given blocking probability \(BP(t), 0 \leq BP(t) \leq 1\) for \(t > 0\), and average busy servers \(E[Q(t)]\). From queueing theory, we have the following relation:

\[
E[Q(t)] = \rho(t) (1 - BP(t)).
\]

Then the offered load to the \(M(t)/M/s/0\) queue is

\[
\rho(t) = \frac{E[Q(t)]}{1 - BP(t)}.
\] (3.3)

If we know approximate functions for \(E[Q(t)]\) and \(BP(t)\), then we can calculate a new function for the offered load, say \(\rho^s(t)\), using equation (3.3). Subsequently, an improved blocking probability function is available by substituting \(\rho^s(t)\) in the expression:

\[
BP(t) = \frac{\rho^s(t)^s/s!}{\sum_{i=0}^{s} \rho^s(t)^i/i!}.
\]
Recall, from section 2.7, that $\rho(t)$ in the MOL method is taken to be $\rho(t) = E[Q(t)]$. Since the MOL method ignores the blocking effect in the system, it underestimates the offered load of the Erlang loss queue. Therefore, the blocking probability obtained by MOL also underestimates the actual blocking probability knowing that $BP(t)$ is monotonically increasing function of $\rho(t)$ with fixed $s$.

In our proposed method we start with an initial value for $BP(t)$ and we then improve it iteratively. The simplest starting choice for $BP(t)$ is the zero blocking probability. Let $BP^0(t)$ be the initial value of the blocking probability. The resulting initial differential equation for $E[Q(t)]$ will reduce to,

$$\frac{d}{dt} E[Q^0(t)] = \lambda(t) - \mu E[Q^0(t)].$$

(3.4)

Equation (3.4) is a basic linear differential equation which can be solved numerically. There are many ways to get a numerical solution for this type of differential equations (see Burden and Faires [2]). We choose to use Euler’s method in our algorithm since it is fast in computations. According to Euler’s method, assume that

$$\frac{d}{dt} E[Q^0(t)] \approx \frac{\Delta}{\Delta t} E[Q^0(t)],$$

where $\Delta t$ is sufficiently small.

$$\Rightarrow \quad \frac{\Delta}{\Delta t} E[Q^0(t)] = \frac{1}{\Delta t} \left( E[Q^0(t + \Delta t)] + E[Q^0(t)] \right)$$

or

$$\frac{1}{\Delta t} \left( E[Q^0(t + \Delta t)] + E[Q^0(t)] \right) = \lambda(t) - \mu E[Q^0(t)]$$

$$E[Q^0(t + \Delta t)] = E[Q^0(t)] + \lambda(t) \Delta t - \mu E[Q^0(t)] \Delta t$$

with the initial choice of $BP^0(t) = 0 \, \forall \, t$, the offered load is $\rho(t)^0 = E[Q^0(t)]$.  

34
Then, a new time-dependent blocking probability $BP^i(t)$ is obtained by the Erlang loss formula with parameter $\rho^0(t)$ and $s$ as follows:

$$BP^i(t) = \frac{[\rho^0(t)]^i / s!}{\sum_{j=0}^{\infty} [\rho^0(t)]^j / j!}.$$ 

$BP^i(t)$ is the first approximation for the time dependent blocking probability of the $M(t)/M/s/0$ queue. Next, $BP^i(t)$ is used in the differential equation (3.2) to obtain the next approximation for the time-dependent average number of busy serves in the $M(t)/M/s/0$ queue, $E[Q^i(t)]$:

$$\frac{d}{dt} E[Q^i(t)] = \lambda(t) (1-BP^i(t)) - \mu E[Q^i(t)].$$

Given the solution $E[Q^i(t)]$, the new offered load of the loss model is

$$\rho^i(t) = \frac{E[Q^i(t)]}{1 - BP^i(t)}$$

The algorithm continues in this manner until the sequence \{BP^k(t): k = 0,1,2, \ldots\} converges. Below are the steps of the algorithm.

**FPA Algorithm**

1. Choose an appropriate $\Delta t$, final time $T_f$ and tolerance $\varepsilon$.
2. Choose initial conditions (i.e. at $t = 0$) for $E[Q(t)]$. Set $E[Q(0)] = 0$.
3. Evaluate $\lambda(t)$ at $t = 0, \Delta t, 2\Delta t, \ldots, T_f$.
4. Start with an initial blocking probability $BP^0(t) = 0$, $t = 0, \Delta t, 2\Delta t, \ldots, T_f$.
5. Set the iteration counter $k = 0$. 

35
6. Solve (3.1) numerically for \( E[Q^k(t)] \) using the following equation:

\[
E[Q^k(t+\Delta t)] = E[Q^k(t)] + \lambda(t) (1-BP^k(t)) \Delta t - \mu E[Q^k(t)] \Delta t.
\]

7. Calculate

\[
\rho^k(t) = \frac{E[Q^k(t)]}{1 - BP^k(t)} , \quad t = 0, \Delta t, 2\Delta t, \ldots , T_f.
\]

8. Update blocking probability

\[
BP^{k+1}(t) = \frac{[\rho^k(t)]^t/s!}{\sum_{i=0}^{\infty} [\rho^k(t)]^i/i!} , \quad t = 0, \Delta t, 2\Delta t, \ldots , T_f.
\]

9. If \( \|BP^k(t) - BP^{k+1}(t)\| < \varepsilon \), then \( BP^k(t) \) converged. STOP.

Else, update the iteration counter \( k = k + 1 \). Go to step 6.

This algorithm takes the arrival rate function as a function defined at certain points on time equally spaced by \( \Delta t \). One of the advantages of this algorithm is that it can consider periodic arrival rate function regardless whether we know its closed-form or not. The algorithm can be easily extended to systems with time-dependent arrival and service rates \( M(t)/M(t)/s/0 \). This extension to the \( M(t)/M(t)/s/0 \) queue is done simply by defining the service rate as a vector corresponding to the same time scale used for the arrival rate. Since this algorithm discretizes the arrival rate function, the continuity and differentiability properties of the arrival rate function are not necessary.

The blocking probability obtained by FPA appears to be exact. We have not been able to show this theoretically, but numerical validations can prove this conclusion. Consequently, the only source of error in this method is due to the discetization method. We expect that the error diminutions as \( \Delta t \to 0 \).
3.3 The Erlang Loss Model with Sinusoidal Arrival Rate

In this section, we use the above proposed algorithm to analyze the $M(t)/M/s/0$ queue assuming a sinusoidal arrival rate function. Consider the following sinusoidal function for the arrival rate:

$$\lambda(t) = \bar{\lambda} + \beta \sin(\gamma t)$$

where $\bar{\lambda}$ is the average arrival rate, $\beta$ is the amplitude and $\gamma$ is the frequency. The frequency $\gamma$ can, equivalently, be taken to be $2\pi/T$, where $T$ is the cycle length.

The sinusoidal arrival rate has been studied extensively in the context of nonstationary queues. For details, see Eick et al [5,6], Green et al [10,11,12,13], Jennings and Massey [13]. This type of arrival rate is of a great interest since it is a typical representation for a periodic nonstationary arrival rate in practice.

One of the properties of the sinusoidal arrival rate $\lambda(t)$ is that it is continuous and differentiable over all $t$. This property made it possible to obtain a closed-form solution for some nonstationary queues. Eick et al [5] provided an explicit formula for the average number of customers in an $M(t)/M/\infty$, $M(t)/D/\infty$, $M(t)/H_k/\infty$ systems with normalized service rate ($\mu = 1$). Another interesting property of the sinusoidal arrival rate function is that it is possible to build an equivalency between some nonstationary queues with sinusoidal arrival rates by changing the parameters of $\lambda(t)$. For example, consider a system with service rate $\mu$ and arrival rate parameters $\bar{\lambda}$, $\beta$, and a one-day cycle. Now, let us consider the same system with arrival rate doubled during the day (i.e. $\bar{\lambda} \rightarrow 2\bar{\lambda}$, $\beta \rightarrow 2\beta$). The new system is equivalent to a system with the same original service rate and arrival rate parameters $\bar{\lambda}$, $\beta$, and a half-a-day cycle (i.e. $T \rightarrow T/2$). This equivalency means that both systems have the same blocking probability.
The blocking probability of an Erlang loss model depends on the following parameters: average arrival rate, degree of variability in the arrival rate (amplitude), and number of servers. We tested the accuracy of the FPA by varying the three parameters and compared the results with the system’s exact solution. In addition, the FPA is compared with the MOL method and the PSA.

It is important to note that the error of each method is a function of time. This is due to the fact that the system is not stationary and, hence, a single value for blocking or any performance measure will never been reached as time goes to infinity. Since the error is a function of time, it is possible to represent it on a graph in order to visualize the accuracy of the approximation. Another way to measure the accuracy is by using the average norm-one function which is average absolute error to obtain a single value for the error of each method. Then the error is defined as:

\[
\text{Average Error } (t) = \frac{\| X(t)_{\text{app.}} - X(t)_{\text{exact}} \|}{D}
\]

where \( X(t)_{\text{app.}} \) and \( X(t)_{\text{exact}} \) are real vectors of dimension \( D \) and \( \| \cdot \|_1 \) is defined as:

\[
\| x \|_1 = | x_1 | + | x_2 | + \cdots + | x_D | \quad \text{for} \quad x \in \mathbb{R}^D.
\]

Analogous to the relative error in the scalar case, the relative error in the vector case is defined as follows:

\[
\text{Relative Error } (t) = \frac{\| X(t)_{\text{app.}} - X(t)_{\text{exact}} \|}{\| X(t)_{\text{exact}} \|_1} \times 100
\]

The exact solution for the performance measure of interest \( X(t) \) is obtained by solving the set of forward differential equations (3.1) numerically. This is being done by choosing an appropriate choice of \( \Delta t \) and solving system (3.1) simultaneously.
3.3.1 Accuracy of FPA as $\Delta t \to \infty$

In this section, we provide a numerical example to demonstrate the effect of decreasing $\Delta t$ on the accuracy of FPA. We consider an Erlang loss queue with $s = 10$, arrival rate $\lambda(t)$, a sinusoidal arrival rate function with parameters $\bar{\lambda} = 20$, $\beta = 15$ and $\gamma = 2$, and service rate $\mu = 1$. The FPA method is applied with tolerance $\varepsilon = 0.01$ and different values of $\Delta t$. The average absolute between the exact and the FPA solutions for the blocking probability and the average number of customers are given in Table 3.1

<table>
<thead>
<tr>
<th>$\Delta t$ (s)</th>
<th>$BP(t)$ (0.0269)</th>
<th>$E[Q(t)]$ (0.0992)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.0269</td>
<td>0.0992</td>
</tr>
<tr>
<td>0.05</td>
<td>0.0165</td>
<td>0.0692</td>
</tr>
<tr>
<td>0.01</td>
<td>0.0165</td>
<td>0.0692</td>
</tr>
<tr>
<td>0.005</td>
<td>0.0118</td>
<td>0.0507</td>
</tr>
<tr>
<td>0.001</td>
<td>0.0115</td>
<td>0.0474</td>
</tr>
<tr>
<td>0.0005</td>
<td>0.0114</td>
<td>0.0447</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.0114</td>
<td>0.0445</td>
</tr>
</tbody>
</table>

Table 3.1 shows that the absolute error decrease as $\Delta t$ decreases. Unfortunately, we were unable to take $\Delta t$ less than 0.0001 due to the CPU and memory limitations.

3.3.2 Accuracy of FPA with Respect to the Number of Servers

Consider an Erlang loss queue with a sinusoidal arrival rate function with parameters $\bar{\lambda} = 10$, $\beta = 5$ and $\gamma = 2$. This queue is solved using four values of identical servers in the system $s = 5, 10, 15, 20$ each with service rate $\mu = 1$. The FPA method is applied with time intervals $\Delta t = 0.001$ and tolerance $\varepsilon = 0.01$. The error and relative error between the exact and the FPA solutions for the blocking probability and the average number of customers is given in Table 3.2. In addition, the error and relative error of the MOL method and the
PSA method for the same performance measures are listed in the following table to compare FPA with the other methods.

Table 3.2 shows that the error of FPA for blocking probability decreases as number of serves increases. On the other hand, the relative error of FPA for the blocking probability increases as the number of servers increases. The reason for that is that the blocking probability gets closer to zero as the number of servers goes up and small absolute differences give rise to large relative error.

<table>
<thead>
<tr>
<th>s</th>
<th>Error</th>
<th>Rel. Error</th>
<th>Error</th>
<th>Rel. Error</th>
<th>Error</th>
<th>Rel. Error</th>
<th>Error</th>
<th>Rel. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.006</td>
<td>1.19</td>
<td>0.019</td>
<td>0.45</td>
<td>0.094</td>
<td>18.94</td>
<td>0.525</td>
<td>12.84</td>
</tr>
<tr>
<td>10</td>
<td>0.013</td>
<td>7.35</td>
<td>0.051</td>
<td>0.72</td>
<td>0.053</td>
<td>30.30</td>
<td>0.293</td>
<td>4.16</td>
</tr>
<tr>
<td>15</td>
<td>0.005</td>
<td>16.06</td>
<td>0.023</td>
<td>0.27</td>
<td>0.009</td>
<td>27.54</td>
<td>0.056</td>
<td>0.66</td>
</tr>
<tr>
<td>20</td>
<td>0.000</td>
<td>16.93</td>
<td>0.003</td>
<td>0.04</td>
<td>0.000</td>
<td>19.22</td>
<td>0.004</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Figures 3.3 to 3.6 demonstrate the accuracy of FPA compared to MOL and PSA for each value of $s$ over time. Figure 3.3(a) shows that the FPA has produced an excellent approximation for the exact blocking probability of the system with $s = 5$ compared to MOL and PSA in Figure 3.3(b). To see how close the approximation to the exact function is, Figure 3.3(c) and (d) plot the error of approximation of each method as a function of time for the blocking probability and average number of customers in the system respectively. Figure 3.3(c) and (d) show the superior accuracy of FPA over time by remaining very close to zero over time. MOL in Figure 2.3 seems to perform poorly due to the high blocking probability which ranges, approximately, between 0.3 and 0.7.
Figure 3.3: Results of Erlang loss system with $\lambda(t)=10+5\sin(2t)$, $\mu=1$, and $s=5$ : (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.

Figure 3.4 shows that FPA preserves its accuracy among the other methods in all plots. In Figure 3.4(b), MOL and PSA do not perform well in estimating the peak time and value of the blocking probability. Although MOL underestimates the blocking probability, it performs better than PSA in estimating the average number of customers in the system as shown in Figure 3.4(d).
Figure 3.4: Results of Erlang loss system with $\lambda(t)=10+5\sin(2t)$, $\mu=1$, and $s=10$: (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.

Needless to say that FPA provides the best results among the other methods in all plots of Figure 3.5. MOL provides its best performance in Figure 3.6 due to the relatively low blocking probability of the system with $s=20$ that makes the system nearly reversible. In contrary, PSA highly over estimate the peak blocking probability with large time lag which makes this is the worst performance of PSA. This, also, can be seen in the error plots in Figure 3.6(c) and (d).
In general, FPA provides the best results among the three methods in all the experiments carried out. MOL and PSA work better for some particular cases. For example, the accuracy of MOL gets better when the number of servers increases making the system behaves like an infinite queue. In contrast, the PSA method works better for loss systems with a small number of servers.
Figure 3.6: Results of Erlang loss system with $\lambda(t)=10+5\sin(2t)$, $\mu=1$, and $s=20$:
(a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.

In all cases of $s$ considered in this analysis, FPA and MOL both provide an approximate behavior of the blocking probability during the transition period. Due to the fact that PSA applies the steady-state results of the stationary system, PSA starts at the approximate steady-state value of blocking probability at time zero. This is, also, true for all later analyses in this section.
3.3.3 Accuracy of FPA with Respect to Average Arrival Rate

In this analysis, the same Erlang loss queue described earlier is analyzed with average arrival rates $\lambda = 5, 10, 15, 20$. The other parameters of the queue are held fixed, i.e. $\beta = 5$, $\gamma = 2$, $\mu = 1$ and $s = 10$. The FPA method is applied with time intervals $\Delta t = 0.001$ and tolerance $\varepsilon = 0.01$. The error and relative error between the exact solution and FPA solution for the blocking probability and the average number of customers is given in Table 3.3. In addition, the error and relative error of the MOL method and the PSA method for the same performance measures are also listed in Table 3.3.

Table 3.3: Error of FPA compared with error MOL and PSA as the average arrival rate changes

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>FPA</th>
<th>MOL</th>
<th>PSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$BP(t)$</td>
<td>$E[Q(t)]$</td>
<td>$BP(t)$</td>
</tr>
<tr>
<td>5</td>
<td>0.005</td>
<td>17.55</td>
<td>0.016</td>
</tr>
<tr>
<td>10</td>
<td>0.013</td>
<td>7.35</td>
<td>0.051</td>
</tr>
<tr>
<td>15</td>
<td>0.003</td>
<td>2.06</td>
<td>0.039</td>
</tr>
<tr>
<td>20</td>
<td>0.042</td>
<td>0.87</td>
<td>0.023</td>
</tr>
</tbody>
</table>

Obviously, increasing the average number of arrivals makes the blocking probability of the system increase. As we saw in the error analysis as the number of servers changes, the error of FPA decreases as the blocking probability increases. In view of this, the error of FPA for the blocking probability and average busy servers decreases as the average arrival rate increases. Plots of $BP(t)$ and $E[Q(t)]$ for each value of $\lambda$ are available in Appendix A.

Comparing the three methods, FPA still provides the best results in response to the change in the average arrival rate. The MOL method has better error than the PSA method. As expected, PSA works better for loss systems with high blocking probability.
3.3.4 Accuracy of FPA with Respect to Amplitude

In this analysis, the same Erlang loss queue described earlier is analyzed with amplitude $\beta = 2, 6, 8, 10$. The other parameters of the queue are held fixed to the values $\lambda = 10, \gamma = 2, \mu = 1$ and $s = 10$. The FPA method is applied with time intervals $\Delta t = 0.001$ and tolerance $\varepsilon = 0.01$. The errors and relative errors are listed in Table 3.4.

Table 3.4: Error of FPA compared with error of MOL and PSA as the amplitude of arrival rate changes

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>FPA</th>
<th>MOL</th>
<th>PSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$BP(t)$</td>
<td>$E[Q(t)]$</td>
<td>$BP(t)$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Error</td>
<td>Rel. Error</td>
<td>Error</td>
</tr>
<tr>
<td>2</td>
<td>0.061</td>
<td>3.57</td>
<td>0.029</td>
</tr>
<tr>
<td>6</td>
<td>0.015</td>
<td>8.29</td>
<td>0.055</td>
</tr>
<tr>
<td>8</td>
<td>0.017</td>
<td>9.35</td>
<td>0.060</td>
</tr>
<tr>
<td>10</td>
<td>0.018</td>
<td>9.33</td>
<td>0.058</td>
</tr>
</tbody>
</table>

The above three types of analyses can be summarized in Table 3.5 that shows the mean and standard deviation of the error and relative error for each method. Table 3.5 represents the average accuracy of each method when all three types of changes occur simultaneously. Plots of $BP(t)$ and $E[Q(t)]$ for each value of $\beta$ are in Appendix A.

Table 3.5: Mean and standard deviation of the error and relative error of each method

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>FPA</th>
<th>MOL</th>
<th>PSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$BP(t)$</td>
<td>$E[Q(t)]$</td>
<td>$BP(t)$</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>Rel. Error</td>
<td>Error</td>
</tr>
<tr>
<td>Mean</td>
<td>0.018</td>
<td>7.33</td>
<td>0.041</td>
</tr>
<tr>
<td>STDV</td>
<td>0.017</td>
<td>5.04</td>
<td>0.017</td>
</tr>
</tbody>
</table>
In practice, approximation methods with relative error less than 10% are considered good methods. The blocking probability is considered a very crucial measure of performance due to its sensitivity to changes in the parameters of the system. Table 3.5 shows that the mean relative error of the blocking probability using FPA is 7.33% which is a very good accuracy. In fact, FPA considered an excellent approximation compared with blocking probability mean relative errors of 27.52% and 67.75% obtained by MOL and PSA, respectively. On the other hand, the FPA produces a better accuracy that is close to exact in approximating the mean relative error of average number of customers in the system with a percentage of 0.56%.

Although the FPA is applied to nonstationary Erlang loss queues with sinusoidal arrival rates, the FPA algorithm can be applied to any periodic or non-periodic arrival rate function since it takes the arrival rate function as a function defined at certain points on time equally spaced by $\Delta t$. In Appendix B, the FPA algorithm is applied to different arrival rate functions.

## 3.4 The Nonstationary Multi-Class Erlang Loss Model

In this model, there are $R$ different streams of arrivals that arrive to a system according to a Poisson process each with time-dependent rate $\lambda_r(t)$ ($r = 1, 2, \ldots, R$). It is assumed that arrivals of the same class are independent of each other as well as arrivals from different classes. Each class $r$ arrival requests an exponential service with mean $1/\mu_r$. The service requested by any arriving customer is performed by a single server. The system has $s$ identical servers each of which can serve a customer of any class. An arriving customer is lost if it finds all $s$ servers in the system are busy. Let $n_r$ be number of class $r$ customers in the system. The state space, $S$, of the system is:

$$S = \{(n_1, n_2, \ldots, n_R) \in \mathbb{N}^R : n_1 + n_2 + \ldots + n_R \leq s\}.$$
Since the arrival and the service processes are Poisson, the probability that the system is in state \( n \) (\( n \in S \)) at time \( t \), \( P(n_1, n_2, \ldots, n_R; t) \), is described by the following set of forward differential equations:

\[
\frac{d}{dt} P(n_1, n_2, \ldots, n_R; t) = \sum_{r=1}^{R} \lambda_r(t) P(\ldots, n_r - 1, \ldots; t) + \sum_{r=1}^{R} (n_r + 1) \mu_r P(\ldots, n_r + 1, \ldots; t) - \sum_{r=1}^{R} (\lambda_r(t) + n_r \mu_r) P(n_1, n_2, \ldots, n_R; t)
\]

Such that:

- \( P(n_1, n_2, \ldots, n_R; t) = 0 \) if \( (n_1, n_2, \ldots, n_R; t) \notin S \),
- \( \lambda_r(t) = 0 \) \( \forall r \) and \( \forall t \) if \( n \in S \) and \( n_1 + n_2 + \ldots + n_R = s, \forall r \),
- \( \mu_r = 0 \) \( \forall r \) if \( n_r = 0 \),
- \( \sum_{n \in S} P(n_1, n_2, \ldots, n_R; t) = 1 \) \( t \geq 0 \),
- \( 0 \leq P(n_1, n_2, \ldots, n_R; t) \leq 1 \) \( \forall t; \forall (n_1, n_2, \ldots, n_R) \in S \),
- \( P(0, 0, \ldots, 0; 0) = 1 \) and \( P(n_1, n_2, \ldots, n_R; 0) = 0 ; \forall (n_1, n_2, \ldots, n_R) \in S \).

A special case of this queue is when \( \lambda_r(t) = \lambda_r, r = 1, 2, \ldots, R \), for all \( t \). As a result, \( \frac{d}{dt}[P(n_1, n_2, \ldots, n_R; t)] \to 0 \) as \( t \to \infty \) for all \( (n_1, n_2, \ldots, n_R) \in S \). The above system of differential equations reduces to a set of linear algebraic equations that provides the following product form solution:

\[
P(n_1, n_2, \ldots, n_R) = \frac{1}{G} \prod_{r=1}^{R} \frac{\rho_r^n}{n_r!} , \quad (n_1, n_2, \ldots, n_R) \in S,
\]

\[
G = \sum_{n \in S} \prod_{r=1}^{R} \frac{\rho_r^n}{n_r!} \quad \text{and} \quad \rho_r = \frac{\lambda_r}{\mu_r} , \quad r = 1, 2, \ldots, R.
\]
This model has been studied extensively and it has a wide range of applications in telecommunications. Refer to Ross [32] and Kelly [19] for further details.

It is worth mentioning that if the capacity of the queue (number of servers) becomes infinity, then the above solution becomes the product-form of $R$ independent Poisson distributions:

$$P(n_1, n_2, \ldots, n_R) = \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!} e^{-\rho_r}.$$  

Note that the solution of the multi-class Erlang loss queue is a truncated solution of the multi-class infinite server queue. This property is due to the reversibility of the infinite queue.

In most applications, the main performance measure of interest is the blocking probability of each class. Due to the independence among arrivals and the fact that each class of arrivals requires only one server, the number of class $r$ customers currently in service has no direct effect on the arrivals of other classes. The total number of customers in the system, *i.e.* the number of busy servers, affects all arrivals only when it reaches its maximum value in the system. In view of this, any arriving customer will be lost if it finds the system is full, *i.e.* all servers are busy, regardless of its class. The blocking probability, $BP_r$, of class $r$ arrivals is

$$BP_r = \sum_{n_1+n_2+\ldots+n_R=s} P(n_1, n_2, \ldots, n_R) = \frac{1}{G} \sum_{n_1+n_2+\ldots+n_R=s} \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!}$$

$$= \frac{1}{s!G} \sum_{n_1+n_2+\ldots+n_R=s} s! \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!}$$

$$= \frac{(\rho_1 + \rho_2 + \cdots + \rho_R)^s}{s!} G^{-1} \text{ by multinomial theorem}$$

$$= \frac{(\rho_1 + \rho_2 + \cdots + \rho_R)^s}{s!} \left[ \sum_{k=0}^{s} \sum_{n_1+n_2+\ldots+n_R=k} \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!} \right]^{-1}$$
\[ BP_r = \frac{(\rho_1 + \rho_2 + \cdots + \rho_R)^s}{s!} \left[ \sum_{k=0}^{s} \frac{1}{k!} \sum_{n_1+\cdots+n_R = k} k! \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!} \right]^{-1} \]

\[ = \frac{(\rho_1 + \rho_2 + \cdots + \rho_R)^s}{s!} \left[ \sum_{k=0}^{s} \frac{1}{k!} \left( \frac{\rho_1 + \rho_2 + \cdots + \rho_R}{k} \right)^k \right]^{-1} \quad \text{by multinomial theorem} \]

Let \( \overline{\rho} = \rho_1 + \rho_2 + \cdots + \rho_R \), then

\[ BP = BP_r = \frac{\overline{\rho}^s}{s!} \left[ \sum_{k=0}^{s} \frac{\overline{\rho}^k}{k!} \right]^{-1}, \quad r = 1, 2, \ldots, R, \]

where \( BP \) is the blocking probability of the whole queue.

Note that the blocking probability of class \( r \) arrivals is the Erlang loss formula with parameters \( \overline{\rho} \) and \( s \). Also, we note that in these blocking probabilities it is that they are independent of the class type. In other words, the steady-state blocking probability is the same for all classes.

Another important measure of performance of this queue is the steady-state average number of class \( r \) customers in the queue. This can be obtained by knowing that the overall average number of customers in the system, \( E[Q] \), is equal to the sum of the average number of class \( r \) customers, \( E[Q_r] \), for \( r = 1, 2, \ldots, R \). Namely,

\[ E[Q] = E[Q_1] + E[Q_2] + \ldots + E[Q_R]. \]

The average number of customers in an Erlang loss model is:

\[ E[Q] = \overline{\rho} (1 - BP) = \rho_1(1 - BP) + \rho_2(1 - BP) + \ldots + \rho_R(1 - BP), \]

and

\[ E[Q_r] = \rho_r(1 - BP), \quad r = 1, 2, \ldots, R. \]
The solution of the multi-class queue described above assumes a time-independent (stationary) arrival process. Most of the real life models, however, do not have time-independent arrival rates. Although the solution of this queue with time-independent arrival rates has an exact solution, this solution is considered an approximation to the real life problems with time-dependent arrival rates. In order to obtain an exact solution that handles time-dependent arrival rates explicitly, one has to solve the set of differential equations as a function of time. Getting the time-dependent stationary probabilities for the queue by solving the forward equations is a very difficult task, particularly, as the capacity and/or number of classes increase. Consequently, approximation techniques have been developed to avoid the direct solution of the set of forward equations. Below, we review the PSA and MOL methods and we also proposed an extension of the FPA method.

3.4.1 The PSA Method for the Nonstationary Multi-Class Erlang Loss Model

As mentioned earlier, a nice advantage of PSA is the ease of its generalization to other queueing models. The PSA solution of the time-dependent multi-class arrival queue preserves the same formulae of the stationary arrivals except for the $\lambda_r$. The arrival rates $\lambda_r$ are taken to be functions of time ($\lambda_r(t)$) as opposed to fixed values. The time-dependent offered load of class $r$ arrivals ($\rho_r(t)$) in PSA becomes the instantaneous offered load ($\lambda_r(t)/\mu_r$) as if the queue is stationary at each instant of time. Then, the time-dependent stationary probabilities, the blocking probability and the average number of customers in the system can be computed as follows:

$$P(n_1, n_2, \ldots, n_R; t) = \frac{1}{G(t)} \prod_{r=1}^{R} \frac{[\rho_r(t)]^{n_r}}{n_r!}, (n_1, n_2, \ldots, n_R) \in S, \ t > 0$$
where
\[ G(t) = \sum_{\forall n_1 \in \mathbb{R}} \prod_{r=1}^{R} \left[ \rho_r(t)^{n_r} \right] \frac{n_r!}{\mu_r} , \text{ and } \rho_r(t) = \frac{\lambda_r(t)}{\mu_r} ; r = 1, 2, ..., R. \]

Then,
\[ BP(t) = \frac{\overline{\rho}(t)}{s!} \left[ \sum_{k=0}^{s} \frac{\overline{\rho}(t)^k}{k!} \right]^{-1} , \overline{\rho}(t) = \rho_1(t) + \rho_2(t) + \cdots + \rho_R(t) , \]

and
\[ E[Q_r(t)] = \rho_r(t) \left( 1 - BP(t) \right) , r = 1, 2, ..., R. \]

### 3.4.2 The MOL Method for Nonstationary Multi-Class Erlang Loss Model

MOL requires more work than PSA to obtain the solution but in most cases it performs better. The MOL defines the time dependent offered load of class \( r \) arrivals as the offered load of the \( M(t)/M/\infty \) with parameter \( \lambda_r(t) \) and \( \mu_r \), as follows:
\[ \frac{d}{dt} \rho_r(t)^{\text{MOL}} = \lambda_r(t) - \mu_r \rho_r(t)^{\text{MOL}} ; r = 1, 2, ..., R. \]

Then,
\[ \overline{\rho}(t) = \rho_2(t)^{\text{MOL}} + \rho_2(t)^{\text{MOL}} + \cdots + \rho_R(t)^{\text{MOL}} \]

Obviously, this differential system consists of \( R \) independent differential equations each solved separately. The resulting offered load functions \( (\rho_r(t)^{\text{MOL}} ; r = 1, 2, ..., R) \) are substituted in the solution formulae above to obtain the desired performance measures.
3.4.3 The FPA Method for the Nonstationary Multi-Class Erlang Loss Model

The FPA algorithm described earlier can be extended to solve the multi-class problem once the expression of $\bar{p}(t)$ is determined. In order to determine $\bar{p}(t)$, the offered load of each class ($\rho_r(t); r = 1, 2, \ldots, R$) has to be determined. Class-$r$ offered load, $r = 1, 2, \ldots, R$, is determined iteratively in the same way it is determined in the single class case, except that there are $R$ differential equations instead of one. Below, we summarize the multi-class FPA algorithm.

Multi-class FPA Algorithm:

1. Choose an appropriate $\varepsilon, \Delta t$ and final time $T_f$.
2. Choose initial conditions (i.e. at $t = 0$) for $E[Q(t)]$. Let $E[Q(0)] = 0$.
3. Start with an initial blocking probability $BP^0(t) = 0$.
4. Set the iteration counter $k = 0$.
5. Solve the following system for $E[Q_r(t)], r = 1, 2, \ldots, R$

$$\frac{d}{dt} E[Q_r^k(t)] = \lambda_r(t) (1 - BP^k(t)) - \mu_r E[Q_r^k(t)], \quad r = 1, 2, \ldots, R$$

6. Let $E[Q^k(t)] = E[Q_1^k(t)] + E[Q_2^k(t)] + \ldots + E[Q_R^k(t)]$
7. Calculate

$$\bar{p}^k(t) = \frac{E[Q^k(t)]}{1 - BP^k(t)}$$
8. Update blocking probability:

\[ BP_{k+1}(t) = \frac{[\rho(t)^{k}]^s}{\sum_{i=0}^{s} [\rho(t)^k]^i i!} \]

9. If \( \|BP_k(t) - BP_{k+1}(t)\| < \epsilon \), then \( BP_k(t) \) converged. STOP.

Else, update the iteration counter \( k = k + 1 \). Go to step 6.

3.5 The Nonstationary Multi-Rate Erlang Loss Model

As in the proceeding model, arrivals to this system follow \( R \) independent Poisson processes with class \( r \) customers arriving at time-dependent rate \( \lambda_r(t) \), \( r = 1, 2, \ldots, R \). The difference in this queue is that a class \( r \) customer requests \( b_r \) servers, \( r = 1, 2, \ldots, R \), to perform its service. Independence among arrivals in a given class is assumed. Class \( r \) arrivals, \( r = 1, 2, \ldots, R \), request a service that requires an exponential amount of time with mean \( 1/\mu_r \). In other words, \( b_r \) servers will be allocated and work together to finish an exponential service with mean \( 1/\mu_r \) for a class \( r \) customer. Upon service completion of a class \( r \) customer, the \( b_r \) servers are released simultaneously. The system has a total of \( s \) identical servers, and each can provide service to any class of arrivals. Let \( \mathbf{n} = (n_1, n_2, \ldots, n_R) \) where \( n_r \) is number of class \( r \) customers in the system, and let \( \mathbf{b} = (b_1, b_2, \ldots, b_R) \). The total number of busy servers in state \( \mathbf{n} \) is

\[ \mathbf{b}^{\mathbf{n}} = b_1n_1 + b_2n_2 + \ldots + b_Rn_R. \]

Then, the set of all possible states of the system can be described as

\[ S^{\mathbf{b}} = \{ \mathbf{n} \in \mathbb{N}^R : \mathbf{b}^{\mathbf{n}} \leq s \}. \]
Since \( n_i \) is defined as in the multi-class problem, the forward differential equations that the system is in state \( n (n \in S^b) \) at time \( t \), \( P(n_1, n_2, \ldots, n_R ; t) \), remains the same as in the previous system except for the boundary conditions.

The boundary conditions for this model are:

\[
P(n_1, n_2, \ldots, n_R ; t) = 0 \quad \text{if} \quad (n_1, n_2, \ldots, n_R ; t) \notin S^b, \\
\lambda_r(t) = 0 \quad \forall r \text{ and } \forall t \quad \text{if} \quad n \in S^b \text{ and } bn^T > s - b_r, \ r = 1, 2, \ldots, R, \\
\mu_r = 0 \quad \forall r \quad \text{if} \quad n_r = 0, \\
\sum_{n \in S} P(n_1, n_2, \ldots, n_R ; t) = 1, \forall t, \\
0 \leq P(n_1, n_2, \ldots, n_R ; t) \leq 1 \quad \forall t ; \forall (n_1, n_2, \ldots, n_R) \in S^b, \\
P(0, 0, \ldots, 0 ; 0) = 1 \quad \text{and} \quad P(n_1, n_2, \ldots, n_R ; 0) = 0 ; \forall (n_1, n_2, \ldots, n_R) \in S^b.
\]

In the case of stationary arrival rates, \( \lambda_r(t) = \lambda_r, \forall r \text{ and } \forall t \), this system has a product-form solution that looks the same as the solution of the single resource request system but normalizing constant \( G \) is different. This difference in the normalizing constants is due to the change of state space \( S^b \) in the multi-rate model has changed.

Unlike the single class model, each class of arrivals experiences a blocking probability that may not be the same for all other classes. For example, consider an Erlang loss queue with two types of arrivals, Type-I and Type-II. Type-I arrivals requires one server where Type-II arrivals requires two servers. In this example, a Type-I arrival gets blocked if it finds no idle server in the system, whereas, a Type-II arrival gets blocked if there is either no idle server or only one idle server. Thus, the blocking probabilities of customers are not the same. Arrivals of different classes will have the same blocking probability if they request the same number of servers, independent of whether the service times are class-dependent. Although classes have different blocking probabilities, all matters to any arrival is the number of busy servers, or equivalently the number of idle servers, in the system.
The biggest challenge in this model is obtaining the blocking probability for each class. Computing the blocking probabilities by directly enumerating all possible states of the system requires an $O(s^8)$ amount of time. This technique is computationally cumbersome and grows exponentially fast even for relatively small systems. Several methods have been presented in the literature to avoid the exponential complexity of the computations. For more details, see Nilsson et al [28] and the references therein.

One of the most powerful methods for obtaining the blocking probabilities was published independently by Kaufman (1981) and Roberts (1981). The Kaufman-Roberts method is a recursive algorithm that has a linear complexity, $O(sR)$, and it is considered the fastest method. The recursive formula is as follows:

$$w(0) = 1,$$
$$w(K) = \frac{1}{K} \sum_{r=1}^{s} \rho_r b_r w(K - b_r), \quad K = 1, 2, \ldots, s.$$  

Then, the blocking probability of class-$r$ arrivals is

$$BP_r = \frac{\sum_{j=s-(b_r-1)}^{s} w(j)}{\sum_{j=0}^{s} w(j)}, \quad r = 1, 2, \ldots, R.$$  

This recursive formula is called the multi-rate Erlang-B formula. It is interesting to know that this formula can be applied to the single resource model, described earlier, as a fast and stable way of obtaining the blocking probability. Given the blocking probabilities, the average number of class-$r$ customers in the system is

$$E[Q_r] = \rho_r (1 - BP_r) \quad ; \quad r = 1, 2, \ldots, R.$$
In this system, the average number of busy servers, $E[Q]$, is another important measure of interest. One may view a single class $r$ as $b_r$ simultaneous arrivals since an arrival of class-$r$ needs $b_r$ servers. Thus, class $r$ arrivals provide a load that is $b_r$ times their normal load. In view of this, the total load in the system $\bar{\rho}$ is

$$\bar{\rho} = b_1\rho_1 + b_2\rho_2 + \cdots + b_R\rho_R$$

and average busy servers in the system is

$$E[Q] = b_1\rho_1(1 - BP_1) + b_2\rho_2(1 - BP_2) + \cdots + b_R\rho_R(1 - BP_R).$$

In the time-dependent case, the exact solution for the multi-rate system does not have the same solution as stationary case. PSA and MOL can be used to approximate the time-dependent multi-rate system. PSA and MOL define their respective offered loads in the same as in the time-dependent multi-class Erlang loss model described above. The time dependent blocking probabilities using these two methods are obtained using their respective offered loads in the following formulae:

$$BP_r(t) = \frac{\sum_{j=0}^{s} w(j; t) b_{r+1}}{\sum_{j=0}^{s} w(j; t)}, \; r = 1, 2, \ldots, R,$$

where $w(0; t) = 1$ and

$$w(K; t) = \frac{1}{K} \sum_{r=1}^{s} \rho_r(t)b_rw(K-b_r; t), \; K = 1, 2, \ldots, s.$$

In addition, the time-dependent average number of class $r$ customers and the time-dependent average busy servers are, respectively

$$E[Q_r(t)] = \rho_r(t) (1 - BP_r(t)) \; ; \; r = 1, 2, \ldots, R.$$
and

\[ E[Q(t)] = b_1 \rho_1(t)(1 - BP_1(t)) + b_2 \rho_2(t)(1 - BP_2(t)) + \cdots + b_R \rho_R(t)(1 - BP_R(t)). \]

### 3.6 FPA for Nonstationary Multi-Rate Erlang Loss Model

The analysis of the stationary case of multi-rate Erlang loss model shows that there are different blocking probabilities for each class of arrivals. As a result, the differential equation of the average number of class \( r \) customers in the system is

\[
\frac{d}{dt} E[Q_r(t)] = \lambda_r(t) (1 - BP_r(t)) - \mu_r E[Q_r(t)],
\]

where

\[
\rho_r(t) = \frac{E[Q_r(t)]}{1 - BP_r(t)}, \quad r = 1, 2, \ldots, R.
\]

In order to determine \( \bar{\rho}(t) \), the offered load of each class \( (\rho_r(t) ; r = 1, 2, \ldots, R) \) has to be determined. The class \( r \) offered load, \( r = 1, 2, \ldots, R, \) is determined iteratively, the same way as it was determined in the single class case, but using \( R \) differential equations instead of one. The stopping rule in the FPA algorithm for this model has to be modified since there is more than one blocking probability.

**Multi-Rate FPA Algorithm:**

1. Choose an appropriate \( \varepsilon, \Delta t \) and final time \( T_f \).
2. Choose initial conditions, for \( E[Q_r(t)], r = 1, 2, \ldots, R \). Let \( E[Q_r(0)] = 0 \).
3. Initialize the blocking probabilities \( BP_r^0(t) = 0 \) for all \( t, r = 1, 2, \ldots, R \).
4. Set the iteration counter \( k = 0 \).
5. Solve the following system for $E[Q^r(t)]$, $r = 1, 2, \ldots, R$

$$\frac{d}{dt}E[Q^r(t)] = \lambda_r(t)(1 - BP^k_r(t)) - \mu_r E[Q^r(t)] ; \quad r = 1, 2, \ldots, R$$

6. Calculate

$$\rho^*_r(t) = \frac{E[Q^r(t)]}{1 - BP^k_r(t)}$$

7. Set $w(0; t) = 1$ for all $t$.

8. For $K = 1, 2, \ldots, s$, compute the recursion

$$w(K; t) = \frac{1}{K} \sum_{i=1}^{K} \rho^*_i(t)w(K-b,t)$$

9. Update the blocking probabilities:

$$BP^k_{r+1}(t) = \frac{\sum_{j=0}^{b-1} w(j; t)}{\sum_{j=0}^{s} w(j; t)}, \quad r = 1, 2, \ldots, R.$$ 

10. If $\|BP^k_{r+1}(t) - BP^k_r(t)\| < \varepsilon$ for all $r$, then $BP^k_r(t)$ has converged. STOP.

   Else, set $k = k + 1$ and go to step 6.

### 3.7 A Numerical Example

In this section we give a numerical example of the solution obtained using FPA of a multi-rate Erlang loss queue. We compare the numerical results against simulation. Below, we first discuss the simulation methodology for nonstationary queues and then give a numerical example.
3.7.1 Simulation Methodology

Simulation is a powerful tool for analyzing queueing models as well as evaluating the accuracy and efficiency of approximate techniques. In fact, for some problems, simulation is the only way to test an approximation method. In many real-life situations, analytic solutions provide an approximation for the actual situation due to the simplifying assumptions imposed on them, such as stationary rates and Markovian property. Simulation makes it possible to model all queueing problems under almost any kind of conditions. In some cases, simulation is used if the mathematical model of the system cannot be easily solved. In this section we use simulation to test the accuracy of the FPA method.

Often, simulation is used assuming stationary processes. In stationary systems, measures of effectiveness provided by simulation are obtained by averaging over time, after the system has reached the steady-state. This simulation procedure provides a single value for the performance measure of interest. Nonstationarity in the arrival process requires nonstandard simulation methodology. The notion of steady-state of the system that grants a fixed value for the performance measure is not present in this case. In contrast, nonstationary simulation seeks for description of a performance measure as a function of time. Thus, instead of a single simulation run with sufficiently long time, a large number of independent replications with relatively short time have to be performed to achieve high precision.

The first step that should be presented in building a simulation model for a nonstationary queue is the random number generator for the time-dependent random Poisson process. Assume that we are interested in sampling from a nonstationary Poisson process (NSPP) with rate $\lambda(t)$. By the definition of NSPP, the number of events occurring within the interval $[t_1, t_2]$ is a Poisson process with mean:

$$\Lambda(t_1, t_2) = \int_{t_1}^{t_2} \lambda(t) \, dt.$$
A well-known method called *thinning* method is, often, used to generating random numbers from this type of distribution. This method starts by finding the maximum value of $\lambda(t)$ over time. Let $\lambda^* = \max\{\lambda(t) : t > 0\}$. The main idea is to generate arrivals constantly at rate $\lambda^*$ at all time. The generated value at time $t$ will be accepted as a NSPP arrival with probability $\lambda(t)/\lambda^*$, otherwise the generated value is rejected indicating that there is no arrivals at time $t$.

In this study, a somewhat different simulation methodology is used to test the accuracy of FPA. To illustrate the idea, let us assume that we are interested in a time-dependent measure of performance, say $X(t)$. A single simulation run consists of multiple replications, say $L$ replications, each of length $T$. Each replication starts with an empty system and the simulation interval $[0, T]$ is divided into subintervals each of length $\delta$ as shown in Figure 3.7. After each $\delta$ period in a replication, the current value of $X(t)$ is calculated. By the end of replication $i$, there will be $T/\delta$ random realizations, $X_i(0), X_i(\delta), X_i(2\delta), \ldots, X_i(T)$. At the end of the simulation run, the time-dependent average of the measure $X$ is obtained by averaging across replications for each time interval, as follows (also see Figure 3.7):

$$
\bar{X}(t) = \frac{\sum_{i=1}^{L} X_i(t)}{L} \quad t = 0, \delta, 2\delta, \ldots, T.
$$

In similar manner, we can compute the time-dependent variance of the measure as follows:

$$
Var_X(t) = \left\{ \frac{\sum_{i=1}^{L} X_i^2(t)}{L} - \left( \bar{X}(t) \right)^2 \right\} \quad t = 0, \delta, 2\delta, \ldots, T.
$$
Obviously, as $\delta$ becomes smaller the accuracy of the simulation output increases. In addition, the sample average of $X$ at time $t$ will be more accurate as the number of independent replications chosen is very large. However, increasing $L$ and $\delta$ may require a large memory and a long CPU time.

Using the time-dependent mean and time-dependent variance of the measure, the $100(1-\alpha)$% time-dependent confidence interval is

$$
\bar{X}(t) \pm t_{L-1,1-\alpha/2} \sqrt{\frac{Var_X(t)}{L}} \\
t = 0, \delta, 2\delta, \ldots, T.
$$

Unlike the stationary simulation, $\bar{X}(t)$ and $Var_X(t)$ do not have the same distributions for all $t$. Therefore, the time-dependent confidence intervals may not be a smooth function.
3.7.2 The Numerical Example

Consider a call center with a capacity of 100 lines. A call may either be class 1 or class 2. Class 1 and class 2 calls arrive according to a nonstationary Poisson processes with rate functions $\lambda_1(t) = 100 + 70 \sin(t)$ and $\lambda_2(t) = 80 + 50 \sin(2t)$, respectively. A class 1 call requires a single line and has an exponentially distributed duration with rate 2 units. A class 2 call requires two lines and has an exponentially distributed duration with rate 3 units.

The FPA algorithm is applied on this example and compared with the output of the simulation model. The values of $\Delta t$, $T_j$ and $\varepsilon$ are used in the approximation are 0.01, 20 and 0.01, respectively. The queue starts empty and idle for both the approximation algorithm and the simulation model. The simulation model is executed for this example with 1000 replications. Each replication spans for 20 units of time with $\delta$ chosen to be 0.01. The measures of performance that were used in the comparison were the blocking probability of each class and the number of busy lines. The results from FPA and simulation output in the interval $[0,10]$ are deleted to avoid the effect of initial conditions in the FPA algorithm and the warm-up period in simulation. Results of the simulation and FPA are presented in Figure 3.8.

Figure 3.8 (a) shows that the curves of average number of busy servers in the system obtained by simulation and FPA are almost the same. Note that the simulation curve in Figure 3.8 (a) is smooth due to the high average arrival rate of each class of calls which makes the states of the system are revisited so often during a very short time. In general, not all performance measures of the same system will have the same smoothness in the simulation. To visualize the accuracy of FPA, the 95% confidence intervals and FPA curve are plotted in Figure 3.8 (b). Although, the confidence intervals are so tight, the FPA curve remains within the confidence intervals.

Figure 3.8 (c) and (e) plot the blocking probability of class 1 and class 2 calls, respectively. Obviously, the FPA curve perfectly follows the simulation output. In addition, the 95% confidence intervals in Figure 3.8 (d) and (f) show that the FPA curves remain
inside the confidence intervals. Precisely, the FPA curves in Figure 3.8 (d) and (f) remain in the middle of the confidence intervals in both plots.

Figure 3.8: Results of FPA and simulation for two-class Erlang loss system with parameters 
\( \lambda_1(t) = 100 + 70 \sin(t) \), \( \lambda_2(t) = 80 + 50 \sin(2t) \), \( \mu_1=2 \), \( \mu_2=3 \) and \( s=100 \)
Chapter 4

Nonstationary Erlang Loss Queueing Networks

4.1 Introduction

In the previous chapter, single Erlang loss queues with nonstationary arrival rates were analyzed. In this chapter, we analyze a queueing network of Erlang loss queues by applying the FPA method. The main focus of this study is to obtain the time-dependent performance measures for this system.

4.2 A Network of Nonstationary Erlang Loss Queues

Consider a queueing network consisting of \( N \) independent queueing nodes. Node \( i, i = 1, 2, \ldots N \), in the network is an Erlang loss queue with time-dependent Poisson stream of external arrivals at rate \( \lambda_i(t) \) and \( s_i \) identical servers. Arrivals to queue \( i \) request a single
service performed by a single server that requires an exponential amount of time with mean $1/\mu_i$. Upon service completion at queue $i$, the customer will move to queue $j$ for the next service with probability $q_{ij}$. The probability that a customer has completed its service at queue $i$ and will leave the network is $q_{i0}$. Hence, the branching probabilities can be represented as a square matrix $B$, where $B_{ij} = q_{ij}$ and $B_{ii} = q_{i0}$ for $i, j = 1, 2, \ldots N$. This type of branching is called referred to as Markovian branching. Any external or internal arrival to queue $i$ finds all servers busy will be lost.

According to the above description, the state space of the system is

$$S = \{(n_1, n_2, \ldots, n_N) \in \mathbb{N}^N : 0 \leq n_i \leq s_i ; i = 1, 2, \ldots, N \}.$$ 

Since the arrival and the service processes are Poisson, the probability that the system is in state $n$ ($n \in S$) at time $t$, $P(n_1, n_2, \ldots, n_N; t)$, is described by the following set of forward differential equations:

$$\frac{d}{dt} P(n_1, n_2, \ldots, n_N; t) = \sum_{i=1}^{N} \lambda_i(t) P(\ldots, n_i - 1, \ldots ; t) + \sum_{i=1}^{N} (n_i + 1) \mu_i q_{i0} P(\ldots, n_i + 1, \ldots ; t)$$

$$+ \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} (n_i + 1) \mu_i q_{ij} P(\ldots, n_i + 1, \ldots, n_j - 1, \ldots ; t)$$

$$+ \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} (n_i + 1) \mu_i q_{ji} P(\ldots, n_i + 1, \ldots, s_j, \ldots ; t)$$

$$- \sum_{i=1}^{N} (\lambda_i(t) + n_i \mu_i) P(n_1, n_2, \ldots, n_N; t). \quad (4.1)$$

Such that:

- $P(n_1, n_2, \ldots, n_N; t) = 0$ if $(n_1, n_2, \ldots, n_N; t) \notin S$,
- $\lambda_i(t) = 0$∀$i$ and ∀$t$ if $n \in S$ and $n_i = s_i$, ∀$i$,
- $\mu_i = 0$∀$i$ if $n_i = 0$,
- $\sum_{n \in S} P(n_1, n_2, \ldots, n_N; t) = 1$∀$t$. 

66
Then the time-dependent blocking probability of queue $j$ at time $t$ is

$$BP_j(t) = \sum_{n \in S} P(n_1, \cdots, n_j = s_j, \cdots, n_N; t),$$

and the time-dependent average number of busy servers in queue $j$ is

$$E[Q_j(t)] = \sum_{n_j=0}^{s_j} n_j \sum_{(n_1, \cdots, n_N) \in S} P(n_1, \cdots, n_N; t).$$

So far, there is no exact closed-form solution reported in the literature for nonstationary Erlang loss network. In fact, the closed-form solution of this queuing network with any stationary rates is not known. However, Abdalla and Boucherie [1] stated that a stationary loss network has an exact solution only if it has arrival and service rates that satisfy the set of necessary conditions (2.2) and (2.3).

A special case of nonstationary loss networks where they have an exact solution is when the number of servers at each queue becomes infinite. We believe that a thorough understanding of nonstationary infinite-server networks provides a helpful insight for solving more complex networks. In the next section, we present the nonstationary infinite-server networks and there solution as a special case of nonstationary loss networks.

### 4.3 The Nonstationary Infinite-Server Queueing Networks

We consider a nonstationary infinite-server queueing networks with the same rates and branching probabilities as the loss networks in section 3.2. The only difference between the Erlang loss queueing networks and the infinite-server queueing networks is that the
arriving customers to a queue in an infinite-server network never get blocked. In other words, blocking probabilities are equal to zero at all time for all queues of the infinite-server network. The state space of the infinite-server network is:

\[ S^\infty = \{ (n_1, n_2, \ldots, n_N) \in \mathbb{N}^N : n_i \geq 0 ; i = 1,2,\ldots,N \} . \]

The forward differential equations for time-dependent state probabilities are:

\[
\frac{d}{dt} P(n_1, n_2, \ldots, n_N; t) = \sum_{i=1}^{N} \lambda_i(t) P(\ldots, n_i - 1, \ldots ; t) + \sum_{i=1}^{N} (n_i + 1) \mu_i q_{ii} P(\ldots, n_i + 1, \ldots ; t) + \sum_{i=1}^{N} \sum_{j \neq i} (n_i + 1) \mu_i q_{ij} P(\ldots, n_i + 1, \ldots, n_j - 1, \ldots ; t) - \sum_{i=1}^{N} (\lambda_i(t) + n_i \mu_i) P(n_1, n_2, \ldots, n_N; t)
\]

Such that:

\[ P(n_1, n_2, \ldots, n_N; t) = 0 \quad \text{if} \quad (n_1, n_2, \ldots, n_N; t) \notin S^\infty \]

\[ \mu_i = 0 \quad \forall i \quad \text{if} \quad n_i = 0 \]

and

\[ \sum_{n \in S} P(n_1, n_2, \ldots, n_N; t) = 1 \quad \forall t \]

The solution \( P^\infty(n ; t) \) to this special case network is a multidimensional Poisson distribution

\[
P^\infty(n ; t) = \prod_{i=1}^{N} \frac{[\rho_i(t)]^{n_i}}{n_i!} e^{-\rho_i(t)}, \quad n \in S^\infty, \quad t > 0.
\]

To obtain the above probabilities, one has to know the offered loads \( \{ \rho_i(t) : i = 1,2,\ldots,N \} \). In general, the rate of change in the offered load of a queue at time \( t \) is the difference between the average rate into the queue and the average rate out of the queue. Consider
queue $j$ in the network, the average arrival rate is the sum of the external arrival rates plus the arrival rates from other queues. The departure rate of queue $j$, $D_j(t)$, depends on number of customers $n_j$ that are in the queue at time $t$. The arrival and departure rates into/from queue $j$ are depicted in Figure 4.1 where $P[n_j ; t]$ and $Q_j(t)$ are the marginal probability that there are $n_j$ busy servers in queue $j$ and average number of busy servers in queue $j$ at time $t$, respectively.

In view of this, the rate of change in the offered load of queue $j$ at time $t$ is given by

$$\frac{d}{dt} \rho_j(t) = \lambda_j(t) + \sum_{i=1}^{N} q_i D_i(t) - \sum_{i=1}^{N} i \mu_j P[n_j = i ; t] , \ t > 0 , \ j = 1, 2, \ldots, N.$$  

Note that, the average number of busy servers in queue $j$ at time $t$, $Q_j(t)$, is

$$E[Q_j(t)] = \sum_{i=1}^{\infty} i P[n_j = i ; t] , \ t > 0 , \ j = 1, 2, \ldots, N.$$  

Then,

$$\mu_j \sum_{i=1}^{\infty} i P[n_j = i ; t] = \mu_j E[Q_j(t)] = D_j(t) , \ j = 1, 2, \ldots, N.$$  

Figure 4.1: Arrival and departure rates of a queue in an infinite-server queueing network

69
This yield

\[
\frac{d}{dt} \rho_j(t) = \lambda_j(t) + \sum_{i=1}^{N} q_{ij} \mu_i E[Q_i(t)] - \mu_j E[Q_j(t)], \quad t > 0, \quad j = 1,2,\ldots,N
\]

The marginal distribution of the number of busy servers at time \( t \) in queue \( j \) in the infinite-server networks is a Poisson distribution with mean \( \rho_j(t) \) as follows:

\[
P^\infty(n_j; t) = \sum_{\forall n_i, i \neq j} P^\infty(n; t) = \frac{[\rho_j(t)]^{n_j}}{n_j!} e^{-\rho_j(t)}, \quad t > 0
\]

The offered load of queue \( j \) at time \( t \), \( \rho_j(t) \), is the average number of busy servers in the queue at time \( t \), \( E[Q_j(t)] \). Finally, the offered loads \( \rho_j(t) \) for \( j = 1,2,\ldots,N \) are obtained from the solution of the differential equations:

\[
\frac{d}{dt} \rho_j(t) = \lambda_j(t) + \sum_{i=1}^{N} q_{ij} \rho_i(t) - \mu_j \rho_j(t), \quad t > 0, \quad j = 1,2,\ldots,N.
\] (4.2)

In section 2.7, we mentioned that differential equations (4.2) are used to approximate Erlang loss networks with time-dependent arrival rates as proposed by Massey and Whitt [24].

### 4.4 The FPA for Nonstationary Erlang Loss

#### Queueing Networks

Consider the Erlang loss networks described in section 4.2. In this network, a customer admitted in a queue is served immediately as in the case of the infinite-server network. Unlike arrivals to a queue in an infinite-server network, arriving customers to a queue in the loss network may get lost if there are no idle servers available. Therefore, the
average arrival rate at time $t$ into a queue in an Erlang loss queueing network does not equal the effective average rate at time $t$ into the queue because of blocking. Thus, the blocking effect should be considered in the traffic equations of the Erlang loss networks. Given that the blocking probability of queue $j$ at time $t$ is $BP_j(t)$, the effective arrival and departure rates corresponding to queue $j$ in a loss network are depicted in the Figure 4.2.

![Figure 4.2: Arrival and departure rates of a queue in an Erlang loss queueing network](image)

As Figure 4.2 shows, the rate of change in the average number of busy servers in queue $j$ at time $t$ in a loss network is given by

$$\frac{d}{dt} E[Q_j(t)] = \left[ \lambda_j(t) + \sum_{i=1}^{N} q_{ij} D_i(t) \right] (1 - BP_j(t)) - \sum_{i=1}^{S_j} i \mu_j P\{n_j = i; t\},$$

$\quad t > 0, \quad j = 1, 2, \ldots, N.$

We know that $E[Q_j(t)]$ for loss queue $j$ is defined as

$$E[Q_j(t)] = \sum_{i=1}^{S_j} i P\{n_j = i; t\}, \quad t > 0, \quad j = 1, 2, \ldots, N \text{ and } n_j = 0$$
Then,
\[
\sum_{i=1}^{n_j} \mu_i P\{n_i = i; t\} = \mu_j E[Q_j(t)] = D_j(t), \quad j = 1, 2, \ldots, N
\]
This yield
\[
\frac{d}{dt} E[Q_j(t)] = \left( \lambda_j(t) + \sum_{i=1}^{N} q_{ij} \mu_i E[Q_j(t)] \right) (1 - BP_j(t))
- \mu_j E[Q_j(t)], \quad t > 0, \quad j = 1, 2, \ldots, N.
\]

(4.3)

In order to solve equations (4.3), we need the solution of \(BP_j(t), j = 1, 2, \ldots, N\), which can be obtained by solving the forward differential equations (4.1) of the Erlang loss network. The solution of the system (4.1) is required for system (4.3) because the average number of busy servers in Erlang loss queues is not equal the offered load as in the infinite-server queues. The equality between the offered load and the average busy servers in the infinite-server queues makes the set of traffic equation (4.3) independent of the forward equations (4.1). In loss networks, the average number of busy servers in queue \(j\) at time \(t\) is
\[
E[Q_j(t)] = \rho_j(t) (1 - BP_j(t)).
\]
This means that the differential equations (4.3) are dependent on forward equation (4.1). However, our proposed approximation technique numerically provides the time-dependent blocking probabilities as well as the time-dependent average busy servers at each queue in the network without solving the forward equations (4.1) in time.

To describe how the algorithm works, let us assume that the nonstationary Erlang loss network has a product form solution, just like the infinite server network. Then, the time-dependent steady-state probability of having \(n\) customers in the system is:
\[
P(n; t) = \prod_{i=1}^{N} \left[ \frac{\rho_i(t)}{n_i!} \right] \sqrt{\sum_{k=0}^{n} \frac{\rho_i(t)^k}{k!}}, \quad n \in S, \quad t > 0.
\]

(4.4)
This means that the time-dependent marginal distribution of the number of busy servers in queue $j$ is:

$$P(n_j; t) = \frac{[\rho_j(t)]^{n_j} / n_j!}{\sum_{k=0}^{s_j} [\rho_j(t)]^k / k!}, \quad n_j = 0, 1, 2, \ldots, s_j, \quad t > 0,$$

(4.5)

where $\rho_j(t)$ is obtained by solving equations (4.3) and substituting the solution in:

$$\rho_j(t) = \frac{E[Q_j(t)]}{1 - BP_j(t)}, \quad j = 1, 2, \ldots, N$$

(4.6)

Given the above assumption, the FPA algorithm starts by choosing an appropriate length of subintervals $\Delta t$, the final time $T_f$ and initial conditions. The values of $\Delta t$ and $T_f$ are used to discretize the arrival rates over time. Since there is no knowledge about the blocking probability of each queue in the network over time initially, the blocking probabilities are set to zero at all time $t$ for all queues in the loss network. Given the initial blocking probabilities, the traffic equations (4.3) are solved numerically to obtain $E[Q_j(t)]$ for all $j$. According to equation (4.6), the initial time-dependent offered load of queue $j$ is equal to $E[Q_j(t)]$ for all $j$. Next, the first approximation of the time-dependent blocking probability at each queue is calculated using (4.5). Once we have nonzero blocking probabilities on hand, the time-dependent offered load of each queue is updated using equation (4.6). The algorithm continues in this manner until the blocking probabilities converge with $\epsilon$ tolerance.

There are number of numerical methods to solve a system of differential equations. Refer to [2] for more details and properties of each method. We solve (4.3) numerically using multidimensional Euler’s methods as follows:

$$E[Q_j(t+\Delta t)] = E[Q_j(t)] + \left[ \lambda_j(t) + \sum_{i=1}^{N} q_{ij} \mu_i E[Q_i(t)] \right] (1 - BP_j(t)) \Delta t$$

$$- \mu_j E[Q_j(t)] \Delta t, \quad \text{for} \quad t = 0, \Delta t, 2\Delta t, \ldots, T_f \quad \text{and} \quad j = 1, 2, \ldots, N.$$
Finally, the FPA algorithm can be summarized in the following steps:

1. Choose an appropriate $\Delta t$, a final time $T_f$ and a tolerance $\varepsilon$.
2. Evaluate $\lambda_j(t)$ at $t = 0, \Delta t, 2\Delta t, \ldots, T_f$ for $j = 1, 2, \ldots, N$.
3. Choose initial conditions (i.e. at $t = 0$) for $E[Q_j(0)]$, $j = 1, 2, \ldots, N$.
   
   Set $E[Q_j(0)] = 0$, $j = 1, 2, \ldots, N$.
4. Set iteration counter $k = 0$.
5. Start with initial blocking probabilities for all $t$; $BP_j^k(t) = 0$.
6. Solve, for all $t$, $E[Q_j(t)]$ for $j = 1, 2, \ldots, N$ the following:

   \[
   E^k[Q_j(t+\Delta t)] = E^k[Q_j(t)] + \left[ \lambda_j(t) + \sum_{i=1}^{N} q_{ji} \mu_j E^k[Q_i(t)] \right] (1 - BP_j^k(t)) \Delta t \\
   - \mu_j E^k[Q_i(t)] \Delta t, \text{ for } j = 1, 2, \ldots, N.
   \]

7. Calculate

   \[
   \rho_j^k(t) = \frac{E^k[Q_j(t)]}{1 - BP_j^k(t)} , t = 0, \Delta t, 2\Delta t, \ldots, T_f \text{ and } j = 1, 2, \ldots, N
   \]

8. Evaluate

   \[
   BP_{j+1}^k(t) = \frac{\rho_j^k(t)^s / s!}{\sum_{i=0}^{t} \rho_j^k(t)^i / i!} , t = 0, \Delta t, 2\Delta t, \ldots, T_f \text{ and } j = 1, 2, \ldots, N
   \]

9. If $\max_{j=1,2,\ldots,N} \left| \|BP_j^k(t) - BP_{j+1}^k(t)\| \right| < \varepsilon$, then $BP_j^k(t)$ converged for all $j$. Stop.

   Else, update the iteration counter $k = k + 1$. Go to step 6.
4.5 The Accuracy of FPA for Nonstationary Erlang Loss Queueing Networks

In this section, we analyze the accuracy of FPA for the loss networks using simulation. The FPA method in this chapter is proposed to solve loss networks with fixed routing probabilities and general topology. Analyzing the accuracy of FPA by considering all possible topologies of a loss network of fixed number of queues is a cumbersome task. Therefore, we will analyze the accuracy of FPA by considering three basic topologies, namely diverging, converging and tandem. Any arbitrary topology can be a mixture of the above three basic topologies. We also examine a small arbitrary topology. The choice of examples in this analysis starts with simple topologies and move toward more complex networks.

The accuracy measure that is used is the average absolute error and the relative absolute error between the simulation output and the approximation method at time \( t \). In our analysis, we focus on the accuracy of approximation for the time-dependent blocking probability since this measure of performance is an important and sensitive measure in network. If \( BP(t)_{\text{app}} \) and \( BP(t)_{\text{sim}} \) are the approximate and simulated blocking probability at time \( t \), the accuracy measures are:

\[
\text{Average Error} = \frac{\left\| BP(t)_{\text{app}} - BP(t)_{\text{sim}} \right\|}{N},
\]

and

\[
\text{Relative Error} = \frac{\left\| BP(t)_{\text{app}} - BP(t)_{\text{sim}} \right\|}{\left\| BP(t)_{\text{sim}} \right\|} \times 100,
\]

where \( \| . \|_1 \) defined as:

\[
\| x \|_1 = |x_1| + |x_2| + \cdots + |x_D| \text{ for } x \in \mathbb{R}^D.
\]
The FPA algorithm in this analysis is applied on the chosen topologies with the parameters $\Delta t$, $T_f$ and $\varepsilon$ equal to 0.01, 20 and 0.01, respectively. All queues in the network are taken to be empty initially in the FPA algorithm. The simulation models are executed with 1000 replications. Each replication starts with empty and idle queues and spans for 20 units of time with $\delta$ chosen to be 0.01. The time-dependent blocking probability and the average number of busy servers of each queue in the network are the measures of performance used in this analysis. The results from FPA and simulation output in the interval [0,10] are deleted to avoid the effect of initial conditions in the FPA algorithm and the warm-up period in simulation.

The blocking probabilities obtained by FPA appear to be exact for all of the numerical examples considered in this section. We have not been able to show that theoretically but the simulation results validate this conclusion. The FPA curves accurately follow the simulation output. In addition, in all of the numerical examples in this section the FPA curves remain almost in the middle of the 95% confidence intervals, for all $t$ and for all queues.

### 4.5.1 A Diverging Topology

The loss network with the diverging topology is a feed-forward network which consists of a single node that is linked in tandem with a number of nodes as shown in Figure 4.3. All queues in the network are Erlang loss queues each of which has an external time-dependent Poisson arrival rate and stationary Poisson service rate. We assume that no departures to the outside from node 1.
To analyze the accuracy of FPA for this topology, we consider the network in Figure 4.3 with the following parameters:

Queue 1: \( \lambda_1(t) = 50 + 40 \sin(t) \), \( s_1 = 20 \), \( \mu_1 = 4 \),
Queue 2: \( \lambda_2(t) = 10 + 5 \sin(0.75t) \), \( s_2 = 10 \), \( \mu_2 = 2 \),
Queue 3: \( \lambda_3(t) = 25 + 20 \sin(t) \), \( s_3 = 10 \), \( \mu_3 = 3 \),
Queue 4: \( \lambda_4(t) = 15 + 15 \sin(2t) \), \( s_4 = 10 \), \( \mu_4 = 2 \),
Queue 5: \( \lambda_5(t) = 20 + 10 \sin(t) \), \( s_5 = 10 \), \( \mu_5 = 3 \).

The branching probabilities from queue 1 to other queues are arbitrarily chosen to perform the experiment. The branching probability matrix for the network is:
The average and relative errors of FPA for each queue is summarized in Table 4.1. Table 4.1 shows that FPA provides an excellent approximation for both performance measures. The errors of blocking probabilities seem higher than the error of average busy servers in each queue because of the randomness in the simulation.

Table 4.1. Error of FPA for five-queue Erlang loss network with the diverging topology

<table>
<thead>
<tr>
<th>Queue</th>
<th>$BP(t)$</th>
<th>$E[Q(t)]$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error</td>
<td>Rel. Error</td>
</tr>
<tr>
<td>1</td>
<td>0.0031</td>
<td>6.03</td>
</tr>
<tr>
<td>2</td>
<td>0.0117</td>
<td>3.46</td>
</tr>
<tr>
<td>3</td>
<td>0.0098</td>
<td>3.88</td>
</tr>
<tr>
<td>4</td>
<td>0.0134</td>
<td>4.92</td>
</tr>
<tr>
<td>5</td>
<td>0.0073</td>
<td>6.16</td>
</tr>
</tbody>
</table>

In Figures 4.4 through 4.8, we present the plots of simulation output and FPA curve for time-dependent blocking probability of each queue since they are more important performance measures and they appear to have higher errors by FPA. For each queue in the network, one plot shows the simulation and approximate blocking probability and the other for shows the approximate blocking probability with respect to 95% confidence interval so the accuracy becomes visual.
Figure 4.4: Simulation and FPA results of queue 1, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.

Figure 4.5: Simulation and FPA results of queue 2, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.

Figure 4.6: Simulation and FPA results of queue 3, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.
As can be seen, the FPA curve follows exactly the simulation outputs for all queues in Figures 4.4 through 4.8. The FPA curve remains within the random observations for all queues even for random observations that have a very small variability around their mean, such Figure 4.7. In addition, the FPA curve falls almost in the middle of the time-dependent confidence interval for all queues throughout the simulation time.
4.5.2 A Converging Topology

The converging topology considered in this section is the opposite of the diverging topology described earlier. The loss network in this topology is, also, a feed-forward network and it consists of a number of nodes in parallel all linked to a single node. All queues in the network are Erlang loss queues each of which has an external time-dependent Poisson arrival rate and stationary Poisson service rate. Figure 4.9 shows an example of a loss network of five queues with the converging topology.

To analyze the accuracy of FPA for this topology, we consider the network in Figure 4.9 with the same parameters used in sections 4.5.1.

Figure 4.9: A five-queue loss network with the converging topology
In this example, we assume that the departures from queue 2, queue 3, queue 4 and queue 5 go to queue 1 with probability one. With this choice of branching probabilities we test the accuracy of FPA in dealing with non-Poisson streams at the extreme case of branching. Then, the branching matrix for the network is as follows:

\[
B = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

Table 4.2 summarizes the average and relative errors of FPA with respect to simulation. The FPA still provides a good approximation for both performance measures in all queues.

Table 4.2. Error of FPA for five-queue loss network with the diverging topology

<table>
<thead>
<tr>
<th>Queue</th>
<th>$BP(t)$</th>
<th>$E[Q(t)]$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error</td>
<td>Rel. Error</td>
</tr>
<tr>
<td>1</td>
<td>0.0053</td>
<td>8.62</td>
</tr>
<tr>
<td>2</td>
<td>0.046</td>
<td>11.06</td>
</tr>
<tr>
<td>3</td>
<td>0.0064</td>
<td>4.70</td>
</tr>
<tr>
<td>4</td>
<td>0.0112</td>
<td>7.27</td>
</tr>
<tr>
<td>5</td>
<td>0.0063</td>
<td>9.13</td>
</tr>
</tbody>
</table>

The last four queues in the network behave like independent queues since the network is a feed-forward network and those queues have only external arrival rates as input rates without interaction with each other. Thus, analyzing the accuracy of FPA for the last four queues will be like analyzing several independent queues which has been analyzed in chapter 2. The most important
queue in the network is queue 1 since the average arrival rate to this queue is affected by the departure rates of the other queues. In Figure 4.10 we plot the simulation output and FPA curves for time-dependent blocking probability and average number busy servers of queue 1.

Figure 4.10: Simulation and FPA results of queue 1, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability, (c) Average number of busy servers using simulation and FPA, (d) Simulation confidence intervals and FPA of average number of busy servers.

Figure 4.10 (a) shows that the FPA curve remains within the time-dependent 95% confidence interval for all time. FPA shows its high accuracy in Figure 4.10(b) by following the simulation output for the random observations that are very tight around the mean of average number of busy servers.
4.5.3 Tandem Topology

Tandem networks have been studied extensively with stationary arrival rates. Very few papers deal with this kind of networks in time-dependent. Refer to Abdalla and Boucherie [1] and Jennings et al.[17] and the references therein.

The loss network in this topology is a feed-forward network with a number of queues arranged in series. The departure stream of any queue or a fraction of it goes to the subsequent queue only. Queues in the network are all loss queues each of which has an external time-dependent Poisson arrival rate and stationary Poisson service rate. Figure 4.11 shows an example of a loss network of five queues with the series topology.

![Figure 4.11: Five-queue loss network with the series topology](image)

We applied FPA on the network in Figure 4.11 with the following parameters:

1. **Queue 1:** \( \lambda_1(t) = 8 + 6 \sin(t) \), \( s_1 = 10 \), \( \mu_1 = 1 \),
2. **Queue 2:** \( \lambda_2(t) = 10 + 5 \sin(0.75t) \), \( s_2 = 10 \), \( \mu_2 = 2 \),
3. **Queue 3:** \( \lambda_3(t) = 25 + 20 \sin(t) \), \( s_3 = 10 \), \( \mu_3 = 3 \),
4. **Queue 4:** \( \lambda_4(t) = 15 + 15 \sin(2t) \), \( s_4 = 10 \), \( \mu_4 = 2 \),
5. **Queue 5:** \( \lambda_5(t) = 20 + 10 \sin(t) \), \( s_5 = 10 \), \( \mu_5 = 3 \).

In this example, we assume that the departures of any queue go to next queue with probability one except queue 5 since it is the last queue. This choice of branching
probabilities tests the accuracy of FPA for series structure under the extreme case. If FPA provides accurate approximation for series networks with this choice of branching probabilities it is expected that FPA will be more accurate with other branching probabilities less than one. Then, the branching matrix for the network is as follows:

\[
B = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 
\end{bmatrix}
\]

Table 4.3 summarizes the average and relative errors of FPA with respect to simulation. The FPA still provides a good approximation for both performance measures in all queues.

Table 4.3: Error of FPA for five-queue loss network with the series topology

<table>
<thead>
<tr>
<th>Queue</th>
<th>BP(t)</th>
<th>E[Q(t)]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error</td>
<td>Rel. Error</td>
</tr>
<tr>
<td>1</td>
<td>0.0090</td>
<td>8.33</td>
</tr>
<tr>
<td>2</td>
<td>0.0084</td>
<td>6.17</td>
</tr>
<tr>
<td>3</td>
<td>0.0102</td>
<td>3.56</td>
</tr>
<tr>
<td>4</td>
<td>0.0126</td>
<td>2.48</td>
</tr>
<tr>
<td>5</td>
<td>0.0129</td>
<td>4.17</td>
</tr>
</tbody>
</table>

Table 4.3 shows that FPA provides a high level of accuracy in approximating the blocking probabilities in this network by having a relative error less than 10%. On the other hand, the accuracy of FPA for average number of busy servers in each queue is excellent to the point that the average relative error never exceeds 0.9%. The plots in Figure 4.12 through 4.16 are the simulation and FPA blocking probabilities over time for each queue. Figure 4.12-16 show that FPA curves are exact for all queues. In addition, the 95% confidence intervals provide a strong evidence for the accuracy of FPA.
Figure 4.12: Simulation and FPA results of queue 1, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.

Figure 4.13: Simulation and FPA results of queue 2, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.

Figure 4.14: Simulation and FPA results of queue 3, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.
Figure 4.15: Simulation and FPA results of queue 4, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.

Figure 4.16: Simulation and FPA results of queue 5, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.

4.5.4 An Arbitrary Erlang Loss Network with Feedback

A more complex network structure is considered in this section by introducing feedback in a loss network. Departures of any queue in the networks considered can either move to a downstream queue or to an upstream queue. There are many ways to construct a feedback network of a fixed number of queues. However, we choose a feedback topology for a network of six loss queues complex enough to measure the accuracy of FPA. The network considered in this analysis is presented in Figure 4.17. It still
assumed that all queues in the network are loss queues with external time-dependent Poisson arrival rates and stationary Poisson service rates. We applied FPA on the network in Figure 3.9 with the following parameters:

Queue 1: \[ \lambda_1(t) = 8 + 6 \sin(t) \], \[ s_1 = 10, \quad \mu_1 = 1, \]
Queue 2: \[ \lambda_2(t) = 10 + 5 \sin(0.75t) \], \[ s_2 = 10, \quad \mu_2 = 2, \]
Queue 3: \[ \lambda_3(t) = 25 + 20 \sin(t) \], \[ s_3 = 10, \quad \mu_3 = 3, \]
Queue 4: \[ \lambda_4(t) = 15 + 15 \sin(2t) \], \[ s_4 = 10, \quad \mu_4 = 2, \]
Queue 5: \[ \lambda_5(t) = 20 + 10 \sin(t) \], \[ s_5 = 10, \quad \mu_5 = 3, \]
Queue 6: \[ \lambda_6(t) = 0 \], \[ s_5 = 10, \quad \mu_5 = 2. \]

Figure 4.17: A Six-queue loss network with the feedback topology

The network in Figure 3.9 has a high feedback structure in it. Moreover, to make the network higher in feedback arrivals, higher probabilities are assigned to feedback branches in the network. The branching probability matrix associated with the network in Figure 4.17 is:
Table 4.4 summarizes the average and relative errors of FPA with respect to simulation. The FPA still provides a good approximation for both performance measures in all queues.

Table 4.4: Error of FPA for Six-queue Erlang loss network with feedback

<table>
<thead>
<tr>
<th>Queue</th>
<th>BP(t)</th>
<th>E[Q(t)]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error</td>
<td>Rel.Error</td>
</tr>
<tr>
<td>1</td>
<td>0.0128</td>
<td>2.41</td>
</tr>
<tr>
<td>2</td>
<td>0.0135</td>
<td>3.35</td>
</tr>
<tr>
<td>3</td>
<td>0.0113</td>
<td>3.70</td>
</tr>
<tr>
<td>4</td>
<td>0.01243</td>
<td>3.53</td>
</tr>
<tr>
<td>5</td>
<td>0.0126</td>
<td>4.19</td>
</tr>
<tr>
<td>6</td>
<td>0.0062</td>
<td>11.80</td>
</tr>
</tbody>
</table>

Table 4.4 gives the relative and absolute error of the blocking probability of each queue in the network. The relative error of FPA for the blocking probabilities is very well under 10% in all queues except for queue 6. Queue 6 has the highest relative error due to the high variability in simulation output corresponds to queue 6. A small average blocking probability of about 0.08 for queue 6 makes the event of blocking rarely occurs producing a high variability in simulation output (see Figure 4.23). Needless to say that FPA provides almost an exact solution for average number of busy servers in each queue. Figure 4.18-23 show that the FPA curves are almost exact for all queues. In addition the 95% confidence intervals provide a strong evidence for the accuracy of FPA.
Figure 4.18: Simulation and FPA results of queue 1, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.

Figure 4.19: Simulation and FPA results of queue 2, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.

Figure 4.20: Simulation and FPA results of queue 3, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.
Figure 4.21: Simulation and FPA results of queue 4, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.

Figure 4.22: Simulation and FPA results of queue 5, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.

Figure 4.23: Simulation and FPA results of queue 6, (a) Blocking probability using simulation and FPA, (b) Simulation confidence intervals and FPA blocking probability.
4.6 FPA for Stationary Erlang Loss Queueing Networks

As mentioned earlier, Erlang loss networks with Markovian branching do not have an exact closed-form solution even in the case of stationary arrival rates. There are some loss networks, however, with special branching probabilities that have a closed form solution but not in general. In this section, we show how the FPA algorithm provides can be used to calculate efficiently the exact solution for loss networks with general Markovian branching.

Let us consider the same Erlang loss queueing network considered in section 4.2 with time-independent Poisson stream of external arrivals at rate \( \lambda_i \), \( i = 1, 2, \ldots, N \).

Recall that, the state space of the system is

\[
S = \{ (n_1, n_2, \ldots, n_N) \in \mathbb{N}^N : 0 \leq n_i \leq s_i ; i = 1, 2, \ldots, N \}.
\]

Since the arrival and the service processes are stationary Poisson processes, the probability that the system is in state \( n \ (n \in S) \) at time \( t \), \( P(n_1, n_2, \ldots, n_N; t) \) becomes independent of time as \( t \) becomes sufficiently large. Let

\[
\lim_{t \to \infty} P(n_1, n_2, \ldots, n_N; t) = p(n_1, n_2, \ldots, n_N) \quad \text{for all} \quad (n_1, n_2, \ldots, n_N) \in S.
\]

Since \( P(n_1, n_2, \ldots, n_N; t) \) becomes independent of time for all \( n \in S \) as \( t \to \infty \), then

\[
\frac{d}{dt} P(n_1, n_2, \ldots, n_N; t) = 0 \quad \text{as} \quad t \to \infty.
\]

Consequently, the set of forward differential equations becomes a set of linear equations as follows:
\[0 = \sum_{i=1}^{N} \lambda_i p(\ldots, n_i-1, \ldots) + \sum_{i=1}^{N} (n_i+1) \mu_i q_i p(\ldots, n_i+1, \ldots) + \sum_{i=1}^{N} \sum_{j=1}^{N} (n_i+1) \mu_i q_{ij} p(\ldots, n_i+1, \ldots, n_j-1, \ldots) + \sum_{i=1}^{N} \sum_{j=1}^{N} (n_i+1) \mu_i q_{ij} p(\ldots, n_i+1, \ldots, s_j, \ldots) - \sum_{i=1}^{N} (\lambda_i + n_i \mu_i) p(n_1, n_2, \ldots, n_N). \] (4.7)

Such that:

\[p(n_1, n_2, \ldots, n_N) = 0 \text{ if } (n_1, n_2, \ldots, n_N; t) \notin S\]

\[\lambda_i = 0 \ \forall i \text{ if } n_i \in S \text{ and } n_i = s_i \ \forall i\]

\[\mu_i = 0 \ \forall i \text{ if } n_i = 0\]

\[\sum_{n \in S} p(n_1, n_2, \ldots, n_N) = 1\]

Then the stationary blocking probability of queue \(j\) at time \(t\) is

\[BP_j = \sum_{(n_i, \ldots, n_N; j) \in S} p(n_1, \ldots, n_j = s_j, \ldots, n_N)\]

and the stationary average number of busy servers in queue \(j\) is

\[E[Q_j] = \sum_{n \in S} n_j \sum_{(n_i, \ldots, n_N) \in S} p(n_1, \ldots, n_N)\]
Our proposed FPA algorithm in section 4.4 iterates on the set of differential equations (4.3) for the time-dependent average number of busy servers in each queue. Due to the stationary arrival and service rates of each queue, the average number of busy servers in each queue in the network will be constant as $t \to \infty$. Let

$$
\lim_{t \to \infty} E[Q_j(t)] = m_j, \ j = 1, 2, \ldots, N,
$$

which implies that

$$
\lim_{t \to \infty} \frac{d}{dt} E[Q_j(t)] = 0, \ j = 1, 2, \ldots, N.
$$

As a result, the set of differential equations (4.3) reduces to a set of linear equations as follows:

$$
0 = \left[ \lambda_j + \sum_{i=1}^{N} q_{ij} \mu_i m_i \right] (1 - BP_j) - \mu_j m_j, \ j = 1, 2, \ldots, N. \quad (4.8)
$$

Equations (4.8) can be rewritten as

$$
\frac{m_j}{1 - BP_j} = \left[ \lambda_j + \sum_{i=1}^{N} q_{ij} \mu_i m_i \right] \frac{1}{\mu_j}, \ j = 1, 2, \ldots, N. \quad (4.9)
$$

The offered load to the Erlang loss queue $j$, as $t \to \infty$, is

$$
\rho_j = \frac{m_j}{1 - BP_j}.
$$

Then, Equations (4.9) become

$$
\rho_j = \frac{\lambda_j}{\mu_j} + \sum_{i=1}^{N} q_{ij} \frac{\mu_i}{\mu_j} \rho_i (1 - BP_j), \ j = 1, 2, \ldots, N \quad (4.10)
$$
Since traffic equations (4.10) are linear equations with $BP_j$ given for all $j$, the algorithm of FPA does not require a numeric solution for (4.10). This feature of stationary rates makes the algorithm very fast. The FPA algorithm in this case becomes as follows:

1. Start with initial blocking probabilities for all $j$; $BP_j = 0$, $j = 1, 2, \ldots, N$
2. Set iteration counter $k = 0$.
3. Evaluate the following equations $\rho^k_j$ for $j = 1, 2, \ldots, N$
   \[
   \rho^k_j = \frac{\lambda_i}{\mu_j} + \sum_{i=1}^N q_{ij} \frac{1}{\mu_j} \rho^k_j (1 - BP^k_j), \quad j = 1, 2, \ldots, N
   \]
   and
   \[
   m^k_j = \rho^k_j (1 - BP^k_j), \quad j = 1, 2, \ldots, N.
   \]
4. Update blocking probabilities
   \[
   BP^k_j = \frac{[\rho^k_j]^s_j}{s_j!} \sum_{i=0}^{\infty} [\rho^k_j]^i / i!, \quad j = 1, 2, \ldots, N
   \]
5. Given tolerance $\epsilon$,
   If $\max_{j=1,2,\ldots,N} \left| BP^k_j - BP^k \right| < \epsilon$, then $BP^k$ converged for all $j$. Stop.
   Else, update the iteration counter $k = k + 1$. Go to step 3.

### 4.7 The FPA for Nonstationary Multi-Rate Erlang Loss Queueing Networks

Arrivals to each queue in the network follow $R$ independent Poisson processes with class $r$ customers arriving to queue $i$ with a time-dependent rate $\lambda_{ri}(t)$, $r = 1, 2, \ldots, R$ and $i = 1, 2, \ldots, N$. A class $r$ arrival requests a service that must be performed by a number of $b_{ri}$ servers and takes an exponential amount of time with mean $1/\mu_{ri}$, $r = 1, 2, \ldots, R$ and $i = 1, 2, \ldots, N$. All types of arrivals are assumed to be independent of each other. Upon
service completion of a class \( r \) customer at queue \( i \), the \( b_r \) servers in queue \( i \) will be released simultaneously and the customer may leave the network with probability \( q_{i0} \) or move to queue \( j \) for another service with probability \( q_{ij} \), \( i \) and \( j = 1, 2, \ldots, N \). Queue \( i \) in the network has a total of \( s_i, i = 1, 2, \ldots, N \), identical servers, and each can provide service to any class of arrivals. Let \( n \) be a vector of length \( NR \) where the component \( n_{ri} \) is the number of class \( r \) customers in queue \( i \) and let \( b = (b_1, b_2, \ldots, b_R) \). Then, the set of all possible states of the system can be described as

\[
S = \{ n \in \mathbb{N}^{NR} : \sum_{r=1}^{R} b_r n_{ri} \leq s_i, 0 \leq n_{ri} \leq s_i, r = 1, 2, \ldots, R, i = 1, 2, \ldots, N \}.
\]

The forward differential equations of probability that the system is in state \( n \) (\( n \in S \)) at time \( t \), \( P(n_{11}, n_{R1}, \ldots, n_{1N}, \ldots, n_{RN}; t) \) are

\[
\frac{d}{dt} P(n_{11}, n_{R1}, \ldots, n_{1N}, \ldots, n_{RN}; t) = \sum_{i=1}^{N} \sum_{r=1}^{R} \lambda_{ri}(t) P(\ldots, n_{ri} - 1, \ldots; t)
+ \sum_{i=1}^{N} \sum_{r=1}^{R} (n_{ri} + 1) \mu_{ri} q_{ii} P(\ldots, n_{ri} + 1, \ldots; t)
+ \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{r=1}^{R} (n_{ri} + 1) \mu_{ri} q_{ij} P(\ldots, n_{ri} + 1, \ldots, n_{rj} - 1, \ldots; t)
+ \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{r=1}^{R} \sum_{k=0}^{b_r-1} (n_{ri} + 1) \mu_{ri} q_{ij} P(\ldots, n_{ri} + 1, \ldots, n_{rj} = s_j - k, \ldots; t)
- \sum_{i=1}^{N} \sum_{r=1}^{R} (\lambda_{ri}(t) + n_{ri} \mu_{ri}) P(n_{11}, n_{R1}, \ldots, n_{1N}, \ldots, n_{RN}; t).
\] (4.11)
Given that

1. \( P(n; t) = 0 \) if \( n \not\in S \) for \( t > 0 \)

2. \( \lambda_r(t) = 0 \) if \( n \in S \) and \( \sum_{j=1}^{R} b_j n_{i_j} > s_j - b_r, r = 1, 2, \ldots, R, i = 1, 2, \ldots, N \) and \( t > 0 \)

3. \( \mu_{ri} = 0 \) if \( n_{ri} = 0, r = 1, 2, \ldots, R, i = 1, 2, \ldots, N \)

4. \( \sum_{n \in S} P(n; t) = 1 \quad \forall t \)

5. \( 0 \leq P(n; t) \leq 1 \quad \forall t \) and \( \forall (n_1, n_2, \ldots, n_R) \in S \)

6. \( P(0, 0, \ldots, 0; 0) = 1 \) and \( P(n; 0) = 0 \); \( \forall n \in S \)

It is worth mentioning that the number of equations grows exponentially fast as the number of queues, number of servers in each queue and the number of classes in the network increase. For instance, let us consider a network of \( N \) Erlang loss queues, where each queue has \( S \) servers and \( R \) classes of arrivals. Obtaining the performance measures as function of time for this network requires solving \( O(RS^N) \) differential equations simultaneously. In most real life problems, the number of queues and the number of servers in each queue are large. Therefore, working with equations (4.11) to obtain the performance measures will be not efficient.

In this section, we show how the FPA algorithm can be used to calculate the time-dependent blocking probabilities without the need to solve (4.11). To construct the multi-rate FPA algorithm, the arrival and departure rates have to be carefully studied. Consider one of the queues, say queue \( j \), in a multi-rate loss network. The rate into and rates out of queue \( j \) are presented in Figure 4.24. Since there are \( R \) classes of arrivals to queue \( j \), there are \( R \) differential equations describe the average number of customers of each class over time in queue \( j \).
From Figure 4.24, the average number of class $r$ customers in queue $j$, $E[Q_r(t)]$, is obtained by solving the following system of differential equations:

$$\frac{d}{dt} E[Q_r(t)] = \left[ \lambda_r(t) + \sum_{i=1}^{N} q_{ij} D_i(t) \right] (1 - BP_{rj}(t)) - \sum_{i=1}^{s_j} \mu_{ij} P\{n_{ij} = i; t\}$$

for $t > 0$, $r = 1, 2, \ldots, R$ and $j = 1, 2, \ldots, N$

By definition, the average number of class-$r$ customer in queue $j$, $E[Q_r(t)]$, is

$$E[Q_r(t)] = \sum_{i=1}^{s_j} i P\{n_{ij} = i; t\}, \quad t > 0, \quad j = 1, 2, \ldots, N$$

where $P\{n_{ij} = i; t\}$ is the marginal distribution of number of class $r$ customers in queue $j$. Then,

$$\mu_{rj} \sum_{i=1}^{s_j} i P\{n_{ij} = i; t\} = \mu_{rj} E[Q_r(t)] = D_r(t), \quad r = 1, 2, \ldots, R \text{ and } j = 1, 2, \ldots, N$$
This yield

\[
\frac{d}{dt} E[Q_\eta(t)] = \left[ \lambda_\eta(t) + \sum_{i=1}^N q_{ij} \mu_i E[Q_{i\eta}(t)] \right] \left( 1 - BP_{\eta}(t) \right) - \mu_{\eta} E[Q_{\eta}(t)].
\]

for \( t > 0, r = 1, 2, \ldots, R \) and \( j = 1, 2, \ldots, N \). (4.12)

The multi-rate FPA algorithm starts by assuming that the time-dependent steady-state probability of having \((n_{11}, \ldots, n_{R1}, \ldots, n_{1N}, \ldots, n_{RN})\) customers in the system is:

\[
P(n_{11}, \ldots, n_{R1}, \ldots, n_{1N}, \ldots, n_{RN}; t) = \prod_{i=1}^N \prod_{r=1}^R \frac{[\rho_{ij}(t)]^{n_{ir}}}{n_{ir}!} G_i(t), \quad n \in S, \quad t > 0.
\]

where

\[
G_i(t) = \sum_{r=1}^R \prod_{r=1}^R \frac{[\rho_{ij}(t)]^{n_{ir}}}{n_{ir}!}, \quad i = 1, 2, \ldots, N.
\]

So, the time-dependent marginal distribution of number of customers in queue \( j \) is:

\[
P(n_{1j}, \ldots, n_{Rj}; t) = \prod_{r=1}^R \frac{[\rho_{ij}(t)]^{n_{ir}}}{n_{ir}!} G_j(t),
\]

where \((n_{1j}, \ldots, n_{Rj}) \in \{n \in \mathbb{N}^R : \sum_{r=1}^R b_r n_{ir} \leq s_j, 0 \leq n_{ir} \leq s_j, r = 1, 2, \ldots, R \}, \quad t > 0.\)

The time-dependent offered load functions \( \rho_{ij}(t) \), \( r = 1, 2, \ldots, R \) and \( j = 1, 2, \ldots, N \), are obtained by solving the traffic equations (3.12) and substituting the solution in:

\[
\rho_{ij}(t) = \frac{E[Q_{ij}(t)]}{1 - BP_{ij}(t)}, \quad r = 1, 2, \ldots, R \text{ and } j = 1, 2, \ldots, N
\] (4.13)
Using Kaufman-Roberts recursive formula, the time-dependent probability of having exactly \( K \) customers in queue \( j \) at time \( t \) is:

\[
w_j(0; t) = 1 \text{ for } t \geq 0,
\]

\[
w_j(K; t) = \frac{1}{K} \sum_{r=1}^{B} \rho_j(t) b_j w_j(K - b_j; t), \quad K = 1, 2, \ldots, s_j, \quad j = 1, 2, \ldots, N, \quad t \geq 0
\]

Then, the time-dependent blocking probability of class-\( r \) arrivals at queue \( j \) is

\[
BP_{r,j}(t) = \frac{\sum_{i=0}^{s_j} w_j(i; t)}{\sum_{i=0}^{s_j} w_j(i; t)}, \quad r = 1, 2, \ldots, R, \quad j = 1, 2, \ldots, N \text{ and } t \geq 0 \quad (3.14)
\]

Given the above assumption, the FPA algorithm starts with zero initial blocking probabilities for each class of arrivals at every queue in the loss network. Given the initial blocking probabilities, the traffic equations \((4.12)\) are solved numerically to obtain \( E[Q_{r,j}(t)] \) for all \( r \) and \( j \). According to equation \((4.13)\), the initial time-dependent offered load of queue \( j \), \( \rho_{r,j}(t) \), is equal to time-dependent average number of class \( r \) customers in queue \( j \), \( E[Q_{r,j}(t)] \), for all \( r \) and \( i \). With the values of \( \rho_{r,j}(t) \) for all \( r \) and \( j \) on hand, the first approximation of the time-dependent blocking probabilities for each class at each queue is calculated using \((4.14)\). Next, the time-dependent offered load of each queue is updated using equation \((4.13)\) and updated blocking probabilities are obtained by \((4.14)\). The algorithm continues in this manner until the blocking probability of each class at each queue converges with tolerance \( \varepsilon \). To maintain the same level of tolerance for all classes and queues, the second norm of the difference between the blocking probability of class \( r \) customers at queue \( j \), \( BP_{r,j}(t) \), corresponds to iteration \( k \) and \( k+1 \) is calculated for all classes at each queue. Then, the FPA algorithm stops when the maximum of the norms is less than tolerance value \( \varepsilon \).
The Algorithm:

1. Choose an appropriate $\varepsilon, \Delta t$ and the final time of integration (say $T_f$).
2. Evaluate $\lambda_r(t)$ at $t = 0, \Delta t, 2\Delta t, \ldots, T_f$ for all $t, r = 1, 2, \ldots, R$ and $j = 1, 2, \ldots, N$.
3. Choose initial conditions for $E[Q_r(0)]$, say $E[Q_r(0)] = 0$, $r = 1, 2, \ldots, R$ and $j = 1, 2, \ldots, N$.
4. Set the iteration counter $k = 0$.
5. Initialize the blocking probabilities $BP_{rj}^k(t) = 0$ for all $t, r = 1, 2, \ldots, R$ and $j = 1, 2, \ldots, N$.
6. Solve the following system for $E^k[Q_r(t)]$,
   \[ E^k[Q_r(t+\Delta t)] = E^k[Q_r(t)] + \left[ \lambda_r(t) + \sum_{i=1}^{N} q_{ij} \mu_i E^k[Q_i(t)] \right](1 - BP_{rj}^k(t))\Delta t 
   - \mu_i E^k[Q_r(t)]\Delta t, \]
   for $t = 0, \Delta t, 2\Delta t, \ldots, T_f$, $r = 1, 2, \ldots, R$ and $j = 1, 2, \ldots, N$.
7. Calculate
   \[ \rho_{rj}^k(t) = \frac{E^k[Q_r(t)]}{1 - BP_{rj}^k(t)}, \]
   $t = 0, \Delta t, 2\Delta t, \ldots, T_f$, $r = 1, 2, \ldots, R$ and $j = 1, 2, \ldots, N$.
8. Set $w_r(0; t) = 1$ for all $t, j = 1, 2, \ldots, N$.
9. Compute the recursion
   \[ w_r(K; t) = \frac{1}{K} \sum_{i=1}^{K} \rho_{rj}^k(t) b_i w_r(K - b_i; t), \]
   $j = 1, 2, \ldots, N$, for $t = 0, \Delta t, 2\Delta t, \ldots, T_f$, $K = 1, 2, \ldots, s_j$ and $j = 1, 2, \ldots, N$.  

101
10. Update the blocking probabilities

$$BP_{rj}^{k+1}(t) = \frac{\sum_{i=0}^{s_j} w_j(i;t)}{\sum_{i=0}^{s_j} w_j(i;t)} \quad r = 1, 2, \ldots, R, j = 1, 2, \ldots, N$$

11. If $\max_{r=1,2,\ldots,R, j=1,2,\ldots,N} \left\| BP_{rj}^{k+1}(t) - BP_{rj}^{k}(t) \right\| < \varepsilon$ then $BP_{rj}^{k}(t)$ has converged for all $r$ and $j$, STOP. Else, set $k = k + 1$ and go to step 6.

### 4.7.1 A Numerical Example

To measure the accuracy of FPA for multi-rate loss networks, the network of six loss queues in Figure 3.9 is considered with three classes of arrivals. The service rates $\mu_{rj}$ of servicing class-$r$ arrivals at queue $j$, for all $r = 1, 2, \ldots, R$ and $j = 1, 2, \ldots, N$, are represented in matrix $M$ where $M_{rj} = \mu_{rj}$ for all $r$ and all $j$. It is assumed that each queue in the network has ten servers. The arrival rate of each class to network, the required number of servers, the branching probabilities matrix $B$ and the service rate matrix $M$ are given as follows:

Class 1: $b_1 = 1$, $\lambda_{11}(t) = 20 + 15 \sin(t)$ and $\lambda_{1i}(t) = 0, \ i = 2,3,4,5,6$,

Class 2: $b_2 = 2$, $\lambda_{22}(t) = \lambda_{26}(t) = 15 + 10 \sin(2t)$ and $\lambda_{2i}(t) = 0, \ i = 1,3,4,5$

Class 3: $b_3 = 3$, $\lambda_{35}(t) = 10 + 10 \sin(3t)$ and $\lambda_{3i}(t) = 0, \ i = 1,2,3,4,6$

$$B = \begin{bmatrix} 0.2 & 0.4 & 0.4 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0.5 & 0.4 & 0 \\ 0.3 & 0 & 0.1 & 0.3 & 0 & 0 \\ 0 & 0 & 0.7 & 0 & 0 & 0.3 \\ 0 & 0.6 & 0 & 0 & 0.1 & 0.3 \\ 0.5 & 0 & 0 & 0.3 & 0.2 & 0 \end{bmatrix}$$

and

$$M = \begin{bmatrix} 4 & 4 & 3 & 4 & 2 & 6 \\ 3 & 4 & 2 & 5 & 8 & 4 \\ 4 & 4 & 5 & 7 & 5 & 6 \end{bmatrix}$$
According to the above parameters, the underlying multi-rate loss network is represented in Figure 4.25.

![Figure 4.25: A six-queue Erlang loss network with three classes of arrivals](image)

The FPA algorithm is executed for the network in Figure 4.25 with $\Delta t$, $T_f$ and $\varepsilon$ equal to 0.01, 20 and 0.01, respectively. The time-dependent blocking probability of each class at each queue in the network is compared with its corresponding output from simulation. All queues in the network are taken to be empty initially in the FPA algorithm. The simulation model is executed with 1000 replications. Each replication starts with empty and idle queues and spans for 20 units of time with $\delta$ chosen to be 0.01. The results from FPA and simulation output in the interval [0,10] are deleted to avoid the effect of initial conditions in the FPA algorithm and the warm-up period in simulation.

Table 4.5 summarizes the average and relative errors in blocking probabilities using FPA with respect to simulation. The small average error of blocking probabilities in Table 4.5 shows that the FPA provides a good approximation for all classes at each queue. In addition, most queues have the relative error of blocking probability that is
below 10% except for class 1 and class 2 blocking at queue 4 which is due to the small average error for both classes at queue 4.

Table 4.5: Error in blocking probabilities using FPA for six-queue multi-rate loss network in Figure 4.25

<table>
<thead>
<tr>
<th>Queue</th>
<th>Classe 1</th>
<th></th>
<th>Classe 2</th>
<th></th>
<th>Classe 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error</td>
<td>Rel. Error</td>
<td>Error</td>
<td>Rel. Error</td>
<td>Error</td>
<td>Rel. Error</td>
</tr>
<tr>
<td>1</td>
<td>0.0102</td>
<td>5.04</td>
<td>0.0136</td>
<td>3.52</td>
<td>0.0167</td>
<td>3.02</td>
</tr>
<tr>
<td>2</td>
<td>0.0119</td>
<td>4.58</td>
<td>0.0156</td>
<td>3.30</td>
<td>0.0153</td>
<td>2.39</td>
</tr>
<tr>
<td>3</td>
<td>0.0142</td>
<td>7.98</td>
<td>0.0268</td>
<td>7.40</td>
<td>0.0358</td>
<td>6.69</td>
</tr>
<tr>
<td>4</td>
<td>0.0041</td>
<td>20.62</td>
<td>0.0063</td>
<td>11.58</td>
<td>0.008</td>
<td>7.25</td>
</tr>
<tr>
<td>5</td>
<td>0.0101</td>
<td>5.65</td>
<td>0.0139</td>
<td>4.02</td>
<td>0.0167</td>
<td>3.36</td>
</tr>
<tr>
<td>6</td>
<td>0.01</td>
<td>5.10</td>
<td>0.0126</td>
<td>3.93</td>
<td>0.0122</td>
<td>2.47</td>
</tr>
</tbody>
</table>

This example has six types of blocking at each class of arrivals. Thus, there are eighteen different types of blocking probabilities. To see performance of FPA over time, class 3 blocking probabilities are plotted in Figure 4.26 since class 3 arrivals request the largest number of servers. The plots in Figure 4.26 show that FPA still performs well in following the simulation output. The FPA curves are almost exact for all queues and romaine within the bounds of the 95% confidence intervals. The worst performance of FPA is for class 3 blocking probability at queue 3. However, the blocking probability curve of FPA for class 3 at queue 3 stays at the lower bound of the 95% confidence interval meaning that FPA still acceptable with acceptable average and relative errors of 0.0358 and 6.69%, respectively.
(a) Class-3 blocking probability at queue 1  
(b) Class-3 blocking probability at queue 2  
(c) Class-3 blocking probability at queue 3  
(d) Class-3 blocking probability at queue 4  
(e) Class-3 blocking probability at queue 5  
(f) Class-3 blocking probability at queue 6

Figure 4.26: Simulation and FPA results blocking for class-3 arrivals
Chapter 5

Constrained Networks of Nonstationary Erlang Loss Queues

5.1 Introduction

In the previous chapter, the nonstationary Erlang loss networks were analyzed assuming that the number of customers at any queue in the network is independent of the number of customers at any other queue. In other words, the only restriction on number of customers in queue \( j \) is that it does not exceed the number of servers at queue \( j \). In practice, there are many loss systems where the number of customers at queue \( j \) is subject to other restrictions in addition to being less than the number of servers at queue \( j \). In this case, we say that the system has a constrained population. The main focus of this chapter is to obtain time-dependent performance measures for the nonstationary Erlang loss networks with a constrained population.
The telecommunication network can be modeled as an Erlang loss network with constrained population. Consider a simple telephone network consists of central office A (CO-A) and B (CO-B), as shown in Figure 5.1. There are three different types of calls: local calls within CO-A, local calls within CO-B area, and long distance calls between CO-A and CO-B. At any point in time, the total number of local and long distance calls that CO-A can support must not exceed its capacity, $C_A$. Similarly, the total number of local and long distance calls that CO-B can support must not exceed its capacity, $C_B$. Finally, the total long distance calls must not exceed the trunk line capacity, $C_{AB}$. Let $n_A$ and $n_B$ be the number of calls which have both callers attached to CO-A and CO-B, respectively, and let $n_{AB}$ be the number of calls which have a caller attached to CO-A and a caller attached to CO-B. Then, vector $(n_A, n_B, n_{AB})$ has to satisfy the following constraints

\[
\begin{align*}
    n_A + n_{AB} & \leq C_A, \\
    n_B + n_{AB} & \leq C_B, \\
    n_{AB} & \leq C_{AB}, \\
    n_A, n_B, n_{AB} & \geq 0.
\end{align*}
\]

Figure 5.1: Simple telephone network
5.2 A Constrained Network of Nonstationary Erlang Loss Queues

Consider a queueing network consisting of $N$ queueing nodes. Node $i$, $i = 1, 2, \ldots, N$, in the network is an Erlang loss queue with $s_i$ identical and independent servers. Customers arrive to the network in different classes each with a time-dependent Poisson process with rate $\lambda_r(t)$, $r = 1, 2, \ldots, R$. A class $r$ customer requests a job that has to be completed by a subset, $J_r$, of queues simultaneously within an exponential service time with mean $1/\mu_r$. In addition, a class $r$ job requires $b_{jr}$ servers simultaneously at queue $j$, $j \in J_r$. A class $r$ arrival is lost if there is some queue $j$, $j \in J_r$, such that the number of idle servers at queue $j$ is less than $b_{jr}$.

The requirements of arrivals are summarized in matrix $A = [a_{jr}]$, such that $a_{jr} = b_{jr}$ if $j \in J_r$ and $a_{jr} = 0$ otherwise, for $j = 1, 2, \ldots, N$ and $r = 1, 2, \ldots, R$. Let $n_r$ be the number of class $r$ customers in the network. Then, the state space of the system is

$$S = \{ (n_1, n_2, \ldots, n_R) \in \mathbb{N}^R : \sum_{j=1}^R a_{jr} n_j \leq s_j, n_r \geq 0 ; j = 1, 2, \ldots, N \text{ and } r = 1, 2, \ldots, R \}.$$

In matrix notation, let $s = (s_1, s_2, \ldots, s_N)$. Then,

$$S(s) = \{ n \in \mathbb{N}^R : An^T \leq s^T , n \geq 0 \}.$$

Since the arrival and the service processes are Poisson, the probability that the system is in state $n$ ($n \in S$) at time $t$, $P(n_1, n_2, \ldots, n_R ; t)$, is described by the following set of forward differential equations:

$$\frac{d}{dt} P(n_1, n_2, \ldots, n_R ; t) = \sum_{r=1}^R \lambda_r(t) P(\ldots, n_r - 1, \ldots ; t) + \sum_{r=1}^R (n_r + 1) \mu_r P(\ldots, n_r + 1, \ldots ; t)$$

$$- \sum_{r=1}^R (\lambda_r(t) + n_r \mu_r) P(n_1, n_2, \ldots, n_R ; t)$$

(5.1)
Given that

1. \( P(n; t) = 0 \) if \( n \not\in S \) for \( t > 0 \)
2. \( \mu_r = 0 \) if \( n_r = 0, r = 1, 2, \ldots, R \)
3. \( \sum_{n \in S} P(n; t) = 1 \) \( \forall t \)
4. \( 0 \leq P(n; t) \leq 1 \) \( \forall t \) and \( \forall (n_1, n_2, \ldots, n_R) \in S \)

Then the time-dependent blocking probability of class \( r \) at time \( t \) is

\[
BP_r(t) = 1 - \sum P(n_r, n_2, \ldots, n_R; t)
\]

where the summation is over all \( n \) such that none of the queues in \( J_r \) is blocked. This is written in a set form as, \( n \in S(s^T - Ae_r)^T \) where \( e_r \) is a unite vector of length \( R \), such that \( e_{ri} = 1 \) if \( i = r \) and zero otherwise. The time-dependent average number class \( r \) is

\[
E[Q_r(t)] = \sum_{n_r \geq 0} n_r \sum_{(n_1, n_2, \ldots, n_R) \in S} P(n_1, n_2, \ldots, n_R; t)
\]

There is no exact closed-form solution in the literature for the Erlang loss network described above. A special case of this system is when \( \lambda_r(t) = \lambda_r, r = 1, 2, \ldots, R \) for all \( t \). As a result, \( d/dt[P(n_1, n_2, \ldots, n_R; t)] \rightarrow 0 \) as \( t \rightarrow \infty \) for all \( (n_1, n_2, \ldots, n_R) \in S(c) \). The above system of differential equations reduces to a set of linear algebraic equations that provides the following product form solution:

\[
P(n_1, n_2, \ldots, n_R) = \frac{1}{G} \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!}, \quad (n_1, n_2, \ldots, n_R) \in S(s),
\]

\[
G = \sum_{n \in S} \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!} \quad \text{and} \quad \rho_r = \frac{\lambda_r}{\mu_r}, \quad r = 1, 2, \ldots, R.
\]

(5.2)
Some methods use the solution of the stationary Erlang loss networks to obtain an approximate closed-form solution for the nonstationary Erlang loss networks. For example, PSA uses the instantaneous offered load in 5.2. The other example is the MOL approximation which uses the offered load of the infinite-server queue in 5.2. As discussed earlier in this thesis, each of these methods performs better under some condition. In this chapter, we introduce a numerical method for solving constrained nonstationary Erlang loss Networks.

5.3 The FPA for Constrained Networks of Nonstationary Erlang Loss Queues

Consider the nonstationary Erlang loss network described in section 5.2 with state space $S(s)$. In this network, a customer requesting path $r$ at time $t$ is admitted if there $a_r$ idle server at each queue along path $r$. If this condition is satisfied, the customer gets immediately served by all servers along path $r$ simultaneously. Otherwise, this customer is blocked and lost. The rate of change in the average number of class $r$ customers, $E[Q_r(t)]$, at time $t$ is the difference between effective arrival rate and the departure rate of class $r$ customers at time $t$. Given that the blocking probability of class $r$ customers at time $t$ is $BP_r(t)$, the time-dependent average number of class $r$ customers is obtained by solving the differential equation

$$\frac{d}{dt} E[Q_r(t)] = \lambda_r(t)(1 - BP_r(t)) - \mu_r E[Q_r(t)], \quad t > 0.$$  \hspace{1cm} (5.3)

Needless to say that, in most of the cases, the time-dependent blocking probability of class $r$ customers is not available. However, an iterative scheme can be developed such that it starts with an initial blocking probability function for class $r$ customers and
improves the blocking probability as the algorithm progresses using differential equation (5.3).

Assume that each Erlang loss queue in the network is independent of the others. In addition, we assume that a class $r$ customer gets serviced in the queues on its path sequentially as opposed to simultaneously. This means that the customer gets its service in the first queue in its path then it moves to the second queue in its path to get the second service and so on. According to these assumptions, each queue in the network will be considered as a multi-rate Erlang loss queue. Next, each queue is solved separately to obtain the blocking probabilities of all classes at each queue. As a result, class $r$ customers will have the set $\{ \pi_{rj}(t) : j \in J_r \}$ of blocking probability where $\pi_{rj}(t)$ is the time-dependent blocking probability of class $r$ customers at queue $j$. The solution of each queue separately requires the time-dependent offered load, $\rho_{rj}$, of class $r$ customers at queue $j$, $j \in J_r$. Due to the independence among the queues, the offered load of class $r$ customers at queue $j$, $j \in J_r$, is

$$\rho_{rj}(t) = \frac{E[Q_r(t)]}{1 - \pi_{rj}(t)} , \quad r = 1, 2, \ldots, R \quad \text{and} \quad j \in J_r. \quad (5.4)$$

where $E[Q_r(t)]$ is the numerical solution of the differential equations (5.3). Finally, the probability that a class $r$ customer is lost at time $t$, $BP_r(t)$, is given by

$$BP_r(t) = \Pr\{\text{there is some queue } j \text{ that is not available for class } r \text{ customer } | \ j \in J_r\}$$

$$= 1 - \Pr\{\text{all queues in } J_r \text{ are available for class } r \text{ customer}\}$$

$$= 1 - \prod_{j \in J_r} (1 - \pi_{rj}(t)) \quad \text{by the independence assumption.}$$

To illustrate how the algorithm works, consider the telephone network in Figure 5.1. According to our assumption, each link in Figure 5.1 is a multi-rate Erlang loss queue. From Figure 5.1, CO-A is used by class A and class AB callers. Thus, CO-A link is
assumed to be an independent multi-rate Erlang loss queue with rates $\lambda_A(t)$ and $\lambda_{AB}(t)$. Similarly, the trunk line link is assumed to be an independent multi-rate Erlang loss queue with rates $\lambda_{AB}(t)$ and CO-B is assumed to be an independent multi-rate Erlang loss queue with rates $\lambda_B(t)$ and $\lambda_{AB}(t)$. To simplify the notation, let CO-A, trunk line, and CO-B be denoted as queue 1, queue 2 and queue 3, respectively. Then, the modified Erlang loss network is depicted in Figure 5.2.

![Figure 5.2: Independent multi-rate Erlang loss queues of the network in Figure 5.1](image)

From Figure 5.1, the set of queues in the route of class A calls is $J_A = \{1\}$, the set of queues in the route of class B calls is $J_B = \{3\}$ and the set of queues in the route of class AB calls is $J_{AB} = \{1,2,3\}$. The requirement matrix $A$ is

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$ 

Then, the state space is

$$S = \{(n_A, n_B, n_{AB}) \in \mathbb{N}^3 : n_A + n_{AB} \leq C_A, n_B + n_{AB} \leq C_B, n_{AB} \leq C_{AB}, n_A, n_B, n_{AB} \geq 0 \}.$$
Also,

\[ S(s) = \{ \mathbf{n} \in \mathbb{N}^3 : \mathbf{An}^T \leq \mathbf{c}^T, \mathbf{n} \geq 0 \} \] where \( \mathbf{c} = (C_A, C_B, C_{AB}) \).

The differential equations of average number customers of each class are

\[
\frac{d}{dt} E[Q_A(t)] = \lambda_A(t) (1 - BP_A(t)) - \mu_A E[Q_A(t)],
\]

\[
\frac{d}{dt} E[Q_B(t)] = \lambda_B(t) (1 - BP_B(t)) - \mu_B E[Q_B(t)],
\]

\[
\frac{d}{dt} E[Q_{AB}(t)] = \lambda_{AB}(t) (1 - BP_{AB}(t)) - \mu_{AB} E[Q_{AB}(t)].
\]

(5.5)

The proposed algorithm begins by assuming that the network is an infinite-server network which means that there is no blocking at all \( t \) for all classes of arrivals. Therefore,

\[ BP_A(t) = BP_B(t) = BP_{AB}(t) = 0, \quad t \geq 0 \]

These initial blocking probabilities make the set of differential equation (5.5) numerically solvable for \( E[Q_A(t)] \), \( E[Q_B(t)] \) and \( E[Q_{AB}(t)] \). Next, the three independent nonstationary multi-rate Erlang loss queues in Figure 5.3 are solved using the following parameters

Queue 1: \( \rho_{A,1}(t) = \frac{E[Q_A(t)]}{1 - \pi_{A,1}(t)}, \quad b_{A,1} = 1 \) and \( \rho_{AB,1}(t) = \frac{E[Q_{AB}(t)]}{1 - \pi_{AB,1}(t)}, b_{AB,1} = 1 \),

Queue 2: \( \rho_{AB,1}(t) = \frac{E[Q_{AB}(t)]}{1 - \pi_{AB,2}(t)}, \quad b_{AB,2} = 1, \)

Queue 3: \( \rho_{B,1}(t) = \frac{E[Q_B(t)]}{1 - \pi_{B,3}(t)}, \quad b_{B,3} = 1, \) and \( \rho_{AB,1}(t) = \frac{E[Q_{AB}(t)]}{1 - \pi_{AB,3}(t)}, b_{AB,3} = 1, \)

(5.6)

where \( b_j \) is number of servers required by class \( r \) calls at queue \( j \).
Recall that we have assumed initially that there is no blocking in the network. This implies that \( \pi_{A,1}(t) = \pi_{AB,1}(t) = \pi_{AB,2}(t) = \pi_{B,3}(t) = \pi_{AB,3}(t) = 0 \), for all \( t \), as initial values. Therefore, the initial offered loads are:

\[
\rho_{A,1}(t) = E[Q_A(t)] , \quad \rho_{AB,1}(t) = E[Q_{AB}(t)] , \\
\rho_{AB,2}(t) = E[Q_{AB}(t)] \\
\rho_{B,3}(t) = E[Q_B(t)] , \quad \rho_{AB,3}(t) = E[Q_{AB}(t)]
\]

The solution of independent queues will provide new functions for the blocking probabilities \( \pi_{A,1}(t) \), \( \pi_{AB,1}(t) \), \( \pi_{AB,2}(t) \), \( \pi_{B,3}(t) \) and \( \pi_{AB,3}(t) \). Next, the functions of blocking probabilities of the network are updated as follows:

\[
BP_A(t) = 1 - (1 - \pi_{A,1}(t)) , \\
BP_B(t) = 1 - (1 - \pi_{B,1}(t)) , \\
BP_{AB}(t) = 1 - (1 - \pi_{AB,1}(t)) (1 - \pi_{AB,2}(t)) (1 - \pi_{B,3}(t)) . \tag{5.7}
\]

The next iteration uses the network blocking probabilities, \( BP_A(t) \), \( BP_B(t) \) and \( BP_{AB}(t) \), obtained by equations (5.7) to substitute them in the differential equations (5.5). The set of differential equations are solved numerically to get the updated time-dependent average number of calls for each class. Next, the time-dependent averages and the current function of \( \pi_{A,1}(t) \), \( \pi_{AB,1}(t) \), \( \pi_{AB,2}(t) \), \( \pi_{B,3}(t) \) and \( \pi_{AB,3}(t) \) are used in equations (5.6). The new system of independent Erlang loss queues is resolved to obtain the new functions of \( \pi_{A,1}(t) \), \( \pi_{AB,1}(t) \), \( \pi_{AB,2}(t) \), \( \pi_{B,3}(t) \) and \( \pi_{AB,3}(t) \). The algorithm continues in this manner until the network blocking probabilities, \( BP_A(t) \), \( BP_B(t) \) and \( BP_{AB}(t) \), converge with \( \epsilon \) tolerance.
The FPA algorithm can be summarized for any constrained network of nonstationary Erlang loss queues in the following steps:

1. Choose an appropriate $\varepsilon, \Delta t$ and the final time of integration (say $T_f$).
2. Evaluate $\lambda_r(t)$ at $t = 0, \Delta t, 2\Delta t, \ldots, T_f$ for all $t, r = 1, 2, \ldots, R$.
3. Choose initial conditions for $E[Q_r(0)]$, say $E[Q_r(0)] = 0$, $r = 1, 2, \ldots, R$.
4. Set the iteration counter $k = 0$.
5. Initialize the blocking probabilities
   
   \[ BP_r^k(t) = 0 \text{ for all } t, r = 1, 2, \ldots, R, \text{ and} \]
   \[ \pi^k_r(t) = 0 \text{ for all } t, r = 1, 2, \ldots, R, \text{ and } j \in J_r. \]

6. Solve the following system for $E^k[Q_r(t)]$,
   
   \[
   E^k[Q_r(t+\Delta t)] = E^k[Q_r(t)] + \lambda_r(t) \left( 1 - BP_r^k(t) \right) \Delta t - \mu_r E^k[Q_r(t)] \Delta t, \]
   \[ \text{for } t = 0, \Delta t, 2\Delta t, \ldots, T_f \text{ and } r = 1, 2, \ldots, R \]

7. Calculate $\rho^k_j(t) = \frac{E^k[Q_r(t)]}{1 - \pi^k_r(t)}$, $t = 0, \Delta t, 2\Delta t, \ldots, T_f$, $r = 1, 2, \ldots, R$ and $j \in J_r$.

8. Set $w_j(0; t) = 1$ for all $t, j \in J_r$.

9. For $t = 0, \Delta t, 2\Delta t, \ldots, T_f$, $K = 1, 2, \ldots, s_j$ and $j \in J_r$, compute the recursion
   
   \[
   w_j(K; t) = \frac{1}{K} \sum_{j=1}^K \rho^k_j(t) b_j w_j(K - b_j; t), \quad j \in J_r
   \]

10. Update the independent queues blocking probabilities
    
    \[
    \pi^{k+1}_r(t) = \frac{\sum_{j=1}^{s_j} w_j(i; t)}{\sum_{j=1}^{s_j} w_j(t; t)}, \quad r = 1, 2, \ldots, R, j \in J_r.
    \]
11. Update the network blocking probabilities

\[ BP_i^{k+1} = 1 - \prod_{j \in J} (1 - \pi_{ij}^{k+1}(t)) \quad r = 1, 2, ..., R. \]

12. If \( \max_{r=1,2,...,R} \left\{ \|BP_i^{k+1}(t) - BP_i^k(t)\| \right\} < \epsilon \) then \( BP_i^k(t) \) has converged for all \( r \) and \( j \in J_r \); STOP.

Else, set \( k = k + 1 \) and go to step 6.

5.4 Applications and Numerical Examples

In this section, we analyze the error of FPA for constrained networks of nonstationary Erlang loss queues by comparing FPA against simulation for two examples. The FPA method in this chapter is proposed to solve nonstationary Erlang loss networks under any constraints. The accuracy measure that is used is the average absolute error and the relative absolute error between the simulation output and the approximation method at time \( t \) defined in section 4.5. In this analysis, we focus on the time-dependent blocking probability of each class of arrivals to the network.

The results of the FPA algorithm given in this section were obtained with the parameters \( \Delta t, T_f \) and \( \epsilon \) equal to 0.01, 20 and 0.01, respectively. All queues in the network were assumed to be empty initially in the FPA algorithm. The independent replications method was used to obtain the simulation results with the total number of independent replications set to 1000. Each replication starts with an empty and idle system and spans for 20 units of time. The results from FPA as well as the simulation output in the interval \([0, 10]\) were deleted to avoid the effect of initial conditions in the FPA algorithm and the warm-up period in simulation. Both FPA and simulation were used to calculate the time-dependent blocking probability of each queue in the network.
5.4.1 A Traffic-Groomed Tandem Nonstationary Optical Network

WDM Optical networks will eventually replace the backbone of the. The optical networks have been studied extensively assuming stationary traffic. In this section, we consider a special WDM optical network with traffic-grooming under nonstationary traffic. Washington and Perros [37] presented a decomposition algorithm for solving this problem under stationary arrival rates.

The traffic-groomed optical network consists of a number of optical nodes arranged in tandem, each node is an add/drop multiplexer (ADM). Each two adjacent ADMs are linked with each other with a single link called a *wavelength*. The wavelength is divided into *sub-wavelength units* to allow unused bandwidth to be available for other traffic. This technique is known as traffic grooming. Calls arrive to the network in different classes according to their source-destination optical nodes and number of sub-wavelength required per class. A call of class *r* from source node *i* to destination node *j* occupies *s_r* sub-wavelengths units from every link between node *i* and node *j* simultaneously. A class *r* is lost if there is at least one link between node *i* and node *j* has idle sub-wavelength units that are strictly less than *s_r*. Refer to Washington and Perros [37] for more formulation details.

To put this system in the context of queueing networks, each link between two adjacent ADMs is modeled as an Erlang loss queue. Therefore, a *N*-node optical network is modeled as (*N*–1)-node network of Erlang loss queues. The sub-wavelengths in each optical node are the servers of each Erlang loss queue. The number of classes of arrivals to the queueing network is equal to the number of all possible source-destination paths. Below, we give numerical example to see how well the FPA method behaves.
5.4.1.1 A Numerical Example

Consider an optical network with six optical nodes connected in tandem. Each link between two adjacent ADMs is a single wavelength that is groomed to carry 12 sub-wavelength units. All possible source-destination requests are shown in Figure 5.3.

![Diagram of a six-node tandem optical network with all source-destination arrivals](image)

Figure 5.3: A six-node tandem optical network with all source-destination arrivals

Calls of length one hop, two hops and four hops require exactly one unit of sub-wavelength units from each wavelength along their source to destination paths. Calls of length three hops require two units of sub-wavelength and five-hop calls require three units of sub-wavelength from each wavelength along their source to destination paths. The service rate is the same for all wavelengths and is equal to 100. The arrival rate of each class is as follows:

One hop arrival rates: \( \lambda_{12}(t) = \lambda_{23}(t) = \lambda_{34}(t) = \lambda_{45}(t) = \lambda_{56}(t) = 100 + 80 \sin(t) \).

Two hops arrival rates: \( \lambda_{13}(t) = \lambda_{24}(t) = \lambda_{35}(t) = \lambda_{46}(t) = 60 + 40 \sin(1.5t) \).

Three hops arrival rates: \( \lambda_{14}(t) = \lambda_{25}(t) = \lambda_{36}(t) = 40 + 40 \sin(t) \).
Four hops arrival rates: \( \lambda_{15}(t) = \lambda_{26}(t) = 40 + 40 \sin(2t) \).

Five hops arrival rate: \( \lambda_{16}(t) = 20 + 20 \sin(t) \).

Define the random variable \( n_{ij} \) as the number of calls that originates at optical node \( i \) and terminates at optical node \( j \). Then, the state space of the network is

\[
S(s) = \{ n \in \mathbb{N}^{15} : An^T \leq s^T, n \geq 0 \}
\]

where \( n = (n_{12}, n_{23}, n_{34}, n_{45}, n_{56}, n_{13}, n_{24}, n_{35}, n_{46}, n_{14}, n_{25}, n_{56}, n_{15}, n_{26}, n_{16}) \),

\[
A = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 3 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 2 & 2 & 0 & 1 & 1 & 3 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 2 & 2 & 2 & 1 & 1 & 3 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 2 & 2 & 1 & 1 & 3 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 1 & 3
\end{bmatrix}
\]

\[
s^T = \begin{bmatrix}
12 \\
12 \\
12 \\
12 \\
12
\end{bmatrix}
\]

Due to the equality in arrival rates among the classes, some of classes have the same blocking probability function. For example, the class (1,2) blocking probability is equal to the class (5,6) blocking probability. Figures 5.4 and 5.5 show the results for some representative classes. Each graph plots the approximate blocking probability and the 95\% confidence interval of simulation for the selected classes.

We note that the blocking probability function has high values for the classes that require a higher number of sub-wavelengths. This is expected due to the fact that classes requesting more than one sub-wavelength are more likely to be blocked. In addition, the blocking probability increases as the number hops increases. Therefore, calls with shorter paths are more likely to go through than those with longer paths. For example, class (2,3) calls require one hop and they have a blocking probability with maximum value of about 0.05, whereas class (2,4) calls require two hops and they have a blocking
probability with maximum value of about 0.13 as can be seen in plots (b) and (d) in Figure 5.4.

The FPA results in most of the plots in Figures 5.4 and 5.4 remains within the 95% confidence intervals, though they seem to be closer to the upper bound of the confidence interval. This means that the FPA tends to over estimate the blocking probabilities.

Figure 5.4: Blocking probability using FPA and the 95% confidence interval of simulation for the classes (1,2), (2,3), (1,3) and (2,4)
Figure 5.5: Blocking probability using FPA and the 95% confidence interval of simulation for the classes (1,4), (2,5), (1,5) and (1,6)

Table 5.1 summarizes the average blocking probability using simulation and the FPA method for each class. Table 5.1 also shows the average absolute and relative error between the simulation and FPA. The overall accuracy of the FPA is good. Although, the relative error seems high in some classes, the average absolute error shows that FPA provides a good approximation. One has to remember that the relative error tends to be large when very small numbers, such as the blocking probability function of class (1,2) and (5,6), are being compared.
Table 5.1: Simulate and approximate average blocking probability and relative and average absolute error for each class in the optical network in Figure 5.4.

<table>
<thead>
<tr>
<th>Class</th>
<th>Average $BP$ (Simulation)</th>
<th>Average $BP$ (Approximation)</th>
<th>Average Abs. Error</th>
<th>Relative Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>0.0012</td>
<td>0.0020</td>
<td>0.0009</td>
<td>77.09</td>
</tr>
<tr>
<td>2-3</td>
<td>0.0108</td>
<td>0.0123</td>
<td>0.0024</td>
<td>22.33</td>
</tr>
<tr>
<td>3-4</td>
<td>0.0249</td>
<td>0.0246</td>
<td>0.0030</td>
<td>12.19</td>
</tr>
<tr>
<td>4-5</td>
<td>0.0108</td>
<td>0.0123</td>
<td>0.0024</td>
<td>21.87</td>
</tr>
<tr>
<td>5-6</td>
<td>0.0012</td>
<td>0.0020</td>
<td>0.0009</td>
<td>75.33</td>
</tr>
<tr>
<td>1-3</td>
<td>0.0117</td>
<td>0.0142</td>
<td>0.0031</td>
<td>26.24</td>
</tr>
<tr>
<td>2-4</td>
<td>0.0328</td>
<td>0.0361</td>
<td>0.0044</td>
<td>13.42</td>
</tr>
<tr>
<td>3-5</td>
<td>0.0328</td>
<td>0.0361</td>
<td>0.0043</td>
<td>13.13</td>
</tr>
<tr>
<td>4-6</td>
<td>0.0117</td>
<td>0.0142</td>
<td>0.0031</td>
<td>26.27</td>
</tr>
<tr>
<td>1-4</td>
<td>0.0774</td>
<td>0.0885</td>
<td>0.0115</td>
<td>14.82</td>
</tr>
<tr>
<td>2-5</td>
<td>0.0899</td>
<td>0.1086</td>
<td>0.0189</td>
<td>21.03</td>
</tr>
<tr>
<td>3-6</td>
<td>0.0775</td>
<td>0.0885</td>
<td>0.0114</td>
<td>14.66</td>
</tr>
<tr>
<td>1-5</td>
<td>0.0409</td>
<td>0.0490</td>
<td>0.084</td>
<td>20.66</td>
</tr>
<tr>
<td>2-6</td>
<td>0.0408</td>
<td>0.0490</td>
<td>0.0085</td>
<td>20.83</td>
</tr>
<tr>
<td>1-6</td>
<td>0.1535</td>
<td>0.1987</td>
<td>0.0455</td>
<td>29.62</td>
</tr>
</tbody>
</table>

5.4.2 A Low Earth Satellite Network on a Single Orbit

Low earth orbit (LEO) satellite networks have been modeled assuming stationary arrival rates (see Zaim et al [41]). In this section, we use FPA to obtain the time-dependent call blocking probability in a LEO satellite networks.

The low earth orbit (LEO) satellite system is a set of identical satellites placed in orbits at an altitude of less than 2000 km. The satellites on the same orbit move in a synchronized way around the earth. Such a set of satellites is referred to as constellation.
of satellites. Each satellite has a fixed size coverage area on the earth called a footprint that moves on the earth over the same trajectory of the satellite.

Each LEO satellite has two types of links: inter-satellite links (ISLs) and up-down links (UDLs). The number of channels in an UDL, $C_{UDL}$, and the number of channels in an ISL, $C_{ISL}$, are fixed and predetermined for each satellite. The UDLs allow users to connect to the LEO satellite to initiate a call. The ISLs permit the satellites to communicate with each other directly. The ISLs are used when two users in different footprints are communicating with each other. A call may occupy two channels of UDL from a LEO satellite when both users remain within the footprint of the satellite during the entire call. Otherwise, the call occupies one of the UDL channels from the LEO satellite of the first user, one of the UDL channels from the satellite of the second user and a number of ISL channels in tandem to connect the first and second satellites. Figure 5.6 illustrate the link occupancy of both cases.

![Figure 5.6: (a) Link occupancy over a single footprint. (b) Link occupancy over two footprints.](image)

In the queueing model of the satellite network, each UDL is modeled as an Erlang loss queue with capacity $C_{UDL}$. Similarly, each ISL is modeled as an Erlang loss
queue with capacity $C_{\text{ISL}}$. A call is admitted if there is a channel available at each link along its path. Otherwise, the call is blocked. The FPA is applied on a numerical example of five satellites network on a single orbit.

### 5.4.2.1 A Numerical Example

Consider a constellation of five identical LEO satellites orbiting around the earth in a clockwise direction as shown in Figure 5.7. Each satellite has a footprint with a capacity of 40 channels. In addition, each satellite is connected to the adjacent satellites with ISL’s each with a capacity of 20 channels. Calls between different footprints are routed over the shortest path from source to destination. For example, a call between footprint 1 and footprint 4 will be routed along ISL 1-5 and ISL 4-5.

Calls arrive to the network in different classes according to their source-destination pair. The set of all possible source-destination pairs is $\{(1,1), (1,2), (1,3), (1,4), (1,5), (2,2), (2,3), (2,4), (2,5), (3,3), (3,4), (3,5), (4,4), (4,5), (5,5)\}$. A call of any class spends an exponential time with rate $\mu = 100$. Let $\tau$ be the time it takes a satellite to travel a distance on earth equal to the diameter of its footprint. Due to the synchronized movement of the satellites, the arrival rate of any class to satellite $i$ are defined as follows:

- $\lambda_{11}(t) = 500 + 300 \sin(0.5t) = \lambda_{35}(t+\tau) = \lambda_{44}(t+2\tau) = \lambda_{33}(t+3\tau) = \lambda_{22}(t+4\tau)$,
- $\lambda_{12}(t) = 300 + 200 \sin(0.5t) = \lambda_{45}(t+\tau) = \lambda_{34}(t+2\tau) = \lambda_{23}(t+3\tau)$,
- $\lambda_{13}(t) = 200 + 100 \sin(0.5t) = \lambda_{35}(t+\tau) = \lambda_{24}(t+2\tau)$,
- $\lambda_{14}(t) = 200 + 100 \sin(0.5t) = \lambda_{25}(t+\tau)$,
- $\lambda_{15}(t) = 300 + 200 \sin(0.5t)$.

In this example, we assume that the there is no handoff calls and $\tau = 0.4$. 

124
Define the random variable $n_{ij}$ as the number of active calls between satellite $i$ and $j$. Then, the state space of the network is

$$S(s) = \{ \mathbf{n} \in \mathbb{N}^{15} : \mathbf{A} \mathbf{n}^\top \leq s^\top, \mathbf{n} \geq 0 \}$$

where $\mathbf{n} = (n_{11}, n_{12}, n_{13}, n_{14}, n_{15}, n_{22}, n_{23}, n_{24}, n_{25}, n_{33}, n_{34}, n_{35}, n_{44}, n_{45}, n_{55})$. 
Figure 5.8 shows the results for all classes of calls that use Satellite 1 in its path. Each graph plots the approximate time-dependent blocking probability and the 95% confidence interval obtained by simulation for the classes (1,1), (1,2), (1,3), (1,4) and (1,5) respectively. In general, the FPA curves follow the shape of the time-dependent blocking probability. This means that FPA calculates the proper values of $t$ for the maxima and minima of the blocking probabilities. In addition, the FPA results lie about in the middle of the confidence intervals in most of the classes. For some classes, such as in plot (e) of Figure 5.8, the FPA curve is close to the upper bound of the confidence intervals, which is still considered an acceptable approximation.

Table 5.2 summarizes the average blocking probability using simulation and FPA for each class. Table 5.2 also shows average absolute and relative error between the simulation and FPA. As before, the relative error is high for those classes that encounter a very small average blocking probability, such as class (2,4).
Figure 5.8: Blocking probability using FPA and the 95% confidence interval of simulation for the classes (1,1), (1,2), (1,3), (1,4) and (1,5)
Table 5.2: Simulate and approximate average blocking probability and relative and average absolute error for each class in the satellite network in Figure 5.7.

<table>
<thead>
<tr>
<th>Class</th>
<th>Average BP (Simulation)</th>
<th>Average BP (Approximation)</th>
<th>Average Abs. Error</th>
<th>Relative Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>0.0193</td>
<td>0.0190</td>
<td>0.0025</td>
<td>13.15</td>
</tr>
<tr>
<td>1-2</td>
<td>0.0146</td>
<td>0.0146</td>
<td>0.0024</td>
<td>16.23</td>
</tr>
<tr>
<td>1-3</td>
<td>0.0168</td>
<td>0.0170</td>
<td>0.0026</td>
<td>15.29</td>
</tr>
<tr>
<td>1-4</td>
<td>0.0178</td>
<td>0.0237</td>
<td>0.0063</td>
<td>35.48</td>
</tr>
<tr>
<td>1-5</td>
<td>0.0175</td>
<td>0.0232</td>
<td>0.0058</td>
<td>33.28</td>
</tr>
<tr>
<td>2-2</td>
<td>0.0119</td>
<td>0.0119</td>
<td>0.0019</td>
<td>15.97</td>
</tr>
<tr>
<td>2-3</td>
<td>0.0126</td>
<td>0.0129</td>
<td>0.0020</td>
<td>15.69</td>
</tr>
<tr>
<td>2-4</td>
<td>0.0091</td>
<td>0.0145</td>
<td>0.0057</td>
<td>63.21</td>
</tr>
<tr>
<td>2-5</td>
<td>0.0149</td>
<td>0.0209</td>
<td>0.0065</td>
<td>43.61</td>
</tr>
<tr>
<td>3-3</td>
<td>0.0159</td>
<td>0.0159</td>
<td>0.0023</td>
<td>14.12</td>
</tr>
<tr>
<td>3-4</td>
<td>0.0154</td>
<td>0.0157</td>
<td>0.0023</td>
<td>14.76</td>
</tr>
<tr>
<td>3-5</td>
<td>0.0165</td>
<td>0.0233</td>
<td>0.0072</td>
<td>43.77</td>
</tr>
<tr>
<td>4-4</td>
<td>0.0179</td>
<td>0.0177</td>
<td>0.0024</td>
<td>13.61</td>
</tr>
<tr>
<td>4-5</td>
<td>0.0168</td>
<td>0.0167</td>
<td>0.0025</td>
<td>14.71</td>
</tr>
<tr>
<td>5-5</td>
<td>0.0186</td>
<td>0.0181</td>
<td>0.0025</td>
<td>13.20</td>
</tr>
</tbody>
</table>

Although there are cases where the FPA over estimates the blocking probability, the CPU time that FPA takes makes it preferable on the simulation models. For example, a problem of size the satellite network in this section took a couple of minutes on a computer with Pentium 4 processor with 1.5 GH of speed. Running the simulation model of the same satellite network took about 45 minutes on the same computer.

It is worth mentioning that the FPA method has limitations. The first limitation to consider is the size of the network. It is expected that as the number of nodes in the network increases the FPA algorithm longer to run. Also, FPA is numerically unstable when insufficiently small $\Delta t$ is used and the arrival and/or service rate is high. Finally,
the value of $\Delta t$ and $T_f$ should be carefully chosen so that the algorithm does not run out of memory or requires unrealistically large CPU time.

As mentioned earlier, the accuracy of FPA seems acceptable for networks with less dependency among the queues of the network. This is due to the fact that the FPA method is based on the independence assumption. In view of this, it is not surprising that the FPA has high relative error when applied on networks with high cross traffic or complicated constraints matrix $A$. On the other hand, FPA would be more accurate if the constraints matrix $A$ is sparse.
Chapter 6

Conclusion and Future Research

There are many real life communication problems that experience nonstationary arrival rates. This gives rise to the need for obtaining time-dependent performance measures. Our focus in this research is on the Erlang loss queue and networks of Erlang loss queues with periodic arrival rates. Specifically, Poisson arrival processes with sinusoidal arrival rate functions.

From Chapter 1, we conclude that most of methods that are used for nonstationary queues rely on queueing theory of stationary queues. Some of these methods, such as the SSA and the PK methods, approximate the nonstationary arrival process by a stationary process which permit them to apply the queueing results of stationary queues in order to obtain performance measures. Other methods, such as the MOL and PSA methods, approximate the offered load of the nonstationary queue and use it in the stationary queue formulas to obtain the time-dependent performance measures.

In this research, we proposed the FPA algorithms that provide time-dependent performance measures for a single Erlang loss queues and networks of Erlang loss queues. The measures of interest obtained by the FPA algorithm do not require the
solution of the forward equations as functions of time. However, the value of $\Delta t$ and $T_f$ should be carefully chosen so that the algorithm does not run out of memory or requires unrealistically long CPU time. The accuracy tests performed on FPA proves that the FPA method provides an exact solution for nonstationary Erlang loss queues and network of nonstationary Erlang loss queues with Markovian branching.

The FPA algorithm was also used to solve approximately population constrained networks of nonstationary Erlang loss queues. Numerical results show that the FPA method has a good accuracy.

6.1 Future Work

Our work on FPA can be extended in two directions. As we mentioned earlier, the nonstationary Erlang loss queues lose the property of insensitivity to service distribution that it has when the arrival rates are stationary. A good extension for the problem would be to consider the nonstationary Erlang loss queues with general service rate, namely $M(t)/G/s/0$. Another available area for research is to obtain time-dependent performance measures for Markovian queues with finite or infinite buffers, namely $M(t)/M/s/k$ where $k \rightarrow \infty$. 
References


Appendices
Appendix A

Plots for Numerical Examples in Chapter 3
Figure A.1: Results of Erlang loss system with $\lambda(t)=5+5\sin(2t)$, $\mu=1$, and $s=10$ : (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.
Figure A.2: Results of Erlang loss system with $\lambda(t)=10+5\sin(2t)$, $\mu=1$, and $s=10$ : (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.
Figure A.3: Results of Erlang loss system with $\lambda(t)=15+5\sin(2t)$, $\mu=1$, and $s=10$: (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.
Figure A.4: Results of Erlang loss system with $\lambda(t) = 20 + 5\sin(2t)$, $\mu = 1$, and $s = 10$: (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.
Figure A.5: Results of Erlang loss system with $\lambda(t) = 10 + 2\sin(2t)$, $\mu = 1$, and $s = 10$ : (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.
Figure A.6: Results of Erlang loss system with $\lambda(t)=10+6\sin(2t)$, $\mu=1$, and $s=10$: (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.
Figure A.7: Results of Erlang loss system with $\lambda(t) = 10 + 8\sin(2t)$, $\mu = 1$, and $s = 10$: (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.
Figure A.8: Results of Erlang loss system with $\lambda(t)=10+10\sin(2t)$, $\mu=1$, and $s=10$: (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.
Appendix B

Examples of Other Time-Dependent Arrival Rates Functions
To demonstrate the accuracy of the FPA algorithm, we have considered sinusoidal arrival rate functions in all of the numerical examples in this thesis. The first derivative of the sinusoidal function is continuous over all $t$. It is of interest to test the accuracy of the FPA algorithm on Erlang loss queues with arrival rate functions that have non-continuous first derivatives over all $t$.

As mentioned earlier, the FPA algorithm can be applied to any periodic or non-periodic arrival rate functions since it takes the arrival rate function as a function defined at certain points on time equally spaced by $\Delta t$. In this section, we give numerical examples for nonstationary Erlang loss queues with different arrival rate functions.

**Example 1**

Consider an $\text{M}(t)/\text{M}/s/0$ queue with arrival rate function given in Figure B.1.

![Figure B.1: plot of arrival rate function of Example 1](image-url)
The arrival rate function in Figure B.1 is defined for $T = 2\pi$ and $i = 0, 1, 2, \ldots$ as

$$\lambda(t) = \begin{cases} 15, &iT \leq t < (i+1)T \\ 5, & (i+1)T \leq t < (i+2)T \end{cases}$$

The queue has 15 identical servers each with service rate $\mu = 1$. The FPA algorithm is applied to this example with $\Delta t = 0.01$ and $\varepsilon = 0.01$.

<table>
<thead>
<tr>
<th>Table B.1: Error of FPA compared with the error of MOL and PSA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FPA</strong></td>
</tr>
<tr>
<td>$BP(t)$</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td><strong>Abs. Error</strong></td>
</tr>
<tr>
<td><strong>Rel. Error (%)</strong></td>
</tr>
</tbody>
</table>

Figure B.2: Results of Erlang loss system in Example 1: (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities.
Figure B.3: Results of Erlang loss system in Example 1: (a) Approximation error of blocking probability using FPA, MOL and PSA. (b) Approximation error of average number of customers in the system using FPA, MOL and PSA.
Example 2

Consider an $M(t)/M/s/0$ queue with arrival rate function given in Figure B.4.

![Arrival Rate Function](image_url)

**Figure B.4**: plot of arrival rate function of Example 2

The arrival rate function in Figure B.4 is defined for $i = 0, 1, 2, \ldots$ as follows:

\[
\lambda(t) = \begin{cases} 
\frac{20}{\pi} t, & i\pi \leq t < (i+1)\pi \\
40 - \frac{20}{\pi} t, & (i+1)\pi \leq t < (i+2)\pi 
\end{cases}
\]

The queue has 15 identical servers each with service rate $\mu = 1$. The FPA algorithm is applied for this example with $\Delta t = 0.01$ and $\varepsilon = 0.01$. 

150
Table B.2: Error of FPA compared with the error of MOL and PSA

<table>
<thead>
<tr>
<th></th>
<th>FPA</th>
<th>MOL</th>
<th>PSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BP(t)</td>
<td>E[Q(t)]</td>
<td>BP(t)</td>
</tr>
<tr>
<td>Abs. Error</td>
<td>0.0062</td>
<td>0.0483</td>
<td>0.0228</td>
</tr>
<tr>
<td>Rel. Error (%)</td>
<td>8.70</td>
<td>0.58</td>
<td>32.04</td>
</tr>
</tbody>
</table>

Figure B.5: Results of Erlang loss system in Example 2: (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.
Example 3

Consider an $M(t)/M/s/0$ queue with arrival rate function given in Figure C.3.

The arrival rate function in Figure C.3 is defined for $i \pi = 2\pi$ and $i = 0, 1, 2, \ldots$ as

$$\lambda(t) = 10 + 5 \sin(t), \quad iT \leq t < (i+1)T$$

$$= 10 + 5 \sin(3t), \quad (i+1)T \leq t < (i+2)T$$

The queue has 15 identical servers each with service rate $\mu = 1$. The FPA algorithm is applied for this example with $\Delta t = 0.01$ and $\varepsilon = 0.01$.

<table>
<thead>
<tr>
<th></th>
<th>FPA</th>
<th>MOL</th>
<th>PSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$BP(t)$</td>
<td>0.0057</td>
<td>0.0112</td>
<td>0.0413</td>
</tr>
<tr>
<td>$E[Q(t)]$</td>
<td>0.0281</td>
<td>0.0719</td>
<td>2.0585</td>
</tr>
<tr>
<td>$BP(t)$</td>
<td>0.0112</td>
<td>0.0719</td>
<td>2.0585</td>
</tr>
<tr>
<td>$E[Q(t)]$</td>
<td>0.0719</td>
<td>0.77</td>
<td>22.52</td>
</tr>
</tbody>
</table>

Table B.3: Error of FPA compared with the error of MOL and PSA
Figure B.6: Results of Erlang loss system in Example 3: (a) Exact and FPA blocking probabilities. (b) Exact, MOL and PSA blocking probabilities. (c) Approximation error of blocking probability using FPA, MOL and PSA. (d) Approximation error of average number of customers in the system using FPA, MOL and PSA.
Appendix C

Plots for Numerical Examples in Chapter 4
Figure C.1: Simulation and FPA results blocking for class-1 arrivals

(a) Class-1 blocking probability at queue 1

(b) Class-1 blocking probability at queue 2

(c) Class-1 blocking probability at queue 3

(d) Class-1 blocking probability at queue 4

(e) Class-1 blocking probability at queue 5

(f) Class-1 blocking probability at queue 6
(a) Class-2 blocking probability at queue 1  
(b) Class-2 blocking probability at queue 2  
(c) Class-2 blocking probability at queue 3  
(d) Class-2 blocking probability at queue 4  
(e) Class-2 blocking probability at queue 5  
(f) Class-2 blocking probability at queue 6

Figure C.2: Simulation and FPA results blocking for class-2 arrivals
Appendix D

Plots for Numerical Examples in Chapter 5
Figure D.1: Blocking probability using FPA and the 95% confidence interval of simulation for the classes (3,4), (4,5), (5,6) and (3,5)
Figure D.2: Blocking probability using FPA and the 95% confidence interval of simulation for the classes (4,6), (3,6) and (2,6)
Figure D.3: Blocking probability using FPA and the 95% confidence interval of simulation for the classes (2,2), (2,3), (2,4), (2,5), (3,3) and (3,4)
Figure D.4: Blocking probability using FPA and the 95% confidence interval of simulation for the classes (3,5), (4,4), (4,5) and (5,5)
Appendix E

MATLAB and Simulation Codes
function [BPr,EQ,EQ_PSA,BPr_PSA,BPr_MOL,EQ_MOL]=Erlang_t(lda,t,s,mu,tol)

% This function is used to calculate the time-dependent blocking probability and the % time-dependent average number of customers in the % in an M(t)/M/s/0 queue using FPA, % MOL and PSA.

% Input parameters:
% lda: arrival rate function
% t: time vector
% s: number of servers in the queue
% mu: service rate
% tol: tolerance value

% Output parameters:
% BPr: vector of blocking probability using FPA
% EQ: vector of average customers in the queue using FPA
% BPr_PSA: vector of blocking probability using PSA
% EQ_PSA: vector of average customers in the queue using PSA
% BPr_MOL: vector of blocking probability using MOL
% EQ_MOL: vector of average customers in the queue using MOL

iteration=0;
BProld=0;
dt=t(2)-t(1);
EQ(1)=0;
for k=2:length(t)
    EQ(k)=EQ(k-1)+(lda(k-1)-mu*EQ(k-1))*dt;
end
r=EQ;
while (1)
    for k=1:length(t)
        den=0;
        for i=0:s
            den=den+r(k)^i/factorial(i);
        end
        BPr(k) = (r(k)^s/factorial(s))/den;
    end
    iteration=iteration+1;
    maxerror(iteration)=norm(BPr-BProld)/norm(BProld);
    BProld=BPr;
    for k=2:length(t)
        EQ(k)=EQ(k-1)+dt*(lda(k-1)*(1-BPr(k-1))-mu*EQ(k-1));
    end
    r=EQ./(1-BPr);
    figure(1)
    str=['Iteration no.',num2str(iteration),'    Max Error = ',num2str(maxerror(iteration))];
    subplot(211),plot(t,EQ),xlabel('Time'),ylabel('EQ(t)'),title(str),grid,drawnow
    subplot(212),plot(t,BPr),xlabel('Time'),ylabel('Ps(t)'),'grid,drawnow
    P(iteration,:)=BPr;
    M(iteration,:)=EQ;
if (maxerror(iteration)<=tol), break, end
end

figure(2)
str=[‘Iteration no.’,num2str(iteration),’    Max Error = ’,num2str(maxerror(iteration))];
subplot(211),plot(t,M),xlabel(‘Time’),ylabel(‘EQ(t)’),title(str),grid,drawnow
subplot(212),plot(t,P),xlabel(‘Time’),ylabel(‘BPr(t)’),grid,drawnow
BPr_MOL=P(1,:);
EQ_MOL=M(1,:);
roh_PSA=lda./mu;
for j=1:length(t)
    den_PSA=0;
    for i=0:s
        den_PSA=den_PSA+roh_PSA(j)^i/factorial(i);
    end
    BPr_PSA(j)=(roh_PSA(j)^s/factorial(s))/den_PSA
    EQ_PSA(j)=roh_PSA(j)*(1-BPr_PSA(j));
end
MATLAB Code for Nonstationary Multi-Rate Erlang Loss Queue

function [BPr,t,EQs,EQC]=ErlangMultiRate_t(dt,Tf,Nr,a,b,g,mu,s,Sr,tol)

% This function is used to calculate the time-dependent blocking probability and average
% number of customers of each class of in a multi-rate M(t)/M/s/0 queue with sinusoidal
% arrival rates.

% Input parameters:
% dt:  time step length
% Tf:  the final time
% Nr:  number of classes
% a,b,g: vectors of parameters for arrival rates of the form a+b*sin(g*t)
% mu:  vector of service rates for each class
% s:  number of servers in the queue
% Sr:  row-vector of number of servers required by class-r customers
% tol:  tolerance value

% Output parameters:
% BPr:  vector of blocking probabilities for each class
% t:  vector contains time steps
% EQ(t,r):average number of class-r customers at time t
% EQS(t): average total number of busy servers at time t
% EQC(t): average total number of customers at time t

% Example [BPr,t,EQ]=ErlangMultiClass(.01,20,2,[10 8],[7 5],[1 2],10,[1 2],.01);

iteration=0;
t=0:dt:Tf;
t=t';
for j=1:Nr
    ArvlRt(:,j)=a(j)+b(j)*sin(g(j)*t);
end
Nt=length(t);
OldBPr=zeros(Nt,Nr);
EQ(1,:)=zeros(1,Nr);
for k=2:Nt
    for j=1:Nr
        EQ(k,j)=EQ((k-1),j)+(ArvlRt((k-1),j)-mu(j)*EQ(k-1,j))*dt;
    end
end
rho=EQ;
while (1)
    for k=1:length(t)
        q(1)=1;
        for i=2:s+1
            q(i)=0;
            for r=1:Nr
                if i>Sr(r)
                    q(i)=q(i)+rho(k,r)*Sr(r)*q(i-Sr(r))/(i-1);
                end
            end
        end
        for j=1:Nr
            BPr(k,j)=sum(q(s-Sr(j)+2:s+1))/sum(q);
        end
    end
end
iteration=iteration+1;
for j=1:Nr
    normNr(j)=norm(BPr(:,j)-OldBPr(:,j));
end
maxerror(iteration)=max(normNr);
OldBPr=BPr;
for k=2:Nt
    for j=1:Nr
        EQ(k,j)=EQ((k-1),j)+(ArvlRt((k-1),j)*(1-BPr(k-1,j))-mu(j)*EQ(k-1,j))*dt;
    end
end
if iteration==1
    EQMOL=EQ;
    BPrMOL=BPr;
end
for j=1:Nr
    rho(:,j)=EQ(:,j)./(1-BPr(:,j));
end
figure(1)
str=['Iteration no.',num2str(iteration),'    Max Error = ',num2str(maxerror(iteration)));
subplot(211),plot(t,EQ),xlabel('Time'),ylabel('EQ(t)'),title(str),grid,drawnow
subplot(212),plot(t,BPr),xlabel('Time'),ylabel('BPr(t)'),grid,drawnow
if (maxerror(iteration)<=tol), break, end
end
EQC=sum(EQ,2);
EQS=sum(EQ*Sr',2);
figure(2)
str=['Total System Performance'
subplot(211),plot(t,EQC),xlabel('Time'),ylabel('Avg. Customers'),title(str),grid,drawnow
subplot(212),plot(t,EQS),xlabel('Time'),ylabel('Avg. Busy Servers'),grid,drawnow
for i=1:Nr
    figure(i+2)
    str=['Class-',num2str(i),' Customers'];
    subplot(211),plot(t,EQ(:,i)),xlabel('Time'),ylabel('Avg. Customers In System'),title(str),grid,drawnow
    subplot(212),plot(t,BPr(:,i)),xlabel('Time'),ylabel('Blocking Pr.'),grid,drawnow
end
function [BPr,t,EQ,BPrMOL,EQMOL]=ErlangNetwork_t(dt,Tf,NN,a,b,g,mu,s,Q,tol)
% This function is used to compute the time-dependent blocking probability of each node
% in a network of M(t)/M/s/0 queues with Makovian branching.

% Input parameters:
% dt:  time step length
% Tf:  the final time
% NN:  number of queueing nodes
% a,b,g: parameters of the arrival rates of the form a+b*sin(g*t)
% mu:  service rate for each node
% s:  vector of number of servers at each node
% Q:  square matrix represents the transitioning probabilities between queues
% Q(i,j): represents probability of moving to queue j after service at i
% Q(i,j): probability that a customer departs the network from i after service
% tol: tolerance value

% Output parameters:
% BPr:  matrix of blocking probabilities
% t: vector of time
% EQ:  matrix of average busy servers in each node

iteration=0;
t=0:dt:Tf;
t=t';
for j=1:NN
    ArvlRt(:,j)=a(j)+b(j)*sin(g(j)*t);
end
Nt=length(t);
OldBPr=zeros(Nt,NN);
EQ(1,:)=zeros(1,NN);
for k=2:Nt
    for j=1:NN
        DepSum=0;
        for i=1:NN
            DepSum=DepSum+mu(i)*Q(i,j)*EQ((k-1),i)*(i~=j);
        end
        EQ(k,j)=EQ((k-1),j)+((ArvlRt((k-1),j)+DepSum)-mu(j)*EQ(k-1,j))*dt;
    end
end
rho=EQ;
while (1)
    for k=1:length(t)
        for j=1:NN
            E=1;
            for i=1:s(j)
                E=rho(k,j)^E/(i+rho(k,j)^E); %den=den*rho(k,j)^(i-1)/factorial(i);
            end
            BPr(k,j)= E; %rho(k,j)^(s(j)/factorial(s(j))/den
        end
    end
end
iteration=iteration+1
for j=1:NN
    normNN(j)=norm(BPr(:,j)-OldBPr(:,j));
end
maxerror(iteration)=max(normNN);
OldBPr=BPr;
for k=2:Nt
    for j=1:NN
        DepSum=0;
        for i=1:NN
            DepSum=DepSum+mu(i)*Q(i,j)*EQ((k-1),i)*(i~=j);
        end
        EQ(k,j)=EQ((k-1),j)+((ArvlRt((k-1),j)+DepSum)*(1-BPr(k-1,j))-mu(j)*EQ(k-1,j))*dt;
    end
end
if iteration==1
    EQMOL=EQ;
    BPrMOL=BPr;
end
for j=1:NN
    rho(:,j)=EQ(:,j)./(1-BPr(:,j))
end
figure(1)
str=['Iteration no.',num2str(iteration),'    Max Error = ',num2str(maxerror(iteration))];
subplot(211),plot(t,EQ),xlabel('Time'),ylabel('EQ(t)'),title(str),grid,drawnow
subplot(212),plot(t,BPr),xlabel('Time'),ylabel('Blocking Pr.'),grid,drawnow
if (maxerror(iteration)<=tol), break, end
end
for i=1:NN
    figure(i+1)
    str=['Node ',num2str(i)];
    subplot(211),plot(t,EQ(:,i)),xlabel('Time'),ylabel('EQ(t)'),title(str),grid,drawnow
    subplot(212),plot(t,BPr(:,i)),xlabel('Time'),ylabel('Blocking Pr.'),grid,drawnow
end
MATLAB Code for Nonstationary Multi-Rate Erlang Loss Network with Markovian Branching

function [BPr,t,EQ_rj,BPr_MOL]=MultiRateNet2(dt,Tf,Nr,NN,a,b,g,mu,S,br,Q,tol)

% This function used to compute the time-dependent performance measures for nonstationary
% multi-rate Erlang loss network

% Input parameter:
% dt: time step length
% Tf: the final time
% Nr: number of classes in the network
% NN: number of queues in the network
% a,b,g: matrices of parameters of arrival rates of each class to each node,
% arrival rates of the form a+b*sin(g*t).
% mu: matrix of service rates for each class at each node
% S: vector of number of servers at each node
% br: row-vector of number of servers required by class-r customers
% tol: tolerance value

% Output parameters:
% BPr: vector of blocking probabilities for each class
% t: time vector
% EQ_rj: average number of class-r customers at queue j

%Example
% dt=0.01 , Tf=10 , Nr=3 , NN= 3 , tol=0.01
% a=[15 15 0 ;
%     10 0 10 ;
%     0 20 20];
% b=[10 10 0 ;
%    10 0 10 ;
%    0 15 15];
% g=[1.0 1.0 1.0;
%    1.5 1.5 1.5;
%    2.0 2.0 2.0];
% mu=[3 4 3;
%    5 3 5;
%    4 5 6];
% br=[1 2 3]; S=[10 10 10];
% Q=[0.4 0.4 0.2;
%    0.0 0.3 0.7;
%    0.4 0.3 0.3];

iteration=0;
t=0:dt:Tf;
t=t';
Nt=length(t);
OldBPr=zeros(Nt,Nr*NN);
EQ_rj(1,:) = zeros(1, Nr*NN);
for k=2:Nt
  v=0;
  u=0;
  for ii=1:Nr
    for jj=1:NN
      v=v+1;
      E0(ii,jj) = EQ_rj(k-1,v);
    end
  end
  for r=1:Nr
    for j=1:NN
      TotRateIn = a(r,j) + b(r,j)*sin(g(r,j)*t(k-1));
      for i=1:NN
        TotRateIn = TotRateIn + Q(i,j)*mu(r,i)*(i~=j);
      end
      u=u+1;
      EQ_rj(k,u) = E0(r,j) + (TotRateIn - mu(r,j)*E0(r,j))*dt;
    end
  end
end
rho = EQ_rj;
while (1)
  for k=1:length(t)
    for j=1:NN
      v=0;
      for ii=1:Nr
        for jj=1:NN
          v=v+1;
          rho_rj(ii,jj) = rho(k,v);
        end
      end
      w(1)=1;
      for K=2:S(j)+1
        w(K)=0;
        for r=1:Nr
          if K>br(r)
            w(K)=w(K) + rho_rj(r,j)*br(r)*w(K-br(r))/(K-1);
          end
        end
        u=j;
        for r=1:Nr
          BPr(k,u) = sum(w(S(j)-br(r)+2:S(j)+1))/sum(w(1:S(j)+1));
          u=u+NN;
        end
      end
  end
iteration = iteration + 1
for j=1:Nr*NN
  normNr(j) = norm(BPr(:,j) - OldBPr(:,j));
end
maxerror(iteration) = max(normNr);
OldBPr = BPr;
u=0;
for k=2:Nt
  v=0;
  for ii=1:Nr
    for jj=1:NN
      v=v+1;
      E0(ii,jj) = EQ_rj(k-1,v);
    end
  end
end
for r=1:Nr

for j=1:NN
    u=u+1;
    TotRateIn=a(r,j)+b(r,j)*sin(g(r,j)*t(k-1));
    for i=1:NN
        TotRateIn=TotRateIn+Q(i,j)*mu(r,i)*E0(r,i)*(i~=j);
    end
    EQ_rj(k,u)=E0(r,j)+TotRateIn*(1-BPr(k,u))*dt-mu(r,j)*E0(r,j)*dt;
end
end
end
if iteration==1
    E_MOL=EQ_rj;
    BPr_MOL=BPr;
end
for j=1:Nr*NN
    rho(:,j)=EQ_rj(:,j)./(1-BPr(:,j));
end
if (maxerror(iteration)<=tol), break, end
end
u=0;
for r=1:Nr
    for j=1:NN
        u=u+1;
        str=['Class-',num2str(r),' Customers at Queue ',num2str(j)];
        subplot(211),plot(t,EQ_rj(:,u)),xlabel('Time'),ylabel('Avg. No. of Customers'),title(str),grid,drawnow
        subplot(212),plot(t,BPr(:,u)),xlabel('Time'),ylabel('Blocking Pr.'),grid,drawnow
    end
end
function [BPr,EQ]=CircuitNetwork7(dt,Tf,A,a,b,g,mu,s,tol)
% This function used to compute the time-dependent performance measures for nonstationary
% Erlang loss networks with constrained population
% Input parameter:
% dt: time step length
% Tf: the final time
% A: square matrix represents the link requirements for each route
% a,b,g: vectors of parameters of the arrival rates, where lambda(t)=a+b*sin(g*t)
% mu : vector of service rates for each class
% s: vector of number of servers in each queue
% tol : the tolerance
% Output parameter:
% BPr: time-dependent blocking probability of each class in the network
% EQ: time-dependent average number of customers of each class in the network
% Example:
% dt=.01, Tf=20, tol=.01
% A=[1 0 0 1 0 1; 0 1 0 1 1 1; 0 0 1 0 1 1],
% a=[5 5 5 4 4 3], b=[4 4 4 3 3 3], g=[1 2 3 1 1 2]
% mu=[1 2 1 2 1 2], s=[10 10 10]

% Example:
% dt=.01, Tf=20, tol=.01
% A=[1 0 0 1 0 1; 0 1 0 1 1 1; 0 0 1 0 1 1],
% a=[5 5 5 4 4 3], b=[4 4 4 3 3 3], g=[1 2 3 1 1 2]
% mu=[1 2 1 2 1 2], s=[10 10 10]

t=0:dt:Tf;
t=t';
NLinks=length(a);
NL=round(NLinks);
Nt=length(t);
BPr=zeros(Nt,NRoutes);
OldBPr=zeros(Nt,NRoutes);
EQ=zeros(Nt,NRoutes);
for i=1:NRoutes
    ArvRt(:,i)=a(i)+b(i)*sin(g(i)*t);
end
TotClasses=0;
for Ln=1:NLinks
    k=find(A(Ln,:));
    TotClasses=TotClasses+length(k);
end
LinkBP=zeros(Nt,TotClasses);
iteration=0;
while (1)
    v=0;
    INPr=ones(Nt,NRoutes);
    for k=2:Nt
        for j=1:NRoutes
            EQ(k,j)=EQ((k-1),j)+(ArvRt((k-1),j)*(1-BPr(k-1,j))-mu(j)*EQ(k-1,j))*dt;
        end
    end
    for Ln=1:NLinks
        k=find(A(Ln,:));
        % k is the vector of indecies corespond to nonzero
        %Links
    end
end
% elements of the i-th row of A
NClasses=length(kk);   % Nr : is number classes uses link i.
 rho=EQ(:,kk)./(1-LinkBP(:,v+1:v+NClasses));
P=MultiClass3(dt,t,NClasses,rho,s(Lnk),A{Lnk,kk});
LinkBP(:,v+1:v+NClasses)=P;
v=v+NClasses;
for j=1:NClasses
    INPr(:,kk(j))=(1-P(:,j)).*INPr(:,kk(j));
end

% Links
BPr=1-INPr;
iteration=iteration+1
for j=1:NRoutes
    normNr(j)=norm(BPr(:,j)-OldBPr(:,j));
end
maxerror(iteration)=max(normNr)
OldBPr=BPr;
if (maxerror(iteration)<=tol), break, end
for i=1:length(a)
    figure(i)
    subplot(211),plot(t,BPr(:,i)),xlabel('Time'),ylabel('BPr(t)'),grid,drawnow
    subplot(212),plot(t,EQ(:,i)),xlabel('Time'),ylabel('EQ(t)'),grid,drawnow
end

************************** separate function **************************
function BPr=MultiClass3(dt,t,Nr,rho,s,Sr)

    for k=1:length(t)
        q(1)=1;
        for i=2:s+1
            q(i)=0;
            for r=1:Nr
                if i>Sr(r)
                    q(i)=q(i)+rho(k,r)*Sr(r)*q(i-Sr(r))/(i-1);
                end
            end
        end
        for j=1:Nr
            BPr(k,j)=sum(q(s-Sr(j)+2:s+1))/sum(q);   % BPr = Blocking Probabilities
        end
    end
end
SIMAN Code for Numerical Example in Section 5.4.2.1
Using ARENA

; Model statements for module: Create 1
146$ CREATE, 1,0:2*3.142/g(1):NEXT(153$);
153$ TRACE, -1,"-Entity Created\n";
147$ ASSIGN: Period1=1:
         T1=0;
150$ ASSIGN: Picture=Default:NEXT(0$);
; Model statements for module: Assign 1
0$ TRACE, -1,"-Making assignments\n";
154$ ASSIGN: ArrivalRate1=AR(1)+B(1)*sin(g(1)*T1):
         Period1=Period1+1:
         T1=T1+dt:NEXT(2$);
; Model statements for module: Delay 2
5$ TRACE, -1,"-Delaying for time dt\n";
156$ DELAY: dt,,Other:NEXT(0$);
; Model statements for module: Create 2
157$ CREATE, 1,0:2*3.142/g(2):NEXT(164$);
164$ TRACE, -1,"-Entity Created\n";
158$ ASSIGN: Period2=1:
         T2=0;
161$ ASSIGN: Picture=Default:NEXT(1$);
; Model statements for module: Assign 2
1$ TRACE, -1,"-Choosing from 1 options\n";
155$ BRANCH, 1: If,Period1<2*3.14159/(g(1)*dt),5$,Yes;
; Model statements for module: Delay 1
4$ TRACE, -1,"-Delaying for time dt\n";
167$ DELAY: dt,,Other:NEXT(1$);
; Model statements for module: Create 10
168$ CREATE, 1,0:2*3.142/g(3):NEXT(175$);
175$ TRACE, -1,"-Entity Created\n";
169$ ASSIGN: Period3=1:
         T3=0;
172$ ASSIGN: Picture=Default:NEXT(6$);
; Model statements for module: Assign 11
6$ TRACE, -1,"-Making assignments\n";
176$ ASSIGN: ArrivalRate3=AR(3)+B(3)*sin(g(3)*T3):
         Period3=Period3+1:
         T3=T3+dt:NEXT(7$);
; Model statements for module: Choose 23
7$ TRACE, -1,"-Choosing from 1 options\n";
177$ BRANCH, 1: If,Period3<2*3.14159/(g(3)*dt),8$,Yes;
; Model statements for module: Delay 11
8$ TRACE, -1,"-Delaying for time dt\n";
178$ DELAY: dt,,Other:NEXT(6$);
; Model statements for module: Resource 1
; Model statements for module: Resource 4
; Model statements for module: Resource 5
; Model statements for module: Create 23
200$ CREATE, 1,0:2*3.142/g(4):NEXT(207$);
207$ TRACE, -1,"-Entity Created\n";
201$ ASSIGN: Period4=1; 
T4=0;
204$ ASSIGN: Picture=Default:NEXT(9$);
9$ Model statements for module: Assign 18
9$ TRACE, -1,"-Making assignments\n";
208$ ASSIGN: ArrivalRate4=AR(4)+B(4)*sin(g(4)*T4):
Period4=Period4+1;
T4=T4+dt:NEXT(10$);
201$ ASSIGN: Period4=1:
T4=0;
204$ ASSIGN: Picture=Default:NEXT(9$);
9$ Model statements for module: Assign 18
9$ TRACE, -1,"-Making assignments\n";
209$ ASSIGN: ArrivalRate5=AR(5)+B(5)*sin(g(5)*T5):
Period5=Period5+1:
T5=T5+dt:NEXT(13$);
211$ CREATE, 1,0:2*3.142/g(5):NEXT(218$);
214$ ASSIGN: ArrivalRate5=AR(5)+B(5)*sin(g(5)*T5):
Period5=Period5+1:
T5=T5+dt:NEXT(13$);
218$ TRACE, -1,"-Entity Created\n";
221$ DELAY: dt,,Other:NEXT(9$);
11$ DELAY: dt,,Other:NEXT(9$);
18$ DELAY: ServiceTime,,Other:NEXT(19$);
32$ DELAY: ServiceTime,,Other:NEXT(19$);
16$ QUEUE, Link1_Q,0,235$:MARK(QueueTime);
32$ TALLY: Link1_Q Queue Time,INT(QueueTime),1:NEXT(18$);
33$ ASSIGN: SAT1,2:NEXT(32$);
10$ BRANCH, -1: 
If,Period4<2*3.14159/(g(4)*dt),11$;Yes;
13$ BRANCH, -1: 
If,Period4<2*3.14159/(g(4)*dt),11$;Yes;
14$ BRANCH, -1: 
If,Period5<2*3.14159/(g(5)*dt),14$;Yes;
15$ BRANCH, -1: 
With,ArrivalRate1/MaxRates(1),16$;Yes;
16$ QUEUE, Link1_Q,0,235$:MARK(QueueTime);
231$ SEIZE, ,Other:
SAT1,2:NEXT(236$);
236$ ASSIGN: j=j;
232$ TALLY: Link1_Q Queue Time,INT(QueueTime),1:NEXT(18$);
235$ DISPOSE: Yes;
18$ TRACE, -1,"-Delaying for time ServiceTime\n";
237$ DELAY: ServiceTime,,Other:NEXT(19$);
19$ TRACE, -1,"-Releasing resources\n";
238$ RELEASE: SAT1,2:NEXT(32$);
24$ TRACE, -1,"-Disposing entity\n";
239$ DISPOSE: Yes;
19$ TRACE, -1,"-Releasing resources\n";
238$ RELEASE: SAT1,2:NEXT(32$);
24$ TRACE, -1,"-Disposing entity\n";
239$ DISPOSE: Yes;

assign: Period6=1;
T6=0;
assign: Picture=Default:NEXT(77$);

model statements for module: Assign 72

trace, -1,"Making assignments\n";
assign: ArrivalRate6=AR(6)+B(6)*sin(g(6)*T6-4*tt):
Period6=Period6+1:
T6=T6+dt:NEXT(78$);

model statements for module: Choose 47

trace, -1,"Choosing from 1 options\n";
branch, 1:
if,Period6<2*3.14159/(g(6)*dt),79$,Yes;

model statements for module: Delay 44

trace, -1,"Delaying for time dt\n";
delay: dt,,Other:NEXT(77$);

model statements for module: Create 45

create, 1,0:2*3.142/g(7):NEXT(272$);
trace, -1,"Entity Created\n";
assign: Period7=1:
T7=0;
assign: Picture=Default:NEXT(80$);

model statements for module: Assign 73

trace, -1,"Making assignments\n";
assign: ArrivalRate7=AR(7)+B(7)*sin(g(7)*T7-3*tt):
Period7=Period7+1:
T7=T7+dt:NEXT(81$);

model statements for module: Choose 48

trace, -1,"Choosing from 1 options\n";
branch, 1:
if,Period7<2*3.14159/(g(7)*dt),82$,Yes;

model statements for module: Delay 45

trace, -1,"Delaying for time dt\n";
delay: dt,,Other:NEXT(80$);

model statements for module: Create 46

create, 1,0:2*3.142/g(8):NEXT(283$);
trace, -1,"Entity Created\n";
assign: Period8=1:
T8=0;
assign: Picture=Default:NEXT(83$);

model statements for module: Assign 74

trace, -1,"Making assignments\n";
assign: ArrivalRate8=AR(8)+B(8)*sin(g(8)*T8-2*tt):
Period8=Period8+1:
T8=T8+dt:NEXT(84$);

model statements for module: Choose 49

trace, -1,"Choosing from 1 options\n";
branch, 1:
if,Period8<2*3.14159/(g(8)*dt),85$,Yes;

model statements for module: Delay 46

trace, -1,"Delaying for time dt\n";
delay: dt,,Other:NEXT(83$);

model statements for module: Create 47

create, 1,0:2*3.142/g(9):NEXT(294$);
trace, -1,"Entity Created\n";
assign: Period9=1:
T9=0;
assign: Picture=Default:NEXT(86$);

model statements for module: Assign 75

trace, -1,"Making assignments\n";
assign: ArrivalRate9=AR(9)+B(9)*sin(g(9)*T9-tt):
Period9=Period9+1:
T9=T9+dt:NEXT(87$);

; Model statements for module: Choose 50

87$ TRACE, 
   -1,"-Choosing from 1 options\n":;
296$ BRANCH, 1:
   If,Period9<2*3.14159/(g(9)*dt),88$,Yes;

; Model statements for module: Delay 47
88$ TRACE, 
   -1,"-Delaying for time dt\n":;
297$ DELAY: 
   dt,,Other:NEXT(86$);

; Model statements for module: Create 48
298$ CREATE, 
   1,0:2*:3.142/g(10):NEXT(305$);
305$ TRACE, 
   -1,"-Entity Created\n":;
299$ ASSIGN: 
   Period10=1:
   T10=0;
302$ ASSIGN: 
   Picture=Default:NEXT(89$);

; Model statements for module: Assign 76
89$ TRACE, 
   -1,"-Making assignments\n":;
306$ ASSIGN: 
   ArrivalRate10=AR(10)+B(10)*sin(g(10)*T10-3*tt):
   Period10=Period10+1:
   T10=T10+dt:NEXT(90$);

; Model statements for module: Choose 51
90$ TRACE, 
   -1,"-Choosing from 1 options\n":;
307$ BRANCH, 1:
   If,Period10<2*3.14159/(g(10)*dt),91$,Yes;

; Model statements for module: Delay 48
91$ TRACE, 
   -1,"-Delaying for time dt\n":;
308$ DELAY: 
   dt,,Other:NEXT(89$);

; Model statements for module: Create 49
309$ CREATE, 
   1,0:2*:3.142/g(11):NEXT(316$);
316$ TRACE, 
   -1,"-Entity Created\n":;
310$ ASSIGN: 
   Period11=1:
   T11=0;
313$ ASSIGN: 
   Picture=Default:NEXT(92$);

; Model statements for module: Assign 77
92$ TRACE, 
   -1,"-Making assignments\n":;
317$ ASSIGN: 
   ArrivalRate11=AR(11)+B(11)*sin(g(11)*T11-2*tt):
   Period11=Period11+1:
   T11=T11+dt:NEXT(93$);

; Model statements for module: Choose 52
93$ TRACE, 
   -1,"-Choosing from 1 options\n":;
318$ BRANCH, 1:
   If,Period11<2*3.14159/(g(11)*dt),94$,Yes;

; Model statements for module: Delay 49
94$ TRACE, 
   -1,"-Delaying for time dt\n":;
319$ DELAY: 
   dt,,Other:NEXT(92$);

; Model statements for module: Create 50
320$ CREATE, 
   1,0:2*:3.142/g(12):NEXT(327$);
327$ TRACE, 
   -1,"-Entity Created\n":;
321$ ASSIGN: 
   Period12=1:
   T12=0;
324$ ASSIGN: 
   Picture=Default:NEXT(95$);

; Model statements for module: Assign 78
95$ TRACE, 
   -1,"-Making assignments\n":;
328$ ASSIGN: 
   ArrivalRate12=AR(12)+B(12)*sin(g(12)*T12-tt):
   Period12=Period12+1:
   T12=T12+dt:NEXT(96$);

; Model statements for module: Choose 53
96$ TRACE, 
   -1,"-Choosing from 1 options\n":;
329$ BRANCH, 1:
   If,Period12<2*3.14159/(g(12)*dt),97$,Yes;

; Model statements for module: Delay 50
97$ TRACE, 
   -1,"-Delaying for time dt\n":;
330$ DELAY: 
   dt,,Other:NEXT(95$);

; Model statements for module: Create 52
331$ CREATE, 
   1,0:2*:3.142/g(13):NEXT(338$);
338$   TRACE, -1,"-Entity Created\n";;
332$   ASSIGN:  Period13=1;
      T13=0;
335$   ASSIGN:  Picture=Default:NEXT(98$);
;   Model statements for module:  Assign 80
98$   TRACE, -1,"-Making assignments\n";;
339$   ASSIGN:  ArrivalRate13=AR(13)+B(13)*sin(g(13)*T13-2*tt):
            Period13=Period13+1;
            T13=T13+dt:NEXT(99$);
;   Model statements for module:  Choose 55
99$   TRACE, -1,"-Choosing from 1 options\n";;
340$   ASSIGN:  ArrivalRate14=AR(14)+B(14)*sin(g(14)*T14-tt):
            Period14=Period14+1;
            T14=T14+dt:NEXT(100$);
;   Model statements for module:  Choose 53
100$  TRACE, -1,"-Choosing from 1 options\n";;
341$   ASSIGN:  ArrivalRate15=AR(15)+B(15)*sin(g(15)*T15-tt):
            Period15=Period15+1:
            T15=T15+dt:NEXT(101$);
;   Model statements for module:  Choose 56
101$  TRACE, -1,"-Choosing from 1 options\n";;
342$   ASSIGN:  ArrivalRate16=AR(16)+B(16)*sin(g(16)*T16-tt):
            Period16=Period16+1:
            T16=T16+dt:NEXT(102$);
;   Model statements for module:  Choose 54
102$  TRACE, -1,"-Choosing from 1 options\n";;
343$   ASSIGN:  ArrivalRate17=AR(17)+B(17)*sin(g(17)*T17-tt):
            Period17=Period17+1:
            T17=T17+dt:NEXT(103$);
;   Model statements for module:  Choose 57
103$  TRACE, -1,"-Choosing from 1 options\n";;
344$   ASSIGN:  ArrivalRate18=AR(18)+B(18)*sin(g(18)*T18-tt):
            Period18=Period18+1:
            T18=T18+dt:NEXT(104$);
;   Model statements for module:  Choose 55
104$  TRACE, -1,"-Choosing from 1 options\n";;
345$   ASSIGN:  ArrivalRate19=AR(19)+B(19)*sin(g(19)*T19-tt):
            Period19=Period19+1:
            T19=T19+dt:NEXT(105$);
;   Model statements for module:  Choose 56
105$  TRACE, -1,"-Choosing from 1 options\n";;
346$   ASSIGN:  ArrivalRate20=AR(20)+B(20)*sin(g(20)*T20-tt):
            Period20=Period20+1:
            T20=T20+dt:NEXT(106$);
;   Model statements for module:  Choose 54
106$  TRACE, -1,"-Choosing from 1 options\n";;
347$   ASSIGN:  ArrivalRate21=AR(21)+B(21)*sin(g(21)*T21-tt):
            Period21=Period21+1:
            T21=T21+dt:NEXT(107$);
;   Model statements for module:  Choose 57
107$  TRACE, -1,"-Choosing from 1 options\n";;
With,ArrivalRate2/MaxRates(2),20$,Yes;
20$
 QUEUE, Link2_Q,0,412$:MARK(QueueTime);
408$
 SEIZE, ,Other:
 SAT1,1:
 Link12,1:
 SAT2,1:NEXT(413$);
413$
 ASSIGN: j=j;
 j=j;
 j=j;
409$
 TALLY: Link2_Q Queue Time,INT(QueueTime),1:NEXT(22$);
412$
 DISPOSE: Yes;
 ; Model statements for module:  Delay 22
22$
 TRACE, -1,"-Delaying for time ServiceTime\n";;
414$
 DELAY: ServiceTime,,Other:NEXT(23$);
 ; Model statements for module:  Release 2
23$
 TRACE, -1,"-Releasing resources\n";;
415$
 RELEASE: SAT1,1:
 Link12,1:
 SAT2,1:NEXT(32$);
 ; Model statements for module:  Create 68
416$
 CREATE, 1,.001:EXPO( 1/MaxRates(3),5):NEXT(423$);
423$
 TRACE, -1,"-Entity Created\n";;
417$
 ASSIGN: ServiceTime=EXPO( 1/100,2);
420$
 ASSIGN: Picture=Default:NEXT(108$);
 ; Model statements for module:  Chance 39
108$
 TRACE, -1,"-Choosing from 1 options\n";;
424$
 BRANCH, 1:
 With,ArrivalRate3/MaxRates(3),24$,Yes;
 ; Model statements for module:  Seize 3
24$
 QUEUE, Link3_Q,0,429$:MARK(QueueTime);
425$
 SEIZE, ,Other:
 SAT1,1:
 Link12,1:
 Link23,1:
 SAT3,1:NEXT(430$);
430$
 ASSIGN: j=j;
 j=j;
 j=j;
426$
 TALLY: Link3_Q Queue Time,INT(QueueTime),1:NEXT(26$);
429$
 DISPOSE: Yes;
 ; Model statements for module:  Delay 23
26$
 TRACE, -1,"-Delaying for time ServiceTime\n";;
431$
 DELAY: ServiceTime,,Other:NEXT(27$);
 ; Model statements for module:  Release 3
27$
 TRACE, -1,"-Releasing resources\n";;
432$
 RELEASE: SAT1,1:
 Link12,1:
 Link23,1:
 SAT3,1:NEXT(32$);
 ; Model statements for module:  Create 69
433$
 CREATE, 1,.001:EXPO( 1/MaxRates(4),6):NEXT(440$);
440$
 TRACE, -1,"-Entity Created\n";;
434$
 ASSIGN: ServiceTime=EXPO( 1/100,2);
437$
 ASSIGN: Picture=Default:NEXT(109$);
 ; Model statements for module:  Chance 40
109$
 TRACE, -1,"-Choosing from 1 options\n";;
441$
 BRANCH, 1:
 With,ArrivalRate4/MaxRates(4),41$,Yes;
 ; Model statements for module:  Seize 7
41$
 QUEUE, Link3_Q1,0,446$:MARK(QueueTime);
442$
 SEIZE, ,Other:
 SAT1,1:
 Link15,1:
447$ Assign: j=j;
448$ Tally: Link3_Q1 Queue Time, INT(QueueTime), 1:NEXT(448$);
449$ Dispose: Yes;

; Model statements for module: Delay 27
439$ Trace, -1,"-Delaying for time ServiceTime\n";
440$ Delay: ServiceTime, Other:NEXT(440$);

; Model statements for module: Release 7
443$ Tally: Link4_Q1 Queue Time, INT(QueueTime), 1:NEXT(443$);
444$ Dispose: Yes;

; Model statements for module: Create 70
450$ Create, 1,.001:EXPO( 1/MaxRates(5), 7):NEXT(450$); 
451$ Trace, -1,"-Entity Created\n";
452$ Assign: ServiceTime=EXPO( 1/100, 2);
453$ Assign: Picture=Default:NEXT(453$);

; Model statements for module: Chance 41
110$ Trace, -1,"-Choosing from 1 options\n";
110$ Branch, 1:
   With, ArrivalRate5/MaxRates(5), 45$: Yes;

; Model statements for module: Seize 8
448$ Queue, Link4_Q2, 0, 463$: MARK(QueueTime);
449$ Seize, , Other:
   SAT1, 1:
   Link15, 1:
   SAT5, 1:NEXT(449$);
450$ Assign: j=j;
451$ Tally: Link4_Q2 Queue Time, INT(QueueTime), 1:NEXT(451$);
452$ Dispose: Yes;

; Model statements for module: Delay 28
470$ Trace, -1,"-Delaying for time ServiceTime\n";
471$ Delay: ServiceTime, Other:NEXT(471$);

; Model statements for module: Release 8
472$ Trace, -1,"-Releasing resources\n";
473$ Release: SAT1, 1:
   Link15, 1:
   SAT5, 1:NEXT(473$);

; Model statements for module: Create 71
474$ Create, 1,.001:EXPO( 1/MaxRates(6), 3):NEXT(474$); 
475$ Trace, -1,"-Entity Created\n";
476$ Assign: ServiceTime=EXPO( 1/100, 2);
477$ Assign: Picture=Default:NEXT(477$);

; Model statements for module: Chance 42
111$ Trace, -1,"-Choosing from 1 options\n";
112$ Branch, 1:
   With, ArrivalRate6/MaxRates(6), 28$: Yes;

; Model statements for module: Seize 4
28$ Queue, Link1_Q3, 0, 480$: MARK(QueueTime);
29$ Seize, , Other:
   SAT2, 2:NEXT(480$);
30$ Assign: j=j;
31$ Tally: Link1_Q3 Queue Time, INT(QueueTime), 1:NEXT(31$);
32$ Dispose: Yes;

; Model statements for module: Delay 24
33$ Trace, -1,"-Delaying for time ServiceTime\n";
34$ Delay: ServiceTime, Other:NEXT(31$);
Model statements for module: Create 72
CREATE, 1,.001:EXPO( 1/MaxRates(7),4):NEXT(491$);
TRACE, -1,"-Entity Created"
ASSIGN: ServiceTime=EXPO( 1/100,2);
ASSIGN: Picture=Default:NEXT(488$);
Model statements for module: Chance 43
TRACE, -1,"-Choosing from 1 options"
BRANCH, 1:
With,ArrivalRate7/MaxRates(7),33$,Yes;
Model statements for module: Seize 5
QUEUE, Link2_Q4,0,497$:MARK(QueueTime);
SEIZE, ,Other:
SAT2,1:
Link23,1:
SAT3,1:NEXT(498$);
ASSIGN: j=j:
j=j:
j=j:
TALLY: Link2_Q4 Queue Time,INT(QueueTime),1:NEXT(494$);
DISPOSE: Yes;
Model statements for module: Delay 25
DELAY: ServiceTime,,Other:NEXT(499$);
Model statements for module: Release 5
TRACE, -1,"-Releasing resources"
RELEASE: SAT2,1:
Link23,1:
SAT3,1:NEXT(32$);
Model statements for module: Create 73
CREATE, 1,.001:EXPO( 1/MaxRates(8),5):NEXT(508$);
TRACE, -1,"-Entity Created"
ASSIGN: ServiceTime=EXPO( 1/100,2);
ASSIGN: Picture=Default:NEXT(505$);
Model statements for module: Chance 44
TRACE, -1,"-Choosing from 1 options"
BRANCH, 1:
With,ArrivalRate8/MaxRates(8),49$,Yes;
Model statements for module: Seize 9
QUEUE, Link2_Q5,0,514$:MARK(QueueTime);
SEIZE, ,Other:
SAT2,1:
Link23,1:
Link34,1:
SAT4,1:NEXT(515$);
ASSIGN: j=j:
j=j:
j=j:
j=j:
TALLY: Link2_Q5 Queue Time,INT(QueueTime),1:NEXT(511$);
DISPOSE: Yes;
Model statements for module: Delay 37
DELAY: ServiceTime,,Other:NEXT(516$);
Model statements for module: Release 9
TRACE, -1,"-Releasing resources"
RELEASE: SAT2,1:
Link23,1:
Link34,1:
SAT4,1:NEXT(32$);
Model statements for module: Create 74
CREATE, 1,.001:EXPO( 1/MaxRates(9),6):NEXT(525$);
TRACE, -1,"-Entity Created"
ASSIGN: ServiceTime=EXPO( 1/100,2);
ASSIGN: Picture=Default:NEXT(114$);

MODEL statements for module: Chance 45

TRACE, -1,"-Choosing from 1 options\n"

BRANCH, 1:
With, ArrivalRate9/MaxRates(9), 53$, Yes;

MODEL statements for module: Seize 10

QUEUE, Link2_Q6, 0, 531$: MARK(QueueTime);

SEIZE, , Other:
SAT2, 1:
Link12, 1:
Link15, 1:
SAT5, 1:NEXT(532$);

ASSIGN: j=j;

TALLY: Link2_Q6 Queue Time, INT(QueueTime), 1:NEXT(55$);

DISPOSE: Yes;

MODEL statements for module: Delay 38

TRACE, -1,"-Delaying for time ServiceTime\n"

DELAY: ServiceTime, Other:NEXT(56$);

MODEL statements for module: Release 10

TRACE, -1, "-Releasing resources\n"

RELEASE: SAT2, 1:
Link12, 1:
Link15, 1:
SAT5, 1:NEXT(32$);

MODEL statements for module: Create 75

CREATE, 1,.001: EXPO( 1/MaxRates(10), 4): NEXT(542$);

TRACE, -1, "-Entity Created\n"

ASSIGN: Picture=Default:NEXT(115$);

MODEL statements for module: Chance 46

TRACE, -1, "-Choosing from 1 options\n"

BRANCH, 1:
With, ArrivalRate10/MaxRates(10), 37$, Yes;

MODEL statements for module: Seize 6

QUEUE, Link3_Q7, 0, 548$: MARK(QueueTime);

SEIZE, , Other:
SAT3, 2:NEXT(549$);

ASSIGN: j=j;

TALLY: Link3_Q7 Queue Time, INT(QueueTime), 1:NEXT(39$);

DISPOSE: Yes;

MODEL statements for module: Delay 26

TRACE, -1, "-Delaying for time ServiceTime\n"

DELAY: ServiceTime, Other:NEXT(40$);

MODEL statements for module: Release 6

TRACE, -1, "-Releasing resources\n"

RELEASE: SAT3, 2:NEXT(32$);

MODEL statements for module: Create 76

CREATE, 1,.001: EXPO( 1/MaxRates(11), 5): NEXT(559$);

TRACE, -1, "-Entity Created\n"

ASSIGN: ServiceTime=EXPO( 1/100, 2);

ASSIGN: Picture=Default:NEXT(116$);

MODEL statements for module: Chance 47

TRACE, -1, "-Choosing from 1 options\n"

BRANCH, 1:
With, ArrivalRate11/MaxRates(11), 57$, Yes;

MODEL statements for module: Seize 11

QUEUE, Link3_Q8, 0, 565$: MARK(QueueTime);

SEIZE, , Other:
SAT3, 1:
Link34, 1:
SAT4, 1:NEXT(566$);

ASSIGN: j=j;
$j=j$

$\text{TALLY:}$  \quad \text{Link3\_Q8 Queue Time, INT(\text{Queue Time}), 1: NEXT(595)}$

$\text{DISPOSE:}$  \quad \text{Yes};$

\text{Model statements for module:  Delay 39}$

$\text{TRACE,}$  \quad -1, "-Delaying for time ServiceTime\n"

$\text{DELAY:}$  \quad \text{ServiceTime, Other: NEXT(605)}$

\text{Model statements for module:  Release 11}$

$\text{TRACE,}$  \quad -1, "-Releasing resources\n"

$\text{RELEASE:}$  \quad \text{SAT3, 1: Link34, 1: SAT4, 1: NEXT(325)}$

\text{Model statements for module:  Create 77}$

$\text{CREATE,}$  \quad 1, 0.001: EXPO( 1/\text{MaxRates(12)}, 6): NEXT(575)$

$\text{TRACE,}$  \quad -1, "-Entity Created\n"

$\text{ASSIGN:}$  \quad \text{ServiceTime=EXPO( 1/100, 2)}$

$\text{ASSIGN:}$  \quad \text{Picture=Default: NEXT(117)}$

\text{Model statements for module:  Chance 48}$

$\text{TRACE,}$  \quad -1, "-Choosing from 1 options\n"

$\text{BRANCH,}$  \quad 1: With, ArrivalRate12/MaxRates(12), 615, Yes;

\text{Model statements for module:  Seize 12}$

$\text{QUEUE,}$  \quad \text{Link3\_Q9, 0, 582: MARK(\text{Queue Time})}$

$\text{SEIZE,}$  \quad , Other: SAT3, 1: Link34, 1: Link45, 1: SAT5, 1: NEXT(583)$

$\text{ASSIGN:}$  \quad j=j$

$\text{TALLY:}$  \quad \text{Link3\_Q9 Queue Time, INT(\text{Queue Time}), 1: NEXT(638)}$

$\text{DISPOSE:}$  \quad \text{Yes};$

\text{Model statements for module:  Delay 40}$

$\text{TRACE,}$  \quad -1, "-Delaying for time ServiceTime\n"

$\text{DELAY:}$  \quad \text{ServiceTime, Other: NEXT(643)}$

\text{Model statements for module:  Release 12}$

$\text{TRACE,}$  \quad -1, "-Releasing resources\n"

$\text{RELEASE:}$  \quad \text{SAT3, 1: Link34, 1: Link45, 1: SAT5, 1: NEXT(325)}$

\text{Model statements for module:  Create 78}$

$\text{CREATE,}$  \quad 1, 0.001: EXPO( 1/\text{MaxRates(13)}, 5): NEXT(593)$

$\text{TRACE,}$  \quad -1, "-Entity Created\n"

$\text{ASSIGN:}$  \quad \text{ServiceTime=EXPO( 1/100, 2)}$

$\text{ASSIGN:}$  \quad \text{Picture=Default: NEXT(118)}$

\text{Model statements for module:  Chance 49}$

$\text{TRACE,}$  \quad -1, "-Choosing from 1 options\n"

$\text{BRANCH,}$  \quad 1: With, ArrivalRate13/MaxRates(13), 655, Yes;

\text{Model statements for module:  Seize 13}$

$\text{QUEUE,}$  \quad \text{Link3\_Q10, 0, 599: MARK(\text{Queue Time})}$

$\text{SEIZE,}$  \quad , Other: SAT4, 2: NEXT(600)$

$\text{ASSIGN:}$  \quad j=j$

$\text{TALLY:}$  \quad \text{Link3\_Q10 Queue Time, INT(\text{Queue Time}), 1: NEXT(675)}$

$\text{DISPOSE:}$  \quad \text{Yes};$

\text{Model statements for module:  Delay 41}$

$\text{TRACE,}$  \quad -1, "-Delaying for time ServiceTime\n"

$\text{DELAY:}$  \quad \text{ServiceTime, Other: NEXT(685)}$

\text{Model statements for module:  Release 13}$

$\text{TRACE,}$  \quad -1, "-Releasing resources\n"

$\text{RELEASE:}$  \quad \text{SAT4, 2: NEXT(325)}$
; Model statements for module: Create 79
CREATE, 1,.001:EXPO(1/MaxRates(14),6):NEXT(610$);
TRACE, -1,\"Entity Created\n\";  
ASSIGN: ServiceTime=EXPO(1/100,2);
ASSIGN: Picture=Default:NEXT(119$);

; Model statements for module: Chance 50
TRACE, -1,\"Choosing from 1 options\n\";  
BRANCH, 1:
With,ArrivalRate14/MaxRates(14),69$,Yes;

; Model statements for module: Seize 14
QUEUE, Link3_Q11,0,616$:MARK(QueueTime);
SEIZE, ,Other:
SAT4,1:
Link45,1:
SAT5,1:NEXT(617$);

ASSIGN: j=j;
j=j;
j=j;

TALLY: Link3_Q11 Queue Time,INT(QueueTime),1:NEXT(71$);
DISPOSE: Yes;

; Model statements for module: Delay 42
DELAY: ServiceTime,,Other:NEXT(72$);

TRACE, -1,\"Delaying for time ServiceTime\n\";  
RELEASE: SAT4,1:
Link45,1:
SAT5,1:NEXT(32$);

; Model statements for module: Create 80
CREATE, 1,.001:EXPO(1/MaxRates(15),6):NEXT(627$);
TRACE, -1,\"Entity Created\n\";  
ASSIGN: ServiceTime=EXPO(1/100,2);
ASSIGN: Picture=Default:NEXT(120$);

; Model statements for module: Chance 51
TRACE, -1,\"Choosing from 1 options\n\";  
BRANCH, 1:
With,ArrivalRate15/MaxRates(15),73$,Yes;

; Model statements for module: Seize 15
QUEUE, Link3_Q12,0,633$:MARK(QueueTime);
SEIZE, ,Other:
SAT5,2:NEXT(634$);
ASSIGN: j=j;

TALLY: Link3_Q12 Queue Time,INT(QueueTime),1:NEXT(75$);
DISPOSE: Yes;

; Model statements for module: Delay 43
DELAY: ServiceTime,,Other:NEXT(76$);

TRACE, -1,\"Delaying for time ServiceTime\n\";  
RELEASE: SAT5,2:NEXT(32$);

; Model statements for module: Release 15
CLOSE, CurrentFull;
CLOSE, FullSUM1;
CLOSE, FullSUM2:NEXT(125$);

; Model statements for module: Create 82
CREATE, 1,FSTCollect::NEXT(644$);
TRACE, -1,\"Entity Created\n\";  
ASSIGN: inival1=0:
n=1;
ASSIGN: Picture=Default:NEXT(138$);
CLOSE, CurrentFull;
CLOSE, FullSUM1;
CLOSE, FullSUM2:NEXT(125$);

; Model statements for module: Choose 64
TRACE, -1,\"Choosing from 2 options\n\";  
BRANCH, 1:
If,NREP==1,137$,Yes:
Else,1415,Yes;
Model statements for module: Write 9
```
137$  TRACE, -1, "-Writing to File CurrentFull

646$  WRITE, CurrentFull:
           Blocking(1), Blocking(2), Blocking(3), Blocking(4), Blocking(5),
           Blocking(6), Blocking(7), Blocking(8), Blocking(9),
           Blocking(10), Blocking(11), Blocking(12), Blocking(13),
           Blocking(14), Blocking(15): NEXT(124$);
```
Model statements for module: Write 7
```
124$  TRACE, -1, "-Writing to File FullSUM1

647$  WRITE, FullSUM1:
           inival1, inival1, inival1, inival1, inival1, inival1,
           inival1, inival1, inival1, inival1, inival1, inival1,
           inival1, inival1, inival1:NEXT(124$);
```
Model statements for module: Write 11
```
142$  TRACE, -1, "-Writing to File FullSUM2

648$  WRITE, FullSUM2:
           inival1, inival1, inival1, inival1, inival1, inival1,
           inival1, inival1, inival1, inival1, inival1, inival1,
           inival1, inival1, inival1:NEXT(124$);
```
Model statements for module: Delay 57
```
121$  TRACE, -1, "-Delaying for time dt

649$  DELAY: dt, Other:NEXT(122$);
```
Model statements for module: Choose 63
```
122$  TRACE, -1, "-Choosing from 2 options

650$  BRANCH, 1:
           If, TFIN-TNOW>dt, 125$, Yes:
           Else, 123$, Yes;

123$  CLOSE, CurrentFull;

127$  CLOSE, FullSUM1;

128$  CLOSE, FullSUM2:NEXT(129$);
```
Model statements for module: Read 5
```
130$  TRACE, -1, "-Reading from CurrentFull

652$  READ, CurrentFull:
           Full1, Full2, Full3, Full4, Full5, Full6, Full7, Full8,
           Full9, Full10, Full11, Full12, Full13, Full14, Full15:
           NEXT(131$);
```
Model statements for module: Read 6
```
131$  TRACE, -1, "-Reading from FullSUM1

653$  READ, FullSUM1:
           oldsum1, oldsum2, oldsum3, oldsum4, oldsum5, oldsum6,
           oldsum7, oldsum8, oldsum9, oldsum10, oldsum11, oldsum12,
           oldsum13, oldsum14, oldsum15:NEXT(132$);
```
Model statements for module: Write 8
```
132$  TRACE, -1, "-Writing to File FullSUM2

654$  WRITE, FullSUM2:
           Full1+oldsum1, Full2+oldsum2, Full3+oldsum3,
           Full4+oldsum4, Full5+oldsum5, Full6+oldsum6,
           Full7+oldsum7, Full8+oldsum8, Full9+oldsum9,
           Full10+oldsum10, Full11+oldsum11, Full12+oldsum12,
           Full13+oldsum13, Full14+oldsum14, Full15+oldsum15:NEXT(134$);
```
Model statements for module: Assign 86
```
134$  TRACE, -1, "-Making assignments

655$  ASSIGN: n=n+1:NEXT(135$);
```
Model statements for module: Choose 66
```
135$  TRACE, -1, "-Choosing from 2 options

656$  BRANCH, 1:
           If, n>(TFIN-FSTCollect)/dt, 32$, Yes:
           Else, 130$, Yes;
```
Model statements for module: Read 7
```
143$  TRACE, -1, "-Reading from CurrentFull

READ, CurrentFull:
Full1, Full2, Full3, Full4, Full5, Full6, Full7, Full8, Full9,
Full10, Full11, Full12, Full13, Full14, Full15:NEXT(144$);

; Model statements for module: Read 8
TRACE, -1, "-Reading from FullSUM2 
";
READ, FullSUM2:
oldsum1, oldsum2, oldsum3, oldsum4, oldsum5, oldsum6, oldsum7, oldsum8, oldsum9, oldsum10, oldsum11, oldsum12, oldsum13, oldsum14, oldsum15:NEXT(145$);

; Model statements for module: Write 12
TRACE, -1, "-Writing to File FullSUM1
";
WRITE, FullSUM1:
Full1+oldsum1, Full2+oldsum2, Full3+oldsum3,
Full4+oldsum4, Full5+oldsum5, Full6+oldsum6,
Full7+oldsum7, Full8+oldsum8, Full9+oldsum9,
Full10+oldsum10, Full11+oldsum11, Full12+oldsum12,
Full13+oldsum13, Full14+oldsum14, Full15+oldsum15:NEXT(133$);

; Model statements for module: Assign 85
TRACE, -1, "-Making assignments 
";
ASSIGN:
n=n+1:NEXT(136$);

; Model statements for module: Choose 67
TRACE, -1, "-Choosing from 2 options 
";
BRANCH, 1:
If, n>=(TFIN-FSTCollect)/dt, 32$, Yes:
Else, 143$, Yes;

; Model statements for module: Write 10
TRACE, -1, "-Writing to File CurrentFull 
";
WRITE, CurrentFull:
Blocking(1), Blocking(2), Blocking(3), Blocking(4),
Blocking(5), Blocking(6), Blocking(7), Blocking(8),
Blocking(9), Blocking(10), Blocking(11), Blocking(12),
Blocking(13), Blocking(14), Blocking(15):NEXT(126$);

; Model statements for module: Delay 58
TRACE, -1, "-Delaying for time dt 
";
DELAY:
dt,,Other:NEXT(122$);